The Machine that Lives Forever

Michael Walton

A thesis submitted in partial fulfilment of the requirements of the University of Essex for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

March 2021

Acknowledgements

I would like to take this opportunity to thank all my friends and colleagues who took the time to help me when I needed it during this research. Dr John Woods, for his mentoring and supervision, Ian Dukes, for his excellent platform building skills, along with his tech team, who have all helped me out at some point or another. My beautiful wife Jangei and, of course, son Samuel both of whom have endured my persistent lack of time. Lastly, to Simon Hardy and Graham Chapman for allowing me the time and sponsorship to follow this path.

Abstract

Design an intelligent micromachine that can self-power and sustain from environmental energy scavenging to achieve an autonomous device that can communicate at will with peers indefinitely.

Explore sleep/wake hibernation strategies coupled with food scavenging off-grid traits to identify the tightest work to sleep efficiency schedule, incorporating adaptive reconfiguration to manage significant environmental impacts.

Capture, store and manage background radiations and stray RF signals to feed on in a continued effort to make intelligent survival decisions and oversee management protocols.

Ensure that every micro Watt of usable energy gets extracted from every part of the harvest and then forward-scheduled it for productive use.

Finally, employ natures tricks and experience to introduce essential personality traits, pursuing maximising survival numbers and increasing dispersal target area sizes of large self-sufficient wireless sensor deployments.

This research intends to provide a closely coupled software-hardware foundation that aids implementers in intelligently harnessing and using tiny amounts of ambient energy in a highly autonomous way.

This platform then continues on to explore ways of maximising the efficient usage of the harvested energy using various hibernation/wake strategies and then making objective comparisons with proposed intelligent energy management protocols.

Finally, the protocol extends to enable the device to manage its personal survival possibilities so the devices can use an evolutional personality-based approach to deal with the unknown environmental situations they will encounter.

This work examines a machine that can self-power and sustain from environmental energy scavenging with the aim to live forever. Living forever implies a brain (microcontroller) that can manage energy and budget for continuous faculty. With these objectives, sleep/wake/hibernation and scavenging strategies get examined to efficiently schedule resources within a transient environment. Example harvesting includes induced and background radiation. Intelligent, biologically-inspired strategies are adopted in forward-scheduling strategies given temporal energy relative to the machine's function (the *Walton*).

Publications

- Paper titled “Efficient charging for batteryless solutions in energy harvesting” published at CEEC 2019, the 11th Computer Science and Electronic Engineering Conference, 2019, https://ieeexplore.ieee.org/document/8974344
- Paper titled “The Machine that Lives Forever – BluBot” published in the COJ Electronic Communications Journal on 10/02/2020.
- Paper titled “Intelligent control of micro power – Immortal machine” published in the Elsevier Nano Energy journal (Volume 72, June 2020), https://www.sciencedirect.com/science/article/abs/pii/S2211285520302561
- Paper titled “A Joint Temporal-Spatial Ensemble Model for Short-Term Traffic Prediction” is currently under review with Science Direct Neurocomputing Journal.

Glossary of Terms

BLE	Bluetooth Low Energy
BT	Bluetooth
FSK	Frequency Shift Keying
IoT	Internet of Things
LPD	Low Power Device
LPN	Low Power Node
RFID	Radio Frequency Identification
S^3	Self-sustaining, Self-learning Subsystems
SAADC	Successive Approximation Analogue-to-Digital Converter
SoC	System On Chip
UWB	Ultra-Wideband
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WMAN	Wireless Metropolitan Area Network
WSN	Wireless Sensor Network

Chapter 1: Introduction and Background

Introduction

Technology is currently advancing at an extraordinary rate throughout the world, certainly accelerated by highly integrated systems being continually condensed into smaller units while pushing their speeds significantly faster; marketing departments boasting reduced energy consumption requirements, smaller footprint, and smaller cost to deployment. From all these recent advancements, it is critically apparent that being mobile and modular, free from cabling constraints, self-contained, and managing and minimising their own energy requirements always benefits.

Energy comes in many forms that continually surround us in every conceivable way, albeit measurably small if not tiny amounts. It is still in existence and ready to be harnessed. From the power suppliers we know and trust like the national power grid, the AAA batteries, the sun beaming down, through to the yet to be effectively tapped sources such as the hydrogen abundance, thermal scavenging, and harvesting the x-rays entering our plant yet originating from other galaxies. Clearly, there are still large swathes of research waiting to be studied.

How this energy is stored, distributed, and used is an unending and entirely separate challenge with many possible paths and pitfalls.

The ultimate goal then; is to efficiently collect and store tiny ‘goblets’ of energy captured from the ambient environment in various ways, securely storing these little pieces of treasure until they are needed, then distribute them as efficiently, intelligently, and carefully as possible. This energy management process is the key to how the energy consumer(s) can exist and function at their most efficient.

This research intends to provide a closely coupled software-hardware foundation that aids implementers in intelligently harnessing and using tiny amounts of ambient energy in a highly autonomous way.

This platform continues on to explore ways of maximising the efficient usage of the harvested energy using various hibernation/wake strategies and then making objective comparisons with proposed intelligent energy management protocols.

Finally, the protocol extends to enable the device to manage its personal survival possibilities so the devices can use an evolutional personality-based approach to deal with the unknown environmental situations they will encounter.

The Device Cycle

Fundamentally, consumer expectations of modern-day intelligent devices consist of some type of physical object that has a somewhat predictable design flow:

- Has a power source of some form
- Has the ability to be activated in some way
- Can do some kind of work
- Can communicate

The difference between a simple device, such as a temperature sensor, versus a complicated device, such as the latest mobile phone offering, in terms of what work they do and offer, is obviously vast; however, these more complicated devices get divided down into subsystems that end up following the same bulleted fundamentals listed above.

If we then introduce ambient, harvestable, renewable energy as the power source for one or more of these devices and allow the device to manage its usage of this energy, we can categorise them as being S-Cube'd Devices, abbreviated to S^3 (self-sustaining, self-learning subsystems).

S^3 devices can be connected together in the sense that they can share their incoming energy with other devices, and they can communicate with each other, potentially learn from each other, work with each other, and solve the problems their deployment addresses.

These concepts open up fundamental questions which this research will be addressing.

Research Questions

1. Is it possible to operate a device intelligently and autonomously from ambient renewable energies indefinitely?
2. Is it possible for the device to adapt to and manage its available ambient energy resources?
3. Is it possible for this autonomous device to communicate via a radio link and continue to sustain its existence?
4. Is it possible to deploy an interconnected mesh of these devices?
5. What are the design considerations needed to achieve a self-sustaining, self-learning subsystem node?
6. Is it possible to draw parallels between animate life and S^3 nodes?

Question 1 is addressed by designing and introducing a bespoke hardware research platform that allows testing of various battery-less self-sustaining framework concepts.

Questions 2 to *5* are then explored in detail, resulting in introducing a low-level protocol that can manage the device's charge/work patterns in an adaptable way.

Various sleep/wake patterns get introduced, evaluated, and compared.

The last *question (6)* then considers the basic life instincts and coronaries found within animate beings and implements them in the form of an S^3 existence protocol; parallels get drawn, which reflect how nature has successfully supported life for so long. These attributes are collected together and considered as essential coronaries:

- **Survival**, Inbuilt instinct above all others to stay alive
- **Work/Life balance**, ensuring the best level of work output to survival needs
- **Sustenance**, based on an available food source, to be able to provide a benefit to society
- **Avoiding starvation**
- **Gluttony**, adjusting its work output to manage its energy input
- **Over-exertion**, frequently over-working will cause difficulties in managing the other coronaries
- **Birth**, managing its initial energy store and being able to establish its running parameters, joining the group
- **Maturity**, Leaning and improving based on its experiences over time
- **Prosper**, getting everything just right to maximise work and enjoy ample energy
- **Accumulate resources**, being able to know the best times and positions to gain basic requirements
- **Rest**, being able to hibernate and sleep in a way that enables energy conservation and recuperation
- **Awareness of surroundings**, neighbours, energy sources
- **Consider the worst**, realising if energy gain/rest cycles are not sufficient to sustain life, and prepare for the eventuality
- **Learn**, both short term and long-term experiences
- **Burn out**, failing to sustain the energy/sleep and work balances to a point where existence is no longer viable.

Problem being addressed

One of the main influencing attributes is time, so one of the main intentions is to study time dilation and its control to deliver electronic-driven services, which otherwise may be prohibitive given access to available resources.

The prize is to have intelligent devices which can achieve more work than the sum of the energies it held during its commissioning.

The conclusions of this research will improve device and sensor networks in the form of introducing the S^3 concept into deployable micro-devices and machines with ultra-low power autonomous designs requiring only scavenged micro-renewables to sustain existence.

Primary Cells and Batteries

Sensor networks are already well established in the research arena, but they come with a considerable negative attribute: *They use batteries*.

This attribute has many knock-on consequences:

- Expensive, not only for the battery procurement but also infers the requirement for a charging system to reduce any disposal/replacement impact.
- Weight, batteries always add considerable weight to any product.
- Maintenance, in the form of charging, replacing, making sure no leakage or degradation has occurred.
- Disposal, eventually, the cell must be disposed of, having a negative environmental impact.
- Current consumption, designs will usually decide a battery capacity and work to achieve an acceptable run-time based on that decision, so once the implementers meet

the design goal targets, no further effort to reduce current requirements is typically made.

- Non-adaptive, the battery is always there, so the unit does not need to change its behaviour as it knows it will eventually have its battery replaced or recharged.
- Environmental impact, battery manufacture itself uses chemicals, some exotic (for example the vanadium found in vanadium redox flow batteries, the cadmium found in nickel cadmium batteries and to some extent even lithium).
- Cost, batteries, and the associated maintenance all come at a considerable cost.
- Deployment, batteries currently suffer considerable transport restrictions.

Introducing a battery into a design immediately gives the device an end-of-life clause. The battery itself will die at some point. The death will be due to total drainage of all its energy, or in the case of rechargeable entities, the breakdown of the batteries internals needed to keep its chemically driven energy creating processes working.

The ultimate goal is to design a device that can out survive these devices and even out survive the generation of designers that created and deployed it – We must eventually sever the dependency we have on these mortal energy devices.

Background

Sensors, supporting devices, and controllers are deployed all around us, and scenarios where this kind of energy managing research can be incorporated are in abundance.

Four unrelated and very different example scenarios get presented below.

Example Scenarios

Pond Pump

Current implementations of a typical pond pump incorporate a solar panel directly connected to a small water pump. When the sun is intense and is directly above the device, the solar panels can collect enough solar rays to sufficiently power the motor and pump water to the fountain spout. As soon as the sun's path becomes obstructed by clouds or shade, the pump and fountain will stop. It is also important to note that if the sun's rays are not quite strong enough to provide adequate solar energy that can meet the motor's requirements, the pumps output speed will be reduced until eventually the power simply gets wasted.

This design is highly inefficient and could benefit massively from some intelligent energy management.

The first and foremost improvement provides a store for the placement of solar energy accumulation. This store allows the capture of energy irrespective of the sun's rays' strength at any given time.

An ultra-low-power microcontroller monitors this energy store, and its charge/discharge profile is then mapped and continuously updated.

The microcontroller then determines the best times and durations to enable the pump, so a more consistent and more extended fountain stream ensues. Communication is implemented via radio meshing between multiple pumps to provide synchronisation or fountain effect patterns.

The intelligence, radio, and microprocessor involved get powered from the same solar energy store.

Mars Rover

The mars rover operated on a remote battery management scheme, where its current charge status was transmitted back (via relays) to a base station on earth. The command centre would use this information to decide what jobs they could safely perform with the available energy. The Rover was put into a sleep state via a command to gain charge within its battery using its solar panel. The command to do this also gave it a wake-up time for which the Rover would wake-up and re-transmit its energy status. If enough energy gain followed, it could perform the list of tasks sent from the command centre; else, it would return to a sleep state for another predetermined length of time.

The delay in sending a message to the Rover, or receive one from it, could be up to 30minutes, largely depending on the planets' position to each other at the time of transmission. Typically, the Rovers receive hardware needed to be powered and consuming energy during any message reception window.

This process demonstrates an energy harvesting/energy management scenario; however, it fails to be truly autonomous. It is the control centre that decides when to wake up and when to perform tasks, and because of this, considerable energy gets consumed waiting for messages generated based on information received by the control centre at an earlier point in time.

Applying autonomy to this project would result in the Rover being able to decide for itself what and when to do the work tasks it is responsible for. It monitors and calculates its own charge and discharge curves and decides to wake up and perform work when it has achieved adequate charge. The control centre only has to issue a list of tasks to do and monitor its status messages.

Distributed Sensor Nodes

CO_2, air quality, and moisture measurements to review environmental changes and impacts get regularly taken in sites of scientific interest and natural beauty such as volcanoes, icebergs, forests, and jungles. Currently, large, expensive battery-powered devices are installed at choice locations and are maintained and monitored, and these locations, frequently found in hostile positions, require significant effort and expense to reach.

To deploy (even via airdrop) a self-sustaining, self-powered mass sensor network into hundreds of tree canopies or over large areas that will have minimal effect on wildlife and ecosystem is highly desirable. Using communication meshing technology, they pass data back to a monitoring point or controller via neighbour hopping radio techniques and allow complete maintenance-free operation. The units self-optimise to provide the best use of the available energy to perform their work tasks and communicate their results. Their useful life is considered indefinite.

Oyster Monitoring

In the mariculture of oysters and other bivalves, producers must time the laying of ‘cultch,’ this being a material that encourages the natural spat-fall such as freshly-ground oyster shells or otherwise disturbed natural or artificial substrates which free themselves from estuarine silts, so becoming accessible to oyster larvae.

This bio-cycle enables the producer to increase their crop using local natural resources instead of buying in post-settlement spat from hatcheries. There is a great incentive to do this both for-profit-margins but increasingly due to the providence of the product and its suitability to grow in a given location, particularly with more specialised products such as the European flat oyster.

However, the timing of cultch-laying, preparation, or other spat collecting activities relative to the spawning behaviour of commercial oyster species is critical for the success of this highly demanding and costly activity.

The shellfish/oyster industry relies either on (i) anecdotal predictions and/or basic measurements (water temperature) or (ii) the much more laborious and expensive counting of free-swimming (planktonic) oyster larvae to time the laying of cultch.

The release of larvae is sensitive to water temperature, and extended growth seasons and summer heat-waves already alter the spawning and settlement behaviour of bivalves, making it increasingly unpredictable for oyster fishers to manage natural stocks.

Late-laying and preparation of cultch will miss the peak in spat abundance, early-laying, however, will result in the degradation and possible dispersal of cultch and, depending on the location, its burial in soft sediments, making it inaccessible to oyster larvae.

To aid in this prediction, quantifying the gaping behaviour in bivalves to predict seasonal spawning activity and release of planktonic larvae using valvometry can be introduced.

The principle of bivalve valvometry is not new. The invention stems from a simplification of the sensing mechanism (a Hall Effect Sensor instead of two electromagnets) and the development of a new sensor unit that can deploy to make the measurements on large batches.

The goal is to use this sensor to measure the oyster's opening and closing and log it periodically. This data then gets post-processed to find a correlation between mating habits, environmental variables, and eating habits.

After enough of the oysters have been monitored and the data collected (current targets aim for 80 oysters being sampled at a 10Hz frequency), it is hoped that the mating window, which is of such critical importance to the oyster farming community, can be predicted.

The sensors monitor a large batch of oysters; they operate for as long as possible, collecting as much logging data as possible. The units get deployed in remote hard to access locations, and they frequently operate in harsh conditions.

The units have a battery; however, they also have a solar panel available to collect energy and charge the battery. The device must manage its energy in a way where it can maximise its existence and perform its logging duties for as long as possible.

The goal is to introduce intelligence where the battery is considered as a secondary, potentially redundant supply, and the device can manage its own incoming solar energy and adjust its sampling frequencies to counter potential failure.

Collaborative Working

Measurements taken in harsh environments such as radioactive, combustible, and extra-orbital are incredibly costly in terms of both finance and effort to implement. In these cases, it is worth introducing redundancy within the system to counter any possible failures which invalidate or prevent the result. Measurements can simply be multiple versions of the same system running entirely independently but parallel with each other. However, this does not cover every eventuality.

Benefits would also arise if these units could inter-communicate to share progress information, results, and current status reports. This visibility would allow the devices to intelligently work together to perform the task, providing redundancy to a loss of energy income.

Suppose the devices could monitor and control their incoming energy resource vs its energy expenditure. In that case, it could, in effect, cooperate with other devices so that their sleep/charge/work patterns are not synchronised, alleviating the possibility of bulk failure.

Chapter Summary

We can best answer this thesis's main question by breaking the research down into three individual layers.

First, we can consider having a device survive and manage its own energy in its simplest form by utilising dynamic sleep/wake cycling based on incoming to outgoing energy balance.

This stage will focus on finding energy, capturing it in tiny quantities, managing the flow and store of energy, and providing a foundation for considering time-dilation and its effects on allowing the device to lose its dependence on finite cell types of energy supply.

A research platform gets developed to study the real effectiveness of creating a foundation that can sustain a digital existence with enough headroom to execute useful work successfully.

The second would be to further this platform by adding a layer of intelligence in the form of a protocol stack that will allow the devices to encapsulate communication and self-management tasks, improve their energy efficiency, and manage their own task completion output energy availability.

Here, the notation of 'containerising' blocks of energy that can be stored, assigned, and consumed according to a dynamic management plan gets proposed.

The protocol stack will allow tasks to be queued and prioritised, where the device will take charge and find the best way to complete its assigned duties within the environment it has been deployed.

A communication protocol is introduced, which is both energy conscious and time driven. The meshing of multiple devices cooperatively working and various ways of timed-based message delivery and scheduling get compared.

Global time and message synchronisation are covered, and both peer-to-master and peer-to-peer(s)-communication are both presented.

Lastly is to consider survival atomicity, in which personality and colliery traits get introduced to provide resilience within unknown environments for prolonged periods of existence.

Light-weight, low energy, and low CPU-burden methods provide intelligent reasoning in dealing with unforeseen circumstances within inaccessible environments.

It is inevitable that during the deployment of enormous swarms of individual sensor devices out in the field for a prolonged period, there will be failure and loss of a deployment batch percentage. Having the devices self-managed and exhibiting less predictable outcomes when overcoming survival-based challenges can dramatically reduce overall group loss. In fact, simulations will show that when all the devices follow the same predictable patterns, a much greater loss will result, and it becomes the deployer's responsibility to find a way of mitigating the problem.

Nature and evolution have themselves harnessed these kinds of rugged cross-generational survival patterns based on the fittest and best-suited characters for their respective environment, typically carrying these traits forward will allow the device the best chance to flourish and prosper.

It will significantly benefit in providing self-managed sensor devices with the ability to evolve in their environment just as we see insects and animals evolving in nature.

For this last phase, the platform gets expanded with a final layer of the protocol stack that allows the device to be furnished with various depths of a digital-personality, guiding it in making its survival choices.

Chapter 2: Literature Review

The following literature review attempts to identify relevant and novel discoveries within the fields surrounding energy harvesting, low-power device design, self-sustainability and cooperative communication technologies.

Figure 1 below illustrates the areas of particular interest, which are the main focus topics of review. The figure highlights the tight coupling between the critical systems required when building self-sustaining devices.

1

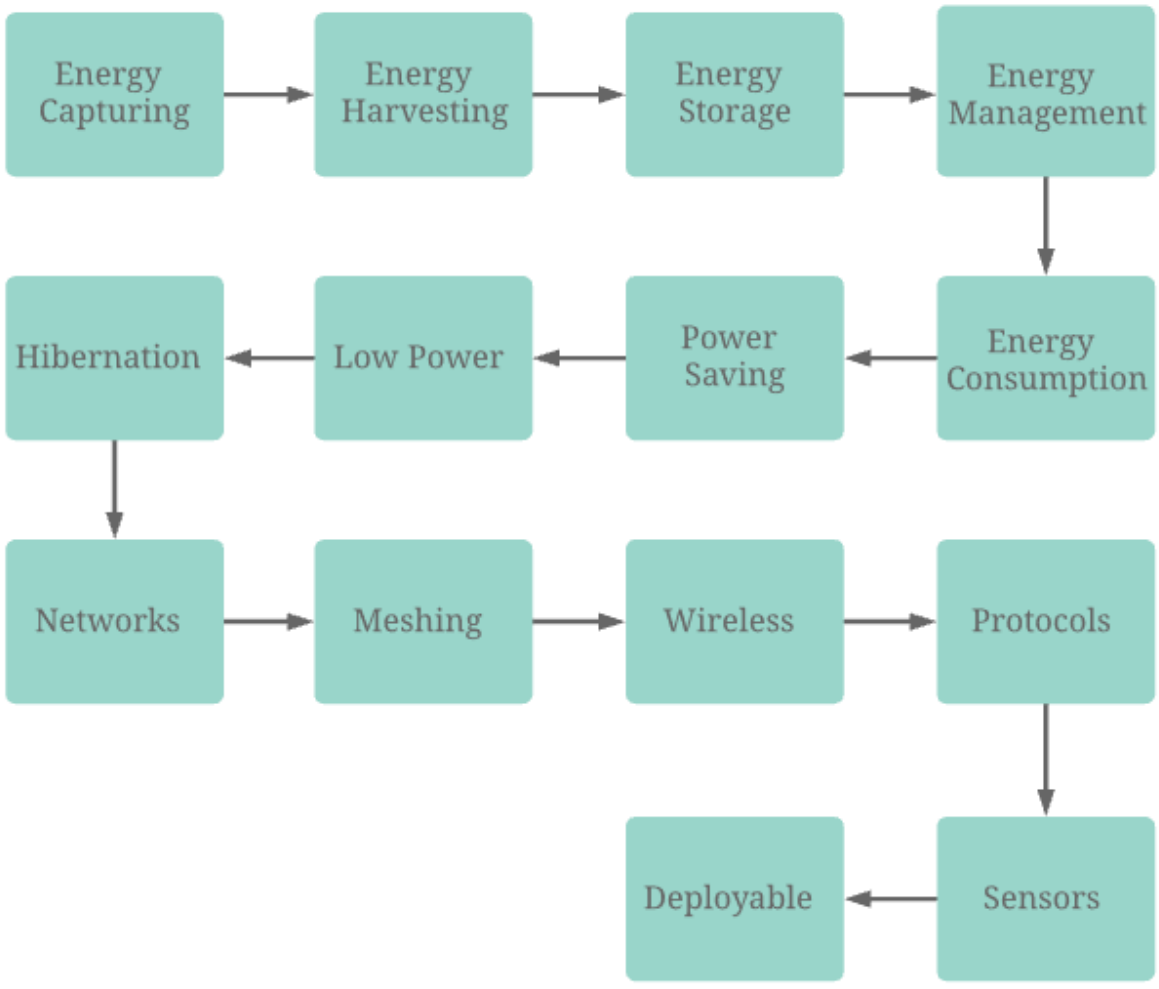


Figure 1: Literature Review Map

Sensor Networks

Wireless sensors, data collection, and global interconnection are essential aspects needed to enable the IoT revolution that is currently firmly underway. To have the ability and technology which allows small, light-weight intelligent sensor arrays to operate indefinitely without the need for human intervention, and for them to be able to acquire and manage their energy source in a non-exhaustible fashion, is a highly relevant area of research with great potential to influence the direction of future sensor deployment.

Having multiple intelligent sensors in environments and situations which are hostile, complicated, or costly to reach or are just impossible to reach creates a situation where self-learning-survival technology is highly sought after. Providing the sensor with the ability to solve its survival issues using learning and evolutional techniques will significantly mitigate this issue.

A deploy-and-forget methodology is a key to this research's foundation; hardware devices containing zero parts or subsystems with a time-to-live dependency break free from the shackles of expensive maintenance commitments and part replacement requirements. LPDs, which can work and communicate together, learning their environmental situation and solving survival issues without the need for external interference, enable a compelling model for large-scale, cost-effective sensor deployment.

Energy Coupling

Magnetic resonant frequency coupling for the enabling of wireless power beaming and transfer has been around for a considerable time with its foundations lying with Nikola Tesla's research in the late 19th century, coil-based solutions are developed and presented in [1]. Energy transfer in the form of near-field wireless charging also has considerable benefits, both for safety, cost-effectiveness, simplicity, and convenience [2]. The health and safety aspects of such energy coupling are considered safe by [3].

Large corporations such as Qualcomm and Apple have been closely monitoring the developments and directions of wireless power transfer ever since scientists at the Massachusetts Institute of Technology (MIT) were able in 2007 to demonstrate transferring tens of watts over a distance greater than two meters [4]. Most research has, however, concentrated on the one transmitter-one receiver approach, and these have concluded that performance is related directly to factors such as distance between coils and transmitted power [5] [6] [7].

Rectenna circuity is critical to the efficient capture and recovery of wireless energy, and many topologies of implementation have been researched and compared [8], such as series single mounted diode approaches, single shunt mounted diodes and diode bridges. Different component designs are explored as described in [9], where a Schottky diode approach gets successfully adopted for millimetre wave rectification. With the design goal of scavenging multiple wireless transmissions being of high importance, previous research evaluates connecting multiband reception antennas to appropriate rectenna designs in the field of energy harvesting [10] [11] [12] increasing the ability to recover efficiently from multiple frequency bands.

Energy Harvesting

Energy harvesting, in general, has come under scrutiny in the sense of its feasibility and effectiveness, especially in the area of ambient radio transmissions [13] [14] and opportunistic radio transmission harvesting [15], where discovery that the leakage currents of the storage capacitors contribute negatively to the overall worthiness was made. However, energy harvesting in itself is still proving to be a great source of interest and research, offering many well-documented techniques and materials [16]. Its promising benefits within the autonomous systems industry are desired as detailed in [17] and in ultra-low powered devices [14]. Harvesting techniques using different transducers are becoming more efficient and practical, such as the adaptive piezoelectric circuit detailed in [18] and the piezoelectric and pyroelectric materials discussed in [19]. Confidence can also be drawn from [20], where the novel idea of harvesting energy from radiation produced by fluorescent lamps was proven.

When considering a very low power resonantly coupled antenna used to charge a supercapacitor, the choice of resonant coupling allows us to tightly monitor the input and output energy using only a signal generator without the need for dc-dc converters to boost the voltage level. It then becomes possible to precisely study the consumption needs and constraints of an LPD (Low Powered Device). The same reasoning applies to small solar or piezo-based renewable sources, which have also been used and produce comparable results.

The performance of the traditional transformer, which, based on inductive magnetic coupling between primary and secondary coils, improves when the two coils resonate at the same frequency, as discussed in [30]. This performance boost forms the basis of most wireless charging systems allowing the transfer of energy across free space at a distance. The amount of power transfer is dependent on the mutual inductance between the two coils, which is inversely proportional to the distance between them [21]. The secondary needs to receive enough energy to enter resonance; otherwise, work becomes impossible.

Traditionally, near and far-field low-power resonant systems have had no significant function. However, systems are available [22], which allow reception, rectification, and then storage of the DC in capacitors.

Another example of energy harvesting using specifically tuned components comes from Dyo, Ajmal, Allen, Jazani and Ivanov [23]; they propose a purpose-designed ferrite rod antenna intended for use with energy harvesting circuitry within the MW bands (526.5kHz to 1606.5kHz inside Europe and the UK). They devised a method to find an optimal configuration for a ferrite rod antenna to harvest energy from a specific frequency band. Their optimisation model was implemented and simulated using the sequential quadratic programming algorithm. Balancing this against the cost of wire needed to provide the optimal windings, they presented the conclusive graph shown in Figure 2, illustrating the principle in practice, focused on a specific and well-documented transmitter.

The test circuitry harvested energy from a BBC Radio 5 transmitter located in Brookmans Park, UK, at a range of distances from 1 to 15 km. This transmitter emits 150 kW of RF power at 909 kHz. The energy harvester is required to deliver 1 mW of power to a 1 kΩ resistor, have 50% power efficiency and have a 1 V voltage across the load.

2

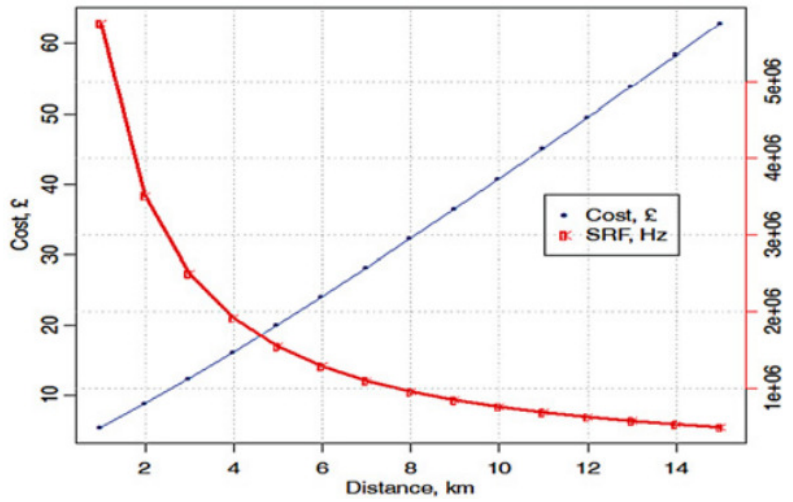


Figure 2: Distance, Self-Resonant Frequency vs Winding Cost [23]

The results clearly show predictable amounts of harvestable energy is available at the different frequencies, and to properly benefit from this stream, proper matching, configuration and optimisation of all hardware components is essential.

Capacitors as Energy Stores

Wireless power transfer offers the possibility to receive power wirelessly and do work without the conventional storage medium of a battery [24]. An example is a battery-free receiver designed by the Powercast Group [25]. This product is the P2110B 915 MHz RF Power Harvester far-field Receiver [26] and designed for sensor networks and active RFID. Distances of 10m are claimed but at the expense of highly directional and powerful transmitters. The process of placing the accumulation of energy into a supercapacitor allows for further organisation and management.

The ubiquitous battery gets replaced with a supercapacitor which has several advantages. It can be charged from zero, is compact, can charge/discharge very rapidly, has duty cycles in excess of 500,000, and has a specific power of the order of 10kW per kg [27].

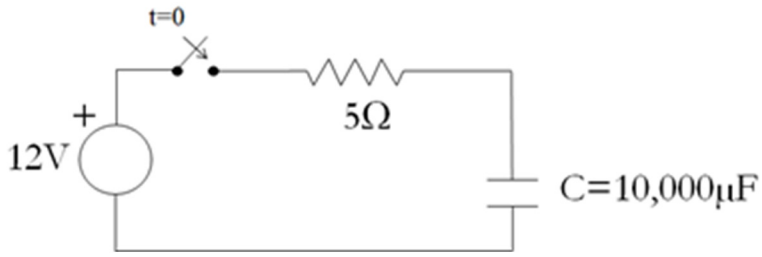
Disadvantages include low voltage ratings; some leakage and supercapacitors are not suited to AC circuits. Applications within low-power wireless networking environments are low voltage (customarily considered 1.6-5v but can be much lower), and capacitors are available in integer multiples of 2.7v and 3.3v. Leakage is important for low power applications, and a good rule of thumb is 1µA/Farad [28], giving a minimum rate at which the capacitor gets charged. The LPD application requires DC, which is primarily the reason supercapacitors get designed. A capacitor stores energy according to:

$$E = \frac{1}{2}CV^2$$

There is, however, a significant caveat involved when choosing a capacitor to use as primary energy storage, which is the fact that only *half* the energy fed into a capacitor for storage will be available to draw on for use. The other 'missing' half of the energy supplied gets dissipated and lost to the resistance that presents itself in the supporting circuitry [29], with a running example referenced from [30] which is re-produced and summarised below.

To demonstrate this principle, we must further look at a simple RC circuit. The switch will be closed at time t=0, and the capacitor C is initially uncharged.

3



$$\tau = RC = 0.05$$

Figure 3: Basic RC Circuit

At the 5τ point (0.25 seconds in this case), the transient is substantially over.

The performance equations for this circuit are:

Equation 1

$$i(t) = \frac{V}{R} e^{\frac{-t}{RC}}$$

Equation 2

$$V_c(t) = V\left(1 - e^{\frac{-t}{RC}}\right)$$

As this example stands, V = 12 volts, R = 5 ohms, and C = 10,000μF, the energy delivered to the resistor and the capacitor can be calculated, and they sum to the energy delivered by the source.

The capacitor will end up asymptotically approaching the source voltage, as such will eventually store within its electric field (U) *0.72 Joules* of energy, this can be proven by the following equations:

Equation 3

$$U_R = i^2(t) \cdot R \Rightarrow \left(\frac{V}{R} e^{\frac{-t}{RC}}\right)^2 \cdot R \Rightarrow \frac{V^2}{R} e^{\frac{-2t}{RC}}$$

$$U_c = V_c(t) \cdot i(t) \Rightarrow V\left(1 - e^{\frac{-t}{RC}}\right) \cdot \frac{V}{R} e^{\frac{-t}{RC}} \Rightarrow \frac{V^2}{R}\left(e^{\frac{-t}{RC}} - e^{\frac{-2t}{RC}}\right)$$

$$(U_R)_{total} = \int_0^\infty \frac{V^2}{R} e^{\frac{-2t}{RC}} dt$$

Integrating using the example values gives us the total energy dissipated in the resistor:

Equation 4

$$\int_0^\infty \frac{144}{5} e^{\frac{-2t}{0.05}} dt$$

$$\int_0^\infty 28.8\, e^{-40t} dt$$

$$= 28.8 \cdot \left[-\frac{1}{40} e^{-40t}\right]_0^\infty \Rightarrow 28.8 \cdot \left[0 + \frac{1}{40} \cdot 1\right]$$

$$= 0.72J$$

A transient simulation tool such as MPLAB Mindi [31] is used to plot graphs showing the energy delivered to both the resistor and the capacitor. Notice how the resistor (red) and capacitor (green) energy traces merge out just past five time-constants. Showing that half the

energy supplied by the source gets delivered to the resistor (which gets dissipated as heat), and the other half is now safely stored in the capacitor's electric field.

4

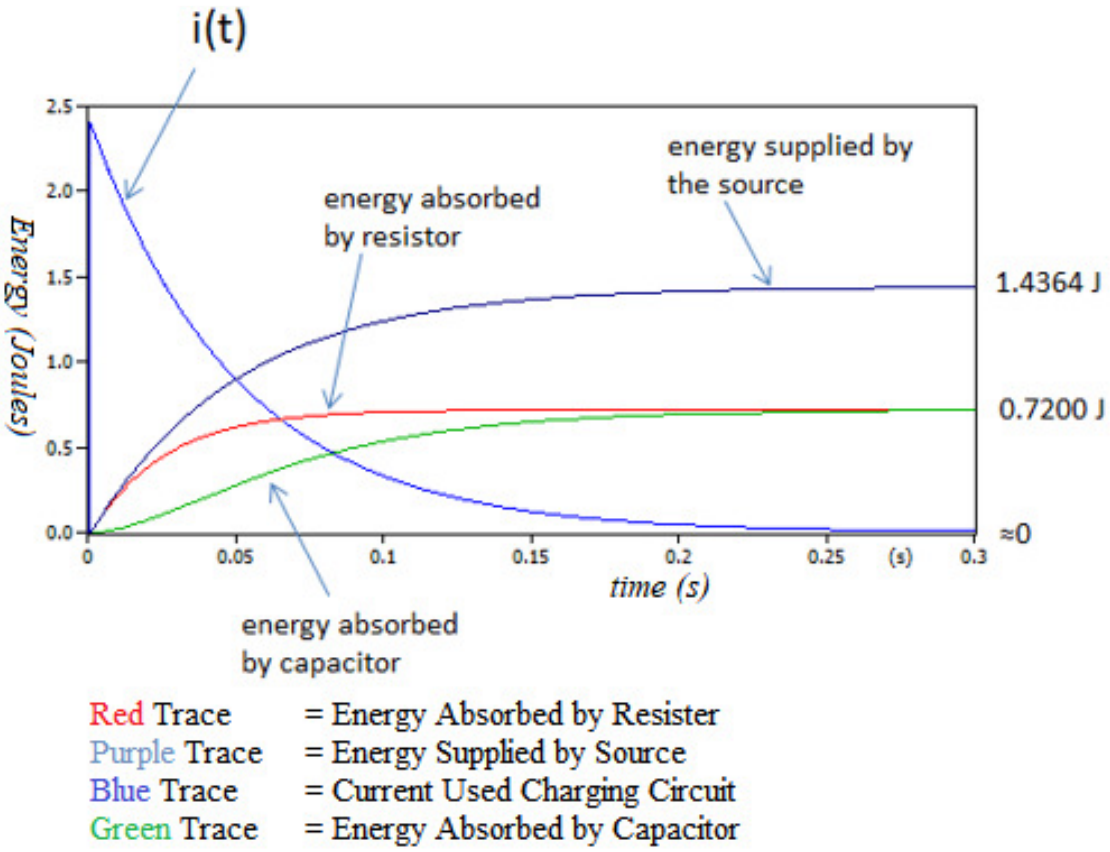


Figure 4: RC Circuit Energy Distribution [30]

The moment the switch contact (shown in Figure 3 above) is made, thus completing the circuit, the capacitor has zero voltage, while the supply has V. This voltage difference generates an electric field that accelerates charges. This acceleration, in turn, sets up a current which flows and charges up the capacitor until its voltage also reaches V.

At this point, there is no voltage difference between the source and capacitor, but the accelerated charges are, in fact, still moving. Half the energy has gone, at this point, into the capacitor and (discounting losses) half has gone into the current in the wire.

The current will continue to flow and continue to charge the capacitor; the capacitor's voltage will overshoot V until the current stops, creating another potential difference. This potential difference will then cause a current to flow back in the opposite direction. This sequence will continue for a while where the current and voltage oscillate. This oscillation behaviour in the circuit gets characterised as ringing.

Resistances present in the circuit will eventually remove this extra energy, leaving only the charged capacitor.

This issue impacts the entire circuit design for low current use. For example, resisters do not resist at all; they do not restrict or push back; they can only lose current, throw it away, and do that by dissipating it as heat.

Any kind of resister in the circuit is fundamentally going to leak current when it has a flow passing within it. This problem becomes further exasperated because every component exhibits some kind of resistive property, intestinally or not. Components also exhibit changes in their resistive parts when encountering different temperatures and different frequencies of operation.

The sum of all these losses represents the leakage current of the device, and if not carefully controlled, can quickly consume any gains introduced by intelligent management techniques. The very best way to constrain energy for use is to hold as much control over these leaks as possible and maintain a balance that allows easy mitigation of these losses by the incoming energy flow. The losses themselves will never be eliminable and will present a constant noise type consumption on the energy store, even during hibernation periods.

Near-Field Lensing

A fascinating and novel research area that surfaces at various points during the experiments conducted for this research involves exploring the near-field coupling 'lensing' effect between two or more antennas during a coupled energy transfer. During previous research [32], it is noted that the actual energy transfer amount present on the reception antenna is somewhat more than the mathematically calculated expectation for the setup. This phenomenon points to the flux lines present during the coupling having a 'pull and acceleration' effect on surrounding flux lines not directly expected to be involved in the coupling process – this 'pull' effect seems to forcibly involve these surrounding flux lines, thus increasing the rate of energy transfer. This process becomes even more intriguing when more antennas get introduced for coupling.

It is suggested here that there will be an optimal positioning arrangement for multiple LPDs all wishing to couple energy from a single source at the same time, the number of coupled LPDs, the size of the reception antenna, the distance between the LPDs, and the distance between the LPDs relative to the transmitting antenna will all need considering. Ultimately an algorithm that the LPDs can individually use to determine this best position would be highly beneficial in many real-world wireless/battery-less sensor/charging networks.

To better understand the observed lensing effects, charge rate logging coupled with LPD position data can be collected and processed to map the flux lines present between the transmitter and coupled LPDs and how they change as the LPD's positions change.

Suppose individual LPDs are aware of the other LPDs surrounding them and the coupling effects they are experiencing. In that case, it is feasible for the LPDs to position themselves in a way where the lensing effect can provide a benefit during the energy transfer cycle, as well as to be able to determine if the desired position may have a negative effect for itself or other LPDs currently coupled.

An exciting idea that can use the multi-device coupling effect to its advantage involves using medically implanted devices. An antenna such as the miniaturised stacked implant antenna proposed by Kaka, Toycan and Walker [33] could be used to couple wireless energy. The study constructs a vertically stacked three layer hybrid Hilbert fractal geometry and serpentine radiator-based patch antenna. This type of LPD allows tuned RF reception from inside biomass using the ISM (Industrial, Scientific and Medical) band located at 2.4-2.48 GHz. Small patch antennas like this can be used to harvest energy intentionally transmitted and directed at them to power the LPD they are attached to, allowing full operation and data communication from within a body indefinitely.

Bluetooth Radio Technology

Bluetooth has become a mature, reliable, and universal communication medium covering multiple topologies, including Point-to-Point, Broadcast, and Mesh [34]. One massive benefit of using Bluetooth is the low energy option it provides (BLE), Ionascu and Marcu [35] carry out an in-depth profile of various BLE designs using well-documented benchmarks and arrive at conclusions that the Nordic nRF system on chip (SOC), along with its hibernation and sleep modes, offers exceptional energy-efficient properties. Ultra-low energy BLE is picking up the pace and gets demonstrated by Ensworth and Reynolds [36] to be operable by using a PSK backscatter modulation to generate a subcarrier in one or more of the Bluetooth channels. RF backscattering represents communication made possible by modulating a devices own reflections of an incident RF signal, as opposed to generating radio waves itself. FSK modulation is then integrated onto it, allowing a data transfer directly injected into the BLE stack. Further comparison-based research performed by Siekkinen, Hiienkari, Nurminen, and Nieminen [37] show how the energy consumption of BLE was studied by measuring real devices with a power monitor and models derived from the raw energy consumption behaviour observed from the measurement results. They contrasted their results by performing similar measurements with ZigBee/802.15.4 LPDs. Their results show that when compared to ZigBee, BLE is indeed very energy efficient in terms of the number of bytes transferred per Joule of energy spent.

Mesh Networking

It is in recognition that the best chance for continued success in deployments of sensor networks is to utilise a mesh networking communication system, specifically a choice of WMN (Wireless Mesh Network). Various surveys into the different WSM offerings and their designs have been commissioned and published. Akyildiz and Wang [38] undertook one such survey where they

identify that WMNs are predicted to both resolve limitations and to significantly improve the performance of ad hoc networks, wireless local area networks (WLANs), wireless personal area networks (WPANs), and wireless metropolitan area networks (WMANs). They continue to foresee that WMNs will deliver wireless services for various applications in personal, local, campus, and metropolitan areas. Their research presents a detailed study of recent advances and open research issues in WMNs. They analyse various system architectures and applications of WMNs and discuss critical factors influencing protocol design. Their work explores theoretical network capacity, and the state-of-the-art protocols currently available for WMNs get explored to point out several open research issues.

The research draws some interesting conclusions that cover some important points about the scalability of mesh networks and other failings that must get considered during implementation. They present the following categories of feature sets and their limitations imposed:

- *Scalability*. Based on existing MAC, routing, and transport protocols, the network performance, indexed by throughput, end-to-end delay, and fairness, is not scalable with either the number of nodes or the number of hops in the network.
- *Self-organisation and self-configuration*. Self-organisation and self-configuration require all protocols in WMNs to be distributive and collaborative. Otherwise, WMNs will lose the autonomic feature.
- *Security*. Due to wireless ad hoc architecture, WMNs are vulnerable to security attacks in various protocol layers. Current security approaches lack a comprehensive mechanism to prevent or counterattacks in different protocol layers.
- *Network integration*. Current WMNs have minimal capabilities for integrating heterogeneous wireless networks. Integrating multiple heterogeneous wireless networks is still an on-going task for WMNs, due to the difficulty in building multiple

wireless interfaces and the corresponding gateway/bridge functions in the same mesh router.

Wireless Radio Protocols

Low power design of wireless technologies is a critical point for the future Internet of things (IoT) applications and will affect deployment momentum as energy starvations are still considered one of the most challenging issues to overcome. Lee, Dong, and Sun [39] produced a preliminary study of low-power wireless communication standards: ZigBee and BLE, evaluating their main features and behaviours in terms of various comparable metrics, including the transmission time, data coding efficiency, power consumption, and delivery ratio. Their paper titled *"A preliminary study of low power wireless technologies: ZigBee and Bluetooth Low Energy."* presents a quantitative evaluation of ZigBee and BLE in terms of the transmission time, data coding efficiency, power consumption, and delivery ratio. They also acknowledge that a clear conclusion of choice is unachievable since the suitability of wireless technology selection is greatly affected by practical situations, such as the realistic environment interferences, chipset prices, and installation cost.

This reasoning is very evident when the technologies get critically compared; the wireless communications offerings, meshing, and others are currently plentiful and growing substantially. Some of the technologies are remarkably similar in how they operate, and it has become these small differences that can match the technology perfectly to a given problem situation.

Wireless Energy Consumption

Also of particular interest from their research is the detailed energy usage mapping correlated with the protocol function area they performed on their hardware devices. This kind of analysis is imperative in achieving an understanding of the consumption costs of individual LPD actions. A high bandwidth calibrated current probe can be connected to the LPD's supply to

achieve this level of accuracy and detail. The probes are typically constructed using field effect type measurements which present no interference to the circuit. The problem is correlating the changes in consumption with the task being executed at that moment in time. Additional hardware is an option that helps resolve this difficulty, allowing software profiling, so the executing code sections are made visible. Using a combination of shared timestamps, external triggering, and software debugging, the effects of individual software functions get mapped to the system's consumption. Figure 5 and Figure 6 below show the graphical representation and mapping, which is achieved using this kind of measurement procedure, as presented by [39].

5

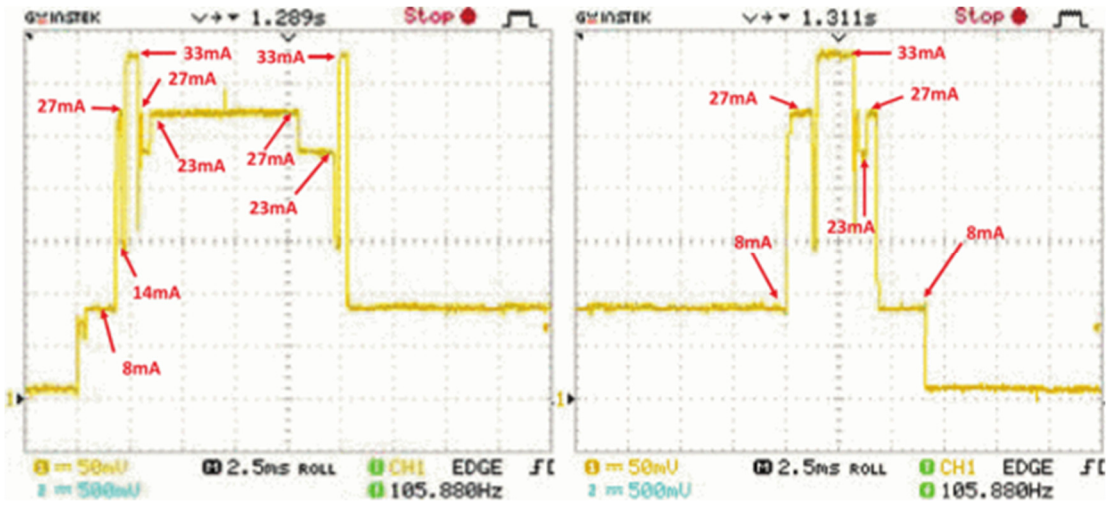


Figure 5: BLE Current analysis

6

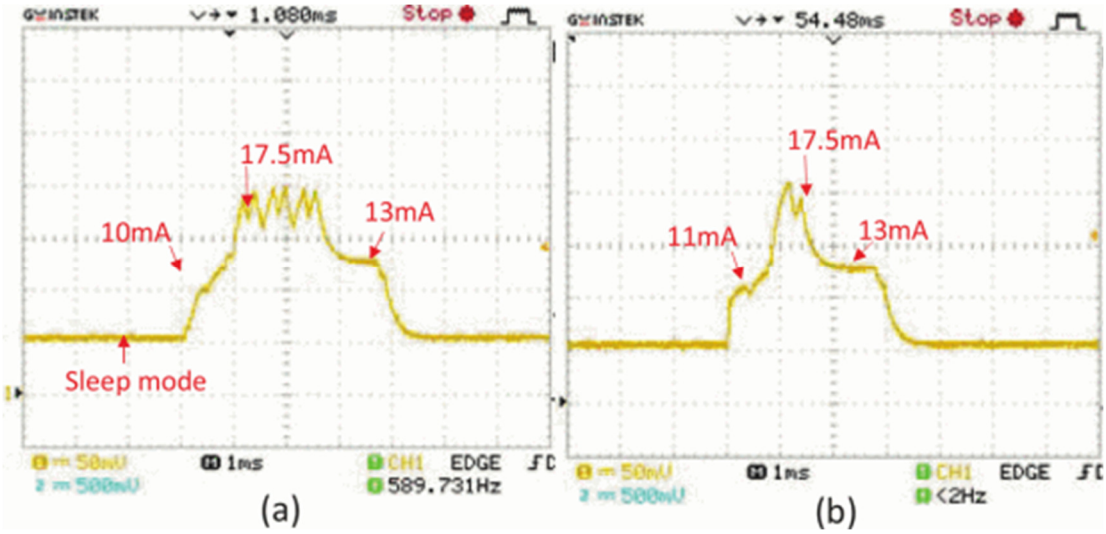


Figure 6: ZigBee current analysis

For ZigBee and Bluetooth, Baker [40] studied their strengths and weaknesses when the technologies get used for industrial applications. Baker claims that "*ZigBee over IEEE 802.15.4 protocol can meet a wider variety of real industrial needs than Bluetooth due to its long-term battery operation, greater useful range, flexibility in a number of dimensions, and reliability of the mesh networking architecture.*"

Lee, Su, and Shen [41] provided a study of Bluetooth, UWB, ZigBee, and WiFi wireless communication standards, evaluating their main features and behaviours in terms of metrics, including the transmission time, data coding efficiency, complexity, and power consumption.

In [42], Georgakakis, Nikolidakis, Vergados, and Douligeris presented a comparison of ZigBee, Bluetooth, and BLE about applications, topology, power consumption, data rate, data encryption, authentication, and modulation. They made critical advantage and disadvantage comparisons between them and summarised their feature offerings. In addition to this study, Tabish, Mnaouer, Touati, and Ghaleb [43] compared the leading wireless technologies according to power requirement, throughput, and latency. Conclusions drawn identified that

BLE showed the most self-resilience towards network obstacles, both BLE and 6LoWPAN have great potentials for such applicability in terms of power demand, bit rate, and latency.

One primary concern that repetitively reappears is the notion of interference in such wireless networks. All of the latest WSN technologies occupy or get configured to occupy the same 2.4 GHz ISM frequency bands. Due to this, in [44], Lin, Talty, and Tonguz compared the received signal strength indicator (RSSI) of ZigBee and BLE under WiFi and Bluetooth interferences. The experiments suggest that BLE outperforms ZigBee in the context of intra-vehicular communication when WiFi interference exists in the car.

Comparisons also get made of ZigBee, BLE, and WiFi transceivers' specifications and power consumption by Shahzad and Oelmann [45]. The results show that the BLE has less minimum energy consumption. In [46], Siekkinen. Studied the energy consumption of BLE by measuring real LPDs with a power monitor and derived models of the necessary energy consumption behaviour observed from the measurement results. They compared their results by performing similar measurements on ZigBee/IEEE 802.15.4 devices. Their results showed that BLE indeed consumes extremely little energy and has a desirable ratio of energy-per-bit transmitted. This analysis gives a good insight into the measurement techniques needed to understand such radio-driven consumption bursts and is discussed in further detail later in this chapter.

Dementyev, Steve, Stuart, and Joshua [47] provided experimental data to compare the power consumption of BLE, ZigBee, and ANT for a cyclic sleep scenario on a low-power wireless sensor node with short-range communication. The results determined a sleep interval at which the trade-off between power consumption and data rate gets optimised.

Wireless Protocol Comparisons

In [48], Gomez, Oller, and Paradells perform an overview and evaluation of BLE. They studied the protocol stack and performed experiments for performance evaluation of energy consumption and latency measurements. They also discussed the main characteristics and features of BLE compared to the other wireless technology offerings currently available.

Higuera, Kartsakli, Valenzuela, Alonso, Laya, and Martinez [49] studied the feasibility and performed experiments using Bluetooth, IEEE 802.15.4, and ZigBee in a high-speed railway environment. They use the technologies to evaluate connectivity times and throughputs at different speeds. Their results concluded that Bluetooth and IEEE 802.15.4 are suitable for ground-to-train connectivity in high-speed train environments.

In [50], Mikhaylov, Plevritakis, and Tervonen perform a study on maximum peer-to-peer throughput, minimum frame turnaround time, and the energy consumption used in BLE, IEEE 802.15.4, and SimpliciTI. Their conclusions agreed with the many others stating that BLE provides an inexpensive and power-efficient solution for wireless communication but still has many limitations restricting the throughput and increasing communication latency.

Further comparisons of efficiency get presented by Shahzad and Oelmann [48], where they do not look at the costs of the transmission protocol itself but rather at the differences in costs when different data processing techniques get used.

As typically realised using IEEE 802.15.4 compatible low-power radio transceivers that offer limited throughput, wireless sensor nodes are generally applicable to low-data rate intermittent monitoring applications. The results highlight the relationship between the amount of data, and frequency of data, against the need to reduce communications and radio usage to preserve energy reserves. To successfully realise high data-rates and increase sample rates for monitoring applications, the transmission of reduced overhead, even raw data, using a high

throughput radio transceiver, or performing potential signal conditioning and computation within the sensor node, resulting in a reduced amount of data to transmit. Shahzad and Oelmann [51] performed a quantitative evaluation of raw data transmission using different short-range wireless technologies and in-sensor processing using energy-constrained wireless sensing nodes. He creates a WSN (Wireless Sensor Network) consisting of spatially distributed sensor nodes and configures them to monitor a range of parameters from the environment they get exposed to. The WSN nodes then form a network through wireless communication to relay the data to a central destination.

To cope with the limited energy budget in such LPNs, wireless communication is typically realised with low-power radio transceivers (i.e., 802.15.4 compatible). With this low-energy consumption and the resulting limited communication throughput, such nodes have traditionally been better suited to low-sample rate intermittent monitoring applications. Realising high-sample rate applications such as image and vibration-based industrial monitoring, both the communication throughput of the low-power radio transceivers typically used in such nodes and their associated energy consumption [52] [53] poses a challenge in attaining a practically feasible solution.

While consuming the minimum energy is a motivating factor in evaluating different standard based wireless communication technologies. In relation to a low-power, low-cost, and compact size wireless sensor node, an analytical evaluation of these technologies enables a choice regarding the most feasible quantitative comparison alternatives.

To further understand the energy consumption intricacies of the three short-range wireless communication technologies ZigBee, BLE, and WiFi, different data loads, were used to make a quantitative evaluation. The study continued the analysis by moving the data pre-processing

into the sensor and making further quantitative evaluations based on the reduced data throughputs.

The study results show that with maximum theoretical throughput, as attainable under ideal channel conditions, ZigBee consumes the least amount of energy when comparisons get made using data loads of up to 500 bytes.

When considering large data payloads measuring 800 kB and more, the study shows that WiFi appears to be best suited. For a wireless transmission load of less than 800 kB and more than 500 bytes, as is the case for typical data-intensive monitoring applications, the results show that the BLE results in minimum energy consumption.

The comparisons get carried out over two different dataset examples; the first was a typical transducer type dataset packets sent regularly and sized around 140 kB of raw data. The second dataset was geared around image transfer and generates dataset packets of sizes more than 250 kB.

Comparing the energy consumption associated with the two different data transmission bandwidths, using the ZigBee, BLE, or WiFi, the results show that the in-sensor processing is the most energy-efficient solution for both of these applications.

Power consumption analysis of Bluetooth Low Energy, ZigBee, and ANT sensor nodes in a cyclic sleep scenario has been studied by Dementyev, Hodges, Taylor, and Smith [54] and provides experimental data comparing power consumption of Bluetooth Low Energy (BLE), ZigBee and ANT protocols for a cyclic sleep scenario. Their setup consisted of a short-range, low-power wireless sensor node that periodically sends a data packet to a remote `hub' with intervening sleep intervals.

For all measured sleep intervals, BLE achieved lower power consumption (10.1 µA, 3.3 V supply at 120 s interval) compared with ZigBee (15.7 µA) and ANT (28.2 µA). Most of the

power consumption differences get attributed to the time taken for a node to connect to the hub after waking up and using sleep/hibernation states between individual RF packets. The study determined that of the three protocols, a sleep interval at which the trade-off between power consumption and data rate is considered the optimal cycle.

The nature of communications between a sensor node and a hub dramatically impacts the power consumed. Unfortunately, much of the literature which compares the power consumption of different radios is theoretical and qualitative [55] or provides generalised experimental data [56] from which it is not possible to draw practical conclusions on power consumption in a given scenario. Furthermore, datasheets do not provide specific parameters or "rule of thumb" that will help a developer balance power consumption and data rate when selecting a radio for an application such as a cyclic sleep sensor node.

The LPNs in this case, which are referred to as slaves within this study, were programmed with a fixed time style hibernation pattern consisting of a cyclic sleep such that they would wake up and transmit an 8-byte packet at the following intervals: 5 sec, 10 sec, 30 sec, 60 sec and 120 sec. The specific cyclic sleep activity at a 5-sec interval is clear from the voltage drop plot across a shunt resistor and shown in Figure 2 for each protocol.

7

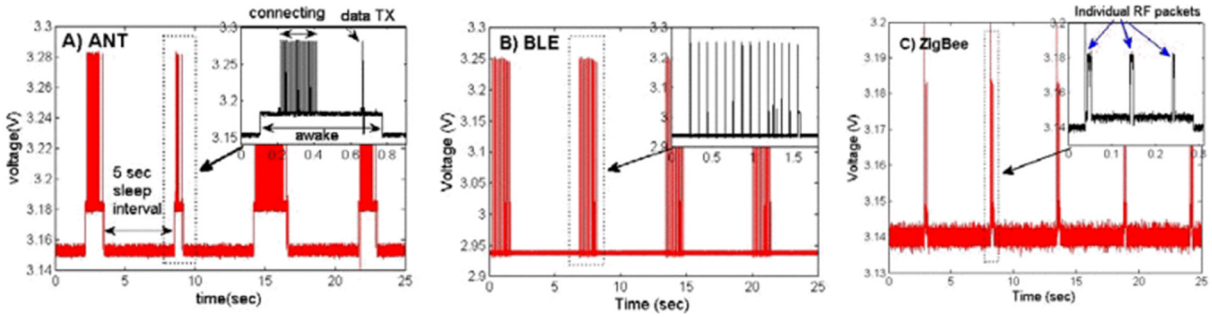


Figure 7: Consumption Characteristics of ANT, BLE, and ZigBee

The LPNs get deployed using a configuration profile that aimed to minimise power consumption by reducing the packet overhead and maximising the time the LPN can remain in an energy reserving hibernation mode. Conversely, the hub (referred to as the master) was optimised to connect with a node as quickly as possible by frequently scanning for new connections.

The conclusions presented from the study suggested that BLE achieved the lowest power consumption, followed by ZigBee and ANT. The study found that the dominating factor that consumed the largest blocks of current was the overhead involved in re-establishing the radio interface upon every wake cycle; this was in contrast to active or sleep currents displayed. It is noted that the duration and frequency of the RF module slept between individual RF packets had effects on the results.

The study suggested that the design and implementation costs in terms of person-hours are that ZigBee was the quickest, ANT took the longest, and BLE was between the two.

Most research and current market offerings point to Bluetooth currently being the winner for the lowest current per useful transmission bit. This outcome is scrutinised in depth by Siekkinen, Hiienkari, Nurminen, and Nieminen [54]. They study the energy consumption of BLE by measuring real LPNs with an integrated power monitor system and then derives models of the basic energy consumption behaviour observed from the measurement results. They continue to investigate the overhead of Ipv6-based communication over BLE, which is relevant for future IoT scenarios that require TCP/IP style communications. The study contrasts their results by performing similar measurements using ZigBee/802.15.4 devices. In addition to concluding that BLE is very energy efficient in terms of the number of bytes transferred per Joule spent, it was also discovered that IPv6 communication energy overhead also remains reasonable and can realistically be operated in a WSN environment.

In terms of accurately modelling the current usages at every stage of the radio cycle, their measurements get performed using a Monsoon Power Monitor [57]. This integrated device provides an adjustable voltage output terminal to which the LPN to be measured can be directly connected.

To characterise and measure the energy consumption of BLE, a BlueGiga BLE112 [58] module (based on TI's CC2540 [59] System-On-Chip) was connected to the power monitor and configured to operate as a slave device. This slave was communicating with a BLE112 USB dongle, which gets configured as a master device. It was ensured that the slave device contained nothing else which would consume energy apart from the bare minimum code required to implement the BLE interface, thus ensuring no bias gets added to the results. In contrast to the nRF BLE stack, this LPN implemented TI's BLE stack (v 1.0) in single-chip mode, and it is considered this represents a realistic example of a sensor where the amount of hardware and power consumption is intentionally kept to a minimum. The slave gets set to generic discoverability and undirected connectability mode.

Similar to the BLE implementation, their ZigBee measurements also used TI's hardware and software offerings. The sensor boards consisted of MSP430F2274 [60] application microcontrollers working with CC2530 [61] ZigBee network processors. The boards are intended for operation in dual-chip configuration (which consists of the MSP430 running the application and CC2530 running network processor firmware). However, the consumption of the CC2530 is the only measurement of interest. Therefore, CC2530 gets flashed with firmware containing the full Z-Stack (v.2.4.0–1.4.0), and the MSP430 was programmed to go to sleep on startup, making this configuration operate as a single-chip solution would.

The comparison is further confirmed valid because the CC2530 SoC is identical to the BLE's equivalent (CC2540); this makes it possible to perform a fair comparison of energy consumption between the two.

The experiment's configuration involved an LPN connected to the power monitor unit, and the LPN then established communications with another LPN, which is also connected to a PC and acting as PAN coordinator. The MAC layer is configured not to use the superframe structure (beacons used only for network discovery); this removes the need for end-devices to listen and wait for the beacon before sending data.

Their results show that BLE indeed consumes extremely little energy and has a desirable ratio of energy-per-bit transmitted. However, the stack used to perform the experiments had certain limitations. The energy efficiency could be further improved by allowing more packet exchanges per connection event and implementing frequency hopping algorithms to combat interference. It is also noted that the discovery energy could be reduced through the design of cooperative mechanisms.

HART [62] [63] is a digital protocol for two-way communication between a host application and smart field instruments, providing access to diagnostics, configuration, and process data. Traditionally, HART specified a physical layer that used frequency-shift keying (FSK) superimposed on the analogue control signal (4–20 mA). As of version 7, HART also incorporates an IEEE 802.15.4-based wireless mesh network as an option for the physical layer. This protocol gets commonly referred to as WirelessHART.

Lennvall, Svensson, and Hekland [64] suggest that ZigBee is not considered suitable for use in most industrial applications, and this unsuitability was the motivation for the development of a new wireless communication standard tailored to industrial needs: WirelessHART.

They continue to point out that many studies have been completed since the availability of all these different wireless technologies, examples of which can be found in [65] [66] [67]. These studies identify the same categories of challenge [68] [69], which surround self-sufficient hardware design.

- Self-sufficiency requires first looking at the low power design of its own operating hardware and consumption
- They must understand the limitations of the resources they have available
- They must be able to network in some way

With these goals in mind, the study proposes an LPD solution that operates around the BLE One Chip Solution CC2540F256 [70] from Texas Instruments (TI). The CC2540F256 consists of a BLE transceiver alongside an integrated 8051 microcontroller running a BLE software stack provided for free from Texas Instruments [71].

Their study successfully creates a low-power device that is self-sufficient by being wholly powered via an energy harvester. The LPD also provided BLE network connectivity with good throughput and very low overhead.

Deep Sleep Hibernation Strategies

Using sleep/hibernation patterns to influence energy consumption is not something new in the microcontroller world, and research conducted by Wendt and Reindi [72] into various wake-up methods used to extend the battery life of wireless sensor nodes.

They identified that to help expand the operational life cycle of LPDs, sleep/wake strategies that significantly affect the controller's consumption by reducing its power need to be utilised. Their work focuses on comparing and evaluating wake-up strategies for wireless sensor applications [73].

To assist in their proper evaluation, a custom demonstration platform capable of wireless communications and current consumption measurements was developed and provided the foundation for this type of study.

Four common multiple access methods suitable for wireless sensor networks [65] get identified and categorised into:

1. Time diversity
2. Frequency diversity
3. Code diversity
4. Space diversity

These strategies represent four very different areas in which LPD consumption rates can be influenced, breaking each of the domains down into four comparable experiments.

Time Domain Strategies

This experiment consisted of sharing and synchronising a common clock source. This clock is used to synchronise the wireless host/controller/hub and LPNs to a standard time base. Every

sensor node in the network is then assigned a specific timeslot in which data gets transmitted to the host. The LPN needs only to be awake and have its radio initialised and ready to utilise its allocated slot. When the slot has ended or the transmission has completed, the LPN returns to a sleep state.

Frequency Domain Strategies

Frequency diversity is achieved using an additional radio frequency or channel to implement a low-power wake-up signal used to bring an LPN out of hibernation. The LPNs are all equipped with a 2.45 GHz frequency band transceiver and additionally with a 125 kHz receiver. The 125 kHz receiver is designed to decode an address value embedded within its message and, if applicable, will power up the 2.45 GHz transceiver ready for data exchange. Communication on the 2.45 GHz band is faster but consumes more power than the 125 kHz receiver. Table 1 below shows a typical difference see in current requirements when comparing operational frequency.

specification	**2.45GHz** RF-Chip ZigBee CC2420 [6] Texas Instruments	**125kHz** RF-Chip ATA5282 [5] ATMEL
maximum data rate	250 kbps	4 kbps
current consumption	18.8 mA at 3.3 V	3.5 µA at 3.3 V

Table 1: Comparison of two typical wireless receivers

Software Domain Strategies

Code diversity represents the most widespread approach to influencing the consumption rates of an LPD. The solution involves the broadcast of every message sent. The broadcast embeds the target node's address, every LPN receives the broadcast and decodes the target address field. At this point, every LPN will return to hibernation apart from the LPN, which gets targeted within the message, and this device then continues to communicate with the host.

This wake and listen method is already widely adopted within the wireless networks domain.

Environmental Domain Strategies

The study's fourth category considers the implications of dividing the radio transmission area down into segments that get targeted using a physically or electrically narrow beam rotatable antenna or multiple input multiple output antenna. This system will scan the segmented areas where communication needs establishing with at most one LPD per segment.

The frequency diversity method is also undoubtedly relevant to this research and presents an exciting option. The idea of having a very low power wake-up addressing channel that could share as much hardware resource as possible as the primary data transfer channel is very desirable. This idea has also been studied by Durante and Mahiknecht [74] in some detail where they present an ultra-low-power 2.4 GHz RF Wakeup Receiver designed primarily for wireless sensor networks nodes. The receiver demodulates On-Off Keying at 100 kbps. A 120 nm CMOS chip includes the analogue front-end and consumes only 7.5 uW from a single 1.5 V supply.

The study states that nearly 75% [73] of the total power consumed from a typical WSN node gets attributed to the radio transceiver. They identify that one leading cause is the receiver,

which gets turned on frequently to listen to possible incoming messages and avoid high latency communication. The radio channel gets sampled using intelligent strategies in the MAC layer, such as those described in [73] or [66], to synchronise communications and optimise power consumption. Using these strategies, very low power consumption is only achieved at the cost of data throughput and latency because communication occurs only during small time windows at low periods (e.g., once per second). In such situations, packets have a worst-case end-to-end delay equivalent ideally to the period of the cycle multiplied by the number of hops.

Wakeup-on-Radio, Wireless Wakeup

Pletcher, Gambini, and Rabaey also present an ultra-low-power RF-based wake-up receiver, using a lower frequency. A complete 1.9 GHz receiver, with BAW (Bulk Acoustic Wave) resonator-referenced input matching network, is designed as a wake-up receiver for wireless sensor networks. This integrated solution consisted of creating a 90 nm CMOS IC with an integral RF amplifier, PGA (Programmable Gain Amplifier), ADC (Analogue Digital Converter), and reference generation peripherals while consuming a mere 65 μW from a single 0.5 V supply. The RF receiver's input bandwidth achieves 7 MHz, providing a maximum data rate of 100 kbps. The receiver exhibits -56 dBm sensitivity for a 90% probability of detection of a 31-bit address sequence.

A tuned RF (TRF) architecture with on-off keying (OOK) gets implemented for simplicity reasons. The TRF architecture by design eliminates the requirement for a local RF oscillator, which is considered expensive in terms of current consumption.

Careful selection of technologies for every system and peripheral on the IC is made to ensure all parts could operate from an aggressively scaled 0.5V supply.

From the amount of research available regarding low-power wake-up signalling channels, it is clear it has become a universally adopted method of maximising the LPD's energy efficiency. Lu, De, Xu, Song, and Cao [75] propose a novel wake-on sensor network design. In this context, we have designed a new sensor platform called TelosW. The wake-on sensing capability of TelosW lets designated sensors wake up the microcontroller (MCU) only on the occurrence of some event with a pre-configurable threshold.

TelosW integrates the CC1101 Wake-On Radio (WOR) hardware, and this enables the MCU to offload the processing needed to continually listen for and decode any address attributes or

threshold data from incoming messages. Upon successfully receiving a targeted message, the CC1101 hardware will assert an interrupt connected to the MCU, allowing it to wake and take appropriate action. This entirely event-driven wake-on sensor network can reduce energy consumption considerably.

Fully Integrated Solutions

The TelosW platform from the study gets equipped with a very precise onboard energy meter capable of performing in-situ energy consumption measurements. As each LPN has the same energy meter, it is possible to collect data and consider all nodes' individual energy states in a network at any point in time. As most mesh-type topologies use a form of flooding protocol, this data provides an insight into how the consumption spreads throughout the network, thus making it possible to analyse and compare protocols for their energy efficiency.

This paper presents a very relevant study concerning the research presented in this thesis. It is shown that predetermining the precise amount of energy needed to exchange a message between two nodes situated at arbitrary routing location points within a mesh network topology is possible.

The energy meter used to take the measurements allowed precise in-situ sampling while incurring very little self-consumption or adding interference into the application. Dutta, Feldmeier, Paradiso, and Culler [76] present an energy meter hardware design geared around 'free-energy' metering. This type of design is implementable in large numbers as part of a distributed WSN. Incorporating the meter design on every LPN allows insight into the consumption patterns of LPNs in a network of any size.

TelosW has a wake-on hardware feature to aid in maximising the duration the LPD hibernates. The wake interface gets configured so any onboard sensors can wake up the MCU based on

configurable event thresholds. Coupling this feature alongside its interrupt-driven wake-on radio hardware enables event-driven wireless communication and minimum energy consumption. The design combines the use of all the current popular energy-saving techniques into a highly integrated SoC.

The wake-on capability of *TelosW* to hibernate until a configured sensor or radio wakes it up allows further peripheral optimisations. These types of selective configurations allow the system to selectively use its ADC module to sample for configured threshold levels during sleep. The wake-on radio also uses a duty-cycle approach to decrease its consumption further.

Unintrusive Energy Metering

The onboard energy meter hardware and the relevant driver (with proper calibration) of *TelosW* allow the facility to track each LPNs real-time energy consumption. This facility enables network-wide energy data collection utilising the network itself to retrieve it. The feature gets used to track energy consumption distribution across the entire network from time to time. It gives a detailed real-time view of the energy consumption incurred by the LPNs throughout a network of any size.

The monitor hardware itself is based on the MAX1724 [77] IC from Maxim Technologies.

8

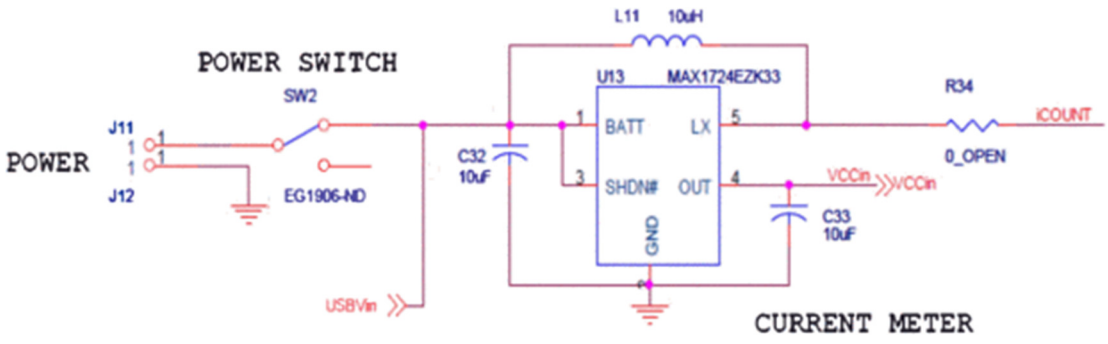


Figure 8: Emergy Meter Implementation

The onboard energy meter of *TelosW* uses the idea and design of *iCount* [76]. *iCount* proposes a hardware design that allows the measurement of energy consumption without adding any loading attributes to the circuit it is trying to measure or by coupling energy using field-effect type measurement techniques. This design can be implemented on ultra-low-power sensor nodes to track and record their energy consumption. The iCount design measures energy usage by counting the regulator's switching frequency instead of interfering with the actual flow of the current path itself. The operation principle utilised here is based on the relationship between the load current and the switching frequency being linear.

The IC implements and maintains an energy meter counter value, representing the current count of the regulator switching frequency. As the switching power-supply used to power the LPN adjusts its switching frequency relative to the current demand presented to it, the energy meter count will change to reflect it – synonymous to how a domestic energy meter tracks user consumption usage rates.

The meter needs to be calibrated before usage to provide the expected linearity. The calibration computes the coefficients which influence the relationship between the counters value and the load current variation against the input voltage. To achieve the calibration, the ADC peripheral gets used to measure the input voltage level. This dependency reveals a limitation with this kind of level measurement implementation in terms of not having a fixed reference to compare the current input level against. Due to this and the fact that the battery will be continuously discharging, frequent re-calibration routines are necessary to maintain accuracy. The *TelosW* implementation has a resolution of 0.1 V when sampling input voltage levels, meaning that re-calibration will occur for every 0.1 V variation in input voltage measurement.

Their study presented Figure 9 below, which shows an energy distribution pattern from a mesh of 17 LPNs, all sharing the same design. It is seen that the closer an LPN is to the controller node (marked 0 in the figure), then the more consumption incurred.

The LPNs located one hop from the controller will carry the highest consumers. In this study, LPN 2 was the highest consumer due to the forwarding responsibilities it has for both LPN 8 and LPN 9. This pattern will continue to be the case as the network continues to grow.

9

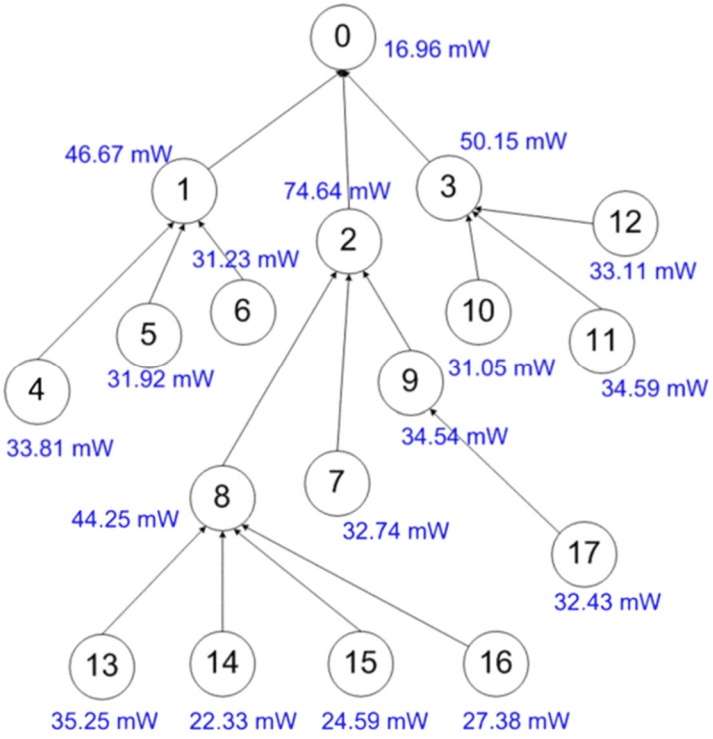


Figure 9: Distributed LPNs Measured Consumption

The controller LPN 0 is surprisingly the lowest consumer, the reason being that the study's implementation required the controller to operate in a receive-only mode without the need to power its transmitter. According to the datasheet, the CC1101 will consume approximately 15.6 mA in receive mode and 32.3 mA in transmit mode (at 250kBaud data rate and 10dBm output power).

Mesh Networking Protocols

During the development of ZigBee, other protocols have spun off, all chasing the goals of lowest energy, easy set-up, reliability, and usefulness. Z-Wave and Z-Wave Plus are good examples of one of these newer offerings. Z-Wave tries to address some of the ZigBee implementation's inadequacies, including introducing various encryption and security levels, gearing nicely towards companies wishing to have an easier deployment to market experience. Z-Wave includes proprietary, closed source code, ensuring any intellectual property, internal operation details, and protocol implementation are all considered secret and not shared with any user base. Although many considered this to be a weakness compared to the open-source ZigBee offerings, it has not led to a lack of large well-known manufacturers adopting the protocol; as of this time, over 700 companies have created an alliance to promote and use the protocol. Z-Wave enjoys much adoption from within the security companies sectors offering wireless security aligned products.

ZigBee has been released under an open-source license scheme, so any manufacturer who wants to utilise the code can obtain the protocol source code, inspect it, adopt it, and even modify it as long as the licence restrictions are adhered to. ZigBee's latest version 3 offering introduced a unified standard that works more alongside how Z-Wave works.

According to Z-Wave marketing material [78], over 35 million different devices sold today use Z-Wave wireless standards. It is a proprietary technology that requires a license for companies to incorporate in their products.

When comparing differences between the two protocols which impact device deployment, it is notable that ZigBee transmits data at a faster 250 kbps compared to Z-Wave's transmission rate of 100 kbps. However, the extra speed comes with the drawback of range; ZigBee has a range of 35 feet, whereas Z-Wave can transmit and receive at up to 100 feet.

Transmission distance gets somewhat enhanced for both protocols by the fact they can mesh. Having more than one ZigBee or Z-Wave device within near-proximity of each other will enable them to additionally act as a repeater or booster, allowing the signal to propagate further.

ZigBee and Z-Wave both offer multiple means of controlling the provisioning of new LPDs into their networks and can add and remove LPDs in a secure manner. Both protocols have an advantage over BLE when it comes to being energy efficient, and this is due to the operating frequency choices made during design time.

Where BLE and ZigBee can both use and occupy the 2.4GHz spectrum, enjoying the fast speeds available, ZigBee can also drop down to 900MHz (868MHz for Europe, 915MHz for the US) band where it can immediately tap into the slower and less energy demanding frequencies. Although throttled severely by speed, the extended range vs lower energy is desirable. The lack of speed (for this research) simply falls into the time dilation solution. The other glaring advantage of this is avoiding radio interference from widespread WiFi and Bluetooth protocols in operation.

When considering these options for deployments of large sensor networks, it must be noted the built-in device maximums which the protocol can provision. Z-Wave has a 232 device limit, which ZigBee overcomes by offering 65000. Any difference in the underlying communication rules significantly impacts speed and performance. Finally, Z-Wave allows only four hops between a device and a hub, while ZigBee imposes no restriction.

Danbatta and Varol conducted research comparing ZigBee, Z-Wave, and Bluetooth in the home automation arena [79]. They identify and evaluate various comparable attributes offered by the technologies and make critical comparisons based on these indices. They consider power consumption, propagation range, cost, ease of use, scalability, and interoperability.

The study showed that for power consumption, Z-Wave, Bluetooth, and ZigBee have a low power consumption of 1mw, 10mw, and 100mw, respectively. While WiFi has high power consumption.

Propagation data shows WiFi has a range of up to 1000m, with Z-Wave having 30m, ZigBee 100m, and Bluetooth with the lowest having 10m. The introduction of Bluetooth 5 long range [34] technologies has greatly increase the operable range for this protocol towards the 50-100m mark.

As for the cost indices, Bluetooth is the cheapest choice, with ZigBee, WiFi, and Z-Wave following, respectively.

The scalability and interoperability indices are somewhat intertwined concerning scaling the network up using products from different manufacturers who have implemented the same protocol. In this situation, it is found that Z-Wave is the best choice because it allows commercial devices and hundreds of nodes with high density.

Table 1 below summarises the results presented in the study.

Indices	*ZigBee*	*Z-Wave*	*Wi-Fi*	*Bluetooth*
Power consumption	100 mw	1 mw	High	10 mw
Range	100 m	30 m	1000 m	10 m
Cost	Low	High	Medium	Very low
Scalability	6000	> 6000	32	20
Interoperability	Same manufacturer	Different manufacturers	Wi-Fi compatible devices	Bluetooth compatible devices

Table 2: Comparison of wireless technologies for home automation

The study concluded that *"Z-Wave is the best choice if the variable of interest is energy saving, or the user is interested in using different devices from different manufacturers, or even*

commercial devices. When the variable of interest is using devices from the same manufacturer with relatively low-cost and power consumption, ZigBee is the best fit in such a situation. In the event ease of use and range is the interest of the user, then the WiFi solution is the most appropriate. However, if cost is the issue, then Bluetooth would provide a better solution."

IoT – Internet of Things

As it stands at the time of writing, many of the alliances that emerged for the creation of both ZigBee and Z-Wave are also in support of an emerging protocol named CHIP (Connected Home Over IP) [80].

As the Internet of Things (IoT) has evolved, the need has become apparent for more substantial unity among brands and ecosystems to enable products within smart environments to work together more efficiently. Working to meet that need, the ZigBee Alliance seeks to promote collaboration in the Internet of Things by creating, evolving, and promoting universal open standards that enable all objects to connect and interact.

The protocol intends to simplify product development for device manufacturers, broaden consumer choice, and ensure easy discoverability, deployment, and engagement to fuel connected living.

The standard uses IP — internet protocol to allow for the broadest inclusion of products and address privacy and security issues.

The technology rollout will begin with Wi-Fi, Wi-Fi 6, Thread, and Bluetooth Low Energy. Also expected to be included will be Ethernet connections, cellular and broadband.

Thread [81] is an IPv6-based networking protocol designed for low-power Internet of Things devices in an IEEE 802.15.4-2006 wireless mesh network or Wireless Personal Area Network

(WPAN). Thread is independent of other 802.15 mesh networking protocols, such a ZigBee, Z-Wave, and BLE.

Thread's primary features include:

- Simplicity — Simple installation, start-up, and operation
- Security — All devices in a Thread network get authenticated, and all communications are encrypted
- Reliability — Self-healing mesh networking, with no single point of failure, and spread spectrum techniques to provide immunity to interference
- Efficiency — Low-power Thread devices can sleep and operate on battery power for years
- Scalability — Thread networks can scale up to hundreds of devices

Thread solves the complexities of the IoT, addressing challenges such as interoperability, security, power, and architecture requirements. It is a low-power wireless mesh networking protocol based on the universally supported Internet Protocol (IP) and built using open and proven standards. Thread networks have no single point of failure and include the ability to self-heal. They are simple to set up, use and will auto-reconfigure when a device gets added or removed. Thread ensures end-to-end communication in most topologies, device-to-device, device-to-mobile, and device-to-cloud.

BLE Meshing

Mackensen and Lai have performed an in-depth feasibility study on using wireless sensors with BLE [82], with references to energy harvesting. They concluded that connection setup could be established in as little as 0.3mJ, with a data transmission consumption of 0.13mJ possible. Coupled with research performed by Mackensen and Wendt [83] regarding possible output

powers of commercially available energy harvesters, RF has been proven to provide mW to W or energy, Solar typically uW to mW. To improve efficiencies further, Feng, Mo, and Li found that the average current of BLE data transmission was reduced significantly by its packaging strategy [84]. They discussed a particular method of transmission which allowed considerable current saving for low data rates.

In 2016, Zenker, Krug, Binhack, and Seitz [85] conducted research to evaluate the meshing capabilities of BLE; their simulations showed great potential for the challenges presented by wireless networks and IoT. This work has been made concrete by releasing an official version with choices of topologies by the Bluetooth controlling body [86]. Building the best type of meshed BLE network has been researched by Chiumento, Reynders, Murillo, and Pollin [87], where they found how each setting such as transmit power, connection interval, and source rate impact overall network performance figures of merit such as end-to-end delay, packet delivery ratio and network build time. BLEMesh is an interesting broadcasting network protocol presented by Kim, Lee, and Jang [88] where the available data payload using Bluetooth Low Energy Generic Access Profile is identified (Non-connectable Advertisement Data for the different number of nodes and packets to send in a batch). Then it is compared with BLEmesh using both a conventional routing method and a flood routing method. Their preliminary evaluation shows that the number of transmissions by BLEmesh is significantly smaller compared to its competitors for some selected network configurations, reducing the aggregated energy consumption within the mesh network.

Further energy savings can be employed as researched by Murillo, Reynders, and Chiumento [89], where the setup's orientation gets considered, and they took a measurement-based comparison of two mesh approaches that fit within BLE operation: flooding and connection oriented networking. Using metrics such as packet delivery ratio (PDR), end-to-end delay, and power consumption, they concluded that the optimal mesh approach depends on the

application. It is shown that for comparable performance in PDR and overhead, flooding could trade a lower end-to-end delay for higher power consumption than the connected mesh. They present a method of automatically switching between the two methods based on a message priority approach.

In addition to these topology studies, an energy-efficient reconfigurable scatternet formation algorithm has been proposed by Yu and Yu [90]. Here a root node propagates parameters *k* and *c* in the downstream direction to construct a tree-shaped subnet and determines new roots. Each new root then sends a request to its upstream master to initiate a return connection to convert the first tree-shaped subnet into a mesh-shaped subnet. Following this, a peak-search method is used in the first root node to locate the optimum *k* layer and generate an optimum mesh-tree topology. This method's simulation results showed that the optimum *k* layer could then be determined and achieve hop-length performance. Another topology formation algorithm for a scatternet gets introduced by Sheng, Qun, Qiang, and Jian [91], where they use a recursively executed sequence of parameter acquisition, node assigning, and link setup to achieve greater robustness and expansibility within the mesh.

BLE Sensors

BLE has also introduced and refined the possibility of using the link to aid distance and direction-finding capabilities. Kajdocsi, Kovács, and Pozna [92] have highlighted a very novel suggestion which potentially improves both accuracy and computational capabilities of indoor positioning using meshed BLE devices. Their work concluded that if the transmitters of position information can create a mesh topology, they may transfer data to each other and the user, thus aiding various algorithms such as dead reckoning and triangulation to be achieved. Further supporting factors as to BLE being a good choice get shown by the performance

comparison done by Yun, Lee, An, Kim, and Kim [93], where positioning schemes are evaluated using Wi-Fi and BLE. They utilised the received signal strength (RSS) of the wireless device and analysed the estimated indoor position's accuracy by using the weighted centroid localisation technique. They concluded that BLE performance is optimal under shorter distances, which is ideal under a mesh network condition. The number of BLE devices aiding the positioning accuracy has also been studied, notably by work done with Ji, Kim, Jeon, and Cho [94], where path loss is analysed and how the introduction of more devices impacts the serviceability.

Low-Power-Devices Design

Considering other research in energy management techniques and efficiency for low voltage sensors and applications, various experiments have been performed to enable maximum use of tiny harvested energies in ways that make capturing even the smallest amounts worthy. Newell, Twohig, and Duffy [95] have performed an in-depth analysis of step-up DC-DC convertors connected to various types of solar-based harvesting inputs. They concluded that there exists an optimal number of series-connected cells where the optimised output power can be identified, resulting in upwards of 15% improvements in the convertor stages.

Donovan, Dewan, Peng, Heo, and Beyenal [96] researched the viability of using a Sediment microbial fuel cell alternative to renewable power, allowing them to enable remote sensors requiring less than 2.5W to function. They developed a power management system that enabled the source to operate the sensor, which effectively stored microbial energy in capacitors, allowing short bursts of power to be realised.

Furthering this idea of alternative energies, Chung-Yang and Nan-Chyuan [97] worked on extracting energy from the human body itself. They experimented with Micro-electromechanical systems (MEMSs) based energy harvesters, characterising various techniques, operation principles, bottlenecks and presenting performance, frequency tuning techniques, and biocompatibility micro energy harvesters.

Some further excellent research is presented by Newell and Duffy [98] again, focusing on the area of improving the efficiency of the conversion chains coupled to the harvester inputs. They analysed the effects of utilising switched capacitor energy buffers injected between the harvester and the boost convertor/load parts for wireless sensors. Performing loss analysis and careful impedance matching allowed them to realise gain increases of up to 10% in terms of efficiency of the power delivered from low-voltage, low-power EH sources over an existing

boost converter. This increase is due to the switched capacitor circuit enabling operation in a similar pulsed manner as the wireless sensor, thus reducing the continuous conversion losses encountered by many commonly available converters.

Ranvijay, Yadav, Kumar, and Agrawal [99] proposed a Dynamic Voltage Scaling solution for an energy harvesting real-time system. They proposed a scheduling algorithm that offers less energy consumption for battery-powered dynamic real-time systems modelled with aperiodic tasks and energy harvesting constraints. The proposed approach has to decide which speed or voltage level is to be selected, leading to a reduction in energy overhead and timing overhead due to the speed switching. The simulation results and examples illustrated that their approach could effectively reduce the overall system energy consumption and improve the system performance in terms of remaining energy and reduce the rejection ratio of aperiodic tasks.

Khairudin and Salleh [100] researched the increased efficiency of energy harvesting after introducing multi-input source harvesting circuits. By exploiting the ambient energy from mechanical vibration, light, and thermal gradient fed into various power management circuits, including LTC 3588-1, LTC 3588-2, and a voltage doubler, they present a simulation, design, fabrication proposal, and testing of double-layer multisource energy harvesting circuit. The result shows that the recommended power management circuit for low power energy harvesting power management is the LTC3588-1 for low power and voltage generated. All the power management circuits can be used together at the same time but require a capacitor to stabilise the output of voltage if any of the input sources have no input voltage, and when all the multiple input source is present, the passive switching will choose the higher available voltage from the circuit.

Further touching on the design issues presented when developing various kinds of low-power sensor systems, Squires and Huff present a white paper [101] that covers many fundamental

design issues which need considering, including Power Budgets, Storage Devices, Harvesting Solutions, MSUs, and connectivity. Using a basic configuration for a standalone sensor system as a reference, their discussion covers sensor/transducer types, power budgets, power sources (especially devices that harvest ambient energy), and energy storage solutions. It also summarises microcontroller requirements and highlights the latest power-management chips and wireless ICs. A good working example is presented, which covers all the issues individually.

Another good working example involving ultra-low power energy harvesting techniques get presented by Gorlatova, Kinget, Kymissis, Rubenstein, Wang, and Zussman with Ultra-Low-Power Energy-Harvesting Active Networked Tags (EnHANTs) [102]. EnHANTs are small, flexible, and self-reliant (in terms of energy) devices that can be attached to objects that are traditionally not networked (e.g., books, clothing, and produce), thereby providing the infrastructure for various novel tracking applications. Examples of these applications include locating misplaced items, monitoring objects (items in a store, boxes in transit), and determining disaster survivors' locations. Recent advances in ultra-low-power wireless communications, ultra-wideband (UWB) circuit design, and organic electronic harvesting techniques will enable the realisation of EnHANTs in the near future. For EnHANTs to rely on harvested energy, they have to spend significantly less energy than Bluetooth, ZigBee, and IEEE 802.15.4a devices. The research explores various energy harvesting techniques. It concludes that EnHANTs necessitate a rethinking of communication and networking principles and require careful examination of ultra-low-power and energy harvesting technologies' particularities. The research shows that the nature of EnHANTs requires a cross-layer approach to enable effective communications and networking between devices with extreme power and harvesting constraints.

Moghe, Divan, and Lambert [103] experimented with using magnetic-field and solar energy harvesting to power low-cost utility sensors in research supported by the National Electric Energy Testing Research and Applications Center (NEETRAC). The research presents a novel 0.2 V to 3.3 V AC-DC boost converter, which converts the magnetic field (H-field) and solar energy to electrical energy used by the sensor electronics. A supercapacitor gets used to operate the sensor even in an outage. Numerous design constraints are identified, and a low-cost power circuit was fabricated, built, and experimentally tested to show functionality under varying operating conditions.

Energy Scavenging

A further twist on powering wireless sensor networks gets presented by Zungeru, Ang, Prabaharan, and Seng [104], where they explore the benefits of radiofrequency energy harvesting. They present a practical approach for RF Energy harvesting and managing the harvested and available energy for wireless sensor networks using the Improved Energy Efficient Ant Based Routing Algorithm (IEEABR). The research covered the RF power density measurement, calculation of the received power, storage of the harvested power, and management of the power in wireless sensor networks. The routing used IEEABR technique for energy management. Practical and real-time implementations of the RF Energy using Powercast™ harvesters and simulations using the energy model of Libelium Waspmote to verify the approach were performed. Comparisons of IEEABR and other traditional energy management techniques get explored while also looking at open research areas of energy harvesting and wireless sensor networks management.

The power management implementations keep appearing in most research done within energy harvesting and are clearly a critical area in achieving useful efficiencies from the tiny amounts of harvestable ambient available. Kansal, Hsu, Zahedi, and Srivastava [105] reinforce this with their research exclusively in the power management and energy harvesting arena. They present various considerations in using an energy harvesting source fundamentally different from using a battery, specifically, rather than limiting the maximum energy. It has a limit on the maximum rate at which the energy can be used. They continue to point out that harvested energy availability typically varies with time in a non-deterministic manner. Simultaneously, a deterministic metric, such as a residual battery, suffices to characterise the energy availability. In the case of batteries, a more sophisticated characterisation may be required for a harvesting source. They also touch on another issue that becomes important in networked systems with multiple harvesting nodes, in that different nodes may have different harvesting opportunities.

The same end-user performance may be achieved in a distributed application using different workload allocations and resultant energy consumptions at multiple nodes. In this case, they show it is crucial to align the workload allocation with the energy availability at the harvesting nodes. The research develops abstractions to characterise such sources' complex time-varying nature with analytically tractable models and uses them to address critical design issues. They continue to develop distributed methods to efficiently use the harvested energy and test the models both in simulation and experimentally on an energy-harvesting sensor network, prototyped especially for this research.

The methods presented in their research enable using environmental energy in a harvesting aware manner and adapting in real-time to energy availability. This yields significantly higher performance levels than the existing approach of using a conservative duty cycle in solar power systems, which is designed for expected worst-case scenarios. The methods discussed addressed some of the more common power-scaling mechanisms and usage scenarios. Several other more sophisticated power scaling techniques are available in more advanced low-power hardware, and these methods may be modified to exploit all such techniques as suitable for maximising performance at the application layer.

Chapter Summary

With such a vast area of topic to cover, the literature review was never going to be small. The volume of available works currently appearing clarifies that wireless meshing networks are genuinely beginning to dominate the low-power distributed device communications network arena.

There is also no lack of studies in energy harvesting topics and the delicately balanced chain of components needed to harness and capture the ambient wireless energy offerings. Rectenna designs and evaluations are plentiful and progressing rapidly, continually finding new, more efficient ways of rectifying, storing and using their harvests.

It is also of interest to see more and more custom silicon solutions presented, suggesting maturity of the problem domain and a willingness to risk larger investments to realise the benefits. Eventually, as Nordic, Silicon Labs [58], and the other big silicon players continue to show, the integrated circuit solution will conquer and provide the fastest, lowest-powered and most implementable solution. As an example, the latest offering from Nordic [106], as of point of writing, is an nRF5340 Dual-core Bluetooth 5.2 SoC (System On Chip) boasting a network processor clocked at 64 MHz and optimised for low power, radio and efficiency.

Chapter 3: The research approach in this thesis (Original Content)

To offer the most flexibility for both this research and the potential continuation of future experimentation, a full-featured prototyping platform has been developed, consisting of controllable energy suppliers, measurable energy consumers, and a communication network to link the consumers together with external control equipment.

As this research concerns *efficiently* using the ambient energy currently available, there is no need to spend considerable time tuning energy conduction systems in search of achieving maximum power transfer. This subject is, of course, touched on and presented due to its importance in the design process; however, it is not the ultimate goal of this research.

The research platform developed can provide and sustain an average lab sized infrastructure where energy deliverance can be controlled and measured and can also provide multiple sources of energy streams, allowing the consumers to choose.

There are far more efficient ways currently available to harvest even smaller amounts of energy than this platform can achieve, which is improving at a considerable rate. The designs presented in this research satisfied all the requirements needed to achieve a stable experimentation platform and deliver consistent, comparable data needed for correct evaluation.

The main goal was to provide an environment that allows the devices under test to be entirely self-sufficient and decoupled as much as possible from any control of the energy sources fed to them. This solution includes the ability to set up a communications network and send messages back and forth between devices and controlling hardware.

Research design

The key points for the hardware requirements include:

- A power controllable wireless energy transmission source
- A power controllable and positionable solar lamp
- A collection of low-power devices capable of harvesting both wireless energy and solar energy
- A radio network that allows the devices to communicate
- A proxy device used to connect the platform with an end-user

Hardware

Both the wireless energy transmitter and solar lamp can have their output power adjusted; the solar lamp can also be re-positioned. Both devices are both classed as energy 'suppliers' and treated as separate uncoupled hardware systems. They provide an artificial version of the sun and ambient wireless power sources.

However, the collection of low-power devices get categorised as energy 'consumers.' Although they have no need to communicate directly with the energy suppliers, they communicate with other consumers and the proxy.

The proxy is neither a supplier nor a consumer, merely an enabler.

Energy Consuming Devices

The *energy consumers* are the devices that receive all the research focus and are the most involved in terms of design and construction.

These devices represent the ultra-low-power work-executing self-sustaining part of the platform, and they get referred to as LPDs (Low Powered Device), or if the device has joined a meshing network, it can also be considered an LPN (Low Powered Node).

Two test LPD designs exist; the first is based on an ATtiny85 [31] and commissioned for the sole purpose of exploring the difference between fixed, threshold-based and adaptive sleep strategies. Thus, this design did not include a radio interface, only a solar cell, capacitor, LED, and MCU. A variant of this LPD using a Microchip PIC16F1823 [107] also exists and capable of performing the same duties (Figure 10). The principle behind this allows comparisons using different vendor offerings; however, recent years have seen Microchip and Atmel merge into a single company.

10

Figure 10: LPD Using PIC16 MCU

The second LPDs design is based around a Nordic Semiconductors Nrf52832 [106], a powerful, highly flexible ultra-low power multiprotocol SoC (System on Chip). With an internal clock speed of 64MHz, 512Kb of flash, and 64Kb of ram, it is more than adequate to interface all required peripherals. Current consumption of 58µA/MHz is possible, with sleep currents as low as 1.9µA. The device gets configured to operate in a low current DC-DC mode,

where it uses an external inductor and capacitor to enable its internal boost circuitry. The primary oscillator is a 32MHz crystal, and the low-frequency clock is enabled using its internal resonator. The operational input voltage is from 1.7V to 3.3V.

Power comes from various ambient energy-receiving transducers connected to a BQ25570 Nano Power Boost Charger and Buck Converter [108]. This device is a highly efficient energy harvesting controller capable of capturing mere microwatts of energy and storing it in an energy store. The BQ25570 offers automated threshold-based monitoring of the energy levels and provides an output line used to power the microcontroller via a MOSFET transistor when sufficient energy has been collected.

The energy store consists of a bank of supercapacitors rated up to 5 volts. The BQ25570 [108] gets configured to prevent the charge voltage from rising above 3.3V; this allows the microcontroller to operate directly from the energy store without further regulation.

Bluetooth 5 Low Energy connectivity [109] comes from the nRF52832 [106], a 2.4GHz RF output is available, which gets connected to a PCB microstrip antenna. Transmission up to +4dBm of power is possible with data rates up to 2Mbps. The microstrip antenna is feed point matched to 50Ω using an impedance matching network configured in a PI arrangement.

Built upon previous work from the author of this thesis, originally named BluBot [110], this hardware platform is re-engineered and continues the development and design produced and detailed in the BluBot paper [110]. The new style stackable configuration breaks down into four independent modular subsystems:

1. Energy harvesting and management – *The Stomach*
2. Central system and communications – *The Brain*
3. Cognitive functions – *The Legs*
4. Universal expansion – *The Hands*

11

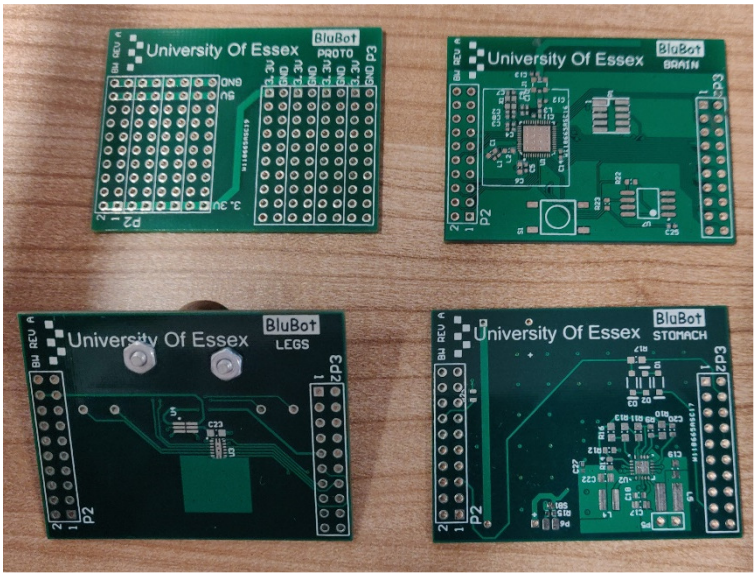


Figure 11: LPD PCB Modules

These modules can be operated independently of each other for testing and characterising purposes; as the research progresses, the complete stack-up connects together and becomes fully operable.

For the design's cognitive aspects, two motors have been installed and controlled via PWM using a DRV8836 dual DC motor driver [111]. Offering a 95nA sleep current and operation from 2V, it provides two H-bridge type outputs possible of delivering 1.5A each. This part takes two PWM inputs where the duty cycle gets used to determine the motors' speed; it also provides two inputs used to change travel direction.

EEPROM inclusion provides a non-volatile memory system allowing for the long-term storage of learned skills and work-task parameters, environmental discoveries, other LPDs, and leadership hierarchy details.

A generic ‘Proto’ board exists, intended to allow flexible design expansion and testing and implement different ambient energy receivers, rectenna circuits, and antennas for critical comparison.

The finished PCB modules are shown above in Figure 11.

Figure 12 below shows the populated and tested CPU module; the EEPROM (U7), planned for a later date, is used to retain non-volatile coefficients, task information, energy consumption details and planned job lists.

The processor's radio hardware has been impedance matched to the microstrip antenna located at the top of the PCB, and a 50Ω high frequency switched connecter soldered down to allow easy connection of tuning hardware.

12

Figure 12: Assembled Brain (CPU) Module

The populated energy harvesting modules shown in Figure 13 below are commissioned with the power level thresholds set using high resistance resistor dividers (thus minimising current consumption of the setup). A supercapacitor mounted in the lower-left corner becomes the energy store, and room made available for additional placement of energy store capacitors if needed.

13

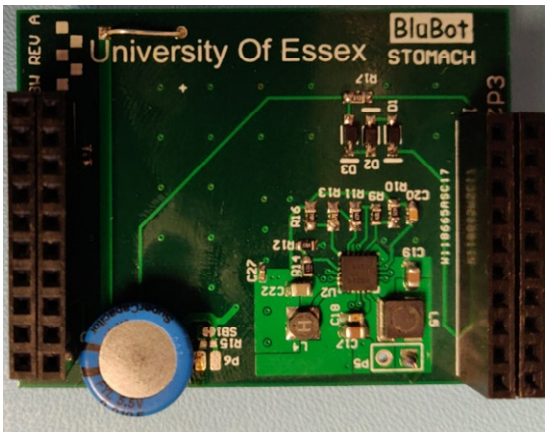


Figure 13: Assembled Stomach (Harvesting) Module

The three modules (CPU, Energy Harvester, and Proto carrying the rectenna and energy reception circuitry) are stacked together to form a modular, expandable, and configurable hardware stack, as shown below in Figure 14.

14

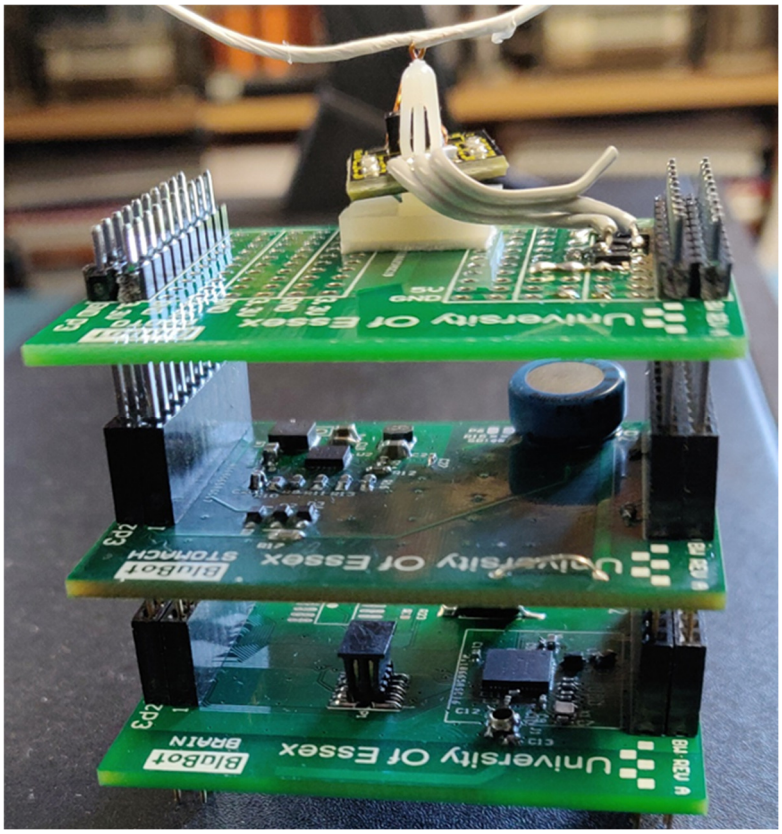


Figure 14: Stacked unit

The rectenna circuitry has been tuned and connected to the coil antenna interface; this is visible in Figure 15 and Figure 16 below. This setup gets replicated to five other devices initially, providing a batch of 6 units for experimentation.

15

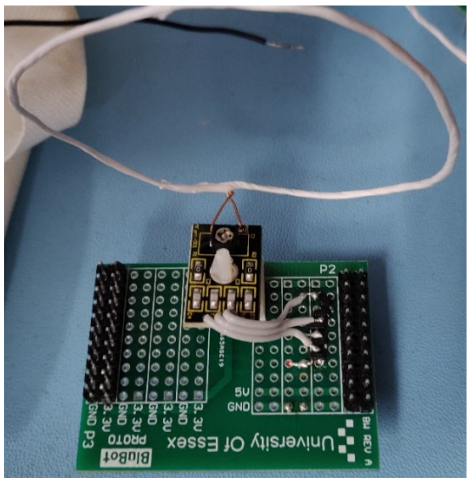


Figure 15: RF Harvesting Antenna

16

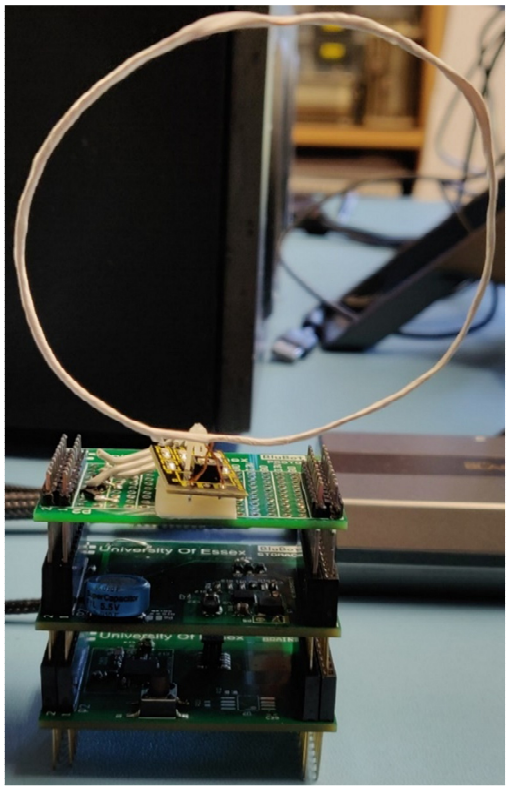


Figure 16: RF Harvesting Stack

Figure 17 and Figure 18 below both show the addition of another form of energy harvesting transducer. Two solar panels and supporting input circuitry are added to the sides of the energy reception module (Prototype board), and this will allow the idea of simultaneously harvesting energy from multiple ambient sources to be considered and evaluated when required.

17

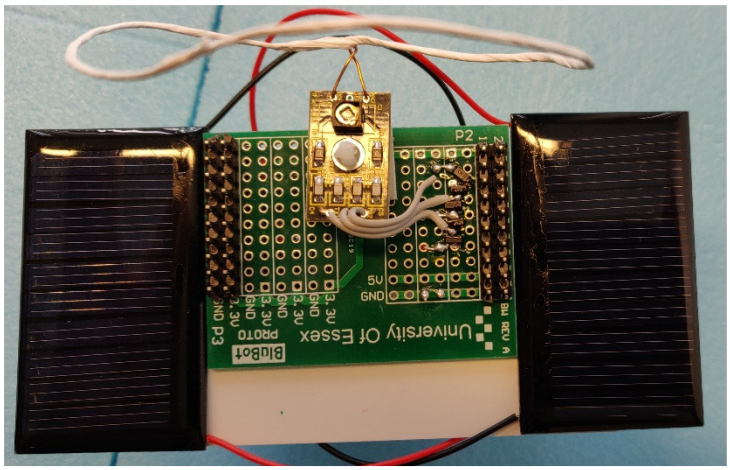


Figure 17: Solar Panels

18

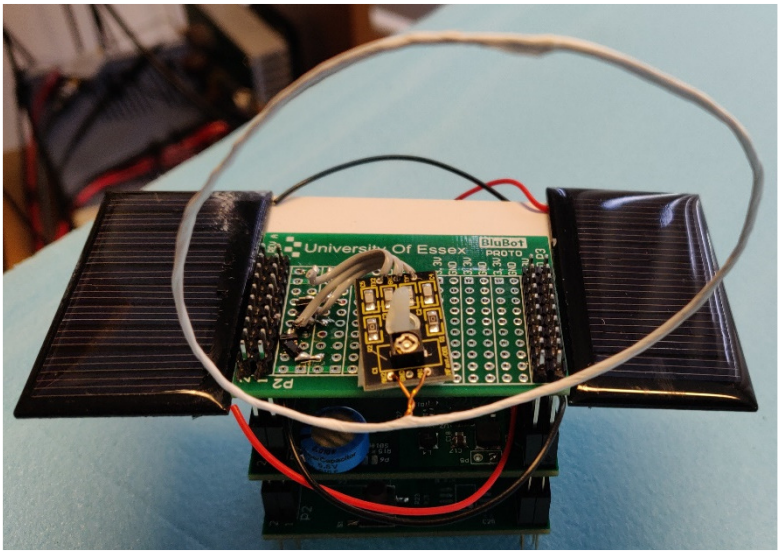


Figure 18: Solar Panel Stack

Both wireless energy and ambient light work in unison to establish the initial operation energy requirements, significantly reducing the wait time needed for the LPD energy stores to charge to a usable level, after which their power settings are adjustable, allowing replication of different environmental simulations. Figure 19, Figure 20, and Figure 21 below show the intended test environment set up with randomly positioned LDPs surrounding the wireless energy transmitting antenna.

Metrics such as charge times, discharge times, radio power consumption, distance from RF source, available workload energy, and message transmission costs are now measurable.

19

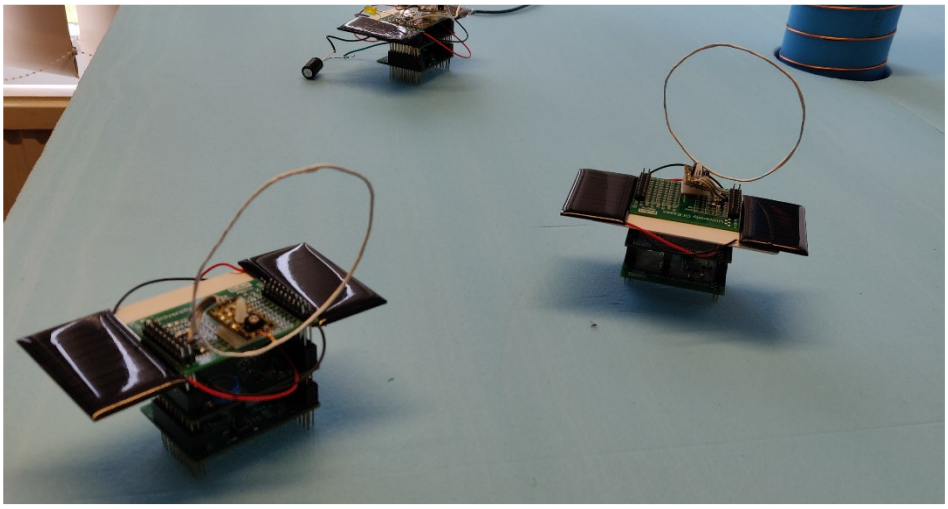


Figure 19: Device Positioning

20

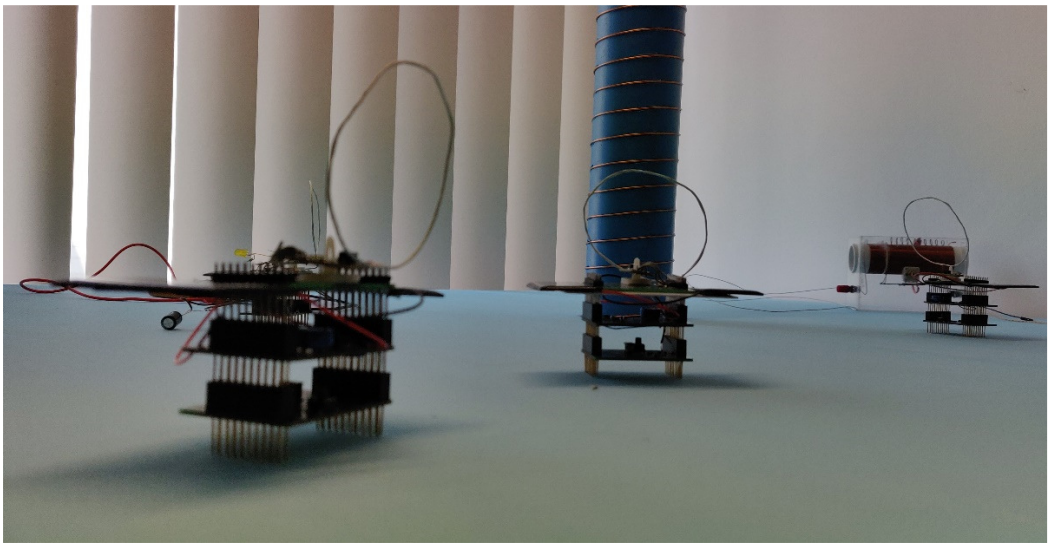


Figure 20: Common RF Power

21

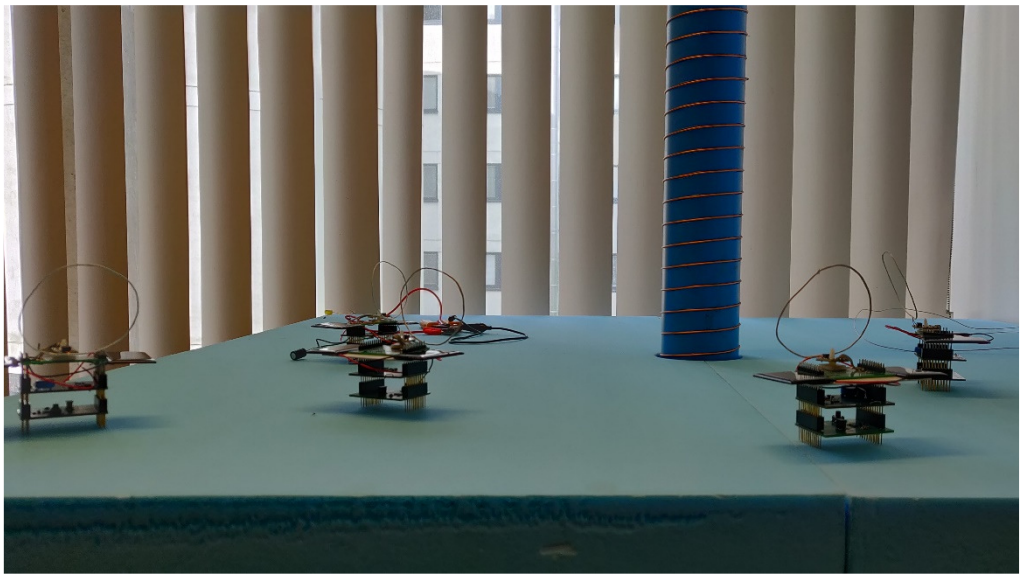


Figure 21: Device Arena

Communication and Meshing

As far as mesh networking is concerned, the supporting devices required to facilitate the network management tasks (such as Controller, Provisioner, and Friend devices) all share the similar hardware development platform shown in Figure 22, allowing quicker time to produce

hardware and thus quicker acquisition of the network's performance metrics. Figure 22 is a standard Nordic Semiconductor [106] nRF52832 evaluation board available for purchase.

22

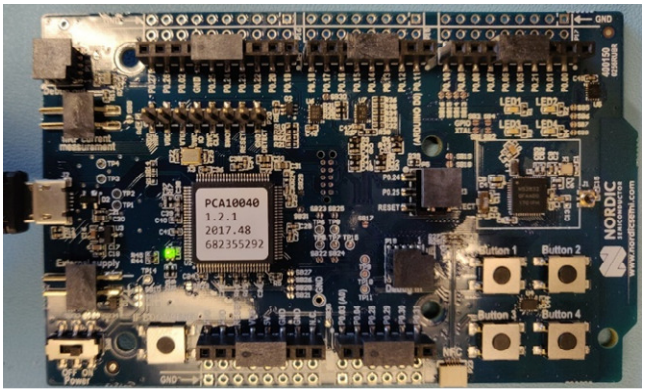


Figure 22: Provisioner, Friend, and Controller

Nordic Semiconductor has also produced and made available for purchase a very sensitive energy monitoring daughterboard (Figure 23). This development device can gets used to measure and correlate precise energy demands with firmware code execution position. Ultimately, this powerful tool will allow full mapping of the energy profile during radio transmissions and work tasks and aid in optimising code for ultra-low energy operation. This energy monitor's measurement output can be captured for long-term logging and trending along with a time-correlated measure of the energy store level, allowing accurate estimations and operation predictions based on available energy store levels over controllable periods.

23

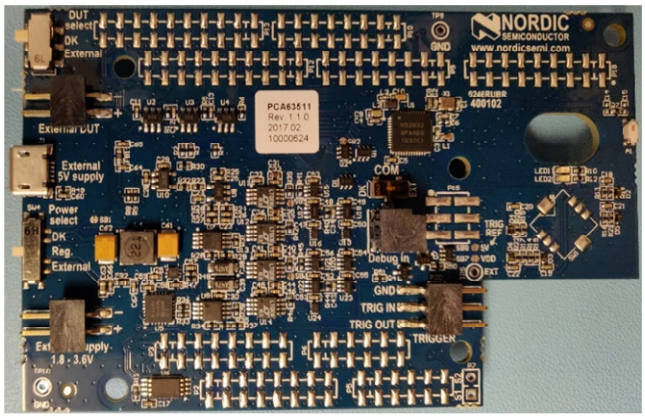


Figure 23: High Sensitivity Power Monitor

Energy Harvesting

The BQ25570 [108] energy harvester IC can be accurately configured to best suit imposed ambient energy conditions, energy store size, and expected expenditure rate. A set of hardware resister voltage-divider networks get manipulated to set and fine-tune its operation.

The BQ25570 [108] also has a highly efficient internal power supply chopper converter that gets used to provide a stable 1.9V output rail when enabled; this gets used to power the CPU, radio, and all other peripherals.

A BQ25570EVM-206 [112] evaluation board shown below in Figure 24 was purchased to understand how to set up and implement the part fully before PCB manufacture; this provided a stable working reference design to ensure the suitability of purpose.

24

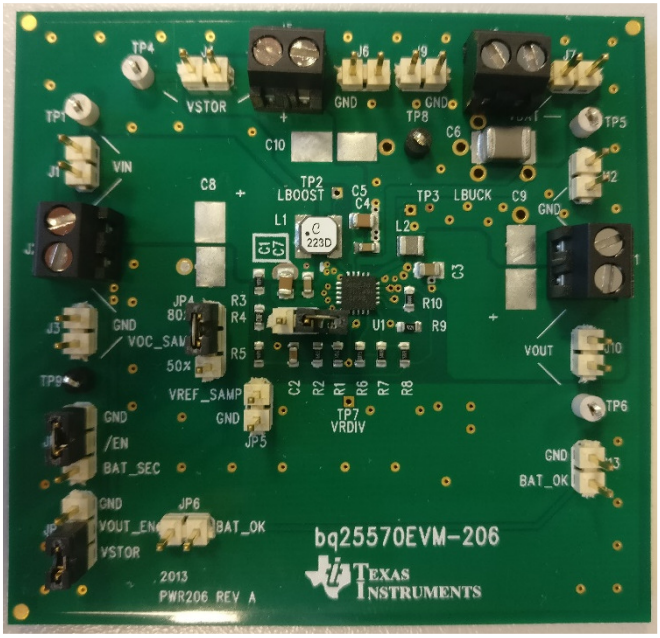


Figure 24: BQ25570EVM

Overcharge protection, MPPT (Maximum Power Point Tracking), Under-volt protection, Voltage OK, and boost voltage output are all set using a collection of high resistance divider networks as detailed in Figure 27 below.

25

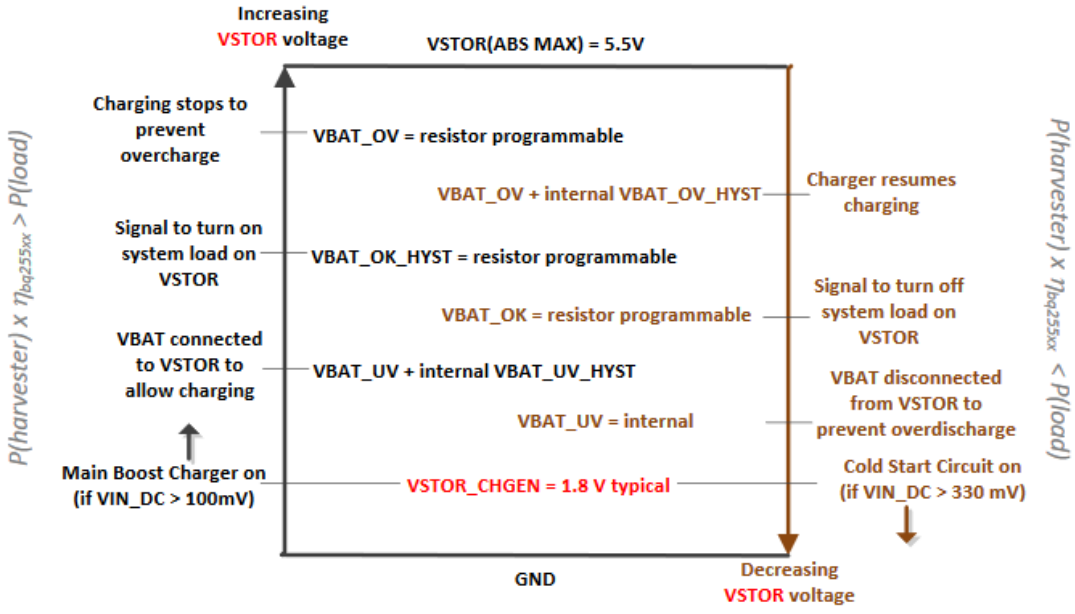


Figure 25: BQ25570 Threshold Voltages

A power level monitor input (Figure 26) is being fed into the microcontroller using one of its SAADC (Successive Approximation Analogue-to-Digital Converter) ports configured with its internal voltage reference of 0.6V enabled. The voltage level will first pass through three diodes, which together effectively drop its voltage by ≈1.5V before being passed through the SAADC gain blocks to reduce it further by ¼. The microcontroller will monitor and detect the energy level and prepare for hibernation when the level reaches ≈1.8V. At 1.7V, the BQ25570 [108] will kill the microcontroller's power and concentrate solely on harvesting more energy. The ideal goal is for the microcontroller to stay powered in a low current hibernation state allowing the BQ25570 [108] to raise the energy store level without switching off the supply.

26

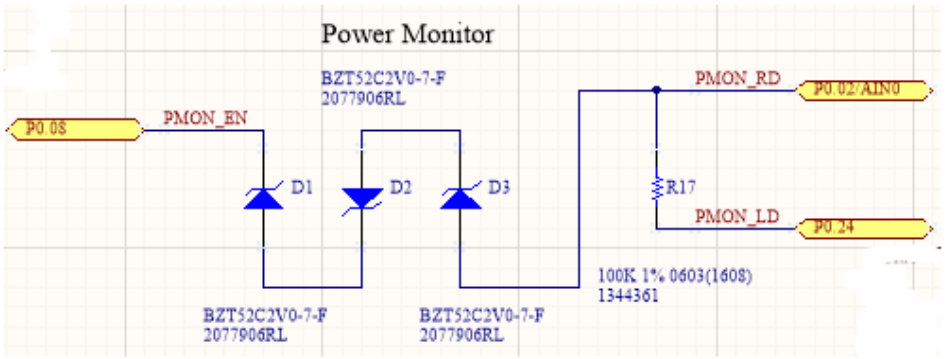


Figure 26: Power Level Monitor

The BQ25570 [108] has three setup stages (shown below in Figure 27) which have been calculated and set using a series of excel spreadsheet formulae solutions.

27

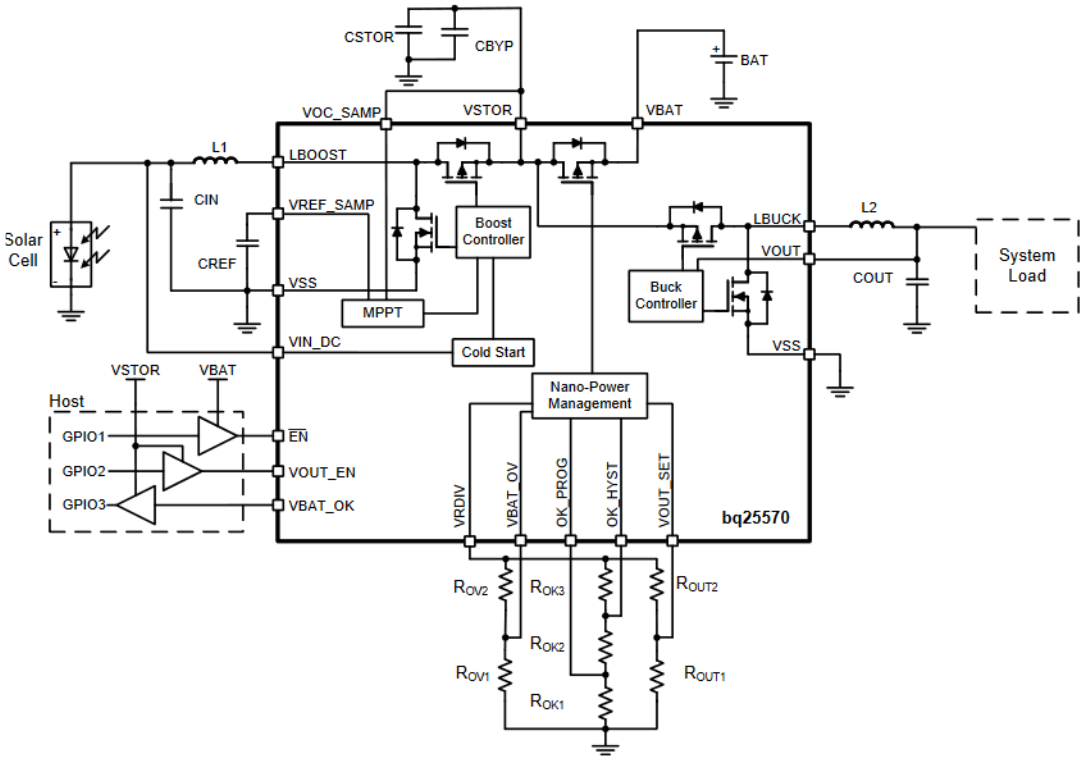


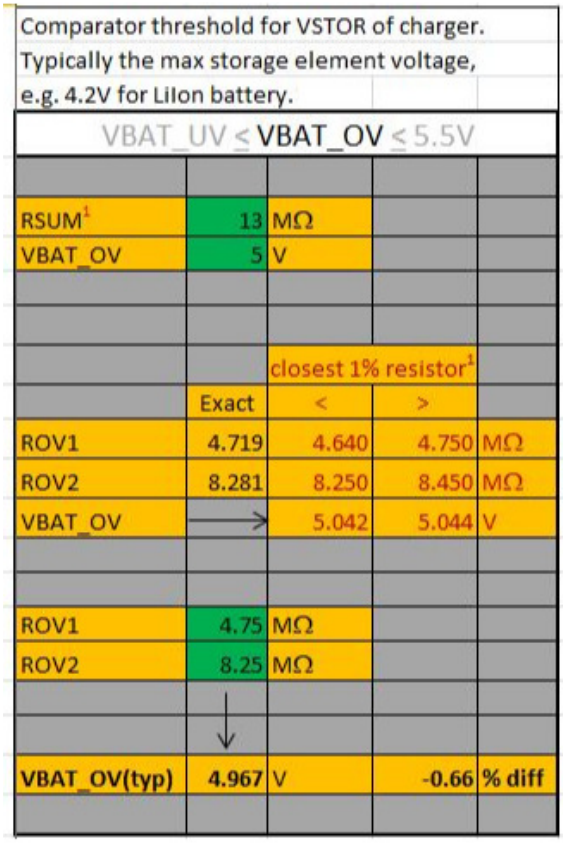
Figure 27: Energy Harvester Configuration Options

The energy store overvoltage protection is set to 5V by a high resistance resistor network with a sum not exceeding 13MΩ (R_{OV1} and R_{OV2}) using the following equation (VBIAS is the internal reference voltage, nominally 1.21V):

Equation 5

$$VBAT_{OV} = \frac{3}{2} VBIAS \left(1 + \frac{R_{OV2}}{R_{OV1}}\right)$$

28



Comparator threshold for VSTOR of charger. Typically the max storage element voltage, e.g. 4.2V for LiIon battery.				
VBAT_UV ≤ VBAT_OV ≤ 5.5V				
RSUM[1]	13	MΩ		
VBAT_OV	5	V		
		closest 1% resistor[1]		
	Exact	<	>	
ROV1	4.719	4.640	4.750	MΩ
ROV2	8.281	8.250	8.450	MΩ
VBAT_OV	→	5.042	5.044	V
ROV1	4.75	MΩ		
ROV2	8.25	MΩ		
	↓			
VBAT_OV(typ)	**4.967**	V	**-0.66**	**% diff**

Figure 28: Overvoltage Actual Calculation

A separate network of high-value resistors (R_{OK1}, R_{OK2}, and R_{OK3}) is used to set the energy store OK level at 2V and its hysteresis at 4.7V using the following formulas:

Equation 6

$$VBAT_{OK} = VBIAS\left(1 + \frac{R_{OK2}}{R_{OK1}}\right)$$

Equation 7

$$VBAT_{HYST} = VBIAS\left(1 + \frac{R_{OK2} + R_{OK3}}{R_{OK1}}\right)$$

29

Comparator threshold voltages indicating when VSTOR has risen above VBAT_OK_HYST or fallen below VBAT_OK.				
VBAT_OV ≥ VBAT_OK_HYST ≥ VBAT_UV				
RSUM[1]	13	MΩ		
VBAT_OK	2	V	> VBAT_UV	
VBAT_OK_HYST	4.7	V	> VBAT_OK	
		closest 1% resistor[1]		
	Exact	<	>	
ROK1	3.347	3.320	3.400	MΩ
ROK2	2.185	2.150	2.210	MΩ
ROK3	7.468	7.320	7.500	MΩ
VBAT_OK	→	1.994	1.997	V
VBAT_OK_HYST	→	4.661	4.666	V
ROK1	3.32	MΩ		
ROK2	2.15	MΩ		
ROK3	7.5	MΩ		
	↓			
VBAT_OK (typ)	**1.994**	**V**	**-0.32**	**% diff**
VBAT_OK_HYST (typ)	**4.727**	**V**	**0.57**	**% diff**

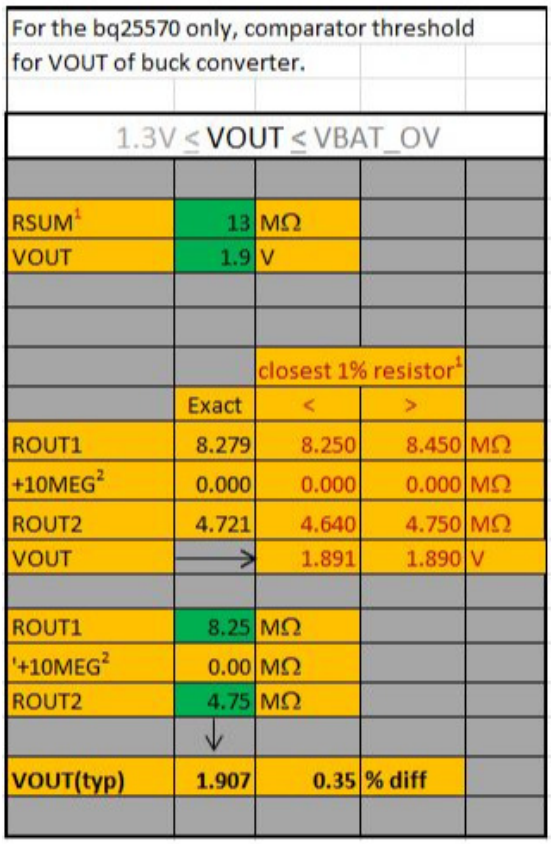
Figure 29: OK and Hyst Voltages Actual Calculations

Finally, a chopped regulated voltage output is set to 1.9V using a third resister network (R_{OUT1} and R_{OUT2})

Equation 8

$$V_{OUT} = VBIAS\left(\frac{R_{OUT2} + R_{OUT1}}{R_{OUT1}}\right)$$

30



For the bq25570 only, comparator threshold for VOUT of buck converter.				
1.3V ≤ VOUT ≤ VBAT_OV				
RSUM[1]	13	MΩ		
VOUT	1.9	V		
		closest 1% resistor[1]		
	Exact	<	>	
ROUT1	8.279	8.250	8.450	MΩ
+10MEG[2]	0.000	0.000	0.000	MΩ
ROUT2	4.721	4.640	4.750	MΩ
VOUT	→	1.891	1.890	V
ROUT1	8.25	MΩ		
'+10MEG[2]	0.00	MΩ		
ROUT2	4.75	MΩ		
	↓			
VOUT(typ)	**1.907**	**0.35**	**% diff**	

Figure 30: Regulated Chopper Voltage Actual Calculation

The energy store consists of a bank of supercapacitors rated up to 5 volts. The BQ25570 gets configured to prevent the charge voltage from rising above 5V; this will prevent damage occurring to the supercapacitors in the event of overcharging and allow the microcontroller to operate directly from the energy store without further regulation.

Energy Store

31

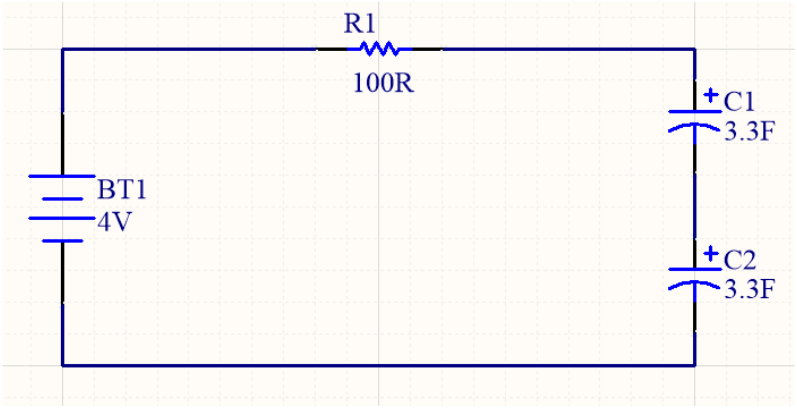


Figure 31: Energy Store Model

Figure 31 above shows the model of the energy store. The capacitors charge time is calculated and graphed using the following formula:

Equation 9

$$\tau = RC$$

$$Vout = Vs\left(1 - e^{\frac{-t}{\tau}}\right)$$

32

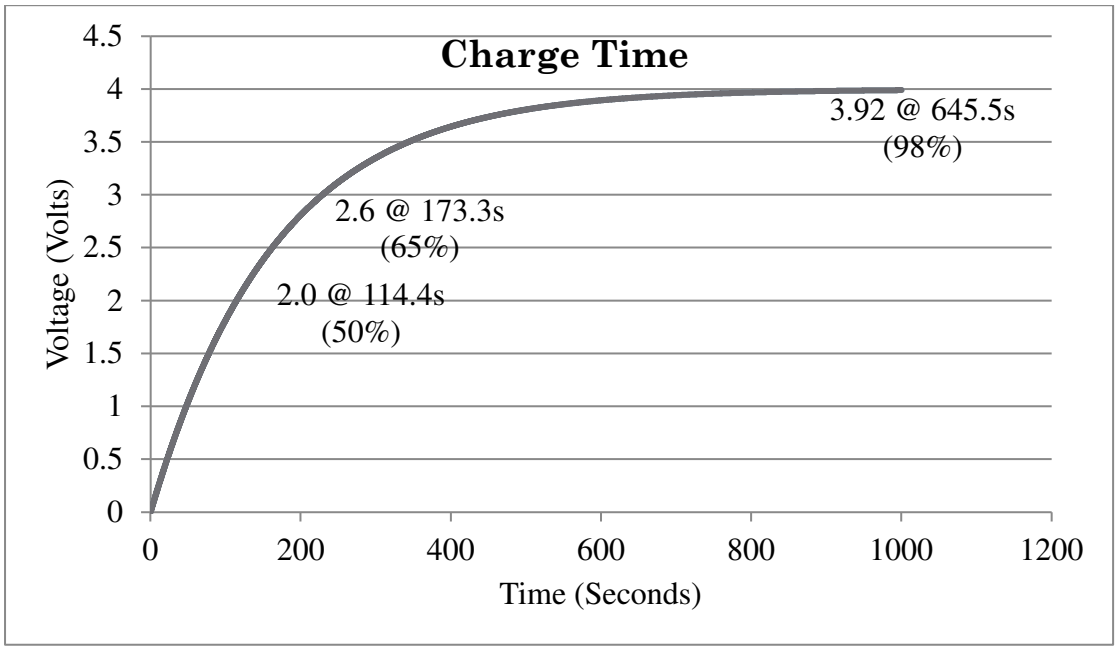


Figure 32: Energy Store Charge Simulation

Figure 32 above shows the expected charge graph for the implemented values.

Wireless Energy Beaming

Initial research focuses on beaming a square wave with a frequency of 13.56MHz via a centre fed ½ wavelength helical dipole antenna as characterised in [113].

The antenna is constructed using a 92mm PTFE pipe approximately 2 meters in length with a wall thickness of 8mm. 11.1 meters of antenna wire is then coiled evenly around the middle section of this aperture. Figure 33 below shows the assembled antenna.

33

Figure 33: Transmitting Antenna

34

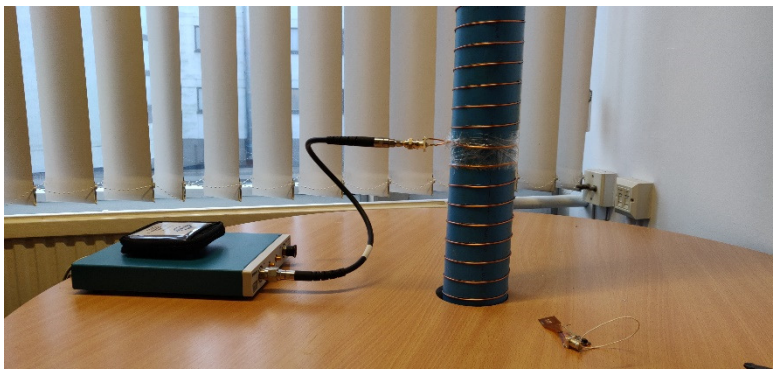


Figure 34: Transmitting Antenna Tuning

A VNA [114] (Vector Network Analyser) is used to tune the antenna's resonant frequency to 13.5MHz with an input impedance close to 50R (Figure 34). This match is achieved using a combination of wire trimming and additional matching circuitry. High power variable

capacitors and inductors (Figure 35) are connected inline for dynamically fine-tuning the match during antenna operation if required.

35

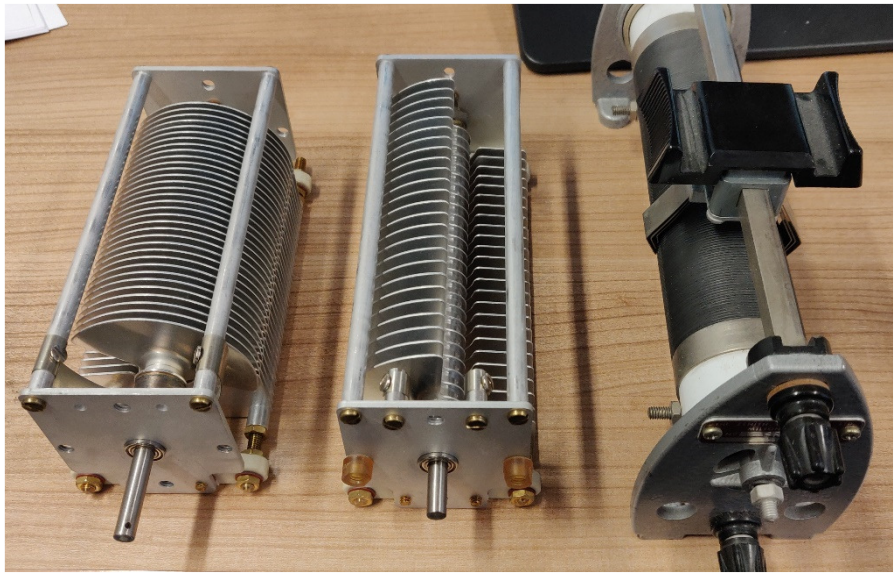


Figure 35: Matching Air Caps and Inductors

A signal generator produces a square wave at the target frequency, which exhibits edge transitions less than 8ns (Figure 36). This signal then gets fed into a pre-amp module (Figure 37). The pre-amp adds up to 1.9dB of gain depending on its adjustable supply voltage.

36

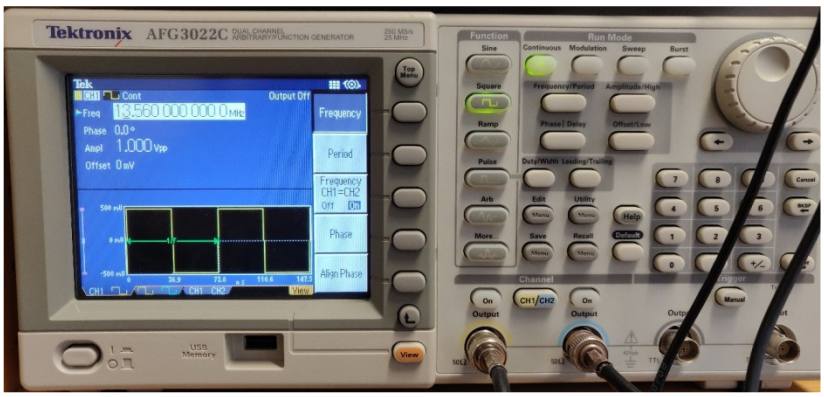


Figure 36: Transmitted Signal Source

37

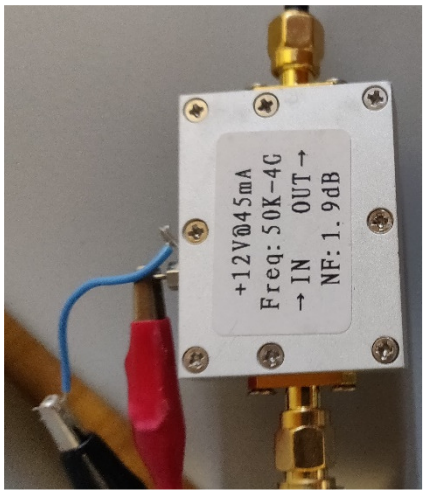


Figure 37: Preamp

The pre-amp's supply voltage is varied to dynamically control the output transmission power level of the RF field, and it is here that any required experiment adjustments get made. This signal propagation continues and gets fed into a power amp module (Figure 38) before being passed onto the antenna structure via a 50R transmission line.

38

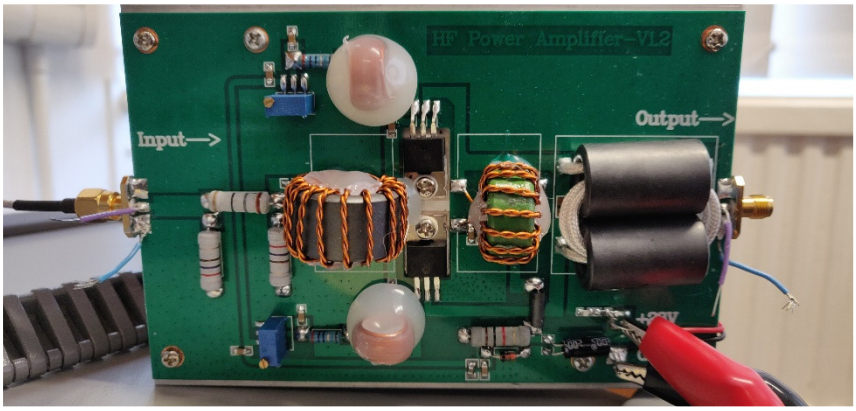


Figure 38: Power Amp

The primary flow of the signal is represented in Figure 39 below. Individual power supplies are used to supply the amplifier stages.

39

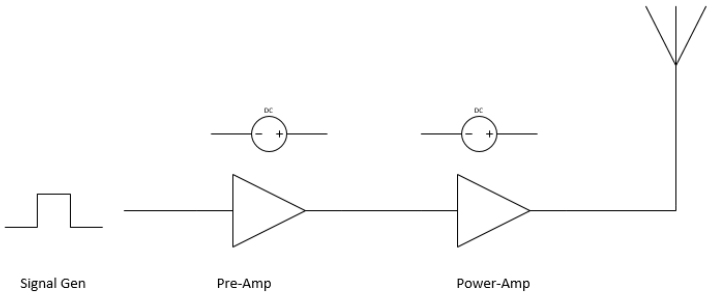


Figure 39: Transmitted Signal Flow

Safety Limitations

Due to the potential of large output wattages being transmitted via the antenna to fully explore the system's scalability, considerable time has been spent obtaining and understanding the legal allowances and precautions of transmitting within the chosen frequency band. OFCOM indicate maximum transmission powers for the band 13.553-13.567MHz to be 42 dBµA/m at 10 m [115] (Figure 41) for a device to be categorised as 'short-range,' this can be interpreted as follows:

An *H-field* of 42 dB µA/m in size is equal to 126 µA/m (in real numbers):

Equation 10

$$dB\mu A/m \ = 20\log(\mu A/m) \Rightarrow \ \mu A/m \ = 10^{\left(\frac{dB\mu A/m}{20}\right)}$$

$$= 125.89\mu A/m$$

Given that the impedance of free space is roughly 377**Ω**, we can convert this to V/m:

Equation 11

$$V/m = I \cdot R \ \Rightarrow \ 125.89 \times 10^{-6} A/m \cdot 377\Omega$$

$$= 0.04746053\ V/m$$

Now conversion into W/m^2 can be made:

Equation 12

$$W/m^2 = \left(\frac{(V/m)^2}{377}\right)$$

$$= 0.005975\ mW/m^2$$

This shows that at 10 meters, the *H-field* gives a power of roughly 6µW. This value is the power per square metre because both *E* and *H-fields* are represented as volts per metre and amps per metre.

Assuming the transmitter emits all the power in a uniformed spherical pattern, ensuring that at any distance, the total power passing through the surface of a sphere is identical, then at a 10m radius, the area of a sphere being $4\pi r^2$ is 1257 square metres. This result equates to the transmitted power being 1257 times greater than the 6 µW (per metre) mentioned above:

Equation 13

$$4\pi r^2 = 4 \times 3.14 \times 100m^2 = 1256m^2$$

$$0.005975\ mW/m^2 \times 1256m^2$$

$$= 7.5046mW$$

The actual power emitted then to achieve 42 dBµA/m at 10m is 7.5 mW; a goal for this research is to operate with transmission powers as low as this allowing the work to get classed within the ‘short-range’ category, thus potentially benefitting from receiving any kind of already heavily attenuated ultra-low-power ambient RF energy.

These calculations all assume an isotropic transmitting antenna exhibiting equal power level transmissions in all directions. An antenna with exhibits gain must have this attribute factored into the equation.

40

13.55 - 13.57 MHz	Fixed (Primary)	5.150 - The following bands: 13553-13567 kHz (centre frequency 13560 kHz), 26957-27283 kHz (centre frequency 27120 kHz), 40.66-40.70 MHz (centre frequency 40.68 MHz), 902-928 MHz in Region 2 (centre frequency 915 MHz), 2400-2500 MHz (centre frequency 2450 MHz), 5725-5875 MHz (centre frequency 5800 MHz), and 24-24.25 GHz (centre frequency 24.125 GHz) are also designated for industrial, scientific and medical (ISM) applications. Radiocommunication services operating within these bands must accept harmful interference which may be caused by these applications. ISM equipment operating in these bands is subject to the provisions of No. 15.13. UK11 - Specific details of frequency bands available for low power devices exempt from licensing are contained in Ofcom's Interface Requirement IR2030. UK6 - The use of industrial, scientific and medical (ISM) applications is allowed in this band providing they do not contravene the provisions of the Wireless Telegraphy Act 2006. Radiocommunication services must accept harmful interference from these devices. EU1 - Commission Decisions 2006/771/EC, 2008/432/EC, 2009/381/EC, 2010/368/EU, 2011/829/EU and 2013/752/EU (harmonised use of spectrum for short range devices (SRDs)) applies. UK2.1 - Responsibility for granting permissions to use frequencies in this Allocation rests with Defence. All frequency permissions are reserved exclusively for Defence use except where assignments for Civil use are agreed with Ofcom.

Figure 40: Ofcom Spectrum Allocation

41

Table 3.1: Minimum requirements for the use of Short Range Devices

	Normative Part								Informative Part
Interface / Notification number / Date	Application	Comments to application	Frequency band	Comments to frequency band	Maximum transmit power / Power spectral density / Field strength	Comments to Maximum transmit power / Power spectral density / Field strength	Channelling	Channel access and occupation rules	Reference
IR2030/1/1 2010/0168/UK Oct 2010	Non-specific short-range devices	Equipment may be used airborne	6765 - 6795 kHz		42 dBµA/m at 10 m				EN 300 330 2013/752/EU Band No. 22b
IR2030/1/2 2010/0168/UK Oct 2010	Non-specific short-range devices	Equipment may be used airborne	13.553 - 13.567 MHz		42 dBµA/m at 10 m				EN 300 330 2013/752/EU Band No.27c

Figure 41: Ofcom IR2030 Extract

Actual measurement and classification of the transmitted signals' levels are explained in the ETSI EN 300 300 standards [116]. A calibrated reference antenna is used to measure the transmission power level emitted for verification against the legislation.

For the purpose of this research, A risk assessment has been completed, which identifies and provides management of the potential risks present during periods of transmission (Figure 42).

42

hazard hazardous event consequence	Who might be harmed	Current controls	Current risk LxC=R	Additional controls
A 13.5MHz source at potentially up to 1W power is applied to a large coil, for transmitting power to small energy harvesting robots. Malfunction or temporary loss of function of an active medical implant or devices operation may occur if the implant is susceptible to EMF/RFI *(e.g. pacemakers, hearing aids)*[3]	Estates Personnel, Cleaners, Visitors.	medical implants or devices. The equipment is powered down when the researcher is not present and running the experiment. In the event that circumstances change, and the experiment is to be left running unattended - then this assessment must be reviewed. Appropriate signage will also be required. The researcher will advise any visitors or persons present that they may be at risk if they have active body worn medical implants or devices		planned, in order to clarify the likelihood of impact on medical devices. It is not expected this will significantly affect this assessment.

Figure 42: Risk Assessment

Beaming Setup

The transmission control system consists of the critical components needed to enable the realisation of wireless energy transmission. Figure 43 shows the control equipment stack, which consists of:

- High current low noise PSU for transmission duties
- Variable low ripple PSU used to control the amplitude via the pre-amp
- Pre-amp stage
- Power amp stage
- Wattmeter, Oscilloscope and high wattage dummy load for verification

43

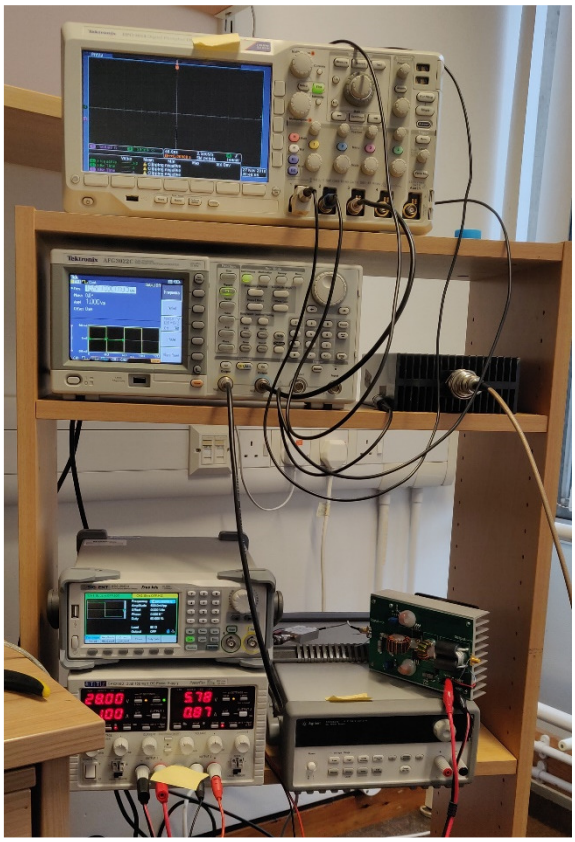


Figure 43: Transmitter Control Setup

Figure 44 below shows an oscilloscope verification trace set captured after the power-up of the energy transmitter. It shows the signal generator's square wave signal in green, passing through and getting amplified by the pre-amp signal coloured in yellow, and finally, the fully amplified signal is visible as it enters the antenna mast area coloured in blue.

44

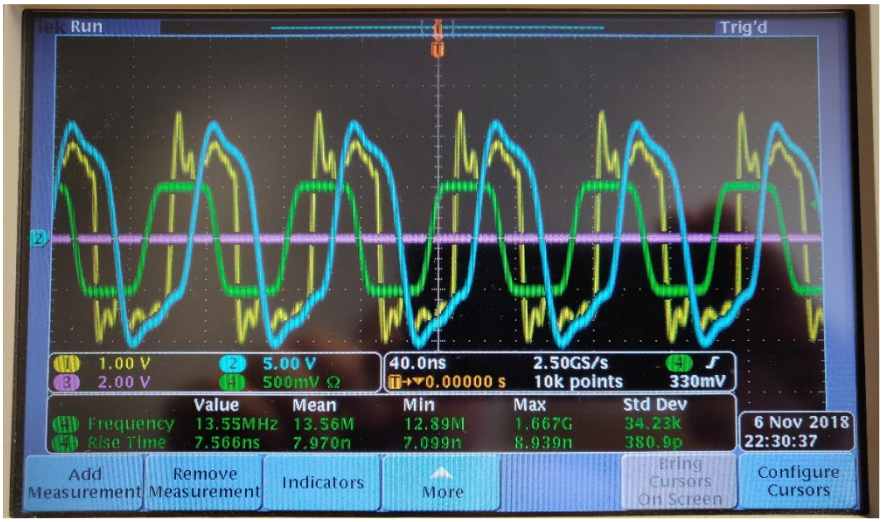


Figure 44: Transmission Amp Gain Stages

The LPD devices are all equipped with a small loop antenna connected directly to a rectenna circuit [117], similar to the design presented in Figure 45, allowing the beamed RF energy to be captured, DC rectified and injected into the energy harvesting IC for accumulation.

45

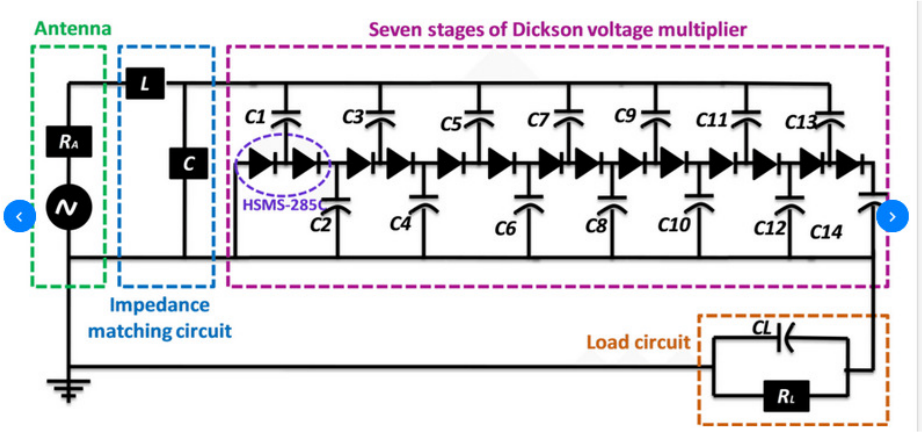


Figure 45: Rectenna Design

The multiple stages of the voltage multipliers are adjusted based on the results obtained from power level measurements done within the near-field area of the transmitting antenna and limited by the maximum voltage input characteristic of the energy harvester IC [108].

The HSMS-285 from Broadcom [118] is a Surface Mount Zero Bias Schottky Detector Diode. It is designed and optimised for use in small signal (Pin <-20 dBm) applications at frequencies below 1.5 GHz where primary (DC bias) power is not available.

This detector diode exhibits an impressively low forward voltage, as illustrated in Figure 46 below. The benefits this characteristic presents for harvesting applications are its ability to rectify tiny amounts of voltage. Based on the harvesters input specifications, this diode part enables the LPD to start 'seeing' usable microwatts for harvesting potential from around 250mV and above.

46

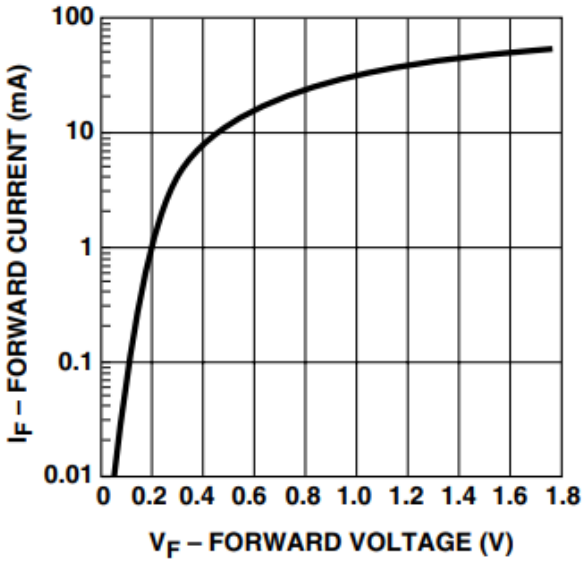


Figure 46: HSMS-285 Forward Voltage Graph

The performance of the HSMS-285 is so good that the Dickson voltage multiplier stages of the rectenna can be entirely omitted and the LPDs are still able to perform as expected.

The reception loop antenna (Figure 47) passes through a matching circuit to maintain optimum power reception delivery. Power transfer efficiencies can be measured directly using an active oscilloscope probe which presents a very low capacitive loading to the device under test.

47

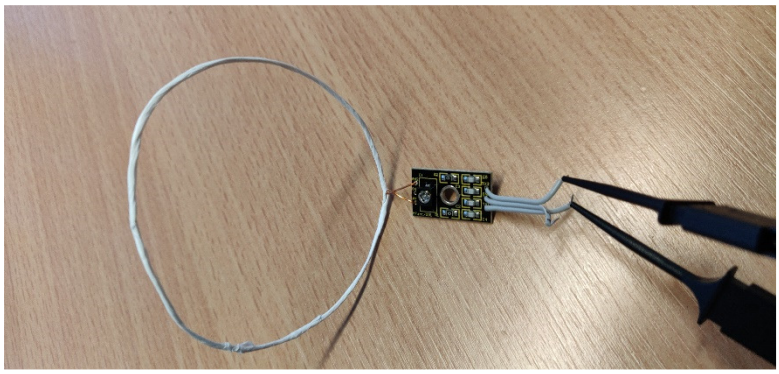


Figure 47: Reception Loop Antenna

Basic reception tests were performed by connecting the probe directly to the reception coil antenna's matching circuity interface (shown below in Figure 48).

48

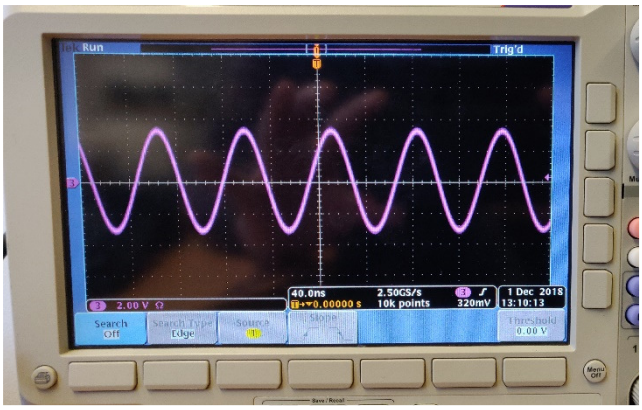


Figure 48: Reception Analysis

Various transmission and reception antennas were reviewed for efficiency and practicality, including Yagi, large coil, Heli-coiled, and dipole constructions. A software interface now exists to quantify the different antennas' effectiveness and efficiencies, allowing quick and easy initiation of essential calculations and performance metrics. Its operation uses antenna aperture comparison points, gain and directivity measurements to help choose the best suitable configuration.

Effective Antenna Aperture (linear gain)

Equation 14

$$A_e = \frac{\lambda^2}{4\pi} G = \frac{C^2}{f^2} \times \frac{G}{4\pi}$$

A_e = Effective Antenna Aperture
λ = Wavelength = $\frac{C}{f}$ (where f = frequency, C = speed of light)
G = Antenna gain (Linear Value)

The same calculation done using logarithmic gain uses the following formulae:

Equation 15

$$A_e = \frac{C^2}{f^2} \times \frac{10 \frac{G(dB)}{10}}{4\pi}$$

The software accepts as input the physical attributes of the antenna under test, along with the electrical characterisation; it then computes the results of the equations and stores the results in the form of a presentable comparison report.

Permabilities

Initial tests made after applying signals to the transmitting antenna were very surprising and somewhat unexpected. The measurement suggested a complete collapse of the signal at any desired frequency regardless of applying any form of impedance tuning. Investigation showed that after removing the antenna assembly from the mounting table, the transmission field returned and behaved as expected.

This result pointed to the table as the cause of the problem; the table consisted of an MDF substrate construction (wood, Formica, resin, glue) with metal legs. This material stack up proves to have a very high dielectric which can quickly sum to a value above 10.

To create a research environment where these kinds of influencing attributes can be measured and controlled, it was decided an entirely new table would be constructed, allowing better measures to be employed regarding the influence the platform has on the experiment's measurements.

The new table almost entirely used Styrofoam for its construction; this material's recorded dielectric is an incredible 1.03 [119], and in Figure 50 below, it can be seen compared against other materials for permittivity.

Although Styrofoam is not representative of a real-world environment, it provides a stable predictable foundation to conduct isolated research experiments.

49

Dielectric Constants of Various Materials

Material	Min.	Max.
Air	1	1
Amber	2.6	2.7
Asbestos fiber	3.1	4.8
Bakelite	5	22
Barium Titanate	100	1250
Beeswax	2.4	2.8
Cambric	4	4
Carbon Tetrachloride	2.17	2.17
Celluloid	4	4
Cellulose Acetate	2.9	4.5
Durite	4.7	5.1
Ebonite	2.7	2.7
Epoxy Resin	3.4	3.7
Ethyl Alcohol	6.5	25
Fiber	5	5
Formica	3.6	6
Glass	3.8	14.5
Glass Pyrex	4.6	5
Gutta Percha	2.4	2.6
Isolantite	6.1	6.1
Kevlar	3.5	4.5
Lucite	2.5	2.5
Mica	4	9
Micarta	3.2	5.5
Mycalex	7.3	9.3
Neoprene	4	6.7

Material	Min.	Max.
Nylon	3.4	22.4
Paper	1.5	3
Paraffin	2	3
Plexiglass	2.6	3.5
Polycarbonate	2.9	3.2
Polyethylene	2.5	2.5
Polyimide	3.4	3.5
Polystyrene	2.4	3
Porcelain	5	6.5
Quartz	5	5
Rubber	2	4
Ruby Mica	5.4	5.4
Selenium	6	6
Shellac	2.9	3.9
Silicone	3.2	4.7
Slate	7	7
Soil dry	2.4	2.9
Steatite	5.2	6.3
Styrofoam	1.03	1.03
Teflon	2.1	2.1
Titanium Dioxide	100	100
Vaseline	2.16	2.16
Vinylite	2.7	7.5
Water distilled	34	78
Waxes, Mineral	2.2	2.3
Wood dry	1.4	2.9

Figure 49: Dielectric Constants of Various Materials

Free air dielectric is 1.0, so Styrofoam is an excellent choice of material to minimise any effects on the experiments.

A power meter is introduced into the measurement chain as a secondary verification monitoring the power amplifier's output power feeding the antenna.

50

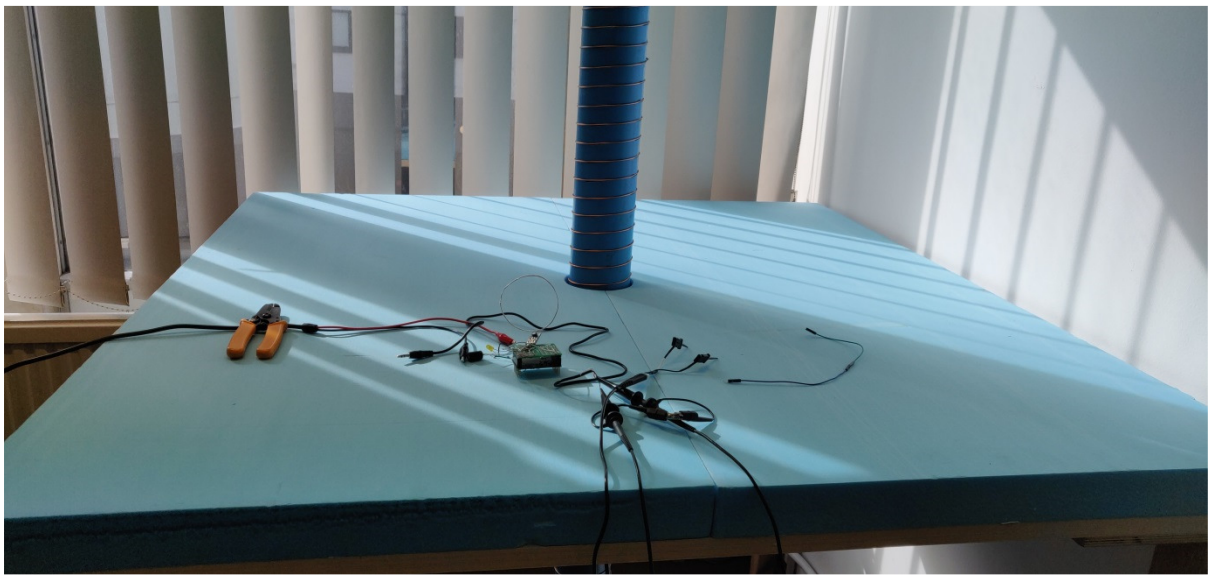


Figure 50: Styrofoam Experiment Table

Verification Experiment

A primitive energy beaming experiment was run to validate the system's ability to wirelessly deliver power which can be harnessed and used by an independent device.

The Energy Harvesting Stomach module is used along with the coil antenna and rectifier together in isolation from the other modules to harvest and store transferred energy in a standard electrolytic capacitor. This accumulated energy then gets released into an LED after the voltage level has breached the thresholds set with the energy harvesters configuration.

The energy store is a 470uF 16V standard electrolytic capacitor. The power supply chopper circuit provided by the BQ25570 [108] gets reassigned so that it takes any available voltage from the store that is above 2.1V and switches it at high speed to an output path which provides a stable 2.1V output continually as the store's charge is consumed and depleted.

This power supply output is triggered internally by the IC when the store's voltage level rises above 4.85V, discharging the capacitors contents directly into a yellow LED, as shown in Figure 52.

51

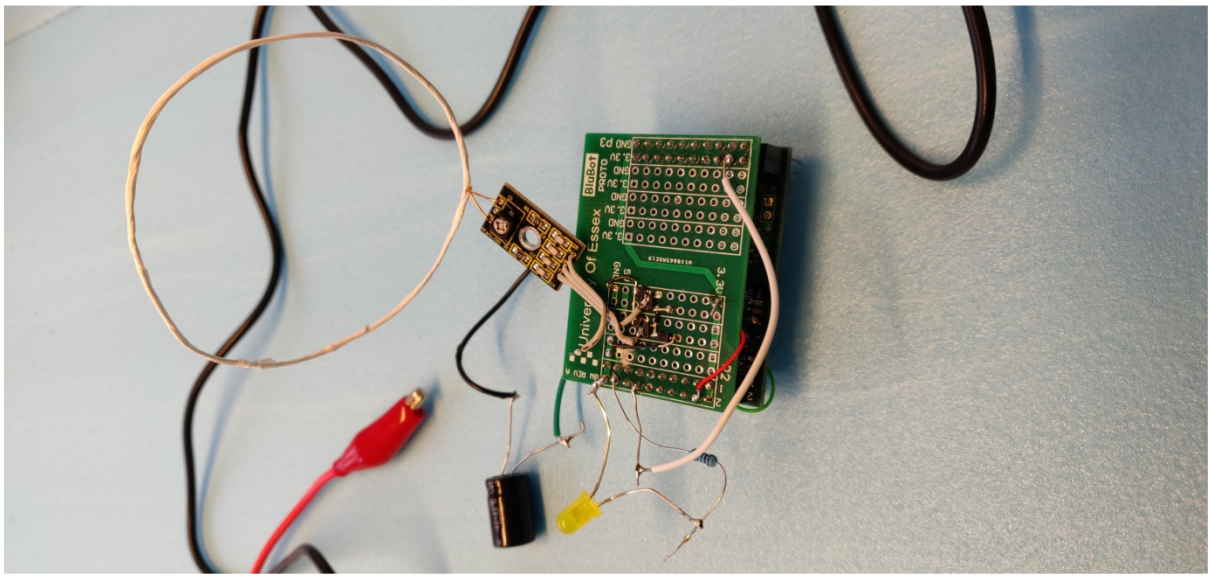


Figure 51: LPD Rectenna and Antenna

As the transmission system is powered up and begins to radiate energy from the antenna area, the LED begins to flash at a rate relative to the strength of the transmission power minus propagation, reception and harvesting losses.

This test, as shown being performed in Figure 53 below, has been adopted as a commissioning test, quickly proving the correct operation of the platform when verifications are needed. It is also frequently used to make informed antenna efficiency comparisons.

52

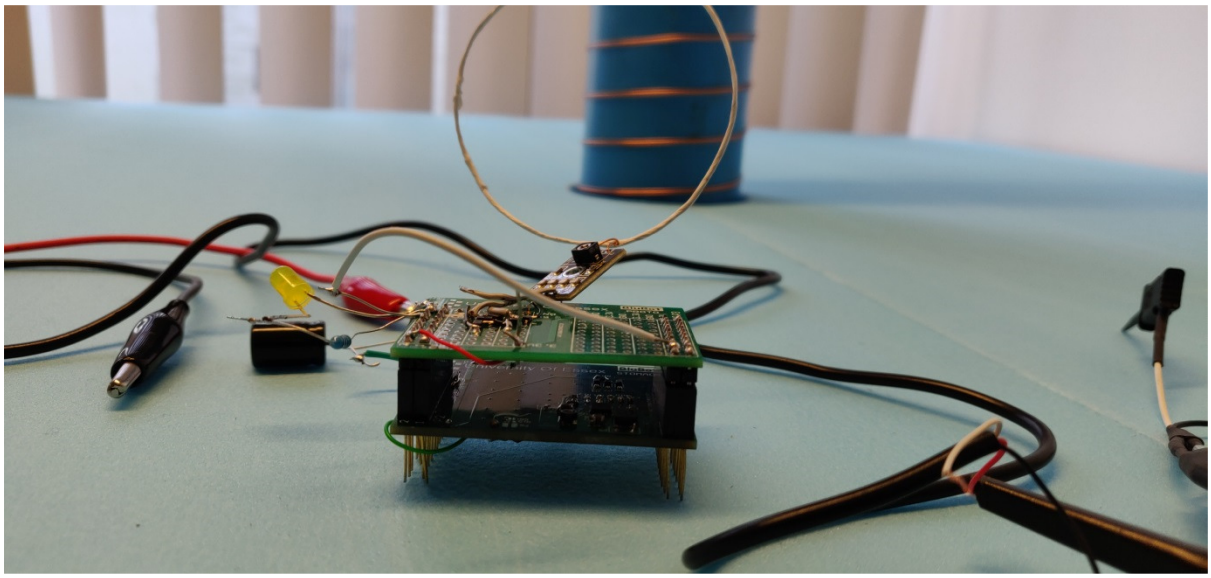


Figure 52: LPD Energy Harvesting Module

Commissioning Test Setup

An initial test cycle proves the hardware and executes the following procedure:

1. Initialise into an initial deep sleep mode when 0.1V of energy has successfully accumulated via the antenna into the store.
2. Continue to collect RF energy and place it into capacitor storage
3. When capacitor storage reaches 4.85 volts, enable chopper circuitry, which turns on the LED, consuming the stored charge
4. When capacitor storage drops to 2.35V, disable the chopper and return to step (2)

The LED flashes at a rate that represents the capacitors charge time and varies depending on the distance the device is located from the transmitting antenna.

This basic test verifies the presented platform's correct operation and aids in choosing which types of antennas best suit the LPNs transmitter and the receiver requirements.

Ensuring the LPDs all stay in the same position, the LED flash period can be recorded and measured. This period then gets used to make direct comparisons regarding the most efficient antenna design combination to continue this research.

The capacitor storage component selection is easily swapped out for supercapacitors when the need arises, and the prototype board allows easy additions of required electronics for other intended experiments. The power supply chopper circuit gets used to power the microcontroller, and the Power Good signal from the harvester gets used to release the microcontroller out of reset. A secondary reason for adopting this configuration is that the microcontroller's wake-up energy surge gets mitigated by allowing the harvesters built-in threshold engine to control the minimum operating level.

A voltage monitor circuit allows the microcontroller to tap into the primary energy store feed point directly for level measurement purposes and will use this to monitor and log the devices charge and consumption rates. This level input also enables the ability to capture metrics related to the speed and efficiency of the energy stream reception.

Multiple LPDs are built for deployment, all using the same design. The charge rate log data of every LPD gets transferred using the radio interface to a communal logging computer along with its current physical position.

This collated data then gets used to map and better understand the lensing and reaction effects exhibited when mutual coupling within the near-field propagation area of a transmitting antenna gets realised. This data further allows the creation of an algorithm that the LPDs can use to best position themselves relative to each other so the maximum benefit of the transmitted power can be captured and realised.

Firmware and Application Software

Firmware development uses the SEGGER Embedded Studio software [120]. All firmware was written using C and made full use of the SDKs (Software Development Kit) provided with the nRF52832 [106]. The SDK covered the low-level communications setup for the Bluetooth radio and provided its relevant certification assurance – this comes in the form of a pre-certified Bluetooth SoftDevice. This binary distributed library ensures that the Bluetooth design conforms to the maintainers' proper protocol legislation and, as such, ensures device interoperability.

A custom written library exposes interfaces to all the external peripherals implemented on the design.

An SWD (Serial-Wire-Debug) and JTAG (Joint Test Action Group) interface located on the PCB allows for debugging, programming and real-time process data exchange with the nRF52832 [106]. A SEGGER JLink [121] (Figure 54) debug probe provides microcontroller flash and debugging facilities. This device fully supports the Embedded Studio software and can be used via the command line for automation purposes.

53

Figure 53: J-Link Programmer

Regarding in-field maintenance and firmware management, the JLink device initially gets used to flash a bootloader and SoftDevice library onto an MCU. Both are binary parts consisting of

firmware components that will not change as the primary firmware develops and extends. Upon application of power, this bootloader transitions into an operational state that powers on the Bluetooth radio and waits for the wireless reception of a primary firmware package. Any future changes to the main firmware can be re-programmed using the same over-the-air method, thus removing the dependency on both the JLink device and any physical connection.

Data Collection

Although EEPROM provisions are available on the MCU module, initial logging data is made non-volatile by writing it to the MCUs internal flash area.

Once experimentation and execution ceases, the LPD gets connected to the JLink device for data retrieval. The JLink reads the relevant memory locations and passes the resulting binary data to the user for further processing and analysis.

Chapter Summary

For this research to successfully investigate different ways of controlling energy, the chosen research platform must support the researchers in making fair and informed comparisons of energy consumption vs task execution output. From a very early stage, it was clear that finding such a platform on the market that also considers very low-power design techniques is near, if not entirely impossible.

When considering satisfaction of the initial research requirements, this platform performs very well in terms of the low current operation, extremely low current hibernations and can accurately sample, control and monitor their own energy stores.

New microcontroller market offerings get introduced every week, which are all boasting lower consumption per MHz of speed, more cores, faster clocks, but the changes are relative when considered alongside the context of this research.

The platform does not include the lowest power latest technology; its research-driven design decisions allow operation with any device and at any speed. This platform's primary design paths are forged for their ability to work in a low RF transmission environment with low levels of light; this way, the sleep/hibernate cycles can easily be artificially influenced to suit the experiments.

Although the chosen microcontroller heavily offers support for the BLE stack, it is also compatible with ZigBee meshing and any type of custom radio interface the designer wishes to implement, and this has come in very useful for the comparisons needed in this research.

Chapter 4: Empirical Research, Phase 1: Energy Harvesting (Original Content)

With the advent of low-power microcontrollers, keeping a computing device alive for extended periods with minimal resources is possible.

Extending this feat to its logical conclusion, it should be possible to power a low-power microcontroller indefinitely if a renewable source can replace the depletion from the power source, the magnitude of which gets discussed later.

However, why would one wish to do such a thing? Self-powered sensor nodes have enjoyed a publishing history in the literature for several years now. Sensor nodes generally take infrequent measurements of data which gets logged for later use. Devices of this type can be run from button cells or small batteries for months, and deplete and replace the batteries is the generally accepted solution. The application of a renewable source implies a desire to be environmentally friendly and/or extend battery life. If a solar panel powers a battery, it will charge during the day and remain in the charged state at night. However, without doing work, the machine serves no purpose. If subjected to a load, i.e., doing work, provided the load is less than the average power generated across the 24 hours, the battery will remain in a charged state. Setting the load to match the average power does not account for the increased power availability during the day. Subsequent voltage regulation or decreased current demand from the battery cells when fully charged will result in useful power not being used to do work.

If intelligent control is employed where the work done reflects the rate at which generation is taking place, the total work done over the 24 hours will be much greater. Intelligent control is achievable using a dedicated piece of circuitry or a microcontroller.

A microcontroller's advantage is clear; the algorithm can be quickly and easily modified, and different adaptation strategies can be applied. Modern microcontrollers can be inactive or standby modes. Active mode is the regular computational operation with access to A/D

converters, GPIO, UARTs, etc. Standby or sleep mode is where the program remains in RAM, and the program counter gets stored. The clock and peripherals assume a suspended state and have negligible consumption. Wakeup techniques use interrupts based on a watchdog timer or external pin changes. Putting the microcontroller into standby/sleep results in very low power consumption and represents little or no burden to the system.

When a cell has been charged and is ready to do useful work, its energy stored over an extended charge cycle can discharge for a short period through a large load (as shown in flowchart Figure 60: Adaptive Decision Process below). This cycle means the system can drive (albeit momentarily) much greater loads than it could from the steady charge alone. Many examples of this principle surround us, e.g., charging a car battery to start a car engine.

A simple example application is a garden pond pump; usually, a solar panel with sufficient output drives the pump motor directly. However, with a small solar panel, the pump motor can never run directly from the sun. Utilising a controlled charge time followed by a shorter, higher power discharge, useful work prevails, and pond aeration commences – just not continuously.

The basis of this work is the addition of a microcontroller to a low-power charging resource to maximise the work done for a given input source. The assumption that the resource is inadequate to drive the load directly is ubiquitous in energy harvesting systems and assumed here. To prevent over-voltage, the load should always be larger than the capability of the generation source.

The microcontroller (or brain) is in itself parasitic to the system but, with proper configuration, consumes very little power (of the order of nanowatts). The advantages outweigh the costs, as seen later. Microcontrollers must operate within strict voltage bounds; typically, in this work, between 1.8 and 5v. Falling below 1.8 results in potential unspecified operations, including corruption of RAM and the device entering into an irrecoverable state. Increasing beyond 5v

will destroy the device. If for a given time-dependent resource, the voltage can be maintained between these two bounds and at the same time do useful work, then the ‘perpetual’ machine has been realised. Work can be either internal or external to the MCU.

Methodology, Phase 1

By using a very low-power resonantly coupled antenna, the supercapacitor is chargeable using wireless energy beaming. The choice of resonant coupling allows us to tightly monitor the input and output energy using only a signal generator without dc-dc converters to transform the voltage. It then becomes possible to study our machine precisely. The same reasoning applies to small solar or piezo-based renewable sources, which have also been used and produce comparable results.

The performance of the traditional transformer, based on inductive magnetic coupling between primary and secondary coils, improves when the two coils resonate at the same frequency [1]. This relationship forms the basis of most wireless charging systems allowing the transfer of energy across free space at a distance. The amount of power transfer is dependent on the mutual inductance between the two coils, which is inversely proportional to the distance between them [2]. The secondary needs to receive enough energy to enter resonance; otherwise, work becomes impossible.

Traditionally, near and far-field low-power resonant systems have had no significant function. However, new systems get introduced frequently [3], which allow reception, rectification, and then storage of the DC in capacitors.

Using wireless power transfer, it is possible to receive power wirelessly and do work without the conventional storage medium of a battery [4]. An example is a battery-free receiver designed by the Powercast Group [5]. This product is the P2110B 915 MHz RF Power Harvester far-field Receiver [6] and designed for sensor networks and active RFID. Distances of 10m are claimed but at the expense of highly directional and powerful transmitters. The process involved the accumulation of energy into a supercapacitor.

This system is distinct from the Powercast offering. Both systems accumulate energy onto a supercapacitor, but we (a) try to keep the microcontroller alive and (b) deliberately design it to run larger instantaneous loads than the source can supply.

Accumulating power and then discharging is a bit like saving up to buy something one cannot immediately afford. This process is achieved by accumulating for a specific length of time, then spending it when a threshold has breached. The analogy continues because the cost of living needs considering to prevent the savings from going below zero.

To accumulate a desired quantity of energy, it is necessary to observe/measure a resource. The act of performing a measurement is in itself parasitic, and the more frequently one measures a resource, the more energy gets consumed without doing useful work. An obvious approach is to perform measurements at regular intervals, but if the system is understood, observations need only be sparse, achieving greater efficiency for a given setup.

The subsequent sections and chapters show and realise intelligent algorithms for power accumulation that optimally adapts to the available source and required load. The power requirements of the microcontroller and the capacitance leakage manifest as an addition to the load.

Sleep Strategies

To illustrate intelligent control of wireless power transfer, an algorithm is created and examined in a testbed. The idea is to periodically run a load from a source with less energy than the continuous load requirement. This extrapolation requires scheduling the load's work time into short bursts or shifts in the duty cycle according to the capacitor's voltage. The steps to achieve this are: turn the load off; wait for energy accumulation on the capacitor; turn the load on for a specific time, deplete the capacitor to a known level; turn off and repeat from the beginning.

Figure 55 illustrates the block diagram of the system. The received power is rectified and stored in the reservoir capacitor. The capacitor represents the DC source of the system. The microcontroller decides to turn the load on or off according to the capacitor's charge state.

54

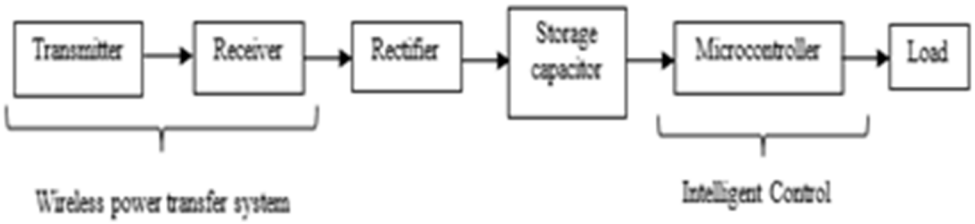


Figure 54: Mutual Induction Process

Figure 56 presents the simplified schematic diagram of the system.

55

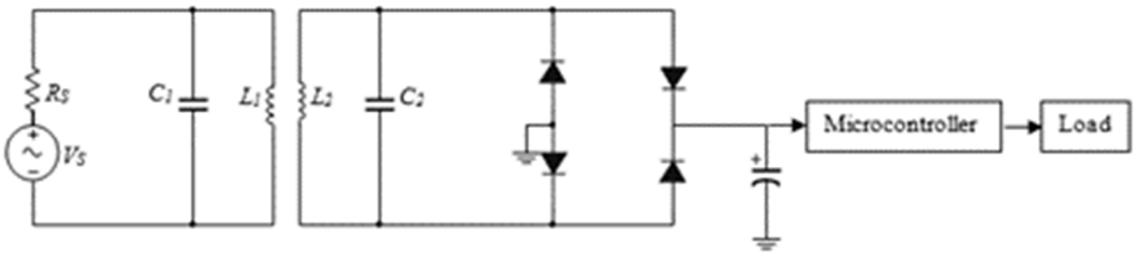


Figure 55: Schematic diagram of the system

The testbed uses a loop antenna, used for convenience due to its impedance characteristics being similar to those used for far-field transceivers. The loop antenna is a radio device that consists of one or more loops. In our design, two loops are wound, as shown in Figure 57. The first one is the coaxial oscillating loop or coupling loop, which has a 50Ω impedance to match the source, a signal generator. The second one is the main loop or the free-running loop (in resonance), representing the transmitter for the overall design. A tuning capacitor allows for

fine adjustments. The receiving coil is a ten-turn PCB-based coil with 1/100th the area of the main loop.

56

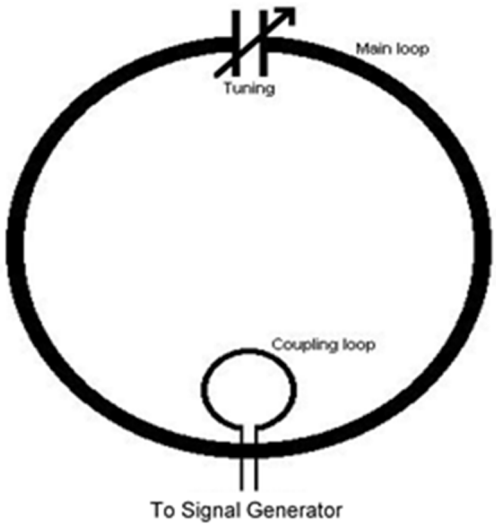


Figure 56: Loop Antenna

A 1v pk-pk sine wave drives the coupling loop at around 4MHz giving a transmit power of approx. 2.5mW assuming perfect matching. Faraday shielding is used in this band to prevent unlicensed radiation.

Hardware Domain

The MCU with the lowest power consumption is a matter of some dispute and depends on the application, duty cycle, sleep cycle etc.

The current state-of-the-art ultra-low-power MCUs quote current requirements as low as 30µA per MHz in active mode and 100nA in sleep mode with operational voltages as low as 1.8v.

The power consumption of MCUs is dependent on the Microcontroller's clock speed, the operational voltage, and the types of technology used but will be in the order of milliamps when in active mode.

The ATtiny85 [31] is an example of a cheap ultra-low-power microcontroller that provides development support for the Arduino platform. During active mode, it draws 10-12mA running at 8MHz. Assuming a CR2032, 3.0v battery has a 500mAh capacity that gives (500mAh/12mA) over 40 hours of run time. This return is pretty good, but this can easily extend by several orders of magnitude with the help of sleep cycles strategies.

Onboard any MCU, there exists several peripherals that consume power. For example, the ATtiny85 has an analogue to digital converter (ADC). When not in use, turning it off can reduce the power budget significantly. Further savings are possible by sleeping the MCU as discussed above and waking it up via a counter within the watchdog interrupt.

The recently introduced MSP430 series from Texas Instruments [60], widely used in power harvesting applications, can be inactive mode at 230μA, 1MHz, 2.2v, or 0.5μA standby, 2.2v. It can also be programmed using Arduino-based code.

This 16-bit microcontroller running from the same CR2032 cell as before could run for 0.5/300u = 1667 hours or nearly 70 days in active mode. Even the most basic sleep strategies could extend life to years.

The above observations give intuitive insight into the large amounts of time available to replenish the source and the small amounts of power required to do so.

For sensor networks and energy harvesting applications, much of the work required is periodic, allowing the device to power down for significant periods. Power down or sleep mode in the MCU allows considerable power savings with current consumptions potentially equivalent to less than the open circuit natural leakage of the batteries used.

For example, it is required to run the ATTiny85, configured as a sensor controller, for one year from a CR2032 cell (which measures a mere 20mm diameter with a 3.2mm thickness). We can calculate its maximum current deliverance to survive the year , that is:

Equation 16

$$\frac{0.5\text{A}}{(24\ hrs \times 365\ days)}$$

An average of ~57µA continuous current/hour. 57µA is not enough to run an ATtiny85 in active mode (approximately 10mA), so a combination of sleep/wake cycles is required to match the available power budget.

By being in active mode for a short period followed by a timed inactive or sleep period, it becomes possible to match the MCU to the resources available. If T_a = the active period, T_s = the sleep period and I_1 = the current required while active, and I_2 = the current required while sleeping, then any average current budget can be expressed as:

Equation 17

$$I_{average} = \frac{T_a I_a + T_s I_s}{T_a + T_s}$$

Equation 18

$$T_a + T_s = \frac{T_a I_a + T_s I_s}{I_{average}}$$

Equation 19

$$T_a - \frac{T_a I_a}{I_{average}} = -T_s + \frac{T_s I_s}{I_{average}}$$

Equation 20

$$\frac{T_s}{T_a} = \frac{I_{average} - I_a}{I_s - I_{average}}$$

With I_a, I_s and $I_{average}$ known, e.g. if I_a = 10mA and I_s = 25µA and $I_{average}$ = 50µA

Equation 21

$$\frac{50 \times 10^{-6} - 10 \times 10^{-3}}{25 \times 10^{-6} - 50 \times 10^{-6}} = 398 = 398{:}\,1$$

Giving a ratio between Ts and Ta of 398:1 for the ATtiny85. To run the MSP430 for a year by the same reasoning would take a 3:1 ratio.

As $I_{average} \rightarrow 0$, the ratio between T_s and T_a is approximately equal to

Equation 22

$$\frac{I_a}{I_s}$$

Thus, the sleep/awake ratio can be found for any known current budget, allowing it to accompany the prescribed energy budget efficiently.

Applications of this technology are typically short duration and periodic such as a sensor node sampling temperature etc. However, the combination of a battery-free active (alive) microcontroller driven from a minimal energy source offers a machine the potential to do useful work into the foreseeable future.

The ubiquitous battery gets replaced with a supercapacitor which has several advantages. It can re-charge from zero; compact, able to charge/discharge very rapidly, has duty cycles in excess of 500,000 and has a specific power of 10kW per kg [9].

Disadvantages include low voltage ratings; some leakage and supercapacitors are not an excellent fit for AC circuits. This application is low voltage (5v), and capacitors are available in integer multiples of 2.7v. Leakage is important for low power applications, and a good rule of thumb is 1µA/Farad [10], giving a minimum rate at which the capacitor can receive a charge. The application here is DC which is primarily where supercapacitors shine. A capacitor stores energy according to:

Equation 23

$$E = \frac{1}{2}CV^2$$

If a 5.4v 2 Farad supercapacitor gets discharged from 5v to 4v, then 9 Joules of energy is released. Division by time gives the power dissipated in Watts.

If a 50mA motor is to be driven at the average voltage of 4.5v; P = I·V and 0.05x4.5 = 0.225W, the motor can drive for 40 seconds given the capacitor conditions above (ignoring the incoming charge).

On the charging cycle, if we assume 1µA of capacitance leakage current, a current source of 1mA, and an average current draw of 10µA from the microcontroller, then according to:

Equation 24

$$i = C\frac{dV}{dt}$$

It will take 2/0.000989 = 2022 seconds or 34 minutes to charge back to the 5v. The above makes no allowance for capacitance leakage or incoming current.

This specific example can be generalised as follows:

If :

C = rating of the capacitor in Farads

$\mathbf{I_{Load}}$ = current drawn by the load

$\mathbf{I_{Leak}}$ = the capacitance leakage current

$\mathbf{V_{before}}$ = voltage on capacitor before charge/discharge

$\mathbf{V_{after}}$ = voltage on capacitor after charge/discharge

$\mathbf{I_{micro}}$ = average current draw of microcontroller

$\mathbf{I_{charge}}$ = charge current from the source

The variables of interest are the charge times and the discharge times T_{charge} and $T_{discharge}$

If the average voltage seen by the motor is:

Equation 25

$$(V_{before} - V_{after})/2$$

Then:

Equation 26

$$T_{Charge} = \frac{C(V_{before} - V_{after})}{(I_{charge} - I_{Leak} - I_{micro})}$$

Equation 27

$$T_{discharge} = \frac{C(V_{before} - V_{after})}{(I_{load} + I_{micro} + I_{leak} - I_{charge})}$$

Clearly from the above, if $I_{micro} + I_{leak} \geq I_{charge,}$ then no useful work is possible, the capacitor will begin to discharge, and I_{micro} must reduce in some way, or the device will *die*.

Software Domain

To manage energy over time, it is necessary to regularly read the capacitor voltage until a target voltage threshold gets breached. Then a decision can be made to run the load. This process also exists using discrete electronics or a low-power microcontroller. The new generation of ultra-low-power controllers such as the Atmel ATTiny 85 and the TI MSP430 have comparable or better power consumption than bespoke electronics with the added advantage of onboard computation.

A microcontroller in the awake state can consume 2-3 orders of magnitude more power than when asleep [7]. Our system moves beyond merely 'sitting' and waiting for the capacitor to charge before doing anything. Two readings are taken of the capacitor voltage at separate times and used to calculate the system's time constant. From this, the required sleep time gets calculated, so the microcontroller awakes at the point where the capacitor has reached the desired charge (recall sleep is much more energy-efficient than awake). Two readings are used in the first cycle, followed by a single reading in subsequent cycles to update the estimated time constant. The outline of the algorithm is as follows:

1. Read V_{cc}
2. Sleep for a short time
3. Read V_{cc}
4. Calculate τ_{ch} (Time constant for charging)
5. Find (t_{sleep}) the required time for sleeping until $target_1$
6. Sleep for t_{sleep}

7. Wake up, read V_{cc} and turn the load on
8. Delay for a short time
9. Read V_{cc}
10. Calculate τ_d (Time constant for discharging)
11. Find (t_{on}) the required time for turning the load on
12. Delay for t_{on}
13. Repeat from step 5

Charging Time Constant

Step 4 in the algorithm needs some clarification, where τ_{ch} is the time constant of charging the system's capacitor. It is known as:

Equation 28

$$\tau_{ch} = R_{ch} \times C$$

Where R_{ch} is the equivalent impedance in series with capacitor C [8]. Figure 58 shows the mechanism of charging and discharging the capacitor.

57

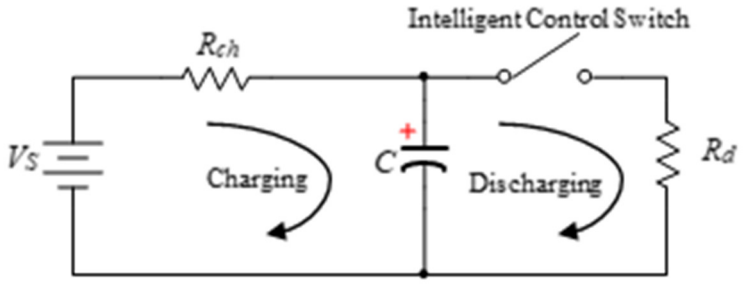


Figure 57: Circuit of charging and discharging a capacitor

In this system, R_{ch} is not a constant impedance. It is the equivalent impedance of the combined circuit before the capacitor and is challenging to calculate as it changes with coupling and load.

Therefore, the time constant needs to be calculated regularly and updated. It is possible to calculate the time constant from two readings of the capacitor's voltage at any given time while it is charging. The time constant for charging can be calculated in this way, starting with the basic capacitor charging formula, as shown below:

Equation 29

$$V_0 = V_{in}\left[1 - e^{t_0/\tau_{ch}}\right]$$

Equation 30

$$V_1 = V_{in}\left[1 - e^{t_1/\tau_{ch}}\right]$$

Where V_1 and V_0 are the voltage readings at t_1 and t_0, respectively.

58

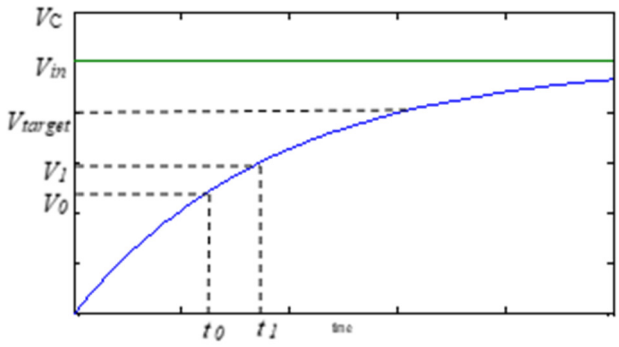


Figure 58: Capacitor Charge Simulation

Where V_{in} is the maximum voltage, as shown in Figure 59. These two equations get rewritten:

Equation 31

$$t_0 = -\tau_{ch}\ ln\left[1 - \frac{V_0}{V_{in}}\right]$$

Equation 32

$$t_1 = -\tau_{ch}\ ln\left[1 - \frac{V_1}{V_{in}}\right]$$

Taking the difference between the two times leads to:

Equation 33

$$t_1 - t_0 = \tau_{ch} ln \left[1 - \frac{V_0}{V_{in}}\right] - \tau_{ch} ln \left[1 - \frac{V_1}{V_{in}}\right]$$

Equation 34

$$\tau_{ch} = \frac{t_1 - t_0}{ln\left[1 - \frac{V_0}{V_{in}}\right] - ln\left[1 - \frac{V_1}{V_{in}}\right]} = \frac{t_1 - t_0}{ln\left[\frac{V_{in} - V_0}{V_{in-V_1}}\right]}$$

This equation gives an estimate for the time constant regardless of where we are on the charging curve.

From the calculated charging time constant, the microcontroller can find the required sleep as in step 5 of the algorithm by:

Equation 35

$$t_{sleep} = \tau_{ch}\, ln\left[\frac{V_{in}}{V_{in} - V_{target1}}\right]$$

Discharging Time Constant

Similar to the above, the time constant for discharging gets calculated as required in step 10. The time constant in the discharging process τ_d depends on the parallel load resistance R_d [8]. In this system, R_d represents the equivalent resistance of the circuit beyond the capacitor and gets defined as:

Equation 36

$$\tau_{\mathrm{d}} = R_{\mathrm{d}} \times C$$

As before, taking two voltage readings (V_2 and V_3 from Figure 59 and Figure 60) is enough to find the discharging time constant. The first reading measured at the highest charged point t_2. The second reading took a short while afterwards, t_3, as shown in Figure 60. Starting with the capacitor discharging formula, τ_d is derived as shown below:

Equation 37

$$V_2 = V_{start}\ e^{-t_2/\tau_d}$$

Equation 38

$$V_3 = V_{start}\, e^{-t_3/\tau_d}$$

These two equations are re-written:

Equation 39

$$t_2 = -\tau_d\, ln\frac{V_2}{V_{start}}$$

Equation 40

$$t_2 = -\tau_d\, ln\frac{V_2}{V_{start}}$$

Taking the difference between the two times leads to:

Equation 41

$$t_3 - t_2 = -\tau_d\, ln\frac{V_3}{V_{start}} + \tau_d\, ln\frac{V_2}{V_{start}}$$

Equation 42

$$\tau_d = -\frac{t_3 - t_2}{ln\left[\frac{V_2}{V_{start}}\right] - ln\left[\frac{V_3}{V_{start}}\right]} = -\frac{t_3 - t_2}{ln\left[\frac{V_2}{V_3}\right]}$$

59

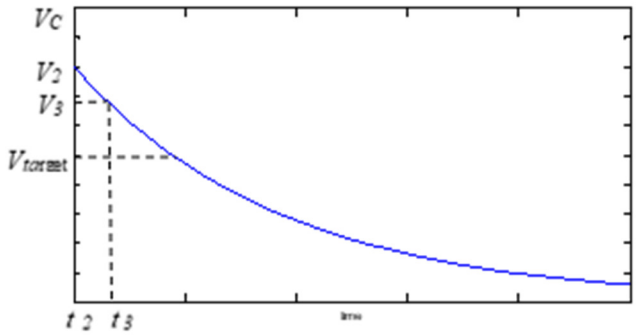


Figure 59: Capacitor Discharge Simulation

Regarding step 11 in the algorithm, the calculated discharge time constant gets used to estimate the required time to spend the accumulated energy as useful work:

Equation 43

$$t_{on} = \tau_d ln\left[\frac{V_{start}}{V_{target2}}\right]$$

The flowchart in Figure 61 provides specific detail of the implementation tested here. The highlighted shapes represent the remaining parts in force in subsequent cycles. This illustration shows that the time constants are calculated only in the first cycle and then updated as new conditions occur. After the first cycle, just one reading is required during charging and one reading during discharging.

60

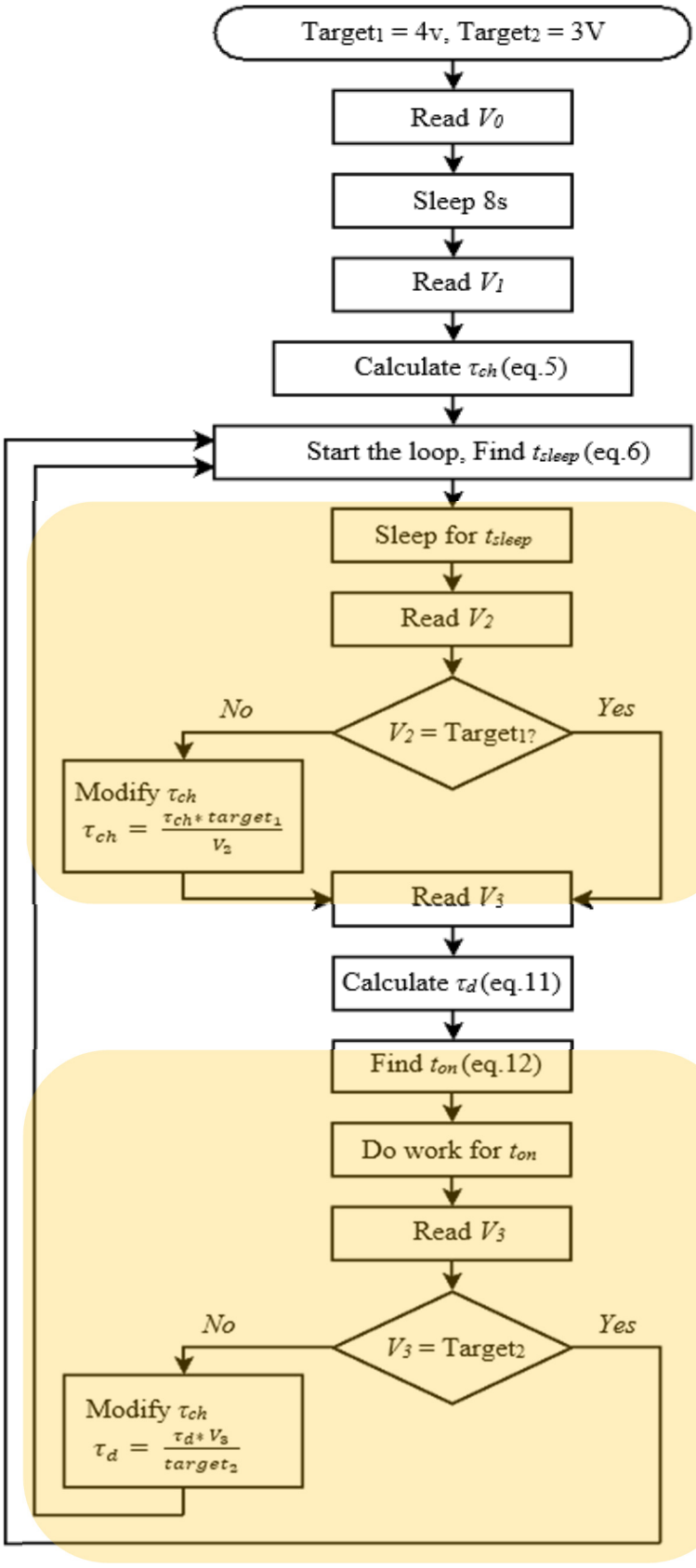


Figure 60: Adaptive Decision Process

Intelligence and Autonomy

If a renewable resource gets managed correctly, it is possible to charge and maintain an accumulator such as a supercapacitor. The addition of a parasitic microcontroller places a load on the accumulated resource, but this can be mitigated by putting the MCU into standby/sleep mode.

If the MCU never sees a voltage level below 1.8v, wakes periodically, and performs a useful function, then the machine that lives forever is born. For this to be possible, the outgoing or consumption of power must be less than the incoming or rate of generation, or the voltage level on the accumulator will progressively decrease. Note capacitive leakage is not being considered here.

Microcontrollers with deep-sleep modes can enter into very low power modes and consequently tailor their behaviour to the resources available to them. In this way, it is possible to run a microcontroller periodically where the input is less than the operational consumption. With the addition of the intelligent controller and awareness of generation rates, it becomes possible to budget according to the current and future perceived resource; the machine should learn from past events.

The overriding objective of a machine of this type is to stay alive, i.e., maintain the voltage above v_min and to do as much work as possible in a given time. Work can fall into various categories, including calculations, driving external loads, and communication with the outside world.

Available energy can be at two extremes: (i) excess or *feast* and (ii) insufficient or *famine*. In (i), the accumulator has reached the desired charge, and the influx rate is fast. In (ii), the accumulator is below its maximum, and the influx is minimal or non-existent.

If energy is bountiful as in (i), long duty cycles of work can be assigned and sleep times reduced. If energy is scarce as in (ii), little or no useful work is possible, and sleep times must be very long. If considering Micro Solar, (i) would be the case during the day and (ii) the case during the night. The MCU would need to ensure sufficient energy available in the accumulator to last from dusk until dawn.

Startup Inertia

An obvious question is why keep the MCU alive at all; if the voltage dips below 1.8v, just let it until sufficient energy becomes available again. While this approach has some merit, MCUs approach their operating voltage from below their minimum specification, causing them to take significant amounts of current and consume more energy than being generated without becoming operational. Low voltage lockout circuits exist, which can prevent this, but they often use more in standby than the MCU while operational! In the event of under-voltage, the current program state becomes lost (in RAM), meaning that any long-term data needs writing to ROM. If no under-voltage protection is employed, there is a genuine possibility of ROM and program memory corruption. Given the tiny amounts of energy involved, the option of keeping the MCU alive is a viable one. For example, the MSP430 series from Texas can sleep for 12 hours for approximately 1uWh

Discussion and Results

Continuously measuring the stored energy on the capacitor can be achieved by regularly reading (e.g., every few seconds) its voltage until it just surpasses the target. Reading voltages in this way is wasteful of power because of the awake state energy requirements of the microcontroller. Logically, reducing the number of readings by increasing the sleep time between subsequent readings will improve the system's efficiency. Nevertheless, there are limitations to maximum sleep times. With arbitrary delays, the voltage could exceed the target and even overvoltage the microcontroller. The action of reading the analogue voltage has the direct effect of also increasing the time to target, so regular reading also requires increased charging times to compensate for the cost of the awake states.

Figure 62 shows the systematic sampling approach for charging and discharging the capacitor. During the charging cycle, the samples are every 8 seconds in this example and require the microcontroller to be awake for 2.5ms. For the rest of the cycle, the microcontroller is sleeping using 100 times less power. If the system has a time constant of 150s, it requires 13 readings to reach target 2 (sampled every 8s) and the samples required during discharging to reach target 1. There are two disadvantages to the traditional sampling approach: as well as decreasing the efficiency; the system requires more time to reach the target because of the increased parasitics.

61

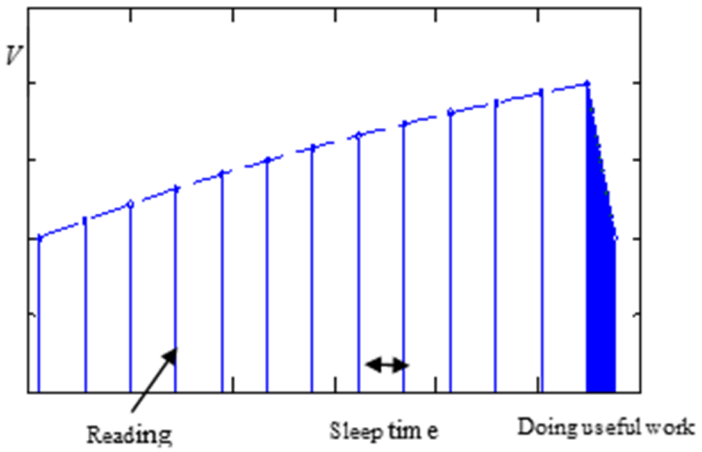


Figure 61: Charging and discharging processes

An accurate estimate of the time constant for charging an appropriate sleep time can be found for the microcontroller, allowing efficient energy accumulation, waking only once when target two has breached. With the associated knowledge of the discharging time, a constant target efficiency can be reached without depleting the capacitor below the microcontroller's minimum operational voltage.

In terms of energy, it is possible to calculate the efficiency of the charging process of the capacitor as follows:

Equation 44

$$Efficiency = \frac{Output\ energy}{Input\ energy} \times 100\%$$

Equation 45

$$Efficiency = \frac{P_{work}.T_d}{P_{work}.T_d + P_s.T_s\ +\ P_r.T_r.N} \times 100\%$$

Where: P_{work} : Useful power or output power.

T_d : Discharging time

P_s : Power consumption of the microcontroller

during sleep time.

T_s : Sleep time.

P_r : Power consumption of the microcontroller for each voltage reading.

T_r : The time it takes for each reading.

N : The number of readings.

62

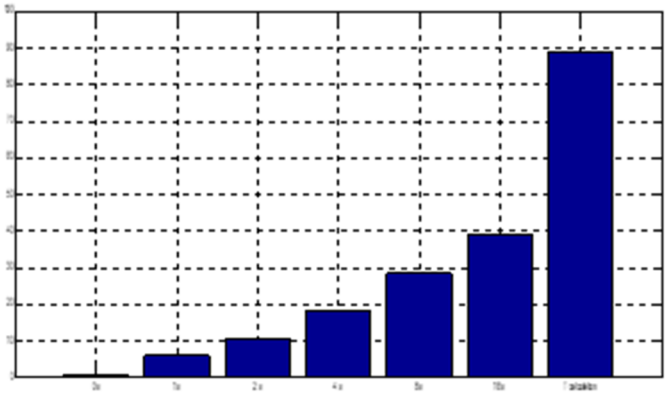


Figure 62: Efficiency vs sleep time

Figure 63 shows the efficiency of the system as a function of sleep time. The figure leads one to believe that longer sleep time gives greater efficiency, but when $T_{sleep} > T_{actual}$ overvoltage occurs, damaging the circuitry, and sub-optimal use gets made of the resources available.

The system shown in Figure 61 has been simulated in Matlab to evaluate the capacitor's charging process controlled by an ATtiny85. The variables in the simulation are shown in Table 3. The two factors of interest are the charging process's efficiency and the extra time required to compensate for the losses incurred from the regular reading during the charging process. Different sleep times are used to show the impact on the efficiency of the system.

$C = 4700\mu F$	The used capacitor
$R_c = 33k\Omega$	The equivalent serial resistance
$R_d = 700\Omega$	The equivalent parallel resistance
$I_s = 5\mu A$	The microcontroller current during sleep mode
$I_r = 5mA$	The average current when the microcontroller is on
$T_{on} = 2.5ms$	The wake-up time for each reading
$V_{target1} = 4V$	The upper level of voltage
$V_{target2} = 3V$	The lower level of voltage

Table 3: Variable Values

Looking now at the discharging, without any interval time between any two readings, the efficiency will be zero for the given setup shown in the table because all the stored energy will get spent driving the microcontroller and performing a continuous reading. Figure 10 shows the effect of three different interval times between readings during discharging. The figure illustrates that the longer the interval time, the higher the efficiency for a specific sleep time. Moreover, increasing sleep time improves the efficiency for a specific interval time.

63

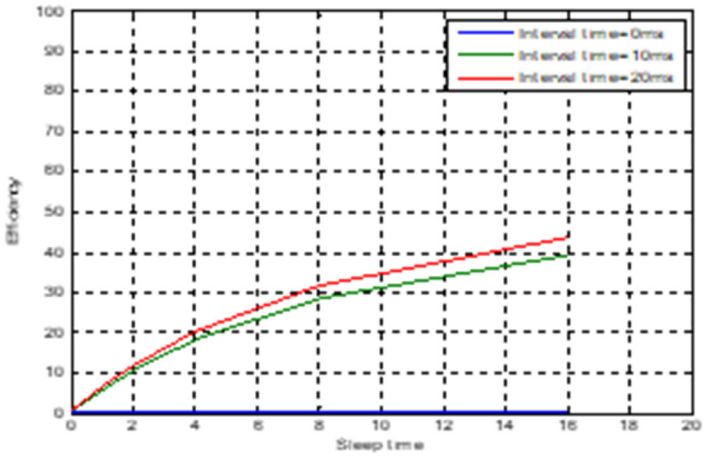


Figure 63: Efficiency of the regular reading Method

The comparison between continuous and our ‘selective time constant’ based method shown in Figure 64. It is clear from the figure that the efficiency of the time constant calculation method

is more than two times that of the regular reading with 16s sleep time and 10ms discharge interval time for the given 150-second Tau setup.

64

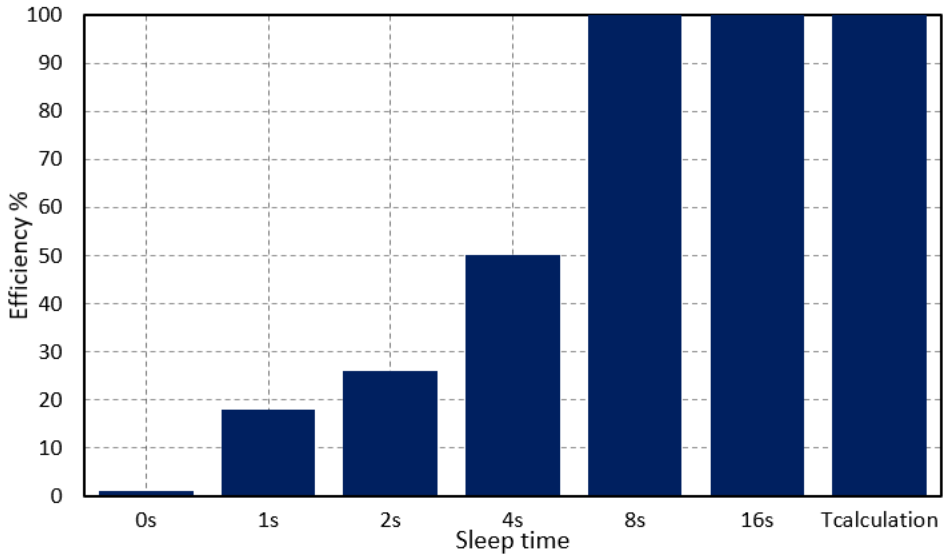


Figure 64: Efficiency of a regular reading method with different sleep times compared to the τ calculation method

Figure 65 presents the percentage of the additional charging time as a result of multiple readings proportional to the required time of our proposed method:

Equation 46

$$Extra\ time\ percent = \frac{t_r - t_p}{t_p}$$

where t_r : Charging time of regular reading method

t_p : Charging time of proposed method

65

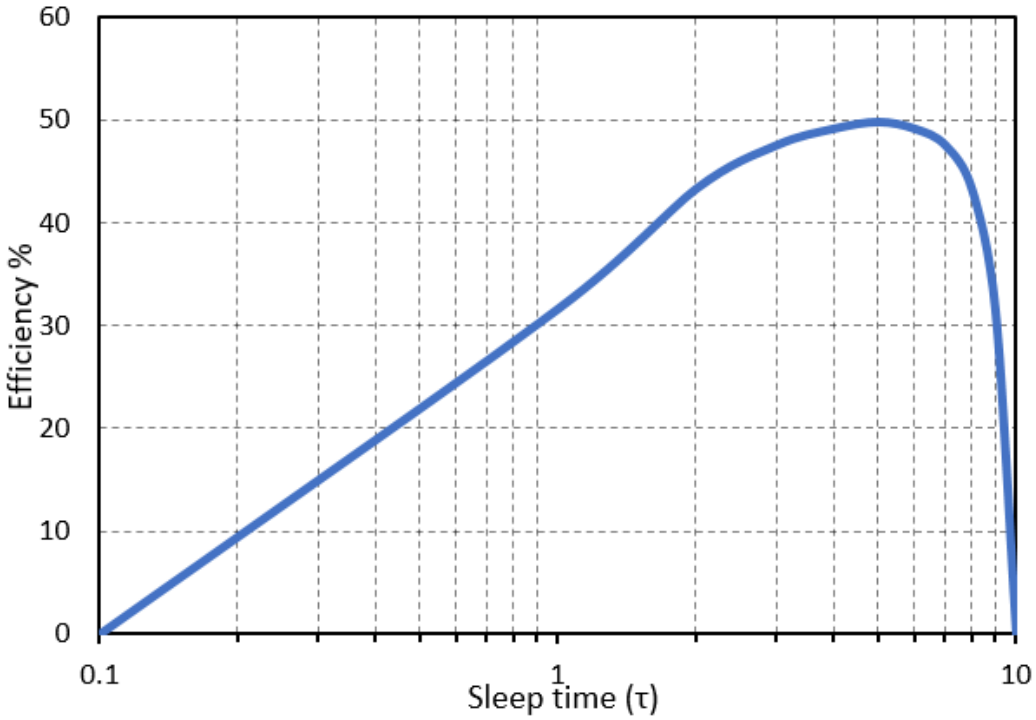


Figure 65: Percentage of extra charging time for regular reading compared to the proposed τ method

Illustrated by Figure 65 and Figure 66, using low sleep times is infeasible due to the low efficiency and increased charging times.

66

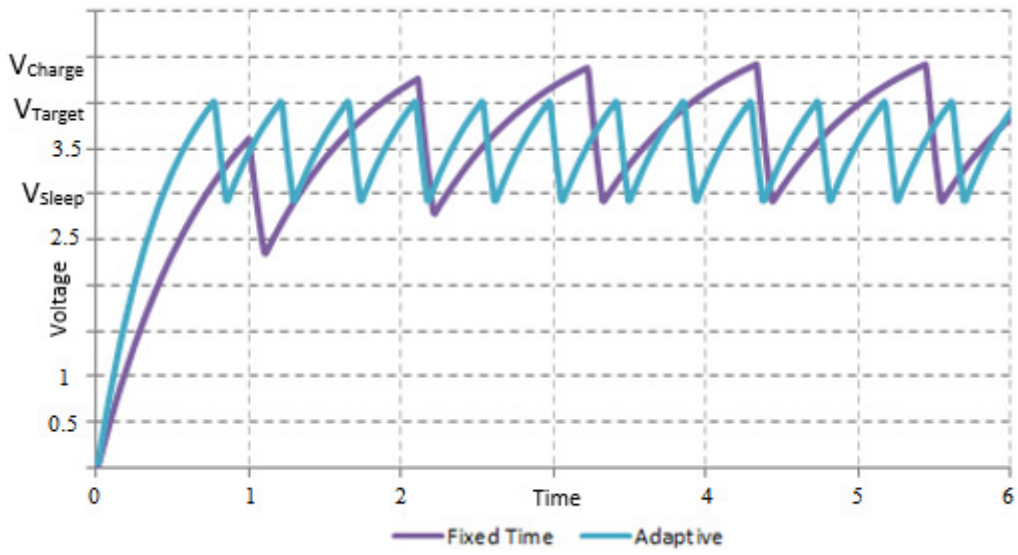


Figure 66: Comparison of Volts over fixed time (Seconds) vs proposed algorithm

Chapter Summary

Power management is achievable by implementing intelligent control in designing a wireless power transfer system but has far-reaching implications for other low power applications. This research presented an algorithm to run a load with a source that has less energy than the load's continuous requirements. The theory evidence comes from using an ultra-low-power microcontroller that makes sleep and awake decisions to accumulate the maximum amount of usable energy. At the appropriate time, the micro wakes up, briefly runs the load, keeping sufficient power to allow it (the micro) to stay alive.

The presented algorithm is easily realised in code and achieves best-case utilisation from just two readings. This contribution represents a significant improvement in efficiency over the periodic analysis case. It keeps charging times fixed without significant expansion and allows increased discharging time to do more useful work.

Chapter 5: Empirical Research, Phase 2. Meshing and Communication

Methodology, Phase 2

2011 saw the Bluetooth SIG [86] release their first deployable implementation of a new Bluetooth version known as Bluetooth Low-Energy (BLE), also referred to as Bluetooth Smart.

The drive behind its creation was a clear market trend targeted towards battery-powered devices. BLE is not constrained by or requires the 'device pairing' protocol initiation procedure found in the previous incarnations of the Bluetooth offerings – now renamed as 'Bluetooth Classic.' BLE is targeted explicitly towards connectivity for the Internet of Things and small sensor type networks, potentially adopted as some kind of broadcast-type sensor offering data samples to other BLE devices, all of which do not need the pairing overhead.

The original Bluetooth offerings revolved around a star topology providing point-to-point type connections between the devices. Consisting of some kind of hub or central device – e.g., a PC or a smartphone performs the pairing operation with one or many different devices. The limitation of the number of pre-paired devices is potentially unlimited; however, the number of these devices that can be simultaneously connected to and used concurrently is limited only by the specific implementation of the software and hardware.

Due to the star-topology of Bluetooth Classic, devices cannot directly connect and communicate with each other, and if the requirement is to pass messages from one device to another, then some kind of Central hardware must be utilised as a relay.

BLE works to overcome some of these pitfalls and resource-hungry limitations. It adds both a point-to-multipoint and a broadcast facility available for short-range navigation beacon tasks such as near object identification and classification, achieved using a combination of RSSI (Received Signal Strength Indicator), Calibrated antennas, and knowledge of the transmission

devices power output. Broadcasting offers an efficient way to send one message to every receiver simultaneously.

The Classic version was again revisited and revamped in 2016 where the release of version 5 was delivered. This version offered a greater choice on data link speeds and coding and providing support for longer-range transmissions.

More interestingly, developed alongside this release was the introduction of a new connectivity model that allowed mesh networking. This model was named the Mesh Profile and is a many to many communication style with integrated provisions for message relay from one device to another. This profile allows creating a very flexible mesh topology with multiple potential paths between nodes, ensuring redundancy, self-healing, and increasing the probability of successful message delivery.

Communication

67

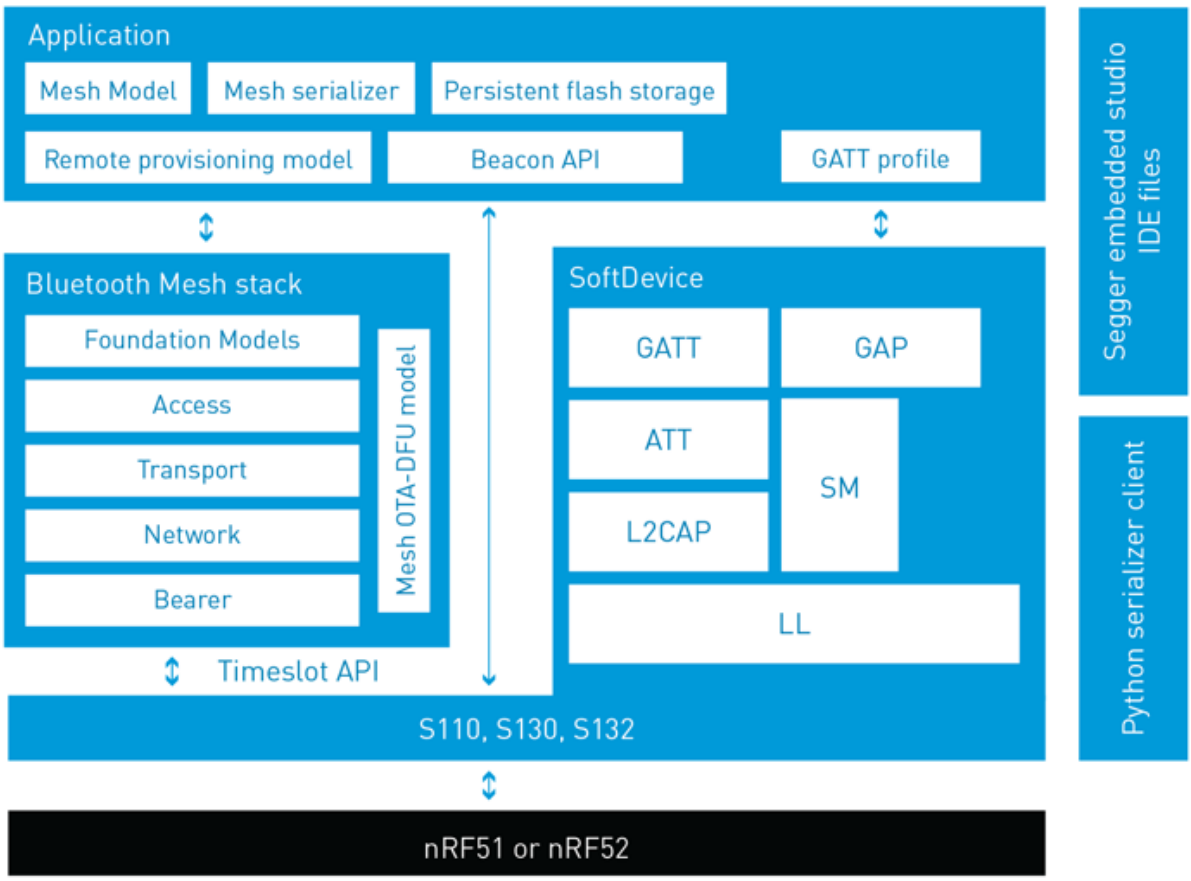


Figure 67: Nordic Semi BLE Mesh [86]

Figure 68 illustrates how the Bluetooth Mesh model fits into the standard BLE protocol stack supplied by Nordic Semiconductors [86]. The Bluetooth Mesh-specific extensions are visible on the left-hand side of the diagram. The Mesh profile was not exclusive to the Bluetooth 5 release as such; its also compatible with Bluetooth 4.2.

BLE Meshing

The Bluetooth mesh profile includes several generic models that can provide solutions for most problems when combined. The generic servers have simple variants such as On/Off, Open/Closed, and Level values. A considerable influence on the Mesh Profile design itself was

the smart home and smart lighting industries. For example, a typical smart light would use the generic On/Off model alongside a generic Level model to manipulate its brightness level.

A device must accept provisioning to allow it to join a particular network segment, and to achieve this, it must execute a sequence of connection setup exchanges with its peer. Specific encryption key sets for the device and the desired part of the network must be exchanged with its peers, after which it can assume its role of network 'node.'

Server Models and Elements

Devices are built by mapping generic server models to every 'feature' offered by the underlying product, and these get referred to as Elements. Every device must provide at least one primary Element and can have one or many secondary Elements.

An example of this Elements hierarchy would be a Temperature controlled bathroom fan. The primary element would be the generic On/Off model representing the fan's power control. A secondary element is introduced, which is a generic level model representing the current sensed temperature value.

Additional elements get added as secondary elements, all of type 'generic level', which represent the other features of the product such as the temperature setpoint to turn on, the setpoint to turn off, maybe even a humidity level. For even more integration, some variants of the same product may also include additional elements allowing for the humidity readings to start and stop the fan.

A Bluetooth Mesh network is capable of connections between 32,767 individual nodes. These nodes are groupable into a maximum of 4096 different subnets allowing broadcast domain control. Provision is made for up to 65535 'Scenes.' A scene is a collection of common elements applied to multiple devices of different types. In the smart home environment, a scene

may well control the levels of numerous bulbs, heaters, and fans, setting them all up to pre-defined settings required for a specific mood. For example, there may be many types of different bulbs from different suppliers; however, they all have a generic level server that can control the brightness.

Mesh Friends

The profile allows for ‘friend’ devices, which facilitate the nodes' ability to remain in an ultra-low power consumption state, yet still retrieve the messages targeted towards the node during its sleep.

The friend device implements a store-and-forward feature to buffer messages to the hibernating node for delivery when the device wakes up. It is the device's responsibility to connect with and check their friends for any pending messages they may have. This power shift moves the entire responsibility and time dependency of retrieving and sending messages to the ultra-low-power device and enables it to be operational for extended periods using extremely low energy levels yet still maintaining an effective communication medium.

68

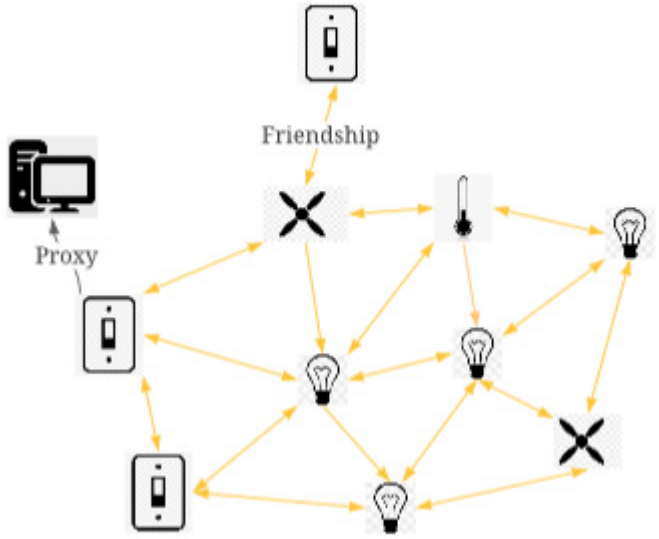


Figure 68: Mesh Network Device Types

Further to the friend device feature, the spec also provisions for a proxy device. Since many user-orientated management nodes do not contain mesh support as standard, the need to pass control messages to and from the network and manage provisioning duties are satisfied by proxy devices.

Mesh Proxies

The proxy allows applications from none-meshing hardware to relay messages in and out of the network. It can also be implemented as an ultra-low-power device using message pushing to communicate with the proxy device.

Friend devices cannot typically operate as a device that can sleep/wake unless network time synchronisation facilities exist to support it.

The mesh uses a managed flood routing protocol to forward messages from one node to another. The messages use a time-to-live (TTL) value to prevent infinite message looping and ensure broadcasts get delivered to every device in the vicinity. The TTL is decremented every time the messages pass through a node until it reaches a zero value which then causes it to drop. The maximum number of hops offered by the Bluetooth Mesh is 126.

By nature of the flood routing and TTL, every device will eventually receive every message; unwanted and unintended message reception can be reduced and controlled using the network segmentation subnet feature.

Managed flood routing allows BLE to scale and cover larger areas quickly and easily and allow internal expansion as the nodes feature requirements dictate; this is a significant addition to the previous Bluetooth specifications.

Control Messaging

A Client-Server architecture operates as the control protocol. The server model is used to track and represent states or states that span elements from within the device. The client model defines a set of messages used by the client to get, set, or request status for a server's various states.

Finally, a control model adds the control logic needed for interactions between the client and server models.

The mesh's communication system is based on passing messages with a maximum payload size of 29 bytes, which is by design intended to be used to surface and publish the device's element states.

State Notifications

There exists a notification system that operates on a publish and subscribe model. Here other devices can send a notification request to a server, which as actors they wish to subscribe to and will be getting, setting, or requesting status from the servers' elements.

As the messages do not traverse a central device or broker of some kind, the mesh's managed flood protocol will ensure every device receives every network message. The penalty here is that all servers will also respond to these messages; careful use of the subnet segmentation reduces this overhead.

Security Layers

Security and encryption are compulsory when joining and using the mesh. A device setup and provisioning procedure must be executed for every node, after which the device gets granted access to join the network.

Provisioning involves exchanging security keys for the network and the device, setting the device's subnet parameters, and then provisioning the device with setup and operational data.

The encryption and authentication of all mesh messages are, if possible via device implementation, done using an Elliptic Curve Diffie-Hellman (ECDH) implementation to establish a shared secret with the provisioner for further possible key exchange.

Alternatively, the provision also exists for an out-of-band key exchange which can take advantage of the QR Code scanning facility. Here, the device gets supplied to the end-user with its shared secret embedded in a barcode. This barcode is scanned by the provisioning device, after which all subsequent traffic will encrypt using the AES-128 symmetric cypher.

This provisioning is a very convenient means of deployment as the barcode can be included in its packaging or attached to the device itself.

The mandatory use of message encryption and authentication is a significant strength of the protocol and enforces the need to have tight security integration from the very beginning. Every device must be authenticated and have securely exchanged keys before any network messaging can occur.

Provisioning commences via a proxy node, with control coming from either the end-user or via a previously successfully provisioned device.

The provisioning process consists of five distinct phases:

1) Beaconing
2) Invitation
3) Public key exchange
4) Authentication
5) Provisioning Data

The unprovisioned new device will begin the process by periodically advertising a beacon message showing its desire to be provisioned.

This beacon gets answered by another device willing to provision the new device by sending it an invitation.

The reception of an invitation will cause the new device to transmit a capabilities protocol data unit (PDU) which describes the following features offered by the new device:

- Number of elements the unprovisioned device contains
- Capabilities in terms of supported security algorithms
- Out of band public key information if available
- Capabilities of this device reporting data
- Capabilities of this device accepting data

Using this PDU, the provisioner will decide how best to exchange the security keys with the new device.

The procedure then securely exchanges several nonces (random numbers) to initiate the authentication stage. Either device generates the nonces, and the PDU contents determine who will take this responsibility. The other device waits to acknowledge the receipt of the nonces. Both devices can also generate a set of their own nonces and exchange them, following up the transmissions with their respective acknowledgements.

Confirmations then follow this process that the nonces are correct and the authentication process can complete, this starts the provisioning phase, and its data gets sent to the new device. This data will include:

- network encryption key
- Unique device key
- Bookkeeping data
- Unicast address

At this point, the new device becomes a ‘node,’ and its unique network address is the Unicast address passed to it during provisioning data transferral. This address points to the device's servers' primary element. The provisioning procedure can now complete, and the node can begin receiving and transmitting mesh messages.

If a device is to leave a network, an un-provisioning procedure gets performed where the node can give up its network encryption keys and gracefully leave the mesh.

In the case of hardware failure where a device becomes completely unresponsive, yet there still exists a requirement for its removal from the network, a facility exists where the other nodes can blacklist the failed device, preventing it from establishing connections to other network nodes from that point on.

After a blacklist has occurred, the provisioner can initiate a Key Refresh procedure that generates a new network encryption key set and securely distributes it to all the network nodes.

This extra security layer prevents an attacker from obtaining a device and somehow removing the encryption key and using it in a custom hardware device to penetrate the network.

The mesh protocol also provides cybersecurity features to protect the integrity of the network's PDU exchanges. These include PDU Sequence numbering and Initialisation vector (IV) index fields, which allow sequence enforcement and audit trails – preventing such intrusion attacks like a reply attack.

Incorporating the BLE Mesh profile into the BluBot hardware platform [110] was straightforward. The layout, to collect and analyse comparable data, consisted of the following hardware and software collective:

69

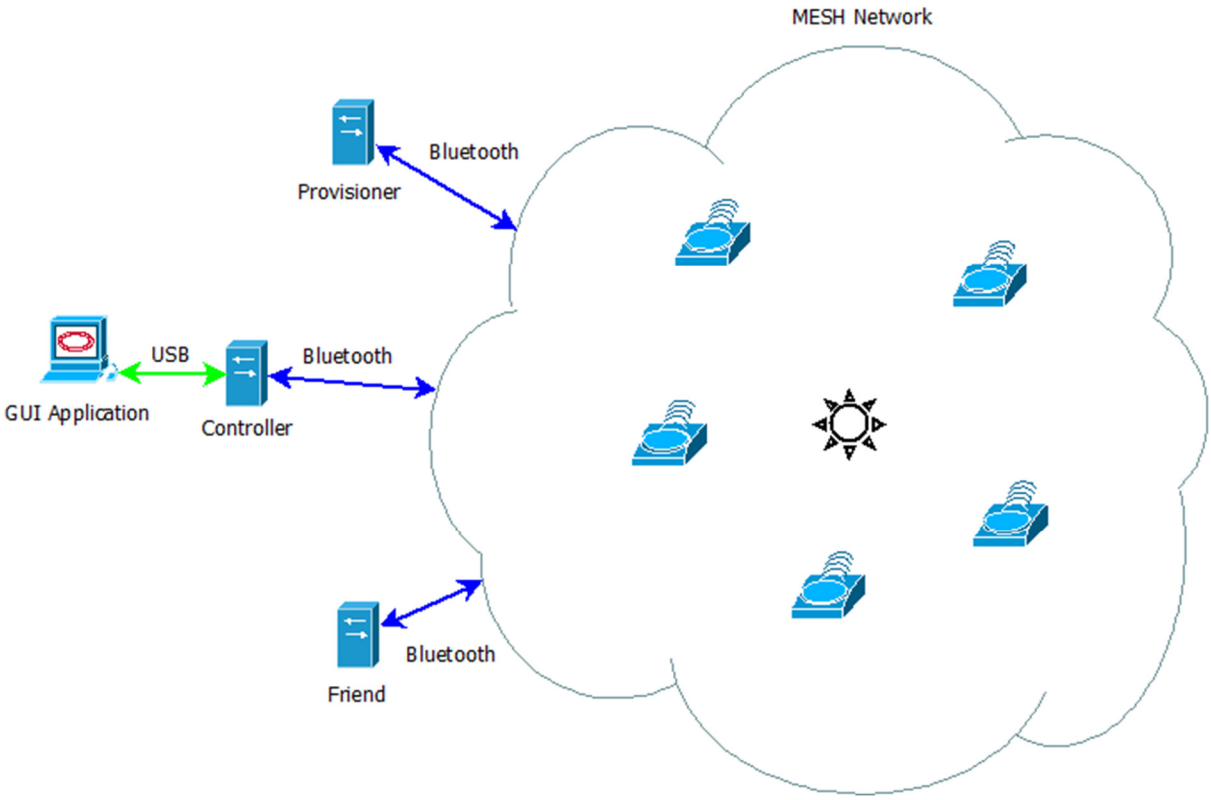


Figure 69: BLE Mesh Layout

1. Provisioner (Enabling the network to perform automatic discovery and provision of the BluDot devices). A single hardware unit deployed alongside the controller. In conjunction with a supporting application, this device enables the end-user to query the network's status and properly remove nodes when required.
2. Mesh Friend (Allows message queueing for low-power devices, enabling them to hibernate). As the devices' coverage area was not large, a single hardware unit was enough to achieve the research goals.
3. Mesh Controller (Hardware interface between network and PC/App). This device acts as the proxy node, allowing messages to traverse between the network and the user. The GUI end implements a USB connection to the Bluetooth Mesh hardware device.

4. Bluetooth Mesh BluBot profiles (Publishes server and client models for the devices, these will facilitate the *task* execution and allow status requests)
5. GUI User Control/Monitor Application, with the facility to capture the network's message data for analysis.
6. Software implementation of the Bluetooth Mesh specification
7. Software implementation of the Mesh Provisioner specification
8. Software implementation of the Mesh Friend specification
9. Several BluBot LPNs (low power nodes) will represent the workforce.

Each stage's implementation uses modified versions of the software offerings from both Nordic semiconductors [106] and the Zephyr mesh project [122].

Provisioner Device

The provisioner is a single standalone hardware device.

The provisioner's role is to provide an entry point into the mesh for new devices and nodes and facilitate nodes wishing to leave the mesh by ensuring they get appropriately decommissioned.

This hardware’s firmware is implemented using the Bluetooth Mesh SoftDevice, and reference design implementation offered by Nordic semiconductors [106]; alongside the nRFConnect app running on Bluetooth enabled Android devices, the provisioning duties operate as per the stock reference specification.

The device performs the following management tasks:

- Scan for unprovisioned mesh devices which are waiting for an invitation
- Checks for compatibility by querying mesh nodes for the correct BluBot services
- It gives the device a unique unicast ID address
- Exchange the network communication keys
- Enumerates and initialises (provisions) the LPN device

Four individual Application Keys get deployed with the devices, each one covering a generic server implementation,

The first covers the generic On/Off server, which controls the LPN's LED.

The second is a generic Boolean value server which represents the current state of the LED.

The third key is to implement a generic level server representing the LPN's current energy store level.

The final key is reserved for a generic level model, allowing the controller to implement a synchronised message wakeup pattern.

Mesh Friend Device

A Mesh Friend is one or many standalone hardware devices, but not low-powered. They get provided with a permanent power supply and, by design, are not intended to be self-sufficient.

At the time of implementation, the idea of an LPN using a friend to store and forward a hibernating device's messages was in the process of being finalised, and as such, a friend solution from Nordic semiconductors [106] was not available.

Therefore, the implementation uses a modified version of the code available from the Zypher mesh project [122]; this code is both open source and available for download. As both

companies fully support the published BLE specifications, both providers' offerings work together without issue.

The implementation is in its raw distributed form, and an LPN must explicitly establish a friendship with the device before any message traversing can occur.

The implemented duties include:

- Continually listens for new BluBot friend requests
- Makes friend offers to unfriended LPN devices
- Establishes and maintains individual message queues for all friended devices
- Provides a service that allows the BluBot LPN devices to poll it for pending messages

Mesh Controller Device

A Mesh Friend is one or many standalone hardware devices, but not low-powered. They get provided with a permanent power supply and, by design, are not intended to be self-sufficient.

The controller part has been implemented using the Nordic Semiconductors [106] Softdevice alongside a bespoke firmware implementing the project-specific details.

The primary duties of the controller cover:

- Network monitoring, size, and load
- Provides feedback to the controlling PC via a wired comms link
- Allows message passing from the user to the network and vice versa
- LED Blink Client Model – Allows the device to request other devices to blink LED
- LED Status Client Model – Allows the device to request other devices LED state
- Power Level Client Model – Allows the device to request power status information from other devices
- Monitors device health, RSSI levels, communication losses etc
- Time sync client model – Allows the controller to sync the wake-up times of the mesh devices

The device allows a copy of the messages sent over the mesh to pass through the hardware USB port destined for the GUI software. It also provides an infrastructure to allow predefined command transmissions to either all or individual LPNs within the mesh.

BluBot LPD

BluBot LPDs are individual standalone ultra-low-power hardware devices with communication abilities and are designed with self-sustainability in mind.

The hardware design supports the following primary requirements:

- LED Blink Server Model – Provides a service that allows other devices to issue a blink LED command
- LED Blink Client Model – Allows the device to request other devices to blink LED
- Power Level Server Model – Provides a service that allows its current energy store level retrieval
- Power Level Client Model – Allows the device to request power level information from other devices
- Low power PWM LED controller
- Implements deep sleep and auto wake cycles
- Implements a service which searches the mesh for nearby friend devices
- Implements a service to request friendships and accept friend requests
- Implements a service used to poll friend devices for pending messages
- Time sync server model – Allows wakeup event synchronisation with other mesh devices.
- Time sync client model – Allows wakeup events to be initiated directly with other mesh devices.

Implementing both the client and server parts of the generic model into the LPN design allows the units to achieve a peer-to-peer messaging and command initiation setup, which significantly reduces the overall traffic flow and the need for the controller to act as a post office.

GUI Application Software

The user software parts of the deliverables consist of a PC based software application and a small supporting collection of mobile device applications.

The GUI application runs on a windows platform and is written using C++. The controlling PC does not need any Bluetooth hardware support or drivers, as the Bluetooth radio, stack, and supporting hardware have already been implemented within the Mesh Controllers hardware assembly.

The GUI application duties comprise of:

- Display all discovered devices currently joined to the mesh network
- Queries and displays power status of these devices
- Provides a means to interact with the network, send and receive messages which initiate commands.
- Allows research and debugging data from devices to be collected and stored.

Various mobile applications are available from Nordic Semiconductors [106] that aid in the initial configuration and deployment of a Mesh network and provide the ability to monitor and diagnose a problematic network if needed.

Radio

The radio interface generates a 2.45GHz carrier when transmitting. The provisioner, friend, and controller devices have their radios permanently enabled. However, the LPNs will only enable the radio when they are unaware of the current communication cycle or expecting to receive or transmit a message. This deployment strategy further illustrates the need for the constantly-powered friend device to act as a go-between.

The flowchart in Figure 71 below shows the overview of how a device behaves when first introduced into the network. The provisioner hardware must initially provision it. Following successful provisioning, it must immediately find and bond with a friend to assist in message passing. When these two setup stages are successfully complete, the device can begin to participate in the low-energy work tasks provided via the network. The actual message flow is detailed in Figure 73 below, and here we can see how the friend can assist in enabling the device to realise a low-energy existence. If the friend were not available, the LPD would need to permanently enable its reception radio due to it being unaware of any message transmission timing, thus consuming considerably more power. The friend allows the device to wake up periodically and request any pending messages for its attention. The LPN, however, can immediately send any responses or transmissions directly to the network, as at this point, the LPD is already in an operational state consuming energy. The friend devices can perform their own energy conservancy and operate in a low energy state if needed, as receiving and storing messages is their only task; they do not undertake any other workload duties. Friends do not need to be mobile or perform any intensive CPU operations, but they do need to have a radio listening and available whenever an LPN or controller message exchange event occurs. Potentially, duty cycling the radio in a predetermined and predictable way could also lead to reduced operational consumption.

70

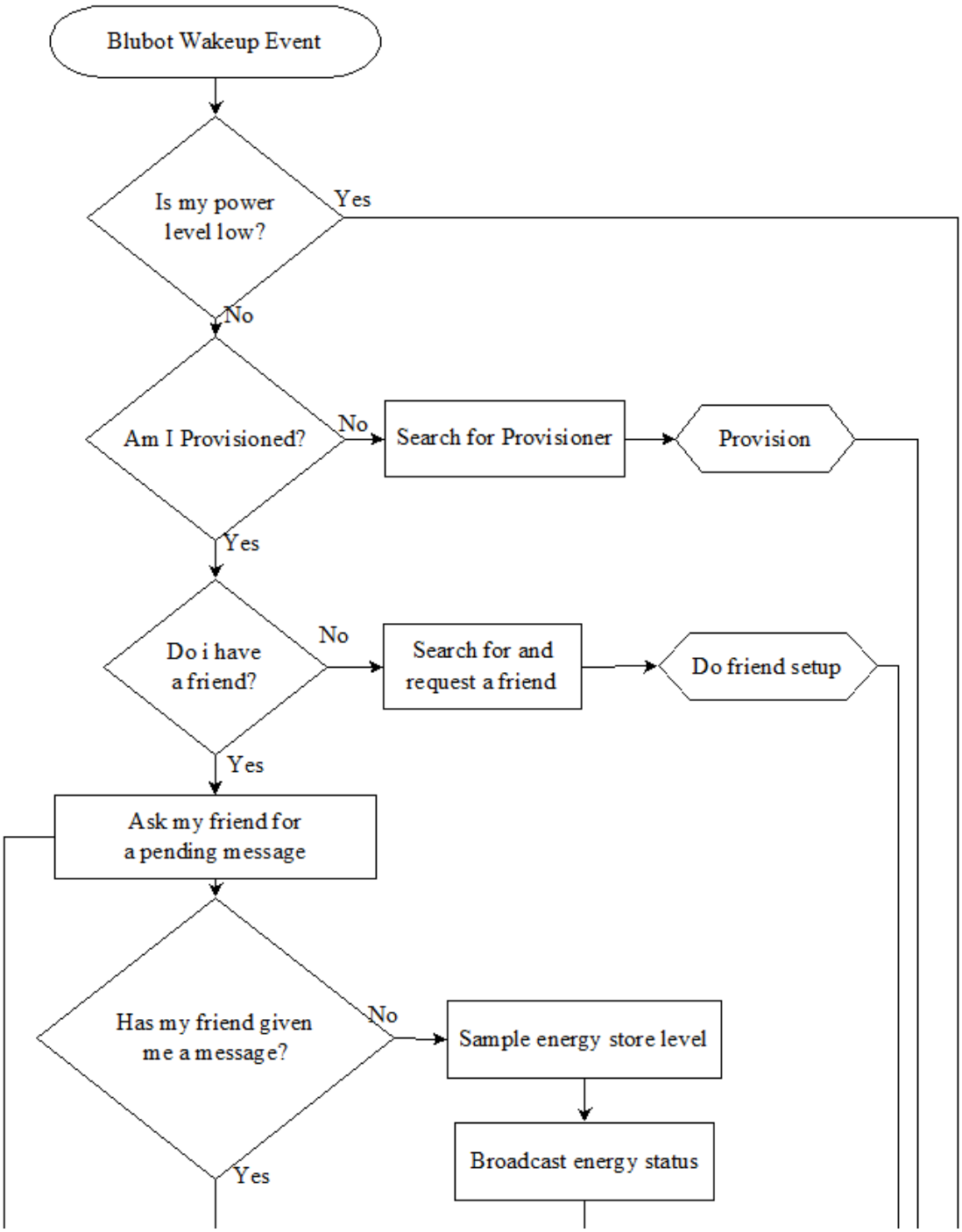


Figure 70: BLE Device Coms Overview

71

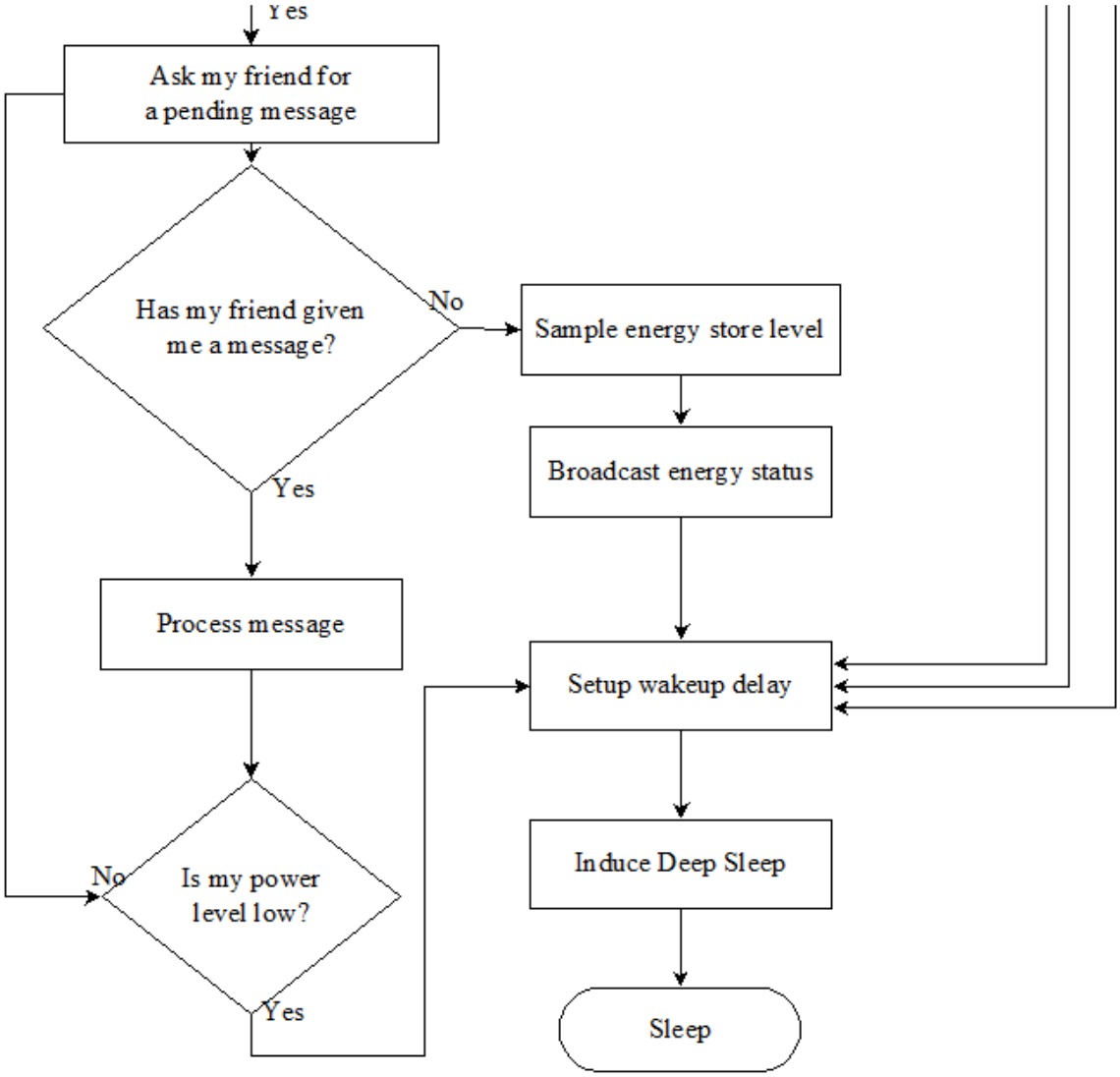


Figure 71: BLE Device Coms Overview Part 2

72

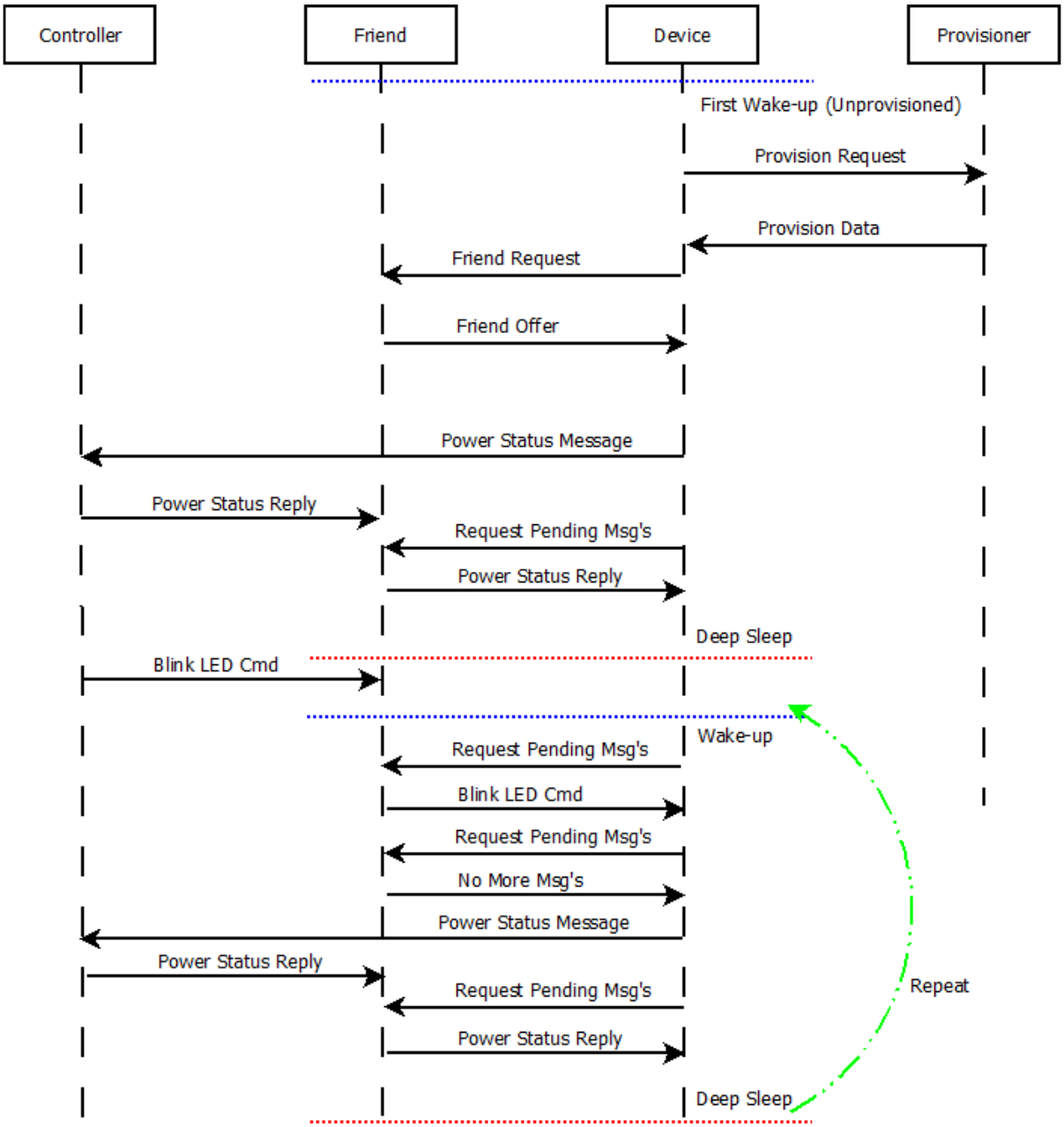


Figure 72: BLE Message Flow Diagram

Chapter Summary

Meshing proves to be a very resilient and capable communication protocol. We can make better use of our energy reserves and be more efficient if we can implement a more direct to peer messaging approach when needed.

Direct communication, self-synchronisation, and independent operation are also highly sought-after attributes.

If we again reflect upon how nature achieves communication, it seldom reveals that it requires everybody hearing what everybody else has to say. Evolution has shown that harnessing all types of messaging protocols and mediums is the best and most diverse way to communicate, and this covers unicast, multicast, and broadcasting.

There are other nature traits we can look at; for example, if we have a message to pass to a friend, we would unlikely hold onto that message in our minds for years. Evolutional messaging systems seem to have achieved ways of losing baggage and only coping with a limited number of internal queues – but it works.

Adopting these kinds of tactics works very efficiently when considering them within the context of 'survival solely to complete tasks'.

If, after some time, a message does not get through, it will naturally be retried at some future point and forgotten. The interval before any re-attempt to pass on the message, at this stage, is dependant on the urgency associated with the message.

The idea is we do not need to throw massive amounts of hardware and energy resource at these types of problems, just as evolutional communicational systems have adapted. It is enough to make a best-effort attempt and keep trying for a reasonable period. The most crucial objective is survival, not message delivery.

When we start to break away from the mesh connectivity idea, or the network becomes segmented down into smaller groups, we almost start to create explorers ready to move away from the comfort and close contact of their siblings and settle in a new group, potentially creating another ‘mini-mesh’ community.

The process of how to make a unit try and detach from the mesh to expand can be solved by having dedicated devices used to join two independent mesh systems together and provide a communication gateway or a proxy.

The more this type of expansion gets consideration, the more its realised that further supporting hardware is needed to ‘prop up’ the platform. Controllers, friends, gateways are all energy consumers and effectively restrain any kind of self-survival.

Devices have a way to contact their home base station by passing messages through other devices and being available to help pass another device's messages onwards to their destination.

The guaranteed delivery aspect of the message is lost, but evolutionally, it seems not to be there anyway.

Time synchronisation is essential and somewhat challenging for this type of operation. Tolerance windows get utilised to counter fluctuations in timing crystals and resonators. This addition will inevitably cause energy loss due to the extra time needed to ensure successful wake synchronisation.

The device must be in a position to be able to decide for itself if it is capable of sustaining communications of any sort based on its current charge level. It must schedule its flow of messages and have a small limit in terms of any type of queue that needs servicing.

Chapter 6: Empirical Research, Phase 3. Intelligent Energy Management Protocols (Original Content)

Methodology, Phase 3 The requirement for Energy Management Protocols

When a device operates without the constraints of a battery and relies only on the energy it can scavenge from its ambient surroundings, every single micro Watt of power successfully plucked from the environment and forced to pay its harvesting costs, followed by having its remainder stashed into a leaky storage reservoir is a remarkable achievement.

Thus, energy has to be considered a valuable resource, and it must be kept, guarded, issued, and used appropriately. Wasting it cannot be tolerated and undermines all the work done to collect it, so the entire accumulation and consumption system needs proper management measures in place.

It is often expected for an LPD to deliver a summed output of useful work that is greater than the primary energy the device held during its commissioning. There must be measures to control the food source's spoils and how it gets distributed to achieve this.

Sleep strategies and intelligent energy management protocols allow devices to operate and survive well beyond their initial design specifications.

Time Dilation Based Protocols

Various types of these energy management protocols have been covered in Chapter 3: The research approach in this thesis (Original Content). They enable the device to look at their available energy reserves and profile their energy expenditure while completing their given *task*. This dataset's contents then make it possible to calculate the optimal times and durations for hibernation and operation, increasing working time efficiency per joule of energy given.

As the incoming energy decreases, time is dilated further by simply hibernating and recharging for longer, potentially even working for shorter times and at different interval patterns.

When time gets managed in this way, and the *task* can be defined as a 'continuous effort needed' type of work, this loop of managed-hibernation followed by task-execution can continue forever. The measurement of *progress* is defined as the percentage of completion of the '*task'* in its entirety (take a sensor-measurement or pass-a-message, for example). The measure of *efficiency* would be the progress percentage against the time it has taken to get there, combined with the overhead cost of self-survival and intelligent energy management protocols, along with encountered false wakes that occurred during the expected execution time.

Equation 47

$$\eta_{Task} = \frac{t_{task\_minimum}}{(t_{finish} - t_{start}) + t_{overhead} + \sum_0^n t_{falsewakes}} \times 100$$

The small cost in terms of energy consumption overhead required by intelligent energy management protocols is mostly a deterministic one. The current *task* gets broken down and energy-costed in terms of completing *usable sub-units* of the overall *task* (Figure 74 below); thus, the overall performance of time dilation-based operation is directly proportional to the energy harvesting input stream.

73

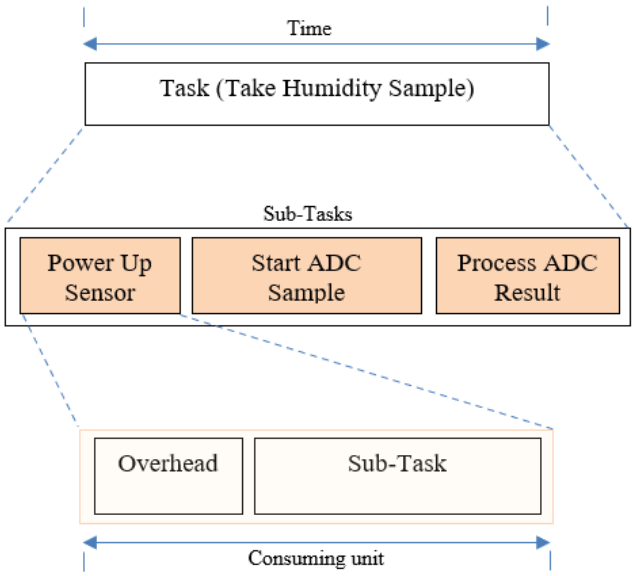


Figure 73: Task to Consuming Unit

The more harvestable energy flowing into our unit will allow the stores to charge quicker, allowing shorter hibernation periods and more frequent *task* sub-unit completion periods (Figure 75 below), increasing overall *task* completion using less time.

74

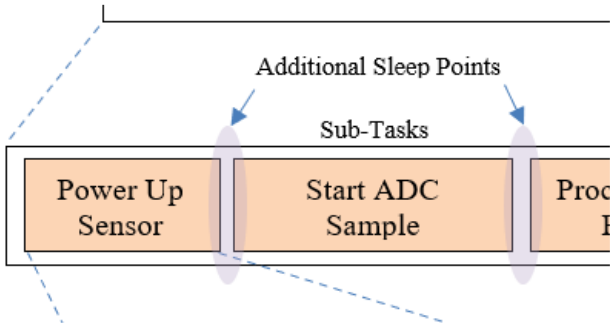


Figure 74: Additional Charge Opportunities

On the opposite side of the coin, if the energy input is so low that only the minimum survival amount is being harvested, thus allowing the device only to wake, assess its levels and return

to immediate re-hibernation, the overall completion rate will become unsustainably slow. At that point, the *task's* success or failure can only reflect a previously defined and imposed time limitation.

A time-dilation-only type of operation will have three basic blocks of energy consumption, illustrated below in Figure 76.

1. The energy consumed to wake-up, turn-on, assess energy store levels and then decide to work or sleep.
2. The energy consumed to assess its *task* and complete a useful work unit from it.
3. The energy consumed to assess its particular work *task* but decide that it cannot complete a useful work unit and should immediately re-hibernate.

75

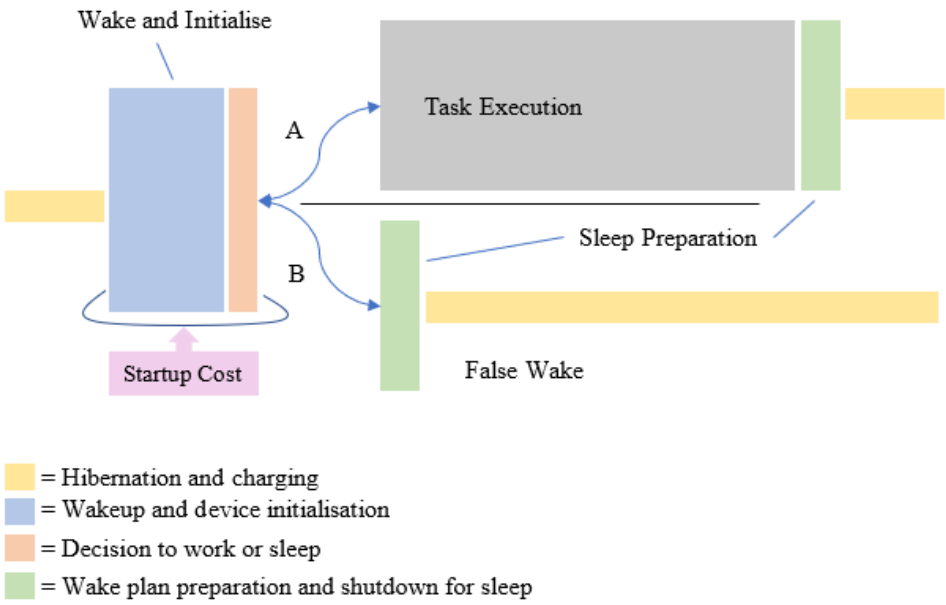


Figure 75: Deterministic Paths of Execution

The first item is largely deterministic and can be considered the absolute minimum requirement to sustain meaningful survival.

The other items in this list can be costed individually in terms of energy consumption. Still, the rate at which either of them occurs will only be deterministic if the harvested energy stream is also deterministic, as each *task* incurs decisions and costs, as illustrated in Figure 77 below.

76

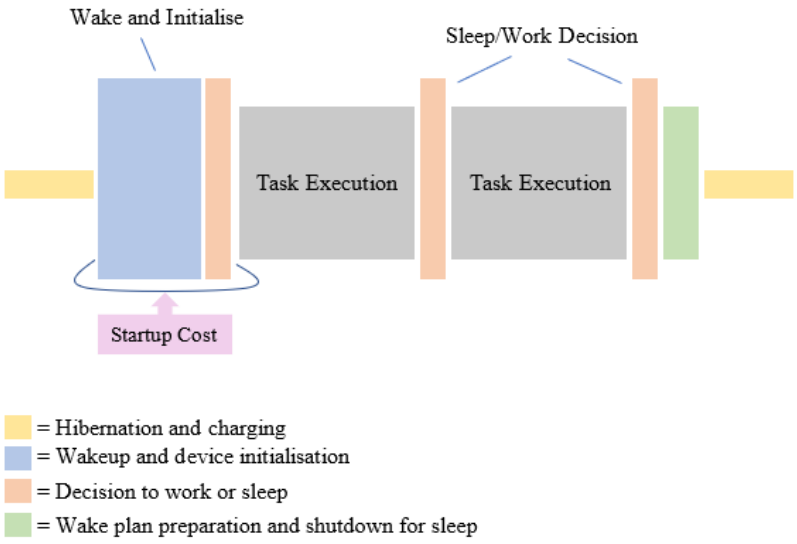


Figure 76: Multiple Tasks Per Wake Slot

If a wake event has occurred, and the energy store level sampling phase returns an unexpectedly low result, then the decision to work or hibernate has already been determined, the device must re- hibernate and incur the wake-up cost loss of this decision. This event is classed as a false-wake and is illustrated below in Figure 79.

When the incoming energy stream fluctuates around the same level as the energy needed to achieve the first item in the list, then the device's energy store will also fluctuate and require extended periods to overcome the balance. This random fluctuation can be considered as an energy noise floor:

77

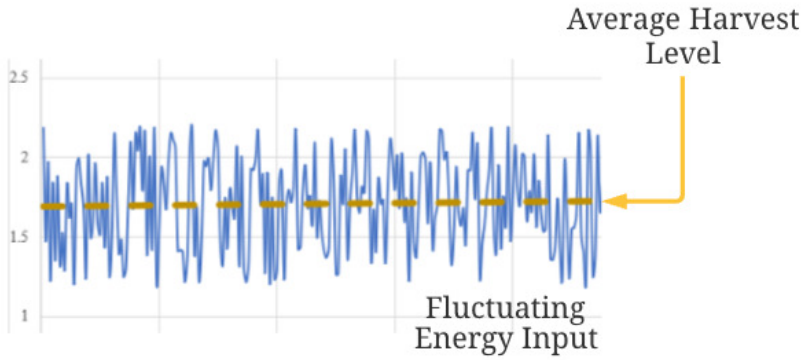


Figure 77: HEFa Example

Figure 78 above shows the effects on the energy store when the device is trying to forage for tiny amounts of energy while sitting on an energy noise floor. Parallels with nature can once again get drawn here when we consider animal foragers. Wake up, nip out, find some food, eat a little, store as much as possible back in the nest and then go back to sleep. Sometimes food is hard to find, and returning to the nest for a hungry sleep is a reality and needs to be managed, preferably with the help of some previously prepared reserves.

Ultimately, the measured HEFa (Harvest Energy Floor average) must, over time, be more significant than our minimum sustainability cost. A larger HEFa must consistently be available to successfully achieve *task* completion within any imposed time limitations.

78

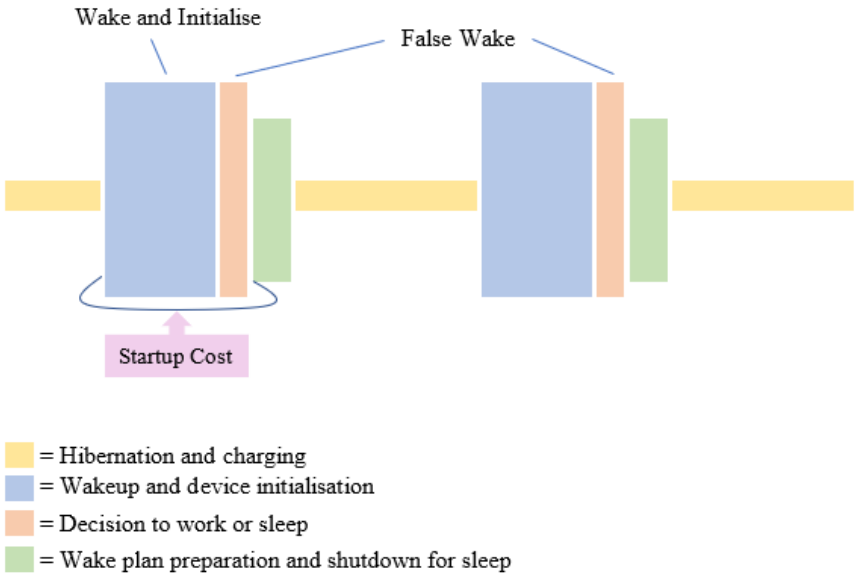


Figure 78: Spiral of Death

The inefficiencies start to become apparent when we look at how the device decides when to hibernate and how long the hibernation period will last.

In its purest non-adaptive form, it would be mostly constant values chosen at design time for both of these parameters. The influencing factors are product responsiveness, environmental circumstances, the complexity of the assigned *task*, and all of which would have been researched and understood beforehand.

The device will perform outstandingly in ideal placement situations, and as expected, smooth, periodic patterns will be displayed consistently, following tightly against all design modelling. It is outside of these ideals where the inefficiencies can quickly build up.

After design considerations get finalised, items 1 and 2 from the above list can be made deterministic for these ideal operating characteristics. As soon as the input energy stream or *task* complexity tracks away from ideal, false wakes will begin.

The event false-wake gets classified as:

The device waking up expecting to complete its task unit; however, the reality is it has not gained enough charge and therefore has no other option but to re-hibernate for a further predetermined hibernation time.

The false-wake costs energy; this energy unit could be significant as everything needed to satisfy item 2 in the previous list must be powered on and used. The negative energy impact sustained from the false-wake gets consumed from the store, further increasing the device's need to re-hibernate and accumulate charge.

Startup Inrush Avoidance

Typical behaviour when initially powering up microcontroller devices, or any kind of discrete device, involves an initial current inrush as the discrete's chosen underlying physical technology traverses past its absolute minimum voltage and current specifications into a startup condition. After this traversal has happened, currents may fall again while the component is waiting for explicit wake commands or configuration. This inrush trait is illustrated in Figure 80 below, which shows a typical startup characteristic from a PIC16F1820 MCU [107]. The yellow trace represents the voltage rise of the devices MCU supply rail, and the green trace represents the current draw. The *B* cursor is positioned at the point where the initial startup minimum conditions have been met and processed. It should be noted that the part was not in an execution state at this point, merely powering up.

79

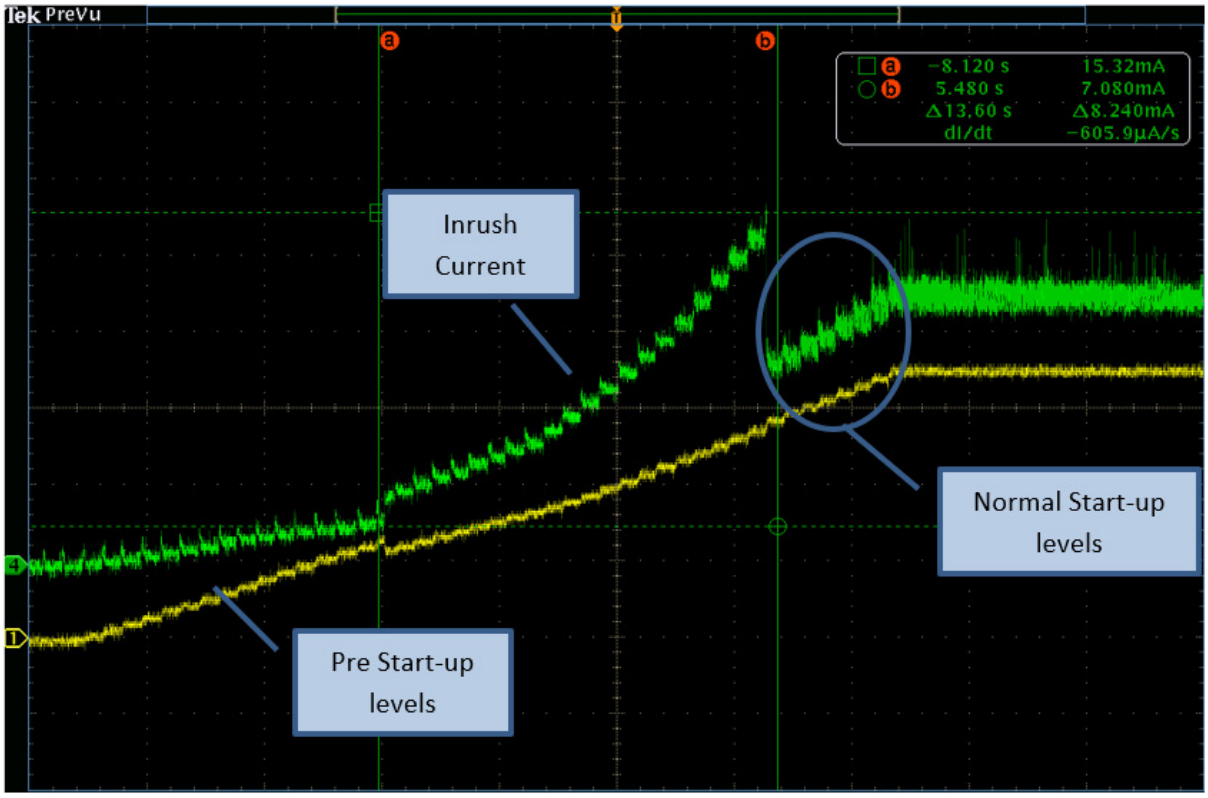


Figure 79: Startup Inrush

Therefore, consideration is needed on how one can reduce as much as possible the need to fund the inrush spike with fresh current supplies. The solution takes two forms depending on the devices design foundation.

If the discrete device is to be entirely powered down, efforts need directing at minimising false-wakes entirely, using prediction and maintained energy reserves.

Suppose the discrete device is to be placed into a specific hibernation mode offered by the part for energy conservation, and its supply is to remain. In that case, efforts need directing at ensuring the device's minimum energy store level never gets breached, allowing the minimum power supply level to be permanently available to the MCU. Figure 81 below illustrates the more consistent consumption demonstrated when the supply voltage fluctuates above the MCU's absolute minimum.

80

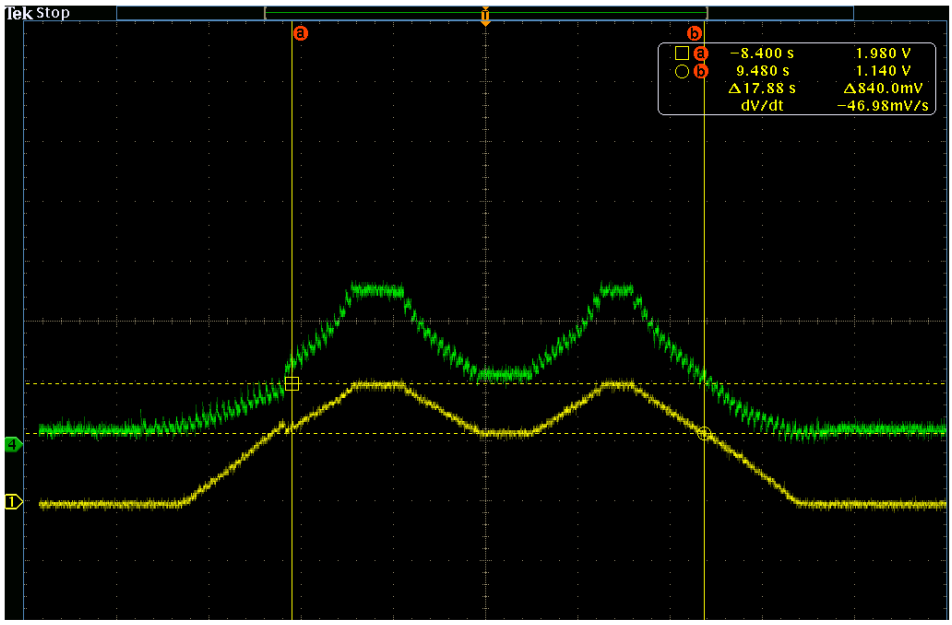


Figure 80: Threshold Sweetspot

Regardless of whether the device's energy store was 1% or 99% insufficient to meet the needs to complete a work unit successfully, the device re-hibernation time will be 100% of the pre-determined wake/work cycle value.

This trait exemplifies the second significant inefficiency with fixed value sleep/wake patterns. The device's existence in itself suggests it has a job it has to do. There are the expected time limits to have the job completed, providing an additional metric for measuring success or failure. The false-wake has not only spent the energy from deciding it had to re-hibernate, but the job completion time frame also takes a negative impact of a size determined by the determined hibernation time it must now also wait.

A real possibility is present of the false-wakes spiralling until the energy store depletes. The incoming energy stream must then increase significantly for the device to overcome the initial energy surge and needs of a potential 'rebirth', as presented in Figure 80.

When the incoming energy stream tends towards being gluttonous in supply, the fixed hibernation period can again prove inefficient. Small dips or losses in the incoming stream can cause sudden energy store level drops, which initiates a hibernation cycle. In this case, regardless of how quickly the energy store replenishes, the hibernation period will need to be entirely spent.

There is also often a job-related reason to hibernate as well to be considered. Sometimes it is not desirable to continuously work, immediately repeating a *task* after its completion or even immediately moving on to the next *task* in hand. Such instances appeal to the hibernation process as a means of spending time in an optimised idle state even if energy charging is not required.

It is also a measure of inefficiency to not be in some kind of deep power-saving state when not needed for work *tasks*. Even as energy scavengers, they must be mindful of wastefulness. Harvesting energy is just that, 'taking' it out of the air, or the heat or the wind. It is not a finite supply in any form. When we consider RF harvesting, we take energy from the airways; this energy was put there initially by someone or something (when considering galactic microwaves). We could well be stealing someone else's power, even in tiny amounts.

Consideration towards other devices needing to 'nab' and harvest little bits of energy floating around is imperative, and each device that feasts will remove a tiny part of this greater stream. Eventually, if enough devices are all drinking from this stream, it will dry up.

If the energy is not needed, it should be left. The device should do everything possible to preserve its energy and hibernate whenever its *task* constraints allow it.

Addressing the inefficiencies associated with time-dilated-wake strategies requires the introduction of specific adaptive measures. These measures influence the hibernation periods

and make decisions weighted more towards their current immediate needs rather than those needs identified and predicted during design-time.

Adaptive Protocols

Adaptive protocols take the time dilatation methods and adjust them dynamically based on other varying and influencing factors.

These kinds of protocols typically perform more efficiently than simple time dilation solutions as they can help avoid the need for 'false-wakes,' which cost energy when waking up too early.

As highlighted in the Time Dilation Based Protocols section above, the most considerable variation in efficiency comes from manipulating the hibernation period variable.

Adapting the hibernation period based on the current environmental surroundings significantly increases the overall efficiency of the device.

Chapter 4: Empirical Research, Phase 1: Energy Harvesting (Original Content) covered an elementary example of manipulating a sleep/wake pattern to minimise false-wakes occurrences.

The essential ingredients needed for this are:

1. A measure of the rate of incoming energy
2. A measure of the current energy store charge
3. A measure of the amount of energy needed to complete a work *task* unit successfully
4. The knowledge of the wake-up decision and current running costs
5. Understanding of the devices charge curve requirements

The most optimum wakeup time is derivable using this information, giving the highest probability of being at the right charge level for the pending *task*.

Even if an influencing environmental factor has changed unknowingly during hibernation, and a false-wake has followed, the loss is manageable. By calculating a quick top-up ‘nap’, the device can minimise the effects of any further wastage.

All of the above-listed items can be easily implemented into a device apart from item 3; it is this area where most of the intelligence gets applied.

Energy Blocking

Devices come in all shapes and sizes and provide one or many functions; each of these functions may or may not need execution at different periods and different recurrence frequencies.

These specific execution schedules cannot be easily constrained and described in a clean, straightforward expression. The size or complexity of any given feature requires quantifying in some form.

In terms of block-diagrams, the device's features become its *jobs*; these *jobs* allow further sub-division down into units representing the ‘smallest amount of useful work’, known as *tasks*.

Tasks represent ‘energy block units’ which move through a processing cycle:

81

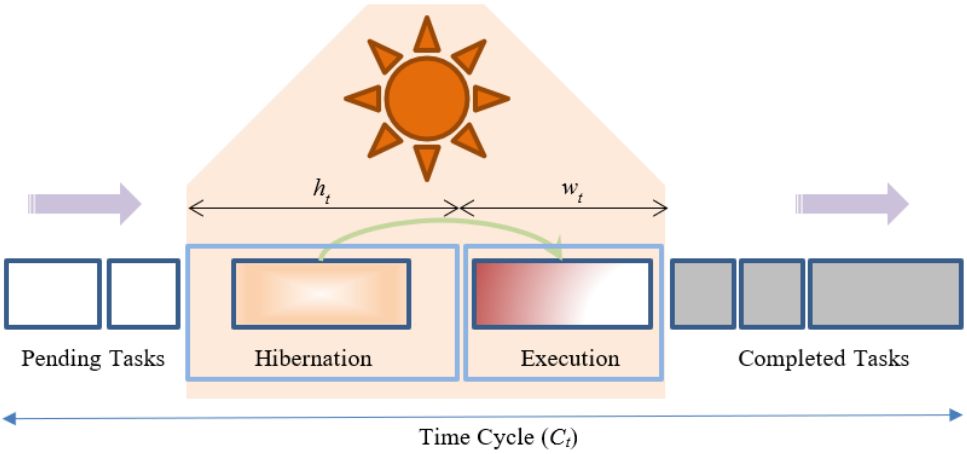


Figure 81: Typical Adaptive Cycle

Figure 82 above shows the relationship time dilation has with regards to the current input energy stream. In this case, the LPDs *job* breaks down into its smallest useful *tasks* sized according to its energy consumption prediction. The *Hibernation* period reflects the time it takes to harvest enough energy to fill the pending *task-block*.

At this point, the *Execution* stage activates, and the *task* consumes the required energy.

When the input energy rate decreases, the *Hibernation* period increases, all other variables retain their existing values, as shown in Figure 83 below.

82

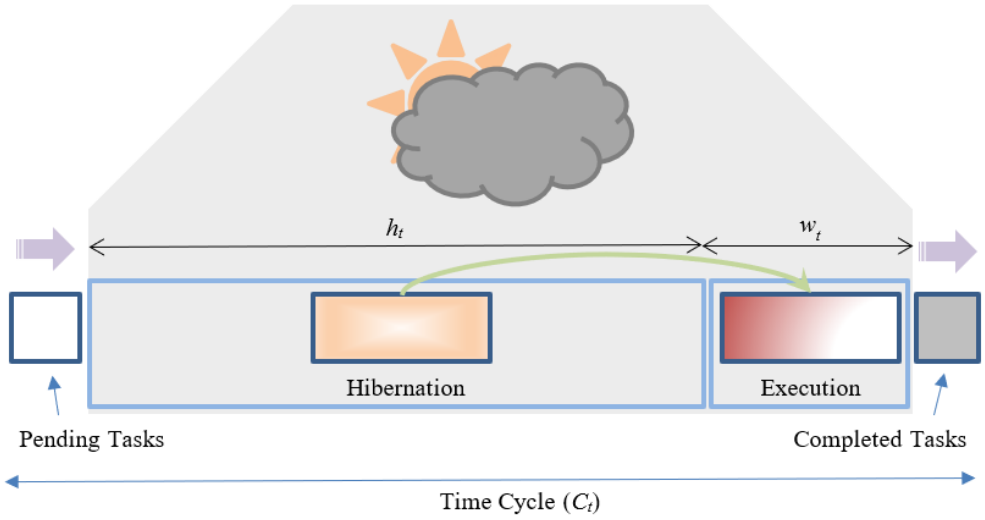


Figure 82: Time Dilated Cycle

Both Figure 82 and Figure 83 share the same C_t period, the difference being that Figure 82 represents better efficiency when considering the output of useful work (or completed *jobs*) from the period.

Quantifying the energy in terms of *task* consumption allows the wake pattern to be dynamically adjusted depending on the awaiting *task*'s predicted energy cost.

The *jobs* and their *tasks* are known beforehand and represent one, or many of the devices feature sets. Each *task* has a predicted energy consumption cost previously determined, and the device is aware of these costs.

A protocol uses this information to provide some kind of work schedule and *task* queuing system to manage the *job's* execution.

The relationship between the *tasks* is one of tight coupling. Any change in the incoming energy supply (item 1 from the list) reflects immediately in all of the other list items.

Item 1 is considered a random variable that must be tolerated within the LPDs design, all of the rules for time dilation apply here, which introduces a time restriction element around the work schedule.

It is clear then, to properly benefit from adaptive power management, there must be a protocol able to deal with the following distinguishing attributes:

- An energy measuring subsystem capable of providing and managing a level monitor sensor for the energy store.
- A *task* information database; detailing known *tasks* and associated costs in terms of energy consumption.
- A *task schedular*; which manages the work *tasks* passing through the device.
- A device manager; responsible for the hibernation duration algorithms.

83

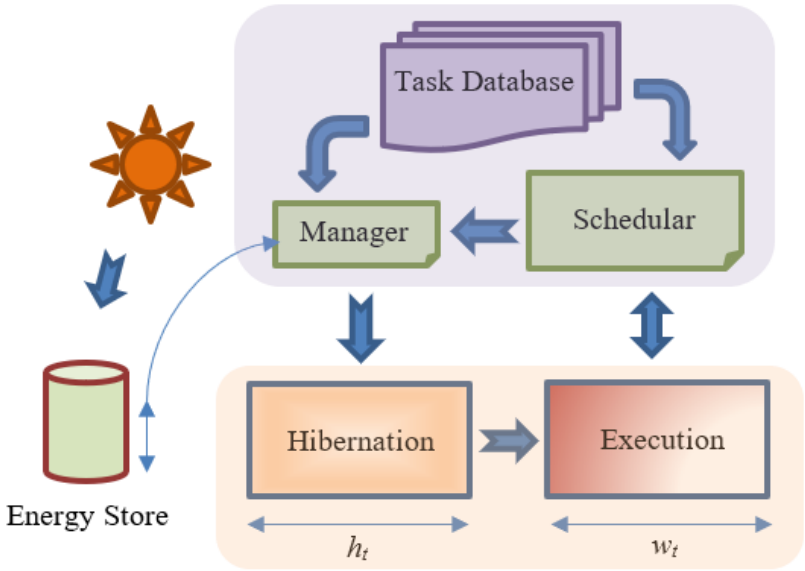


Figure 83: Protocol Module Outline

In addition to these essential services illustrated above in Figure 84, various supporting offerings should also be considered, such as measuring a *tasks'* actual consumption and a mechanism allowing for the introduction of new *tasks* to the device.

To provide a complete system, inter-device communication and user interaction also need to be considered and properly managed.

A protocol stack enables the implementation of these identified subsystems at the hardware's lowest level.

It surfaces the discussion that "any kind of energy prediction could significantly benefit from various AI methods and calculations"; however, the consumption requirements, space, processing power and storage for AI type learning data quickly render the energy savings gained irrelevant. Much simpler adaptive measures benefitting from regular feedback and self-correction loops proves substantially adequate.

The aim is to use significantly less energy than the savings its implementation can provide, be small enough not to impact device hardware characteristics, and be efficient enough not to drain any CPU resource and interfere with regular operation.

WIMP (Walton Intelligent Management Protocol) Protocol Stack

The WIMP stack attempts to blend the worlds of both time dilation and adaptive energy management protocols to provide an autonomous encapsulated platform capable of supporting the higher-level application.

The stack is made aware of the dynamic inputs of the device it is operating within by its implementor. During operation, the stack provides the application with notifications of when to sleep and for how long, what *task* to run next, and how to deal with any false-wakes gracefully.

As expected, the implementor must design in the intelligence to execute the *tasks* at hand.

Typically, an individual stack implementation operates on every microprocessor equipped with energy-saving features found within a device. However, designs where a *task* may use other microprocessors or controllers without possessing the stack's underlying knowledge to aid in completing their *task* are also operable. They will, however, be unable to exhibit the full potential of the energy management system.

The *Calibrator* implementation discussed below automatically determines a particular *task's* consumption cost by performing a run-and-measure type calibration cycle. This autonomy aids in achieving necessary feedback adjustments computed during operation in the field.

When designing protocols that largely control the availability of the device they are working within, one must also consider the design's communications aspects. Wake patterns and message exchanges must attain tight synchronisation in some way.

Basic Operation

The stack itself integrates with the higher-level application using a simple set of signals interfaced within the *Task-Receptor* service.

Two-way messaging is achieved through the *Task-Receptor* calling an application function and passing it a standard set of parameters:

- *taskExecute(task)*
- *taskTerminate(task)*
- *SampleEnergyStore()*
- *Hibernate(status, duration)*

It is critically important that the *Hibernate* function immediately puts the device into a deep sleep state. Any further execution or energy expenditure between the Hibernate request and actual hibernation will incur greater false-wake probabilities. If further processing is needed here to process any rejected or failed *tasks*, this process time should be kept as constant as possible and incurred regardless of the status or return attribute. The more this time can be made deterministic; the more efficient the algorithms will become. Figure 85 below highlights this critical timing area.

84

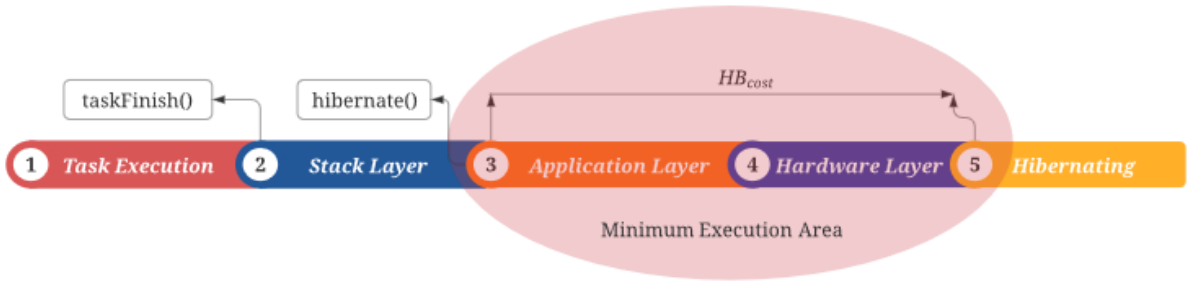


Figure 84: Minimum Execution Area

When the execution of a *task* has ended, and execution control gets passed back into the stack, the *Hibernate* signal gets generated. This signal allows information to be passed back to the application software, indicating the result of the *task's* execution. If a calibration or rejection result has occurred, it will be included here as part of the status value.

The application can either retry failed or rejected *tasks* or prepare alternative *tasks* designed to deal with rejections and failures.

The application will respond to or initiate requests by calling the *Task-Receptor* Service *Signal(message)* function, where the message attribute can take the form of:

- *Woken*
- *taskFinish*
- *taskError*
- *taskCalibrate*
- *taskAdd(priority,…)*
- *taskRemove*
- *Idle*
- *Tick*

Similar to the *Hibernate* function, the *Woken* signal needs sending as soon as reasonably possible after the device has spent its hibernation period.

The *taskFinish* and *taskError* signals occur appropriately as soon as the application has completed or decided it has given up on its current *task* execution.

There is also a current status retrieval facility that will return the *task* the stack is currently expecting to execute.

A primary message diagram (as shown in Figure 86 below) illustrates the intended flow of messages in an ideal operating environment.

85

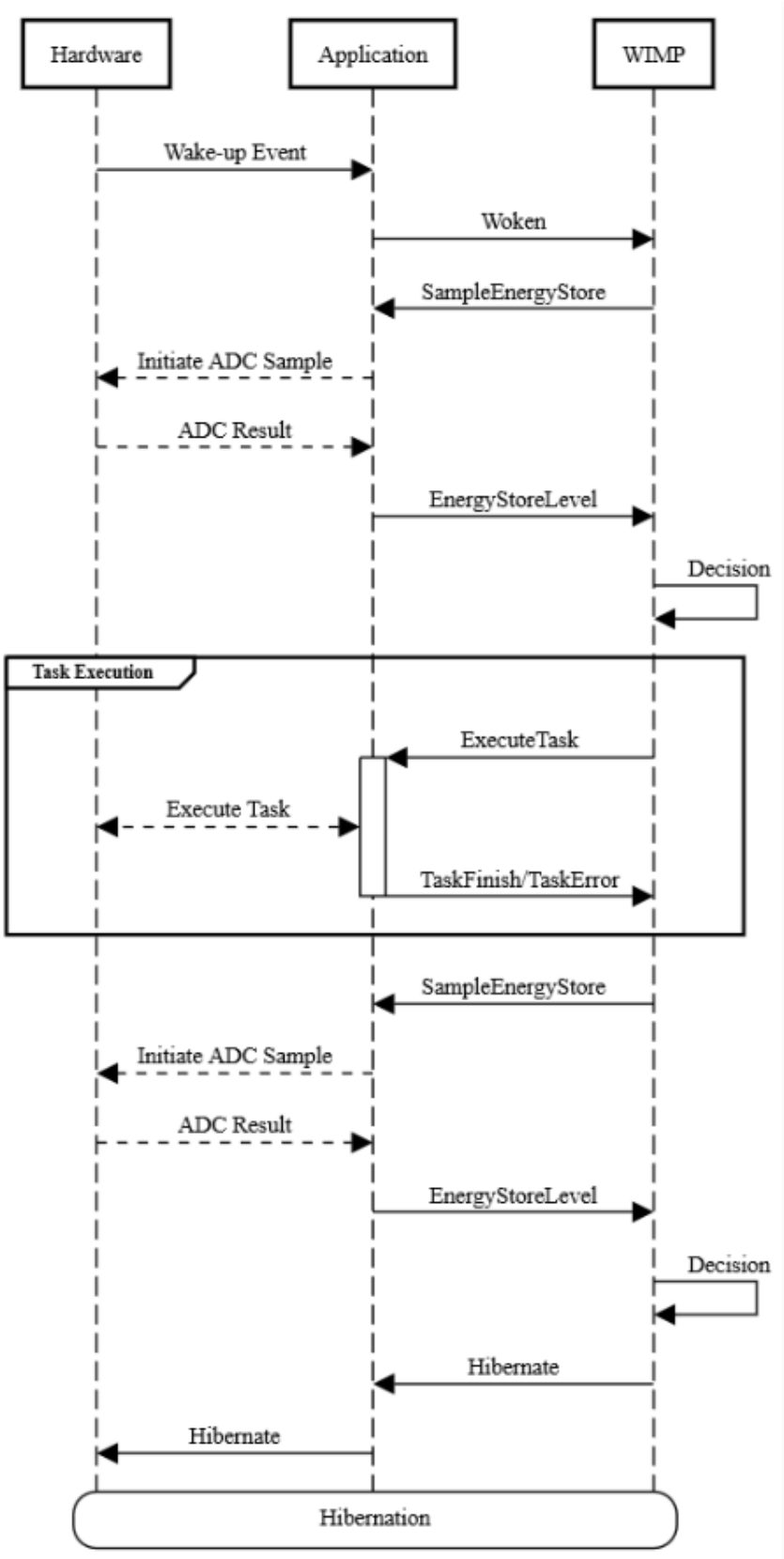


Figure 85: Message Flow Overview

The stack does not know the contents of the *task* it will be scheduling. It does not have any direct input into the execution and completion of *tasks*. These processes are controlled entirely by the application software and hardware. The stack only knows a prediction of how much energy the *task* will consume, and the only real control it wields is requesting to the application software what *tasks* start at what time. However, it can request to the application that a *task's* process terminates due to it not adhering to the energy predictions expected of it.

Figure 86 shows an ideal operating environment after all *tasks* have been calibrated; it highlights the energy level sample points both before and after the *task's* execution. The decision to work or sleep is coupled directly with the result of the measurement.

An energy level sample will always immediately follow a wake event and be performed just before entering a state of hibernation.

Primary Modules and Consumption

The entire process divides up into three distinct subsystems:

1. Wakeup/Hibernate management
2. Decision-Making management
3. *Task* execution management

Each of these subsystems will consume energy, and stack operation will commonly see pairs of these subsystems working together. This commonality allows the categorisation of the consumption parts into measurable units that are deterministic in design, for example:

- A wake-up event will always consume *Wake Cost* + *Decision Cost*.
- A hibernation event will always consume *Decision Cost* + *Hibernate Cost.*

Additional *Decision Cost* deductions happen when a *task* reaches completion; however, if this decision is to initiate hibernation instead of continued *task* execution, then the last *Decision Cost* becomes part of the following *Hibernate Cost*, as two successive energy level store readings serve no purpose.

Hibernation itself also consumes energy. Although markedly reduced compared to *task* execution costs, it can be challenging to ensure it remains deterministic. A lot of this consumption will be from various component leakage characteristics and environmental influences, coupled with device components experiencing continuous temperature, light and humidity changes, all of which have additional knock-on effects.

Predictable Losses

Resisters, diodes, and capacitors are common causes for the lack of precise long-term deterministic time properties and exhibiting varying energy consumptions. Physics is at play here; these components' primary characteristics relate to their operating temperatures and conditions.

Suppose we take resistance; its ability to resist accurately at its specified value fluctuates with relation to its operating temperature. Ohms Law proves resistance is directly related to current with *V/R;* in that case, we can immediately comprehend that environmental temperature changes will affect the individual components' resistance, which will affect the amount of current flowing through them. This potentially extra current can only be satisfied from the energy store.

These tiny fluctuating changes may not be a problem for the device's *tasks*, and indeed in most cases are designed to average themselves out, but they will have a combined effect, small or large, on the energy predictions and the accuracy of how deterministic they are.

The tolerance choices of these components during design time have an enormous impact during attempts to model the devices' potential energy usage. Wherever possible, during the low-current design stages, these types of considerations need to be addressed, especially when considering long-term effects as discussed by Zhai, Zhou and Ye when they looked at a tolerance design method for electronic circuits based on performance degradation [123].

Energy Storage

Energy store implementation takes the form of a capacitor, and again the leakage of the capacitor must be understood at design time. Supercapacitors usually offer much smaller loss levels over much more extended periods than exhibited by traditional capacitors [124].

Another point of tolerance consideration would be for oscillators and resonators. Every logic system needs some kind of clock, and this clock will determine all types of timing inaccuracies which in turn have both positive and negative effects on current consumption.

Oscillator and resonator start-up times and accuracy can be influenced considerably by temperature and temperature change.

These leakages and tolerances are essential contributions to the accurate energy consumption predictions required for proper stack execution. Allowances introduced at every part of the energy management process for the combined tolerance requirement provide resilience against randomness. The running total of all these allowed tolerances, presented by all the hardware components, increases the consumption prediction. However, incurring a false-wake due to over-tight tolerance calculations will inevitably cost more than having tiny amounts of surplus energy leftover from successful *task* execution, which would then be fed back into the system for reuse.

Allowances for the differences in consumption potentially caused by environmental influences needs addressing. These types of interference are tough to predict for both strength and time, as such a percentage increase of the consumption prediction provides a mechanism to offer reserve.

Test runs found that slight over predictions given to the deterministic *tasks* had minimal effect on overall efficiency measurements. This trait is due to the regular energy store level samples allowing rescheduling of the surplus energy reserved for remaining pending *tasks*. However, causing a false-wake has the opposite effect and induces a comparably more extended recovery period to replenish the wasted energy.

Wake Cost	WK_{cost}
Decision Cost	D_{cost}
Hibernate Cost	HB_{cost}
Hibernation Cost	Z_{cost}
Execution Cost	X_{cost}
Tolerance Cost Addition	TOL_{cost}
Tick Cost	TK_{cost}
Task Predicted Cost	J_{cost}

Table 4: Variable Mapping

Equation 48

$$\text{Startup Cost} \rightarrow S_{cost} = (WK_{cost} + D_{cost}) \times TOL_{cost}$$

Equation 49

$$\text{Sleep Cost} \rightarrow E_{cost} = (D_{cost} + HB_{cost}) \times TOL_{cost}$$

A typical execution cycle CYC_{cost} will contain

Equation 50

$$CYC_{cost} = S_{cost} + X_{cost} + E_{cost}$$

86

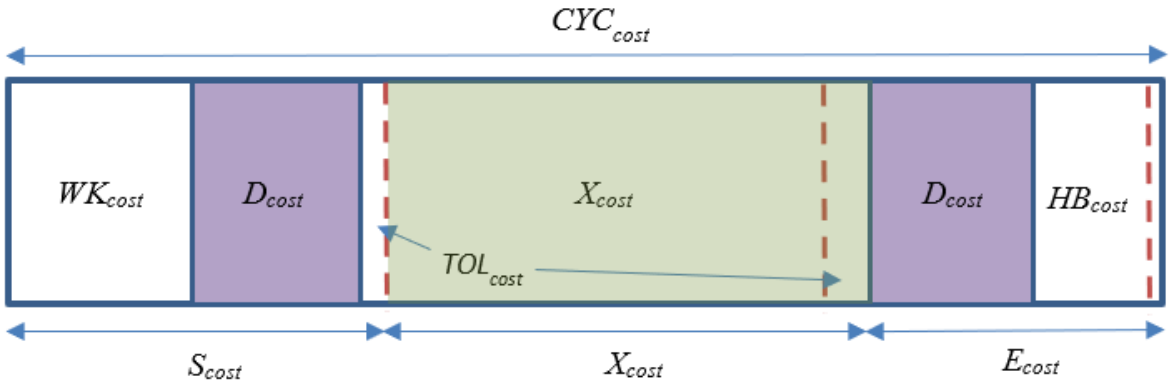


Figure 86: CYCcost Breakdown

Figure 87 above shows how a CYC_{cost} breaks down into its sub-parts; the dashed red vertical lines represent the allowances added to the consumption predictions for tolerance and environmental influences.

X_{cost} gets further broken down into its sub-parts, as shown in Figure 88 below.

Depending on how many *tasks* the device plans to execute, which will depend on how much energy it has accumulated in the store, an X_{cost} will consist of at least one J_{cost}.

X_{cost} may have many J_{cost} parts and include some *Ticks* to monitor the real-time energy situation; this will cost one or more TK_{cost} units, mathematically represented as:

Equation 51

$$X_{cost} = \sum_{n=1}^{N}((J_{cost}[n] + TOL_{cost}[n]) + \sum_{m=0}^{Jcost[n]:M} TK_{cost}[m])$$

87

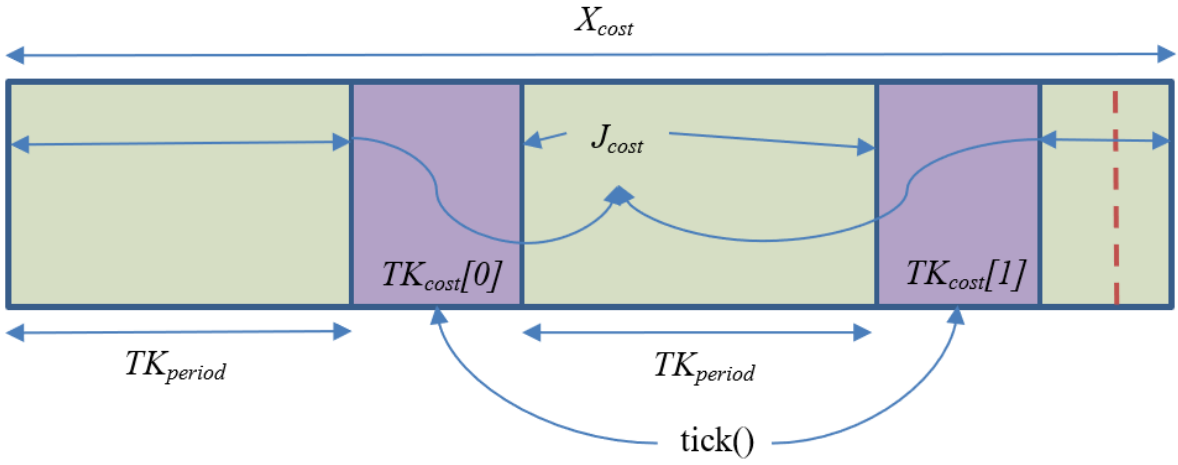


Figure 87: Xcost Breakdown

Figure 88 above shows the most straightforward X_{cost} setup consisting of a single *job* J_{cost}, and two *Tick* calls. The *Ticks* are regular, and their interval is application-specific.

An X_{cost} can have many J_{costs} within it; Figure 89 below illustrates this point visually. This figure has a *Tick* event that falls between the finish of $J_{cost}[0]$ and the start of $J_{cost}[1]$. This *Tick* event can also provide the sample reading data to decide on further *task* execution or hibernation.

88

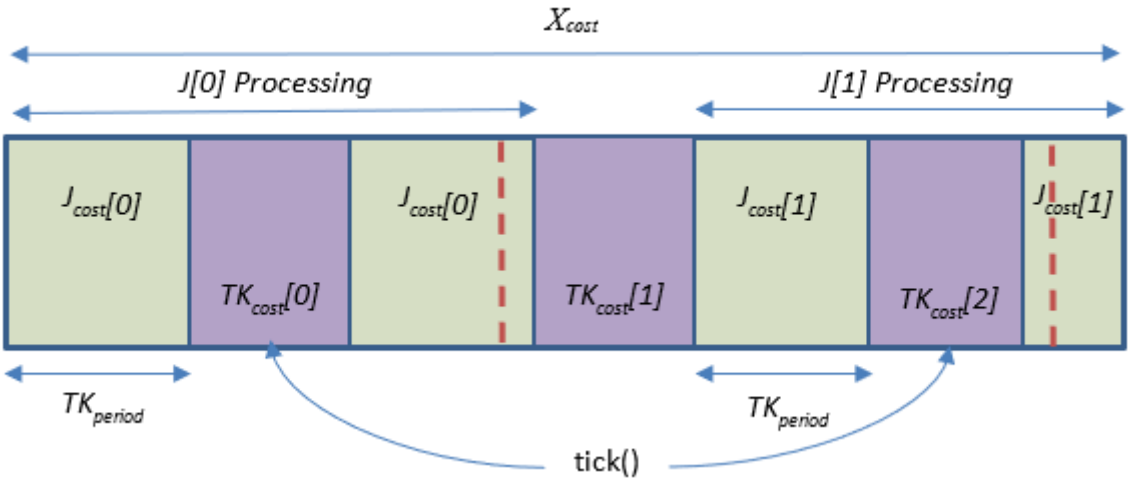


Figure 88: Multiple Jcosts

Most of these identified consumption costs can easily be self-discovered at runtime, barring an accurate measurement of WK_{cost} and HB_{cost}. The correct procedure for measuring these two parameters involves using some high quality current sensitive measurement equipment. The point at which the measurement gets made is critical, and the following guidelines describe the optimum sequence:

- For $WK_{cost,}$ the measurement must be started at the exact point the devices' primary clocking system starts and will end when the device is up to operating speed, stable, and the application software has made the *woken()* call into the WIMP stack (Figure 90 below).
- For $HB_{cost,}$ the measurement starts when the *hibernate()* request was made from the stack and ends when every clocking system within the device has finished their winddown sequences and settled into their lowest consumption settings (Figure 91 below).

89

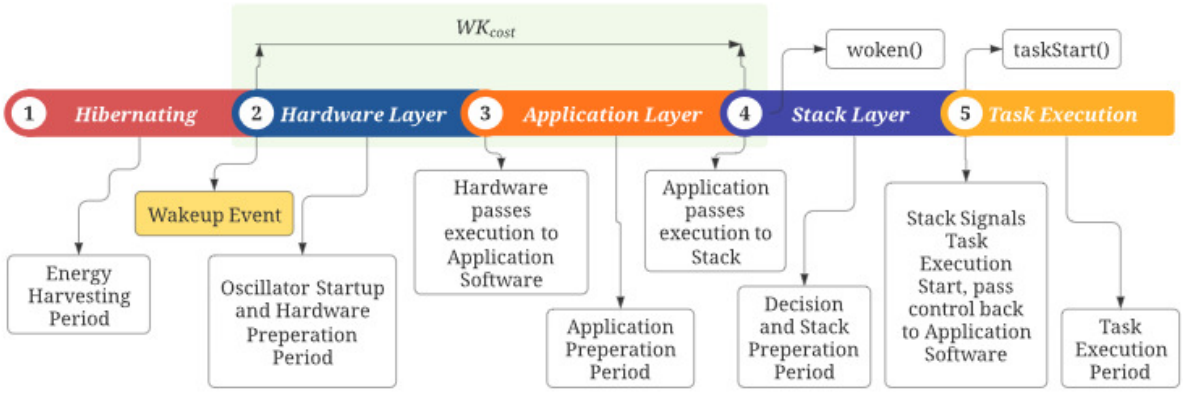


Figure 89: WKcost Measure Points

90

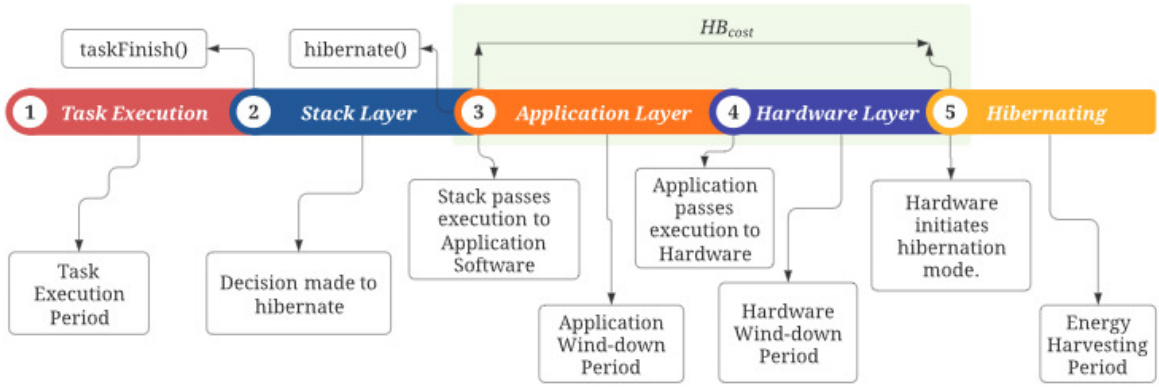


Figure 90: HBcost Measure Points

Figure 92 and Figure 93 below illustrates the various costs grouped around the specific stack features they support and how they flow through the modules.

The boxing around the functions represents the types of services the stack expects execution for during that period.

91

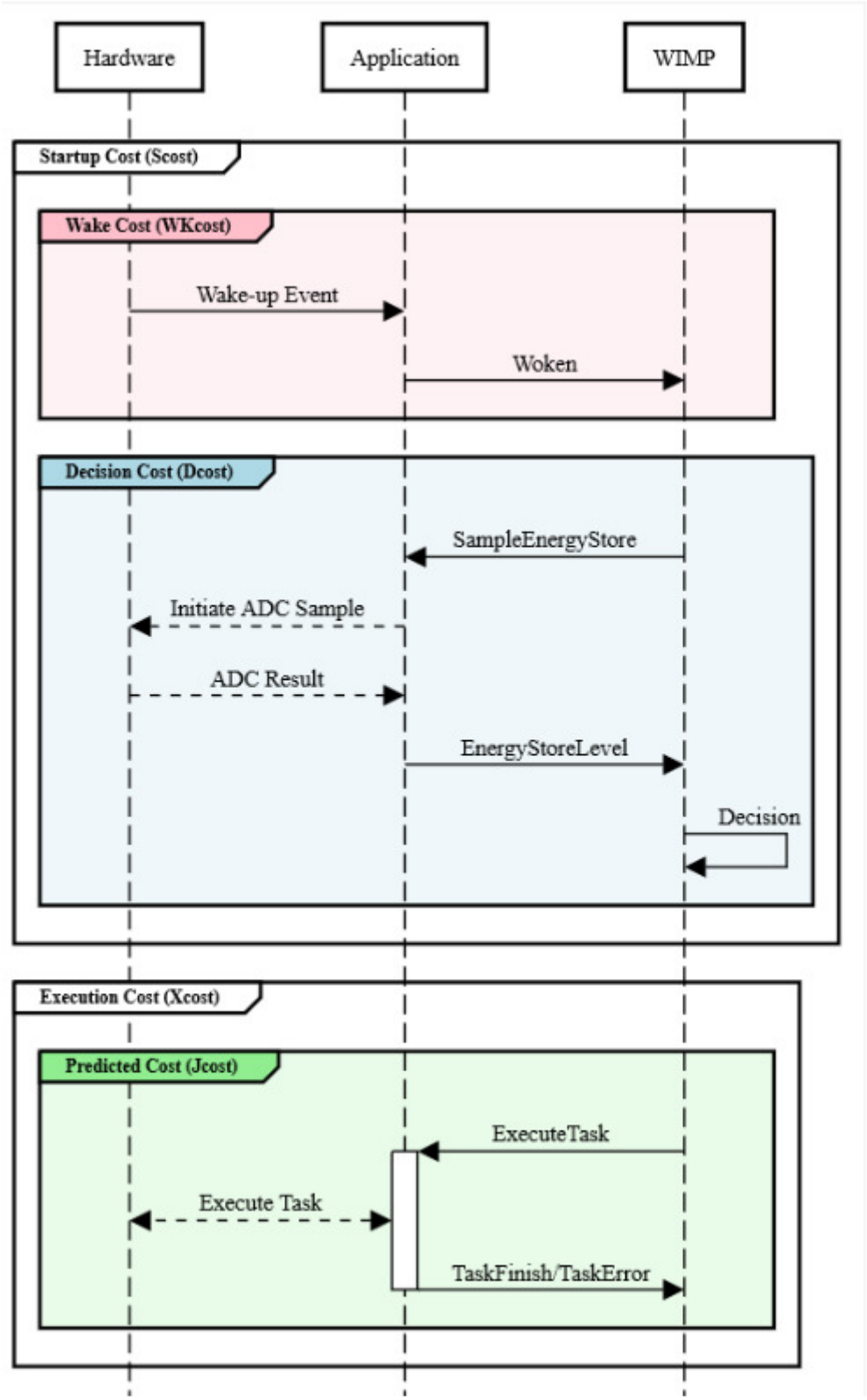


Figure 91: Costed Message Flow (Part A)

92

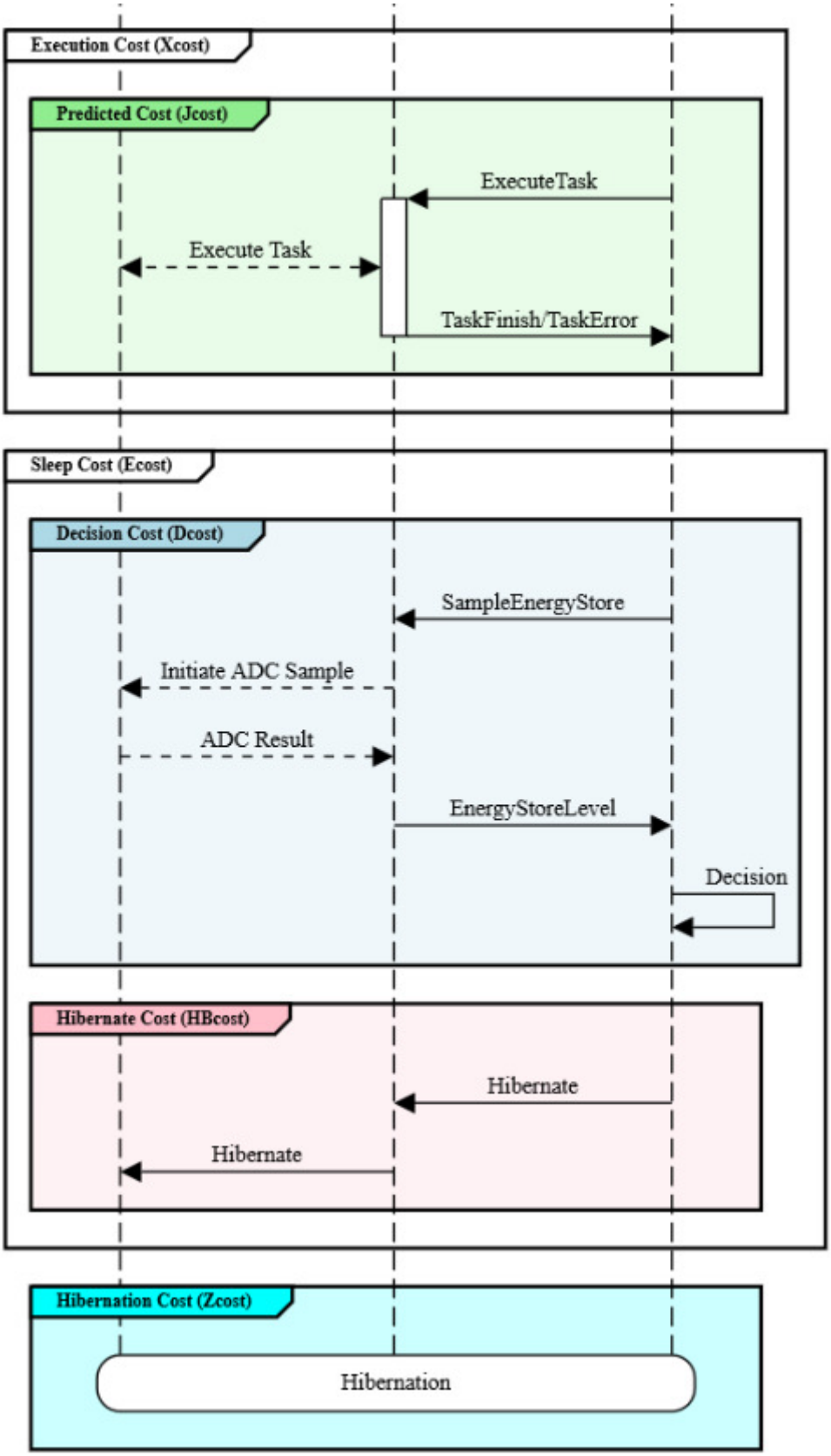


Figure 92: Costed Message Flow (Part B)

Level Offsets and Topologies

The stack can make an offset adjustment to every energy store level sample it makes, and this allows the application to control how the stack 'sees' the contents of the store. It can enforce a minimum store level that is always kept and considered untouchable by the stack.

Alongside this, a 'full' level variable to control the maximum amount of level the stack sees is available, and this allows multiple instances of the stack to run concurrently as part of a bigger system; yet at the same time enabling the sharing of the same energy store.

Each implementation of the stack can be configured only to use a portion of a single larger store via their primary energy stores. The communal energy store must feed into the smaller individual primary energy stores so the stack can monitor its actual usage accurately, feed this back into its adjustment algorithms, and detect any consumption overruns (Figure 94 below).

93

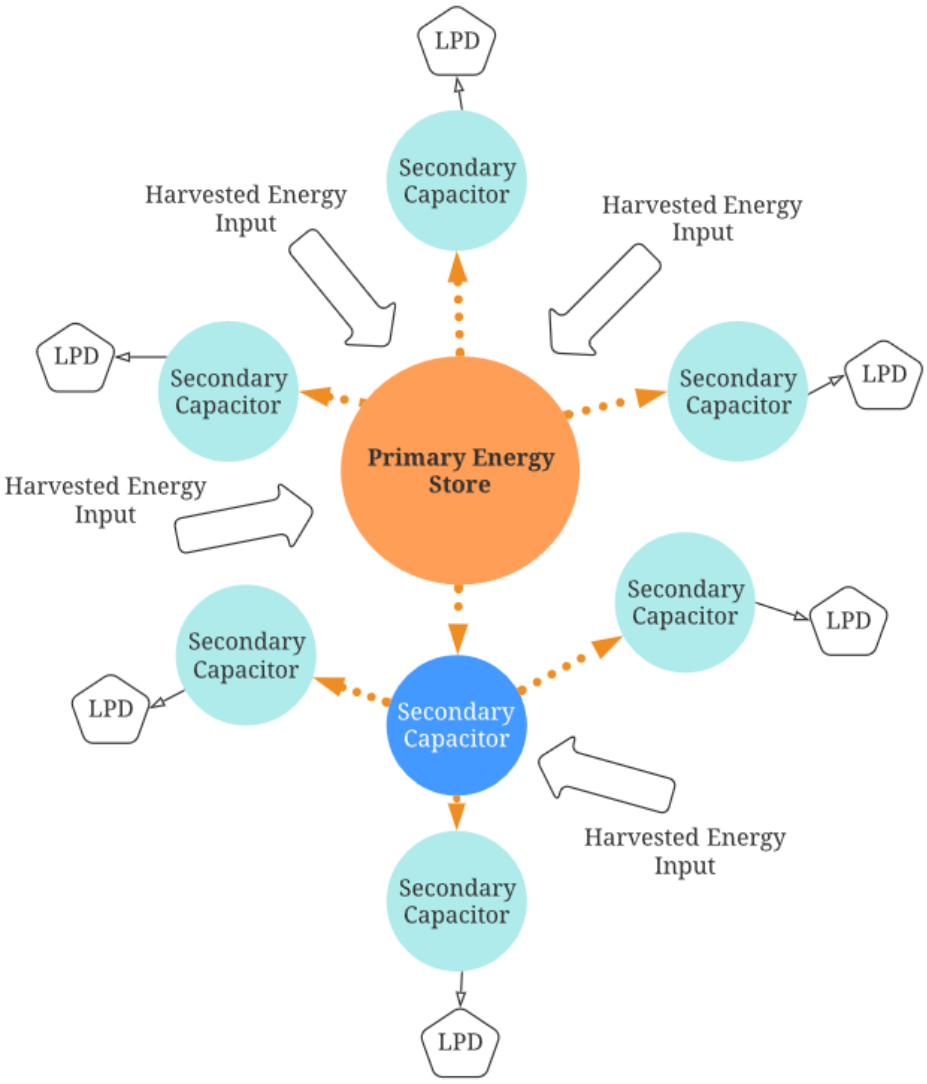


Figure 93: Distribution Topology

This setup is easily implementable and, in most cases, follows acceptable bulk-capacitance requirement design practices.

In these types of energy distribution topology, the communal energy store can do a secondary job of smoothing the incoming harvested energy and then distributing the accumulation either equally or by a resisted priority-based supply setup to all of the individual subsystems.

Initial commissioning of this shared store topology type is slightly more involved because the individual subsystems need a temporary external energy source for accurate consumption measurements, commissioning and initial calibration. Some kind of supply isolation is needed.

There is a consideration in the usefulness of an intelligent ‘marshalling’ type of device to sit between the primary energy store and sub-energy stores and wielding power to decide which subsystem gets what energy and at what rate (Figure 95 below illustrates this topology). After some brief modelling, it showed the consumption cost outweighing the benefits provided and any intelligence in this area would only provide distribution management, not energy reductions.

94

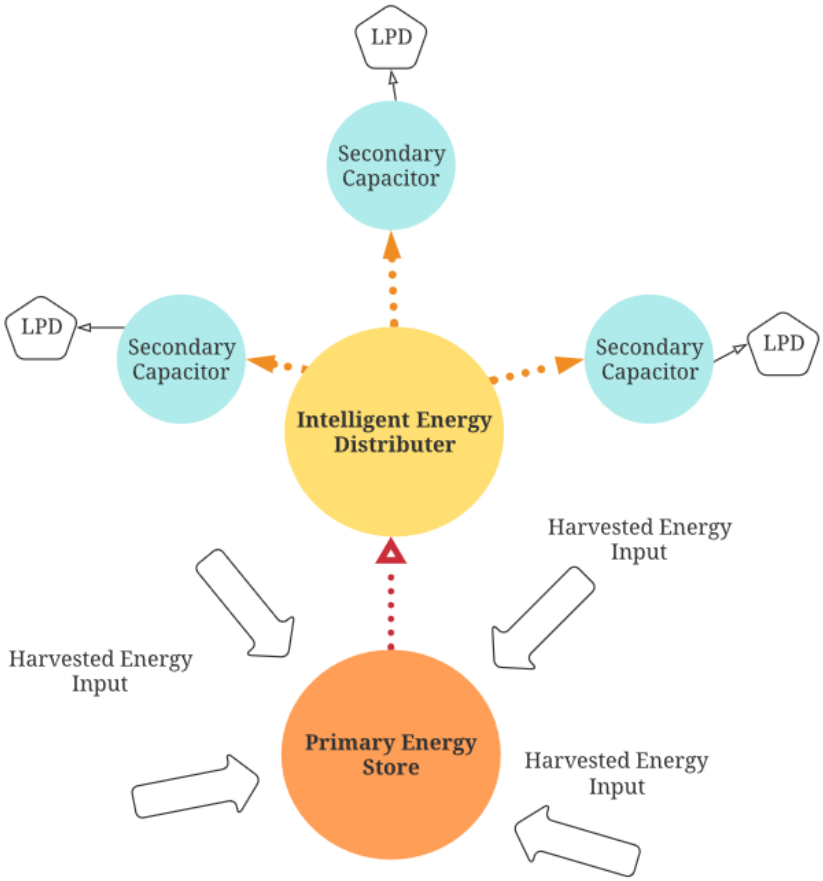


Figure 94: Intelligent Energy Distribution

A simple equal energy distribution path performs as expected; the overall input rate is the combined efforts of all available energy harvesters. As each subsystem executes its *tasks*, their different individual consumption rates cause the stacks in each subsystem to start adapting differently.

When device communication is part of the design requirements, this type of topology can efficiently synchronise wake-up and communicate type messaging protocols.

Internal Status Monitoring

Once all the *tasks* have been assigned and calibrated, the device will enter a sleep/work routine, adapting dynamically to satisfy as best as possible its input energy stream against its output *task* execution rate.

The *tick* function manages a relatively slow interval clock for the stack to progress its internal monitoring functions. When the device is awake and executing *tasks*, this *tick* function is called by the application software at regular intervals, using a period specific to the devices desired responsiveness and requirements. The primary need for this interval is to progress the status monitoring features implemented by the *Power-Trender* service and prevent a *task* unknowingly feasting from the energy store.

Figure 96 below shows a message flow diagram for executing a single *task* and pausing the execution at regular intervals to service the tick call sent from the application software. It shows that every *tick* call will trigger an energy level sample via the underlying hardware subsystem; this sample's result becomes the supporting data for the pending sleep/wake decision that will then occur.

95

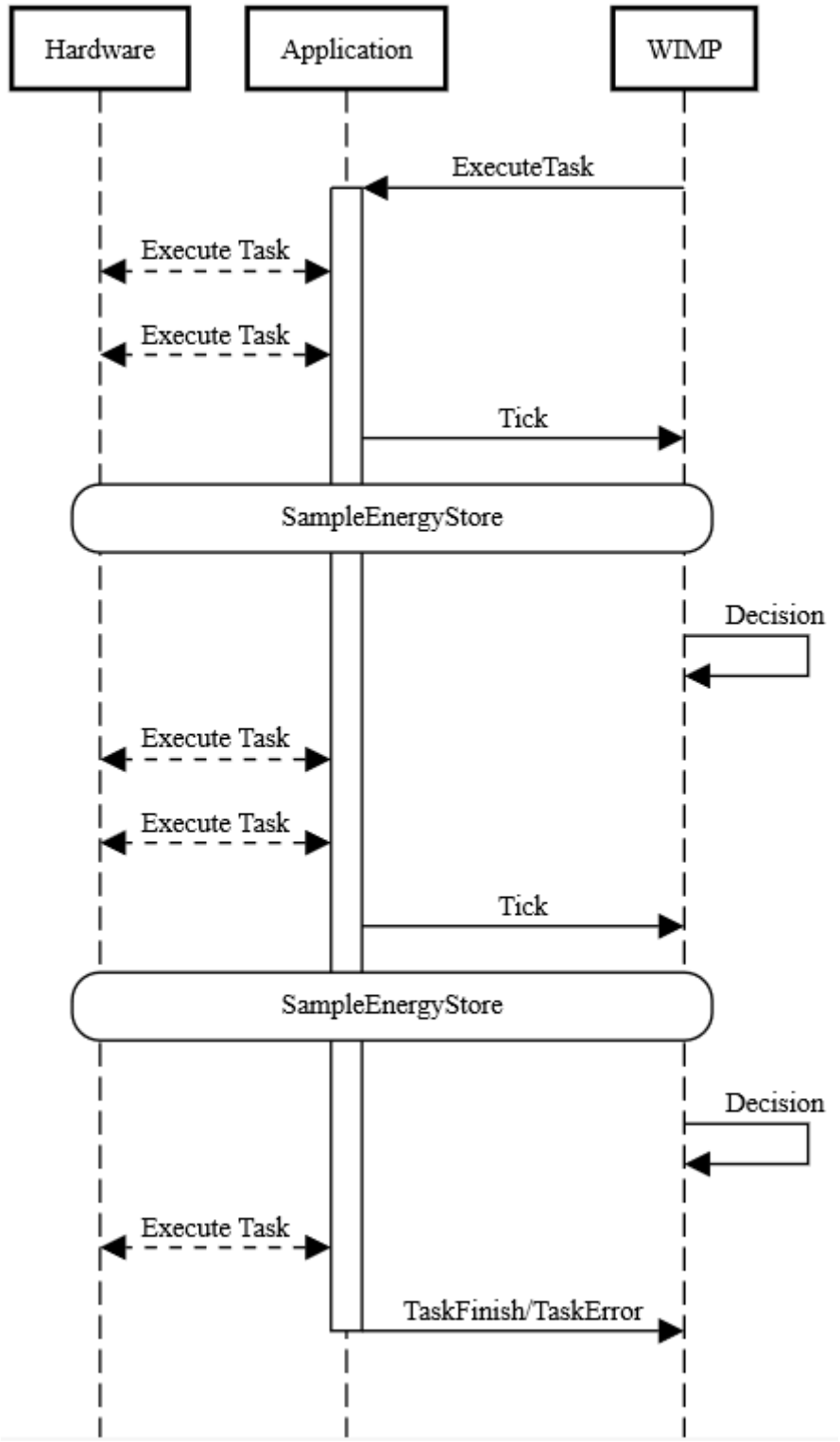


Figure 95: Execution Flow with Ticks

The *tick* interval is chosen carefully to find a balance suitable for the application. Calling *tick* too frequently will cause overhead consumption to increase. However, the stack will detect and intercept any consumption overrun situation quickly and gracefully deal with the problem incurring minimum wastage of remaining energy.

Slower intervals are more efficient but risk a more considerable energy loss during an overrun event before detection. This setup type is best suited for devices that guarantee that their *tasks'* energy predictions will be accurate.

Removing the *tick* altogether is also an option but will be heavily application dependant and will remove the ability to perform priority-based scheduling and communication synchronisation.

Figure 97 below illustrates what parts of the message exchange are managed by TK_{cost}, and how they fall within the Task-Execution periods, The CYC_{cost} is extended due to every TK_{cost}, demonstrating the importance of careful consideration for the *tick* period decision.

96

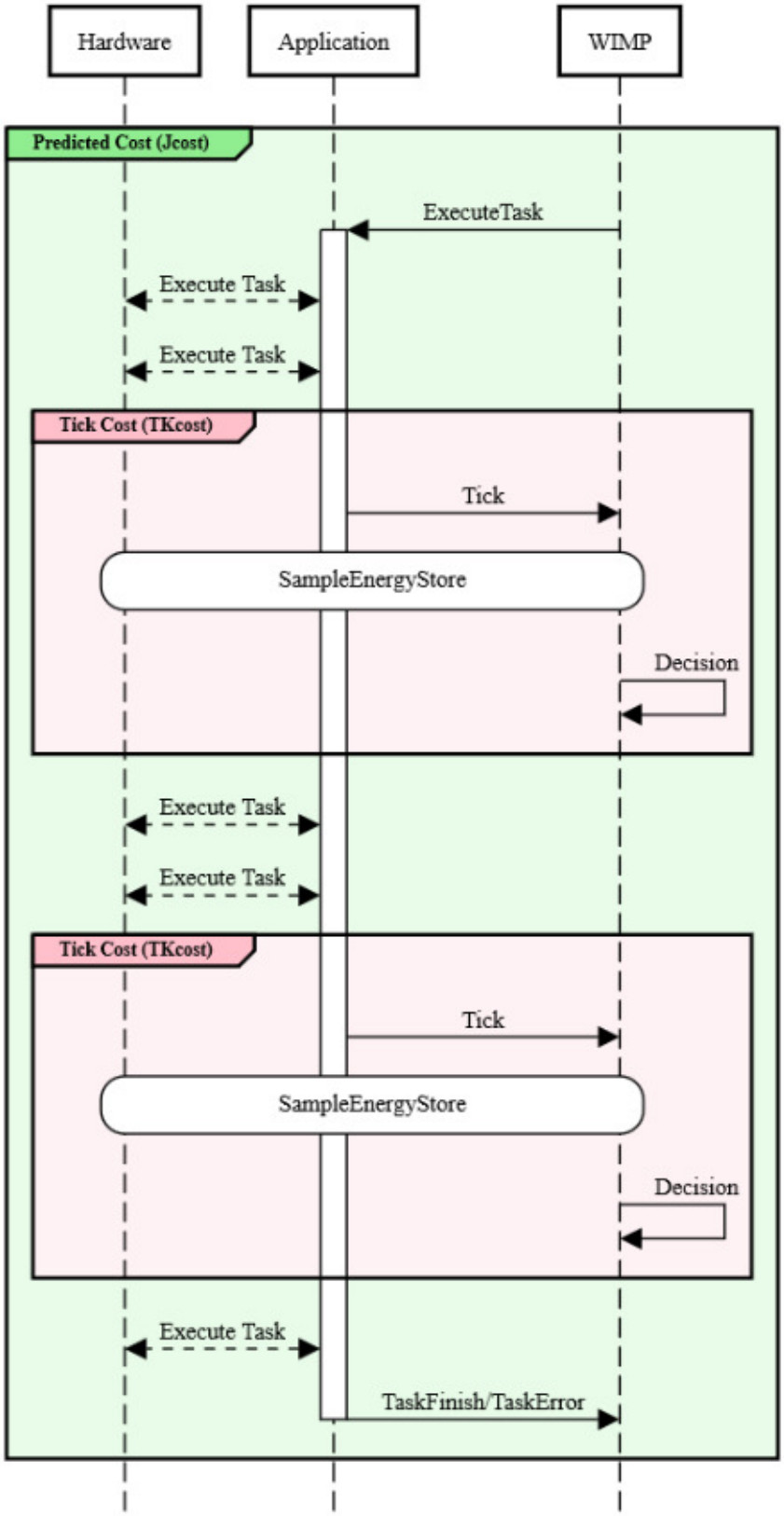


Figure 96: Tick Costing

Decision to Work or Sleep

The *Decision* block represents the stack choosing between executing a *task* or hibernating. Although based on numerous factors, the decision itself mainly follows the adaptive wake calculations outlined in the Sleep Strategies Chapters.

The other influencing terms driving the decision include the devices pending *tasks*, the current energy store level, the next scheduled *task's* predicted costs, and the average incoming energy rate.

Having a regular clock signal like this also enables the stack to obey time scheduling limits to the planned *tasks* currently on the queue.

When a *task* gets added to the stack, it also accepts two scheduler influencing parameters, priority and a TTL (Time-To-Live) value. This TTL value represents the maximum allowed number of *ticks* (or multiples of) that can pass before this *task* must complete its execution.

The *priority* represents how quickly (or urgently) the new *task* should be scheduled for operation in relation to the other currently queued events.

Zero represents the highest priority *task*, and 255 the lowest. Multiple *tasks* can share the same priority. Suppose the planner is currently scheduling *tasks* of an importance level that represents multiple possible *task* choices. The choice then of how best to utilise the available work slot involves evaluating combinations of *task* energy consumption prediction values.

If there is no way of completing the last *tasks* of the highest priority together in one work time slot, but a combination of all but one *task* at the highest priority and a *task* from the next highest priority, this is considered an acceptable choice only if explicitly allowed via stack option configuration. When enforcing priority-based execution, the surplus energy is used to reduce the impending hibernation period.

The TTL is only valid for non-zero priorities, and its operation will raise the priority of the *task* as the preconfigured time limit reduces.

The TTL operation principle is that its value gets divided by the priority value passed with the *taskAdd* request. This calculation results in a *JumpCount* value representing the number of *ticks* that will pass before the *task's* priority gets upgraded to the next level. At this point, the *task* parameters get reevaluated.

Equation 52

$$JumpCount = \frac{TTL}{Priority}$$

When the *task's* priority reaches zero, the *TTL* becomes irrelevant as the *task* at this point will have the highest possibility of being scheduled alongside the other *tasks* about to gain service. The only other influencing factor for the schedular to consider is the *task's* energy consumption prediction size and how it can best get scheduled for execution, ultimately allowing its removal from the stack.

TTL usage affects the natural-priority and balance of intended *task* execution; thus, considering the overall *task* flow is critical when assigning its values. If *tasks* are being added to the device dynamically during runtime, this TTL will ensure that any previous *tasks* will not be pushed back for unacceptable amounts of time.

The *JumpCount* must undergo sleep period correction, which leads to the possibility of a *tasks* TTL expiring during a long hibernation period. This limitation is acceptable because the predefined delay is adjustable for this possibility and the cost of processing the extra intelligence of dealing with the corner case is not worth the gains.

Stack Services

The stack itself gets broken down into several different services which encapsulate both required and optional modules. The interface is simplistic in design for the sole reason of operating within energy constraints far smaller than those it saves.

Figure 98 below shows the general outline of the services and how they integrate. The stack consists of seven modular services working together to provide an execution management system that overviews the work expected to be done by the application software.

97

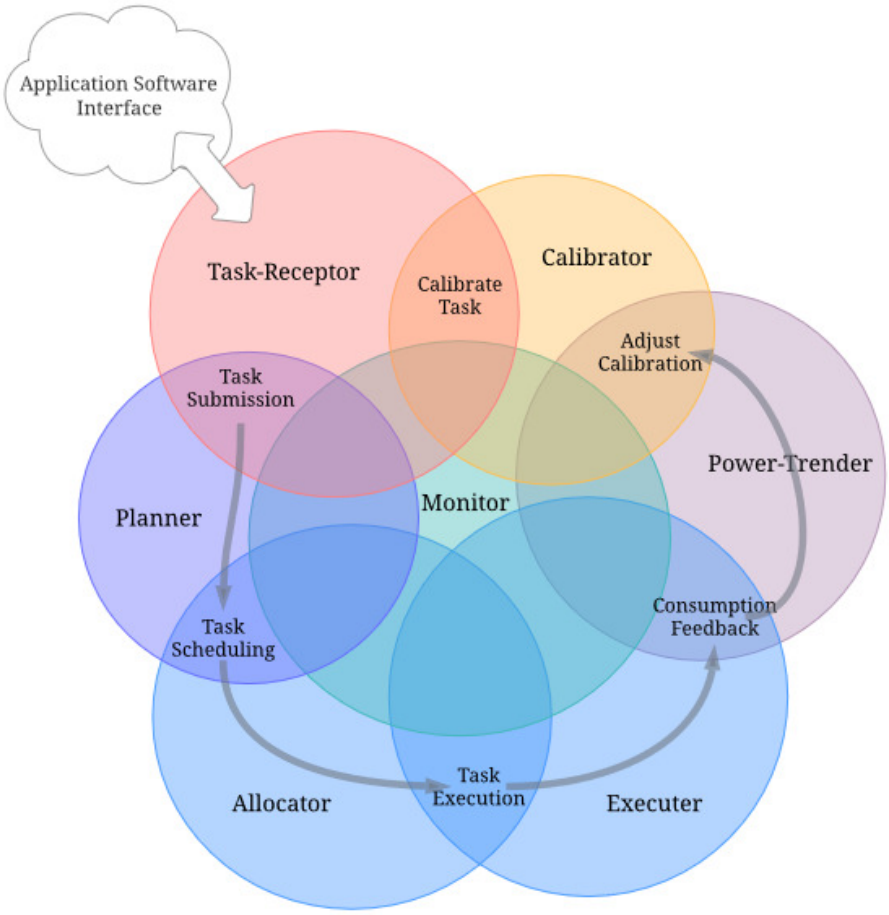


Figure 97: Stack Services Overview

- **Calibrator**: Provides an interface to measure and maintain *tasks* actual energy consumption characteristics against a common scale.
- **Task-Receptor**: Surfaces an interface to the application software, allowing *task* management message exchanges, hibernation control and result feedback.
- **Power-Trender**: Measures and maintains real-time energy consumption.
- **Planner**: Organises incoming *tasks* against priorities, requirements and restrictions.
- **Allocator**: Controls the flow of the planned *tasks*, assisted by a schedular, through the executor.
- **Executer**: Executes the *task*, keeps track of it and processes its end status.
- **Monitor**: Provides logging and data collection facilities allowing analysis of the stacks operating efficiency.

Calibrator Service

The Calibrator module dramatically expands the flexibility of which *tasks* (jobs) the device can and cannot perform. It allows the device to measure and normalise every *task's* expected consumption rates against a unified scale. It moves the complication of providing the *task* consumption predictions from design-time into run-time, creating more autonomy.

The simplest solution to energy costings is to pre-document the consumption costs of all the *tasks* the device will support and provision the device with this dataset during the design phase. However, there are a few corner-case situations that have the potential to cause field failure:

1. Execution of the *task* exhibits considerable consumption difference from the predicted cost due to unexpected environmental impacts.
2. Receiving a new *task* while operational in the field; of which it has no prior knowledge of consumption requirements.

Environmental impacts have to be taken into account as they could significantly change the amount of energy needed to complete the *task*. These impacts could be down to weather, terrain, light levels, distances, and any other influencing factor that will cause the *task* currently executing to consume additional energy.

This change in consumption cost for a *task* will impact the planning and scheduling of it and the other *tasks* the device has to maintain. A feedback loop is needed, so tolerances for the differences between planned usage and actual usage reconcile; a separate Power-Trender service provides this feature set.

However, if the change's impact is more significant than frequent planning tweaks can satisfy whilst in its current environment, a mechanism has been proposed to recalibrate the *task* in-situ.

In the case of recalibration, the device will perform the *task* in its entirety in its current environment.

The successful execution of the calibration cycle must have access to more energy than the *task* is expected to use to acquire a completion status. If the *task's* rates are truly unknown, the energy store charge level must be maximum before calibration executes.

98

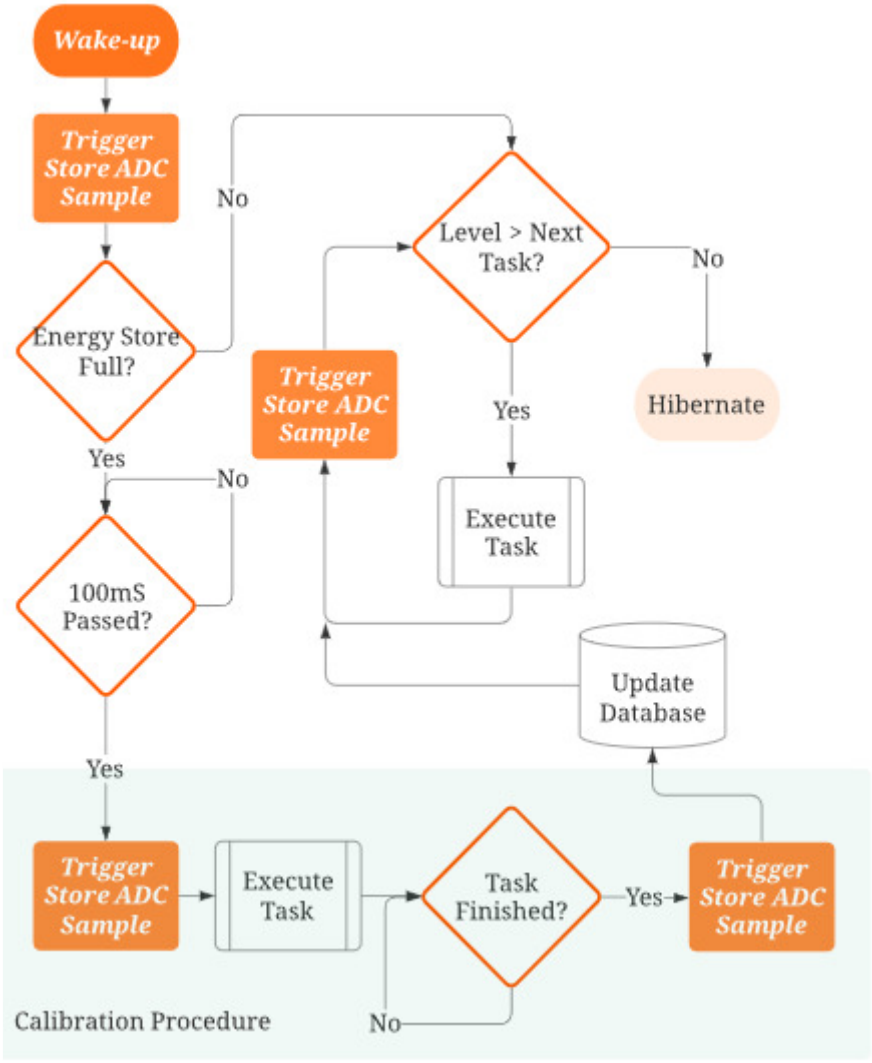


Figure 98: Calibration Procedure

The calibration sequence shown in Figure 99 above illustrates the following steps:

1. Wake-up and check energy store level requirements are satisfied
2. Wait a predetermined amount of time
3. Take an energy store sample level
4. Signal application to execute the *task*
5. Wait for the application to signal successful *task* completion

6. Take an energy store sample level
7. Update the *task* operating specifications of the *task* to the new value
8. Hibernate or continue further *task* execution as appropriate

Suppose the *Calibrator* cannot finish the *task* execution with the energy constraints the device has given it. In that case, the *task* calibration will fail, and either the previous prediction value gets retained, or in the case of a new unknown *task*, the *task* gets entirely rejected.

Using the C*alibrator* service nullifies the need to measure and provide *task* predicted consumption costs during the device commissioning stage. It allows the device to be passed a *task* and do a self-calibration within itself before its initial deployment, thus mostly aiding the production and preparation stages. Prior knowledge of the maximum possible consumption of a *task* must still be estimated. This method should cater to worst-case scenarios as it only enforces control after consuming the most energy the designer permits for that *task* ever to have.

The benefits of performing this initial self-calibration run during commissioning include supplying the device with an alternative permanent power source, thus removing the fluctuating incoming energy stream additions from the harvesting systems during the calibration process.

The *Calibration* service also has its minimum costs in terms of energy consumption. It must also pass the extra parameter of maximum energy allowed for a *task*, and this is the value that allows the device to determine calibration success or failure for the *task* and schedule the calibration routine based on its energy needs.

Initiation of the calibration procedure is through the *Task-Receptor* service. Automatic *task* recalibration can be forced by the system when a running *task* gets forcibly terminated due to a significant overrun of its predicted energy consumption.

Self-Calibration

As an internal function, the *Calibrator* is also used to measure the stack itself's consumption costs. By following a predefined self-calibration procedure, It can self discover consumption predictions for:

- Hibernating and waking up for a known interval
- Sampling the energy store
- Scheduling *tasks*
- The cost of a false-wake

The internal calibration routine gets run during commissioning and has access to an area of non-volatile storage where these variables can be stored and retrieved. The routine steps through each of the required measurements using dummy *task* profiles requiring the application to signal *task* completion at specific time durations.

99

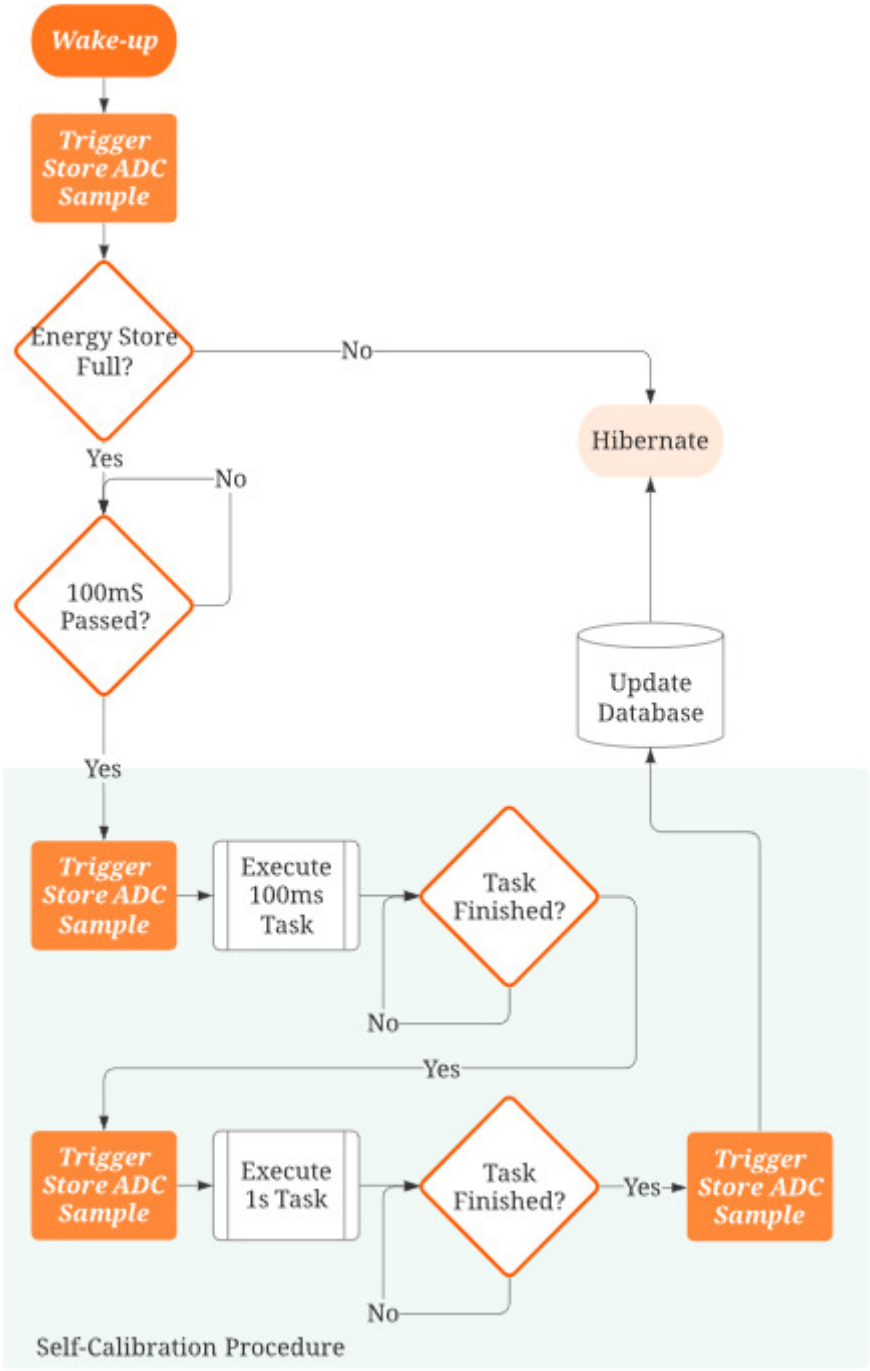


Figure 99: Self-Calibration Loop

The first step of the self-calibration routine, as illustrated in Figure 100 above is to measure the current rate of incoming energy. This rate measurement involves issuing two consecutive

SampleEnergyStore requests spaced at a predefined duration, assuming that this incoming energy rate will not change significantly until the end of the calibration cycle.

The need to characterise the incoming energy stream at this point is due to the continual charging element it places on the energy store. If manual intervention is available, then removing the incoming energy supply before the process begins can provide the most accurate results, as long as the store can hold a charge large enough to achieve *task* completion.

100

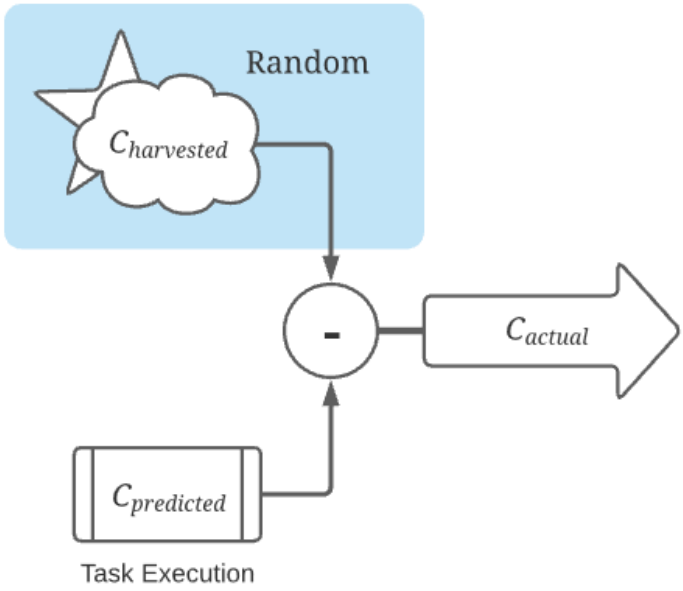


Figure 100: Cactual Flow

Figure 101 above illustrates the constant charging effect. The currently executing *task* will be consuming from the store ($C_{predicted}$) while the incoming energy flow is trying to charge it ($C_{harvested}$), making it very hard to measure the *task* consumption in isolation accurately.

Equation 53

$$C_{actual} = C_{predicted} - C_{harvested}$$

If $C_{harvested}$ is deterministic, the cost measurement can be adjusted to account for it. If this is a problem, running the analysis multiple times with incoming energy stream measurements performed in between, then averaging the result can smooth out any significant discrepancies delivered from the energy harvesting systems. The effects of increasing accuracy beyond this are considered diminished returns. As the stack can adapt to differences and tweak its parameters as it runs, it can self-mitigate many initial calibration errors over time.

This method's secondary benefit is the accurate costing of the application's overhead consumed when initiating a *task* and signalling back to the application software regarding completion.

Figure 100 above illustrates this function's principal idea; it consists of executing a *task* called 100ms and a *task* called 1s, which both get scheduled and consecutively signalled to start by the stack.

The application must then wait a time duration in a loop of either 100ms or 1s, depending on the executing *task*. After the period, the application must immediately signal to the stack that *task* completion has been successful.

By sampling the energy store levels at specific times during the process, enough data gets collected to enable stack profiling of its interface requirements and consumptions.

The aim is to internally self-model the *platform* the stack is implemented on and then make the most accurate hibernation and schedule predictions for its *tasks*.

It is clear from the context that the need to obtain an accurate measure of the energy store's current level is critical to the stack's operation, and this requirement is solely a hardware-related one that needs resolving during the design process.

Battery level monitoring and battery management are already widespread in devices on the market, and most will provide the ability to pass their primary energy store level in some form

or another. The complication in implementing the stack is that accurate measurement of the current energy level is directly related to the interface's efficient operation.

Energy Store Level Monitoring

At its simplest form, an energy store voltage measurement gets sampled and converted into the digital domain. This process will likely involve an ADC (Analog to Digital Converter) device quantising and digitising its analogue input level signal.

It is at this point that the design decisions must be analysed carefully. ADC peripherals have various limitations which constrain their performance characteristics. The three variables which have the most impact are:

1. Resolution of the ADC in the form of bits
2. Noise and bit-error characteristics of the module
3. The reference voltage source(s) associated with the ADC module

The ADC module's resolution will directly impact how efficiently the stack is able to utilise every drop of incoming energy.

When considering the ADC's incoming voltage level, the stack must also be aware of the 'Empty' sample value and the 'Full', which may well fall between two arbitrary values. This dynamic range will already reduce the effective bit rate offered by the ADC module.

The ADC's bit width determines the quantisation steps possible between the modules' negative and positive reference points. This working window, in turn, dictates the size of the smallest unit of energy the stack would be able to manipulate effectively. The quantisation step process over a rising voltage is illustrated in Figure 102 below.

101

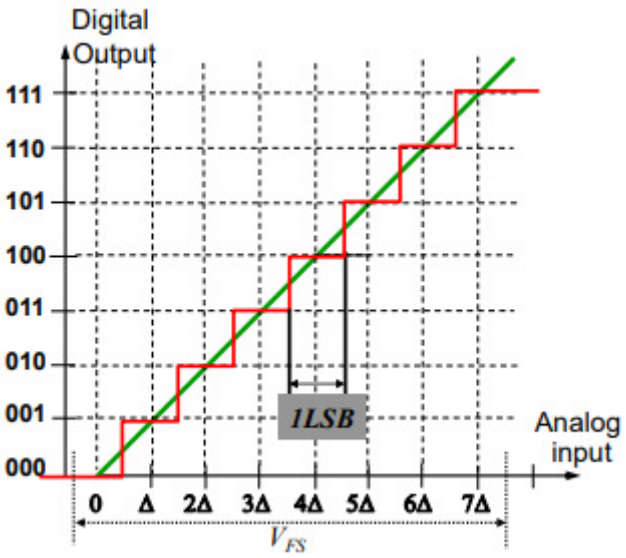


Figure 101: Quantisation Step Rise

Taking a particularly poor example shows how the effects of the incoming signals' dynamic range not satisfactorily fitting the window provided between the positive and negative references coupled to the ADC module decrease the module's effective bit rate. Adding to this the inadequate amount of bits available to do reasonable quantisation, the energy level precision quickly deteriorates.

Minimum Store Level	ST_{min}	0	V
Maximum Store Level	ST_{max}	10	V
Current Store Level	LVL_{cur}	var	V
Minimum Voltage Level	LVL_{min}	2	V
Maximum Voltage Level	LVL_{max}	8	V
Negative ADC Reference	ADC_{neg}	0	V
Positive ADC Reference	ADC_{pos}	1	V
ADC Bit Width	ADC_{width}	8	bits

Table 5: Test Variable Values

For simplicity, let us assume some very high-value resistors are used in a divider network to drop the input voltage level down to one that will fall between the reference voltage window of 0 – 1V. The equivalent circuit is drawn below in Figure 103.

Equation 54

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

102

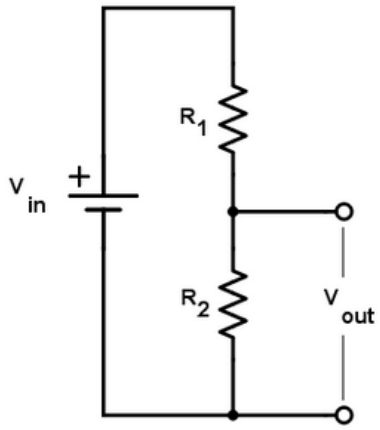


Figure 102: Basic Voltage Divider

Giving R_1 12 MΩ and R_2 1.33 MΩ, a 10V V_{in} will give roughly 0.9977V at V_{out}

LVL_{min} and LVL_{max} define the usable region of the energy store for which the services will work. Figure 104 below illustrates the breakdown of the store levels.

103

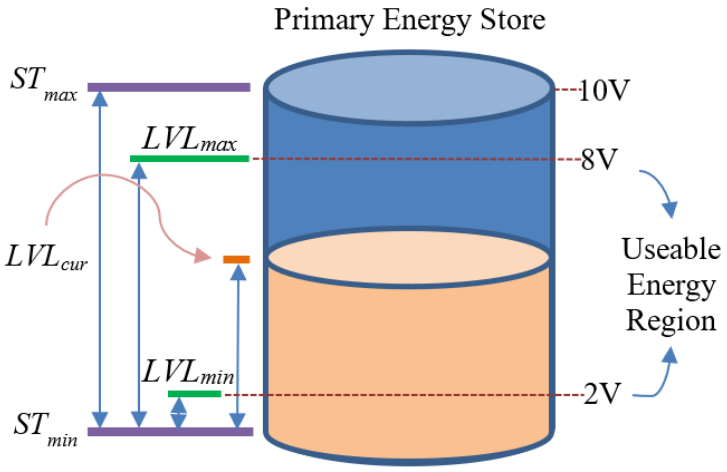


Figure 103: Energy Store Level Detail

The ADC device divides the 0-1V operating window, denoted by the potential difference between the negative and positive ADC references, into 256 quantisation steps. In this particular case, each quantisation step can detect a change between levels as small as 0.003906V.

Equation 55

$$ST_{step} = \frac{(ADC_{pos} - ADC_{neg})}{ADC_{steps}}$$

The example case's LVL_{min} level sample value will be 51.2 and LVL_{max} 204.8, giving us 153 useful quantisation steps over the 2-8V level range available for use by the stack.

In *task* execution terms, this equates to the smallest *task* the system can most efficiently handle, which consumes energy in measures that fit neatly in blocks of this size. Figure 105 below shows the variable mappings provided by the ADC device for the test case.

104

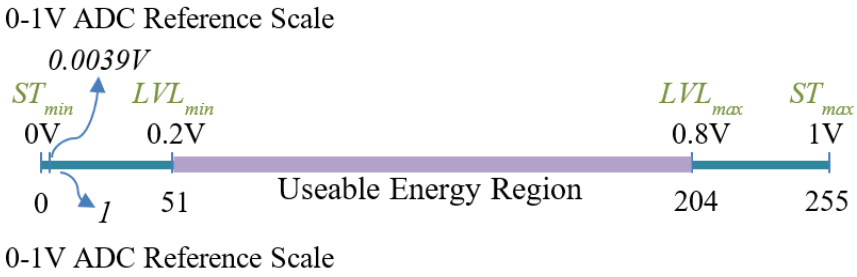


Figure 104: 8bit ADC Values

Increasing the bit resolution of the ADC to just 10 bits (ADC_{steps} = 1024) changes the smallest measurable unit (ST_{step}) down to 0.011V, providing a significant increase in accuracy and performance; Figure 106 below shows the ADC value difference such a change would incur.

105

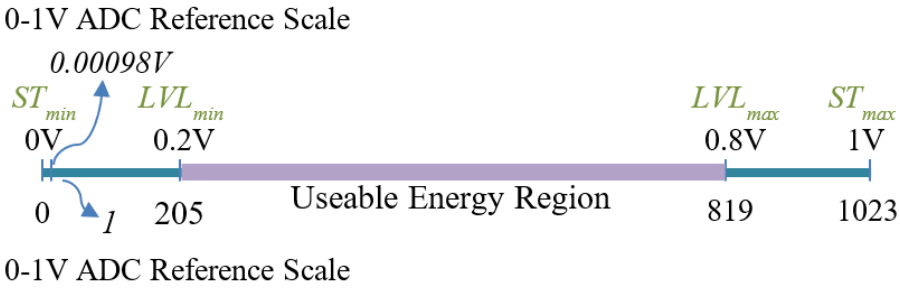


Figure 105: 10bit ADC Values

To further any measurement error present, the ADC module itself will have a noise tolerance level which will also need factoring into any calculation, typically measured in (loss of) bits.

The quantisation noise level of the ADC and its SNR (signal-to-noise ratio) are the characteristics that significantly govern the effectiveness of the transfer function from the

analogue domain to the digital. Figure 107 below shows an illustration representing the best possible transfer function [125]:

106

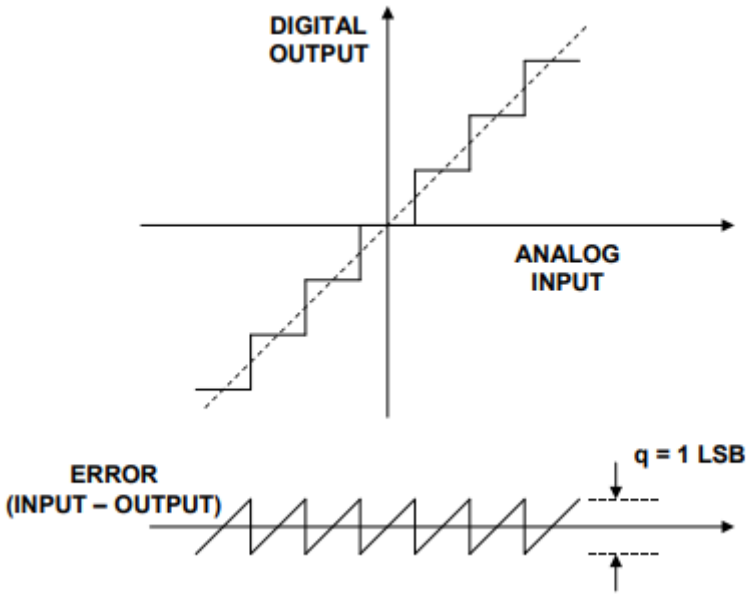


Figure 106: Ideal Quantisation Noise

In this ideal model, any value of error is equally likely, creating a uniform distribution range of:

$$-\frac{Q}{2}\ to + \frac{Q}{2}$$

The quantisation error becomes:

Equation 56

$$V_{qnoise} = \frac{Q}{\sqrt{12}}$$

The relation between the bits of the digital representation and the originating signal is expressed as the SNR (Signal to Noise Ratio). Assuming the full dynamic range of the difference between

the positive and negative ADC references being used to represent a sinusoidal signal whose peak-to-peak value equals the ADC reference levels, its RMS value is:

Equation 57

$$V_{rms} = \frac{V_{ref}}{\sqrt[2]{2}} = \frac{2^N Q}{\sqrt[2]{2}}$$

In this equation, *N* represents the number of ADC bits, and *Q* represents each Quantum.

The SNR is the RMS of the input signal dived by the RMS of the quantisation noise:

Equation 58

$$SNR = 20\log\left(\frac{V_{rms}}{V_{qnoise}}\right) = 20\log\left(\frac{\frac{2^N Q}{\sqrt[2]{2}}}{\frac{Q}{\sqrt{12}}}\right) = 20\log\left(\frac{2^N\sqrt{12}}{\sqrt[2]{2}}\right)$$

Equation 59

$$\rightarrow 20\log(2^N) + 20\log\left(\frac{\sqrt{6}}{2}\right)$$

Equation 60

$$SNR = 6.02N + 1.76(dB)$$

Further analysis of ADC error functions is available from [126].

Signal Conditioning

The most desirable design goal is providing some type of input conditioning enabling the energy store level sensor output to scale into the same operating window as the ADC references span, thus maximising the ADCs dynamic measurement range and creating the smallest minimum useful measurement size.

As the heart of the stack revolves around the periodic *tick* signal sent from the application software, its rate gets factored into the predicted *task* energy expenditure measurements and the minimum energy block unit capable of being measured.

Equation 61

$$Consumption = task\ cost - Incoming\ energy$$

The incoming energy rate will be continuously charging the store, and the *task* execution will be consuming from the store. The *tick* call will sample the store's current level. It is desirable to have the tick interval set to ensure the *task* execution has consumed at least one complete minimum measurable unit. If the tick rate is too fast, the sample's digitised value may not have changed as the ADC engine's step thresholds may not have yet to be breached. The speed of change here is not critical to the stack's correct operation, so this fast sample rate returning unchanged values is considered a deficiency.

It may be that the host hardware can provide a CPU off-loaded, interrupt-driven autonomous periodic ADC sample of the store, which proves to be the most efficient way to operate the sampling engine; in these cases, the oversampling occurring may not be of issue.

107

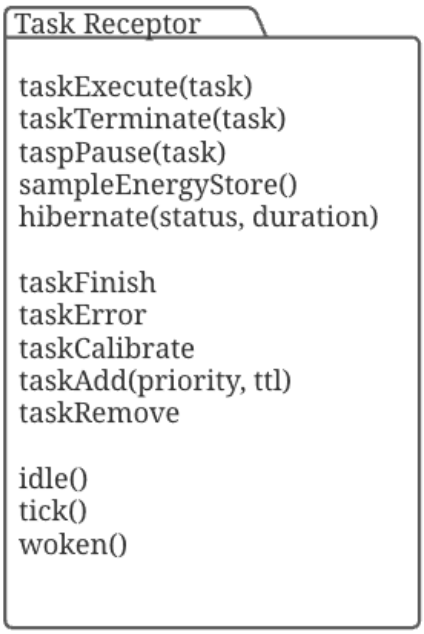


Figure 107: Task-Receptor Interface Outline

The *task-receptor* service receives a high-level *task* request from the devices application software. The service micro-manages the *task*'s lifecycle via the other supporting services interfaces until its execution has come to an end. Management of the *task* includes retrieving its current status and cancelling its execution prematurely.

When a *task* is received, it assumes the initialising *PENDING* state. As the *task* passes through the stack, it will traverse several other states defined by Figure 109 below:

108

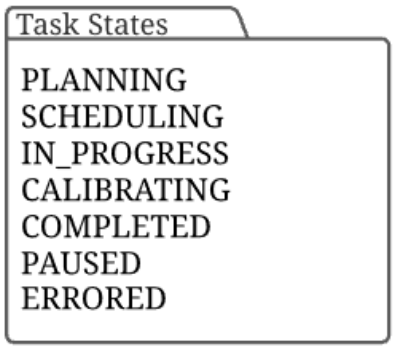


Figure 108: Task State Enumeration

The *Task-Receptor* is considered the interface with the application software where a particular device's list of *tasks* is assigned. *Tasks* get identified by ID number, and once a *task* has been added via the respective interface, calibration and execution attributes get created and resources reserved accordingly.

The *tasks* are tracked and managed throughout their life by the *Task-Receptor* service; the stack is ultimately responsible for providing and releasing a planned energy slot for the application to execute the *task* within.

Other interfaces surfaced by the service include status monitoring querying, *task* cancelling, pausing, resuming and adding.

Internally, the service maintains a *task* list that stores the *task* ID, the *task's* initial and current priority, calibration state, and current status. By default, a newly created *task* will assume either a *PLANNING* or *CALIBRATING* state.

Reception of a new *task* causes the receptor to search its list and see if it has previously encountered or executed said *task*, and if it has, the *task* is scheduled based on the associated incoming priority. From previous encounters with the requested *task*, the device will have already established predicted consumption data.

If the *task* is a new one, it must either supply accurate energy consumption predictions or allow the device to initiate a *task* calibration routine.

When a *task* requires calibration, the stack will schedule this timeslot based on the requested priority and the priority of the currently executing *task* schedule. As no previous data will be available for the *tasks* predicted consumption, it uses the maximum allowable charge level for the *tasks* initial execution. Rejection is possible of a new *task*, but this will only be known if the *task* requires and fails a calibration cycle.

Any rejection or failure of the *task* at hand triggers the stack to signal the issue back to the application software.

Power-Trender Service

The *Power-Trending* service actively tracks and trends the 'actual' power consumption induced by the stacks overhead, *task* execution and the underlying application software and hardware consumption costs – this allows the stack to retrieve a predicted power demand for a particular *task*. This prediction, in turn, is used to assign a costing to a particular *task* request. The costing allows a planning service to schedule the *task* for execution. The *Power-Trender* also keeps track of the current charge level of the energy store and its current charge change rate. This information is calculated and provided by the independent *Supervisor* service.

The trending process allows the energy costs for particular *tasks* to be dynamically adjusted based on the device's current environmental impact. An example of where this is significant is when considering a typical 'journey' to the shops, one of which happens on a warm summer's day, another during a hurricane and storm. The environmental impacts that affect the journey during the storm cause the traveller to burn far more energy to achieve the same journey.

The *Power-Trender* needs to consider these kinds of real-time changing environmental impacts; all considered random events.

Monitoring the progress of the currently executing *task* is achieved by regular samples of the energy store. The period of this monitoring is dependent on how often the application software calls the *Tick* function. This *Tick* function initiates an energy store level sample and consumes its energy cost. It then allows the *Power-Trender* to analyse its current progress against its predictions. The *Power-Trender* will allow a certain amount of leeway when enforcing its energy allowances. However, if the current *task* uses considerably more energy than predicted, the *Power-Trender* can terminate the *task*.

During the termination progress, the *Power-Trender* may also mark the *task* as needing a re-calibration, allowing a possible way to re-start the *task's* execution in the future.

The *Power-Trender* will always initiate an energy store level sample at the end of every *task* execution, regardless of the success state. This strategy allows an accurate update of the current situation before hibernation and the ability to feed the actual *task* energy consumption data back into the stacks feedback loop. Every time a *task* achieves successful completion, it will average its actual energy cost with the currently associated energy estimate, allowing future executions of the same *task* using more realistic energy consumption predictions.

Planning Service

The *Planning* Service organises *tasks* in priority and energy cost order, and its job is to map all the *tasks* the stack needs runtime execution information for. This organisation allows the *task* allocation to best satisfy the energy store capability at the point of the wake.

The *Planning* Service is also responsible for governing the length of time the next device hibernation will consume. It allows for rearranging its pre-hibernation plans if circumstances

changed during hibernation, which impacted its charge accumulation potential, either negatively or positively.

The latest hibernation plan is re-assessed under current operating conditions, and new decisions on which *task* or collection of *tasks* get execution time during the wake cycle are made. As such, it knows how long the hibernation period should be to achieve the necessary charge. Upon wake, the energy store level sample will reveal if the device is in a position to be able to achieve its pre-hibernation plan fully.

Suppose the store level charge is higher than expected. In that case, the planner can rearrange the planned *tasks* of the same priority level, adding additional *tasks* to utilise the extra resource surplus.

If, on the other hand, the charge level has not risen as expected, the planner can take other appropriate action which it decides is more beneficial to overall efficiency than accepting a false-wake and trying to correct the level deficiently with further hibernation. Remedial action to prevent this may include swapping the *task* for a smaller *task* of equal priority but younger in terms of its pending execution wait or processing a predefined fail-safe *task* that requires some kind of regular servicing in any case.

The stack assumes that the variables needing persistency get stored in an area of RAM that can retain its data during deep hibernation states. If this is not the case, steps are needed to provide some kind of non-volatility for the current operation. The *Planning* Service contains such critical variables, as it will need to be aware upon the wake of the pre-hibernation *task* execution plan devised of which it intends to follow.

If other means of non-volatile storage are needed, this can be incorporated in the wake and hibernate areas of the message flow. Here it can be assumed that a table of variables gets flushed to a non-volatile storage area once directly before hibernation occurs. As the dataset's

size should be constant, this process should be considered deterministic and adjust the consumption costings related to these stages.

It is necessary to read the data from the storage medium and re-instate the correct variable values to ensure the stacks continued proper operation upon wake. This process is considered deterministic and occurs as soon as possible directly after a wake event.

The stack places no restrictions on what application code gets executed during these stages, but they must have an accurately predictable current consumption. Having large amounts of processing in these areas is generally not advised as they will incur potentially large consumption losses during false-wake events.

The planner size represents the total amount of forward-planning the system can schedule. If the stack's desired operation mode is to wake, perform as many *tasks* as possible, then hibernate, the planner size denotes how many of these *tasks* can be prepared and scheduled.

After every *task* completion, the *Planning Service* gets refreshed, allowing the released work-slot occupied by the previous *task* to be immediately re-available. The plan list utilises a FIFO (First In First Out) type buffer (Figure 110 below) where the *Schedular Service* removes *tasks* pushed on by the *Planning Service* as their execution time depletes.

109

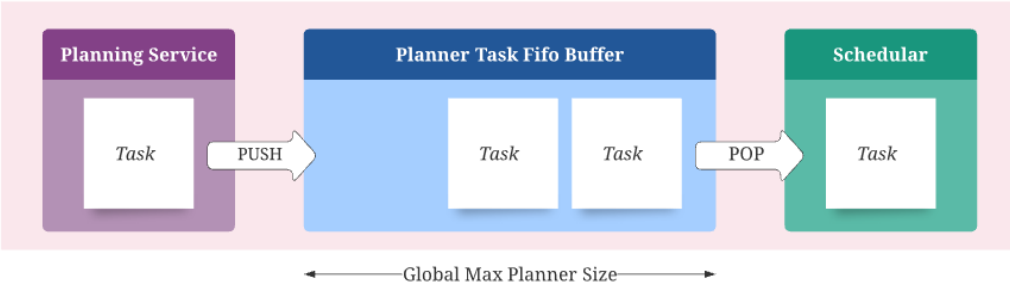


Figure 109: Planner FIFO

The *Planning Service* receives *task* messages. The contents of which include a predefined *taskID* and any *task* execution constraints. The service will analyse the message and, by using the *taskID* parameter, determine if the request involves a 'single executable *task'* or a 'group of executable *tasks'* of which successful execution of all grouped *tasks* in the correct order must occur before completion gets recorded. These collections of *subtasks* empower the implementer to reuse common and shared consumption predictions for shared hardware resources. The *Power-Trending Service* is responsible for establishing the 'actual energy cost' needed for each *subtask* in the same way it would a regular *task*. This grouping feature allows the application software to specify the *tasks* where time-dilation introduction is entirely acceptable.

A *task* delivered to the *Task-Receptor* service is considered the highest-level or *'parent task'*. The application software knows how it can physically solve the *tasks* expected of it and has prepared the stack; therefore, the stack knows if the *task* is or is not a grouping *task* by interrogating a list maintained by the *Task-Receptor Service* (Figure 111 below).

110

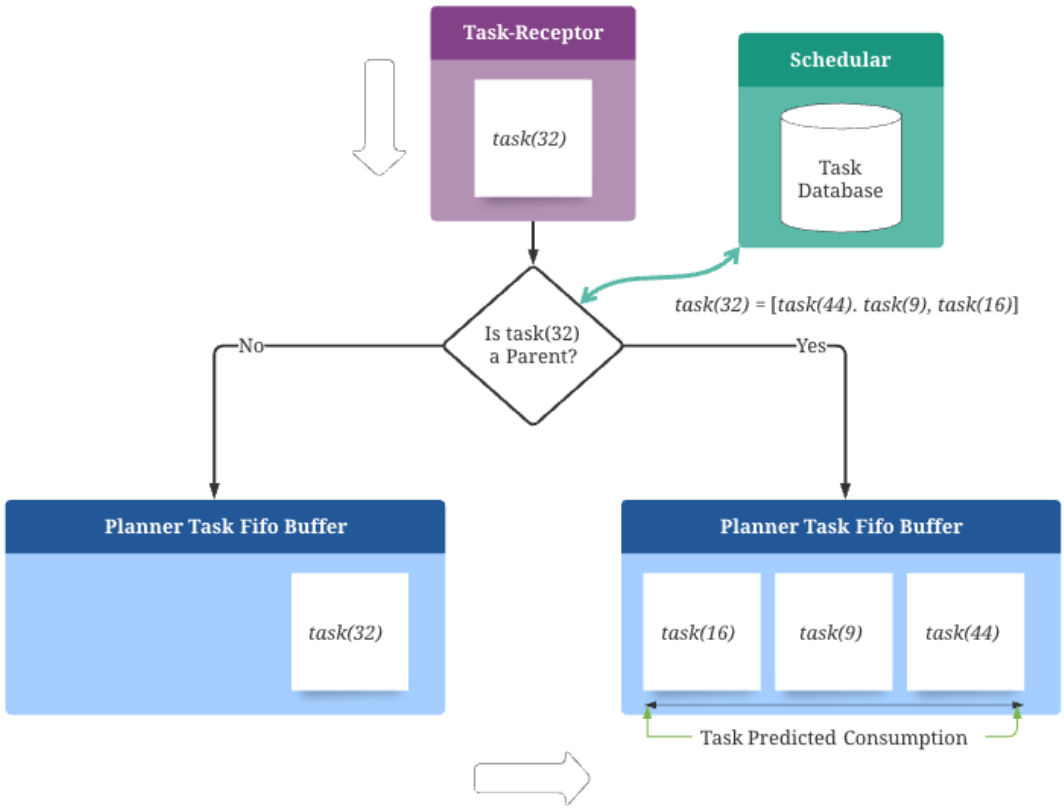


Figure 110: Parent Tasks

Grouped *tasks* will only be marked as complete when all the individual *subtasks* have reached completion without error. Grouping *tasks* in this way offers significant resource-saving benefits.

The planner in its current state will only allow work on a single *task* at any one time, and as such, there is only ever one work *task* that's currently in a transitional stage towards completion; all the other pending *tasks* within the *Planning Service* are queued and made ready for execution. The *Allocator Service* has the power to mark a *task* as 'paused', forcing the planner to reconsider the plan.

The services use FIFO buffer type queues where new *tasks* are continually added to the bottom of the queue as space is made available from processing and removing the *tasks* at the top of the queue.

This singular *task* execution thread restriction does not in itself pose a problem for devices that implement Real-Time Operating Systems (RTOS), as the *task* can represent a time window where the RTOS schedules its operations using the CPU time as it wishes (illustrated in Figure 112 below) with knowledge of how much time it has remaining.

111

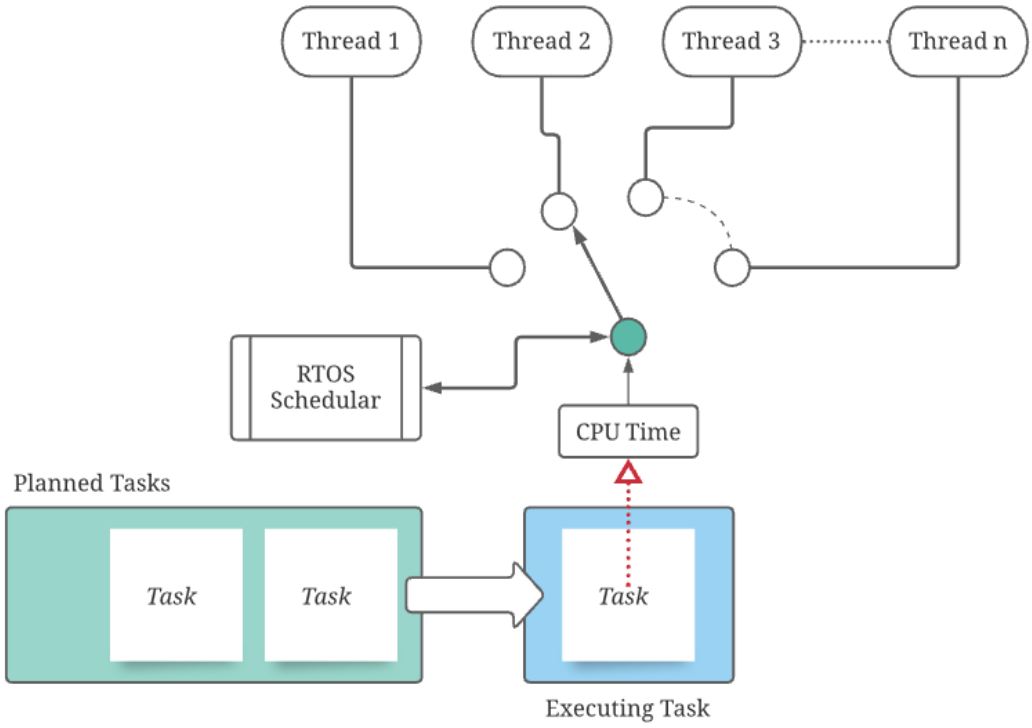


Figure 111: RTOS Topology

When the wake event causes the RTOS to start or resume multiple simultaneous threads of execution, each trying to complete their requirements successfully, the *task* completion status is only updated when every one of these threads signal termination or suspension. In this configuration, the duty of ensuring proper thread suspension before energy depletion is that of

the RTOS executive or application software. Failure to properly manage this will result in forced *task* suspension and unplanned hibernation by the stack

Supervisor Service

The supervisor service creates, allocates and controls work slots. A work slot is a time/energy cost-specified window of execution. The quantity and size of the available work slots get determined by data provided by the *Planning Service*, *Power Trending Service*, data collected from known environmental circumstances, energy charge rates, and the charging rate trend over periods. The dataset is processed using various self-modifying algorithms to provide the most efficient wake-up/sleep cycles. Each of the wake cycles has one or more work slots assigned to it. The *Supervisor Service* monitors the allocation of these work slots, the work slots' order, the *tasks* assigned to the slots, the state of the work slots, and their completion management.

A finite number of work slots are available, depending on the success of the previous charge period. The *Supervisor Service* maintains this list of prepared work slots and passes them to the *Allocator Service* for *task* assignment as needed.

The service tracks *task* predicted consumption costs needed to complete the planned *task* list, and it will also optionally evaluate other constraining variables which may influence the way the *task* gets passed through the stack for execution. These variables may include time limits that determine the maximum amount of time spent working on the job in its entirety (including the time taken to hibernate to recharge energy). Breaching these limits causes the *Supervisor* Service to declare the job's failure in part or in full, mark it as such, and move on to the next *task* pending.

Other execution constraints may accompany the pending *tasks*, such as periods where the *task* execution is either allowed or prohibited, dependencies of the prior completion of other jobs in the queue, or dependencies involving jobs that first must be completed by other devices working in co-operation.

Environmental changes may also have implications, both positively and negatively, on executing the *tasks* queued, and this must be detected and fed back for adjustments by the *Power-Trender* Service.

Allocation Service

The *Allocation Service* effectively allocates the work *tasks* to the CPU via the application software for execution. This service receives a *task* and its subtasks from the planning service by looking at the top of the planning queue and taking the next available item, and at this point, the planning service has already mapped the estimated power cost of each of the individual subtasks. The implementer will then allocate the *tasks* into empty work slots provided by the *Supervisor Service*. These work slots represent scheduled operation slots and originate from the S*upervisor* service. Once every subtask has successfully been mapped to a work slot, the *task* gets marked as in-progress.

The *Allocation Service* can reorder the slots for execution if needed; however, it will always try and follow any predefined priorities and sequence ordering as far as possible.

There may be instances where the device has woken after a pre-determined hibernation time, expecting that it will have a predicted energy store size ready for work utilisation. This outcome may not happen if the ambient energy source currently being suckled for energy has disappeared or dried up. In this case, the device may well find itself in a position that the execution plan becomes unachievable due to the lack of energy reserves.

Two options are available in this instance. Firstly, the device can recalculate its hibernation period based on the new ambient energy levels and go back to sleep, thus losing any chance of completing any work *task* during the current cycle.

The second option involves a little more intelligence, where the energy store level is evaluated against the energy level requirements from the other pending *tasks* waiting for execution time. If it finds one that it can complete or progress using the energy stores remaining charge, the service can re-arrange its *tasks* to achieve reasonable progress during the remainder of the cycle.

The allocator must also be aware of job priorities and re-evaluate its *task* priorities to some level during every wake cycle. The high-level user or controller may add, remove, re-prioritise, or re-order the jobs currently in the queue at any 'wake' time. The Allocator Service needs to respond to these changes and re-evaluate any imminent allocation accordingly; whilst it is nearly always inefficient to cancel an uncomplete job, this may be unavoidable during some kinds of critical operations. The overall assumption is that a work units size becomes the smallest unit available to deliver useful contributions to the overall *tasks*, so cancelling a currently executed *task* should not be needed.

Finally, the Allocation Service is aware of any *tasks* showing signs of error or failure and must either remove said *task* or find a way to continue execution while consciously avoiding potential looping or stuck job issues that could result.

Executor Service

The *Executor Service* duties include mapping *tasks* found in the currently scheduled work slot to the actual hardware subsystems by passing control back to the application software. The *Executor* prioritises any time limits and energy limits imposed by the *Supervisor Service* for

the executing *task*, and it evaluates any environmental, dependent or period restriction imposed by the constraints of the *task*. The *Executor Service* works closely with the *Power-Trender* and the *Supervisor Service* to achieve the goals required to progress the *task* state towards completion. The service periodically updates the power usage and trending calculations with the real-time consumption caused by the hardware needed to achieve this goal.

This system represents the feedback (Figure 113 below) loop that makes the power consumption adjustments imposed by the environmental conditions currently being experienced.

112

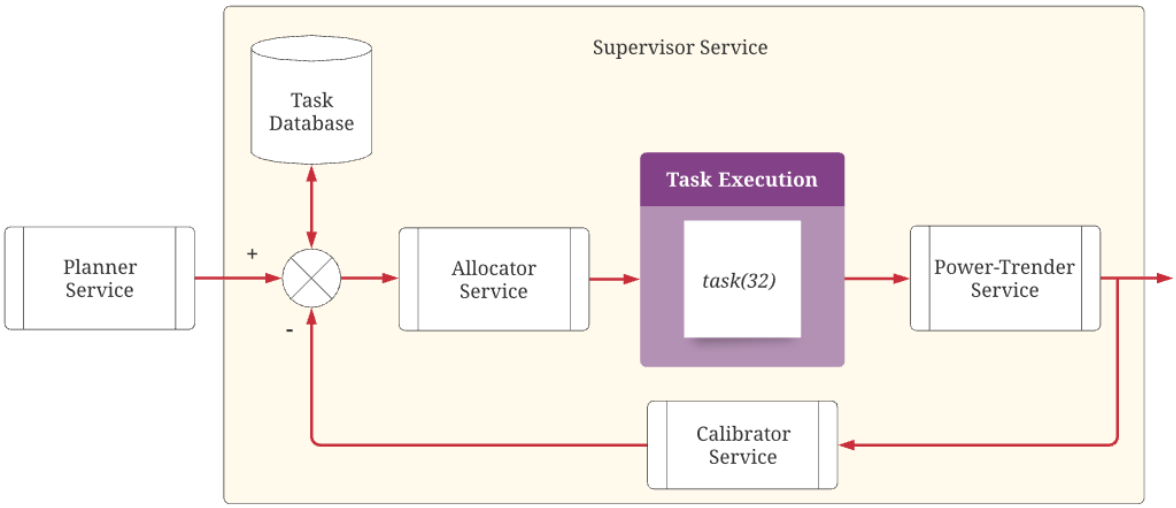


Figure 112: Feedback Correction Loop

The *Executor Service* will accept a single *task*, analyse its job parameters and constraints, mark its status as *IN-PROGRESS*, and pass control to the application software to execute the *task*.

The executor sends the *taskStart* signal to the application software and passes the *taskID* of the *task* it is expecting execution.

The stack waits for the application software to signal back, indicating an end of execution has occurred, accompanied by the result status code.

The only monitoring that the Executor Service performs is on the current *tasks'* energy store level samples every time the application software calls the stacks *Tick* function.

The executor will compare the energy store level drop to the expected consumption of the *task* in hand; if an overshoot situation occurs breaching the stacks permitted tolerance, then the stack will signal to the application software that it should forcibly stop the execution of the *task* as something unexpected has occurred. The application software should honour the request as soon as possible and allow the stack to mitigate the situation best. This damage-limitation typically involves requesting re-calibration services for the *task* at the next possible opportunity and going into a hibernation state to replenish its energy consumption losses.

Consumption Rate Monitoring

The stack in its current form does not see time as a metric for monitoring; it merely sees a drop in the energy store level. The rate of drop is the consequence of processing done by the application software and hardware executing the *task* at hand. To include energy rate consumption monitoring in this area would enable the stack to decide better if the *task* at hand is *becoming* unmanageable, can cease the execution of the *task* at an earlier stage and thus reduce its incurred losses; the overhead to manage this feature, however, is considerable.

For this research, the feature's implementation contains the following conditions; Individual *task* consumption analysis performed whilst the device is attached to an oscilloscope and high accuracy current clamp probe allows accurate plotting of the current consumption at the different stages of the *task*. The areas where a consumption step change is significantly

noticeable gets broken down into smaller subtasks. Each subtask has a time duration, and its consumption level is plotted and divided into a hierarchical structure.

The plan organisation stays the same, and the planning is still performed based on the *task* consumption prediction without regard to these newly identified subtasks, ensuring certainty that if a *task* gets scheduled, there is adequate energy available in the store to execute all of the *tasks* subtasks fully.

As the *task* execution begins, the executor will send the individual subtasks information to the application software for processing and comparatively monitor the consumption rate changes against the predefined rate measurements taken during commissioning.

The complication with this implementation comes when the overhead of these extra monitoring and measuring requirements consume more energy than they save. Inevitable, this type of monitoring will only pay justice if there are frequent occurrences of *tasks* overshooting their predicted consumption rates, which points to the system not performing as intended.

RTOS Costings

When using the stack with an RTOS, it is unreasonable to expect an accurate prediction of consumption rates when *tasks* could consist of work distributed to multiple unmanaged threads. In this case, the timeslot of execution relates to its energy usage versus charge level.

This proposal centres around two particular cooperation strategies; the first involves treating each *task* as having identical consumptions and providing average consumption predictions via multiple executions. The efficiency of execution depends on the accuracy of this average versus the actual energy consumed for all the RTOS execution threads running. If the consumption overshoots too often, then the gains of management diminish, so in this scenario,

it is essential to try and have all the threads running balanced loads. Even if the threads can be deterministic, it is still unknown if any other threads get scheduled by the RTOS executive in the same period.

The second method uses the *tick* function to monitor a level usage threshold defined at the point of wake. This threshold allows the RTOS to self-manage energy and gets signalled when it is time to hibernate. At this point, the *Planning Service* can utilise its algorithms to determine the best charge time based on the incoming energy stream, which will reflect how much time the RTOS will next be given to service its threads.

This average costing method also allows for the RTOS to signal to the stack that all of its threads have suspended and the system no longer needs to be in the 'wake' state. The executor will then arrange to go into early hibernation. The unused energy at this point gets rescheduled for use during the next wake period.

The RTOS configuration must ensure that the signal sent from the stack requesting execution termination followed by hibernation gets treated as the highest priority interrupt. The application software must carefully manage the time between this signal and the call-back to the stack, which will initiate device hibernation. Enforcing this can be a significant complication as the threads will have an unknown amount of processing left to execute. In such situations, thread pausing is a viable option if the hardware peripherals can be isolated and put into the same state they would have assumed during a calibration cycle. The main point to note here is that if paused *tasks* cause additional parts of the system to remain consuming energy during the hibernation period, then the charge predictions will be incorrect, and the risk of a false-wake significantly increased.

If pausing the thread and disabling the systems associated with the execution will cause the thread to terminate unexpectedly, then a graceful exit routine must be implemented to close down the situation as soon as possible.

Communicator Service

An interface exists which allows for the control of synchronised message flow between devices or base stations. The interface allows management and initiation of synchronisation, reception, and transmission functions; however, the messages' content is entirely application-specific.

The physical building, transmission, reception, and decoding of messages is the application software's responsibility, and the stack will merely coordinate timing, add some additional control data, and pass messaging requests to and from the application software.

The messaging interface consists of the following functions:

- *addMsg()*

The stack will send the following signal requests originating from the *signal()* call:

- *requestSendMsg*
- *requestRecieveMsg*

The configuration constants available to tweak this stacks responsiveness to the device's minimum design requirements include manipulating the maximum frequency of message sending and maximum duration to wait for an expected incoming message to arrive.

The *tick* function of the stack takes on a more essential and managerial role when synchronised communication is required. The *tick* function allows the stack to schedule message

transmission requests and service predicted receptions, enabling the foundation to trigger a timely response to messages it has scheduled to receive.

For the communication synchronisation, the periodic *tick* call from the application software to the stack during *task* execution increments an internal *tick* counter, providing and maintaining a time scheduling facility.

The stack will schedule wake-up times to exchange messages with other devices using this *tick* counter and ensure the process does not breach the global settings' maximum transmission rate.

For devices to communicate directly with each other (peer-to-peer), they have two options. The first will require the devices to share precisely the same *Tick* repetition rates among them. The second involves providing a conversion routine.

During a communication setup from either a base station or a peer-to-peer direct message transfer, the units will synchronise their times to achieve a coordinated wake event. This synchronisation does not necessarily enforce the designer to implement a timekeeper routine or real-time clock. It can be done efficiently with basic tick counting and wait times which will ideally coincide with *task* execution events.

Time conversion routines allow the stack to perform a ticks-to-time translation. The stack can then schedule a communications slot for 15 seconds by using this routine to calculate the number of ticks needed to satisfy the period relative to its specific hardware platform running frequencies.

The message waiting period provides a mechanism for dealing with the inaccuracies or differences of timekeeping between devices. An essential goal of the planner is to keep track of any scheduled messaging slots that are pending and make sure that the device is in a wake state at the appropriate time. Ideally, the device will have already woken at its most efficient point, and *task* execution will be underway, coinciding with the scheduled communication

timing slots. These slots need regular serving and have enough additional execution time remaining to ensure either completion of the message transfer or expiry of the predefined maximum wait period. If device wake must occur when energy stores are yet to reach a successful *task* execution level, then the additional cost of the wake will significantly reduce efficiency calculations in terms of useful work done versus energy input. The planner will use its algorithms to maximise the possibilities of making sure the communication slot timing always occurs during a useful *task* execution period, and this is a metric used to evaluate the stack's performance within its application.

During hibernation state, the *Planning Service* must keep track of the tick count it would have accrued if awake and executing *tasks*. This loss needs recovering, and a time-to-ticks conversion routine or the application-specific method of keeping time during hibernation can accomplish this and replenish the counter's value. The *Planner Service* must ensure that after it has a viable plan and has calculated its hibernation period regarding its incoming energy stream, there are no scheduled message slots that need servicing during the sleep.

Discovering a clash where a scheduled radio event will miss service causes the planner to either try a different combination of same level *tasks* and sleep patterns or, as a last resort, schedule a dedicated wake event intended solely to service the communication, causing the device to consume additional energy.

Wake Time Latency

Another reason for carefully considering the predefined maximum message wait period reflects the latency introduced by the application software between the stack requesting a *sendMessage* signal and the actual message bitstream electronically transmitting through the air. If not a base station, the reception side will inevitably also need time between a request for

message reception and the actual point the hardware would be capable of electrically receiving and processing it. The *Planner Service* must take wake time latency into account, as different types of devices do not know the scheduled wake times of other devices. A predefined tolerance value is added to the scheduled wake time to cater to these variances and unknown influences, as shown in below in Figure 114.

If the device schedules for a message reception event, it should wake at the scheduled time minus its wake-time-latency at the absolute minimum.

113

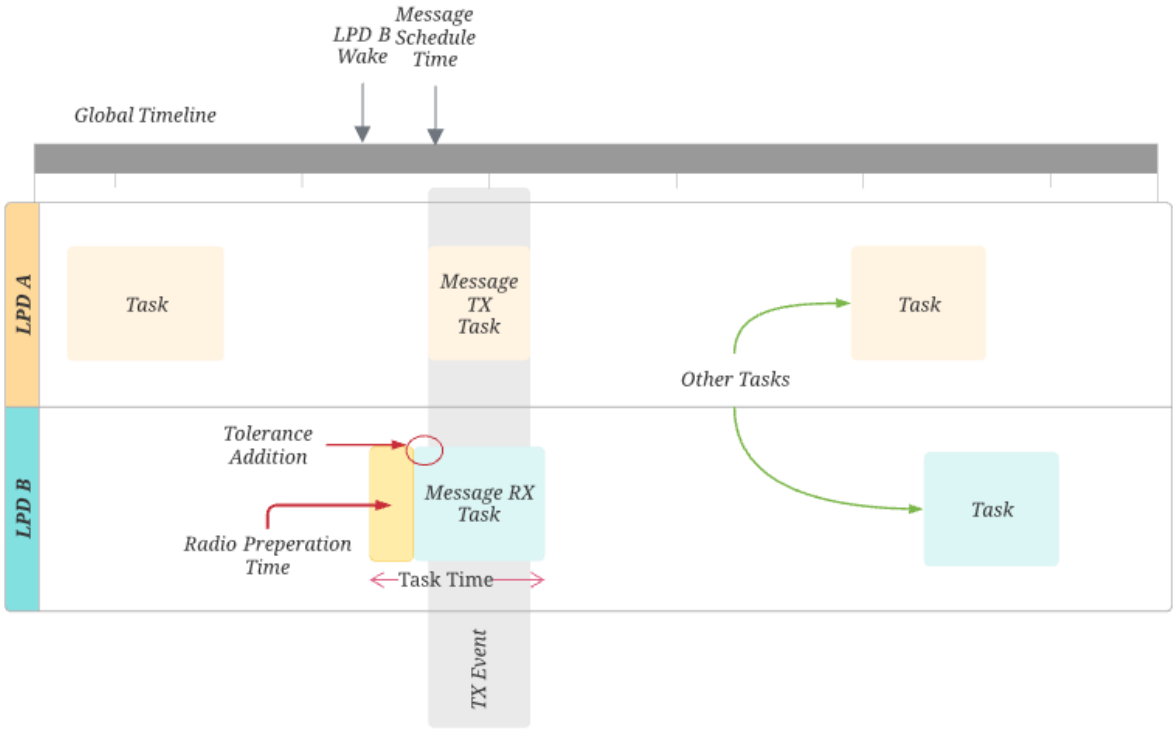


Figure 113: Radio Overhead Tolerance

Monitoring Service

The *Monitoring Service* allows long-term data collection of all performance and efficiency metrics. The service allows the mesh controller to retrieve and analyse every hardware unit's

performance within the network. This information allows a real-time GUI user interface where the *task* execution and hardware units become visualised.

It also provides a messaging foundation to pass certain events back to the controllers, such as survival messages, scheduled hibernation times, scheduled hibernation durations, and remaining work time and energy datasets.

Before the device goes into hibernation, it will send a status-packet message destined towards the controller's logging service. This packet summarises the devices decisions made, the current environmental state, and the current health state since the last log packet transmission. It also contains the current energy store level and the average rate of change this source is currently experiencing alongside the rate of charge accumulated during the hibernation period.

Message Management

Messaging between devices and between the controller and user are done using a synchronised time message delivery system. Both ‘pass-your-message’ exchange styles and direct-messaging are supported using a calendar scheduling service.

Messages passed to the device are serviced during wake periods, and message exchanges scheduled using the log message transmission process. Message exchanges originating from a device and targeted to the controller are allowed anytime the device is in a wake state. The controller acts as a post office; if inter-device communication is required, the message is addressed to the destination device and passed to the controller for forwarded delivery when possible.

When the device passes its messages, it also asks if it has any pending messages waiting and proceeds to retrieve them. The controller will pass all broadcast messages and all pending messages holding the device's address during this message exchange.

The devices are also able to communicate directly with one another after creating a friendship bond. This bond is achieved by initially asking the controller to be introduced to the target destination device.

If the controller allows this activity based on its predetermined security and privacy settings, it will pass a message back to the device, revealing the target devices next wake-up time. It will also pass a message to the target device giving details of the requested friendship bond. Figure 115 below shows the message flow diagram outlining the process; the meet time gets agreed in this case as *@1*.

When the target device next enters its wake period, it will request its messages from the controller and see the pending friendship request. At this point, it will schedule and turn on its receiver, waiting at the prearranged time for the potential contact.

114

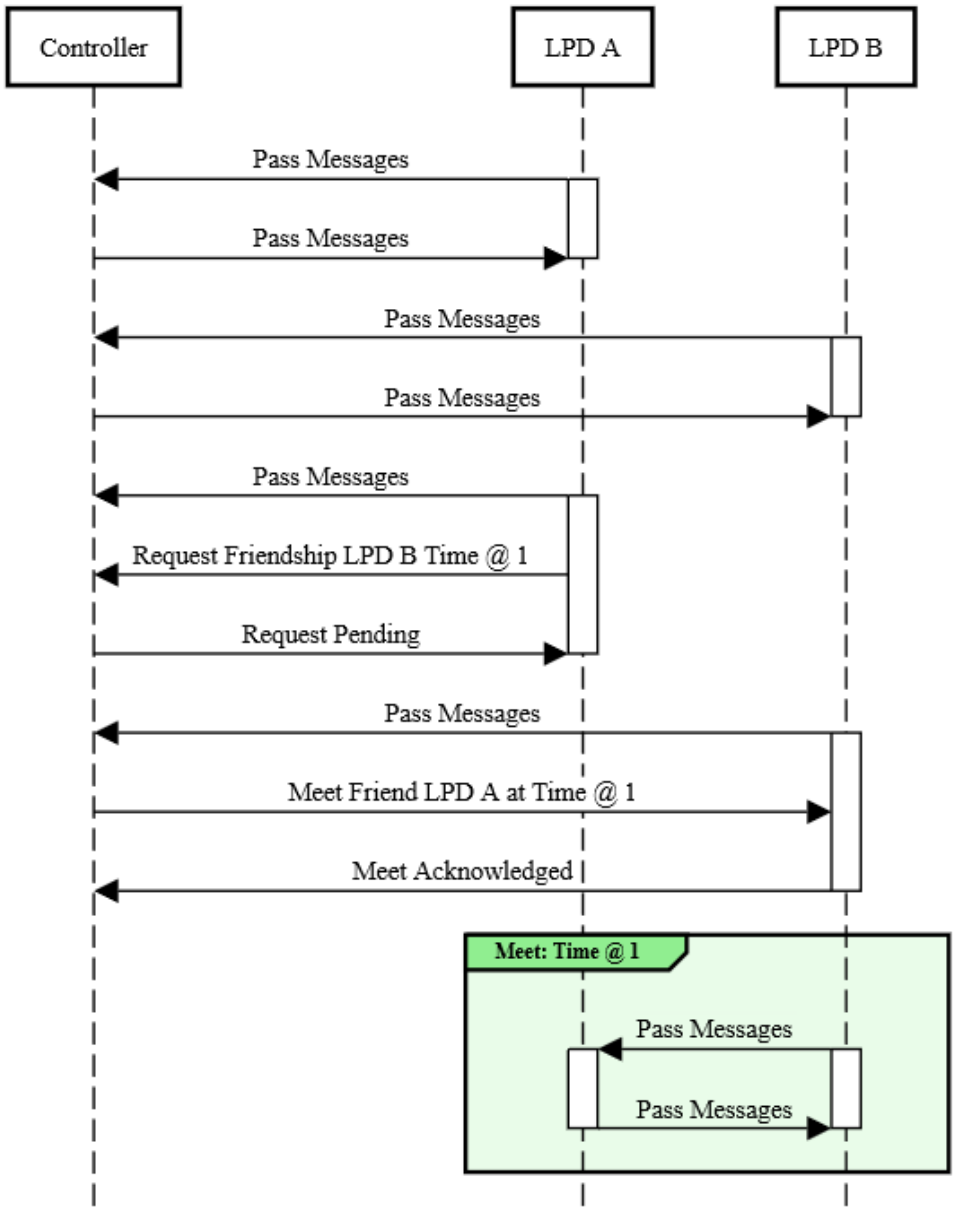


Figure 114: Peer-To-Peer Message Exchange Setup

Meanwhile, the transmitting device received the reply from the controller that its friendship request is pending with the target device, and it now knows the next time the targeted device will be listening for incoming messages.

The transmitting device must now schedule itself to wake simultaneously, so the two devices effectively synchronise their wake patterns for the next message exchange.

When the wake cycle occurs, the transmitting device can repeatedly send its message until it receives a reply from the target. At this point, the friendship bond becomes stable (Figure 116 below illustrates this sequence), and they can then resync their next wake-up message exchange directly with each other; no further arbitration from the controller necessary.

115

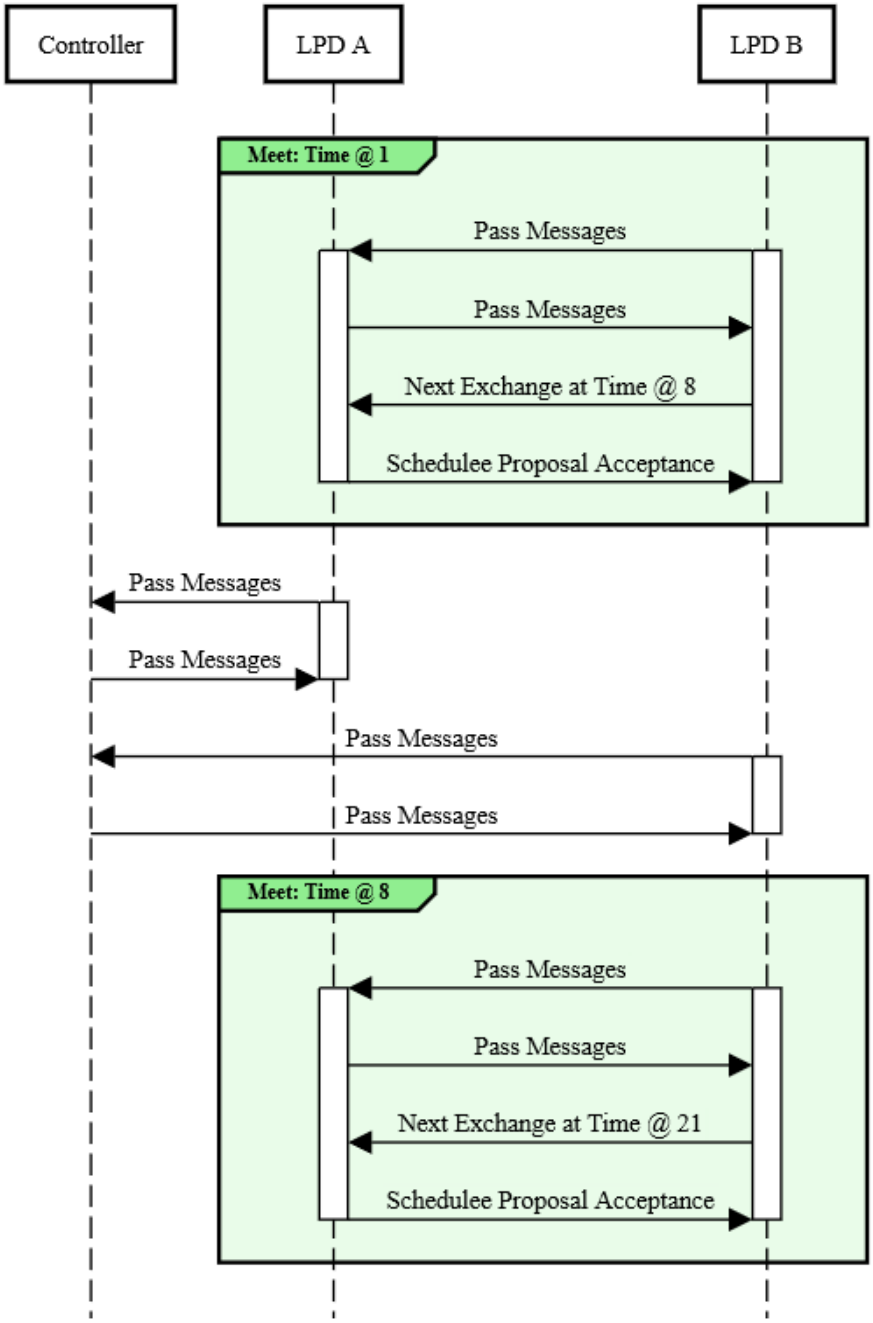


Figure 115: Continued Friendship Meetings

Ultimately this direct messaging system is the most energy-efficient way of sustained communications. The only real commitment needed is the synchronisation of a mutually

agreed wake-up time. Complications arise due to the energy store predictions that need taking into consideration.

A two-way mutually agreeable system needs establishing, which deals with either device's need to reject a wake-up proposal due to its personal constraints, and the arbitration process reworked until an acceptable timing strategy for both units is agreed.

Wake failure also needs consideration. The entire communication process has an obvious energy expenditure cost, which should be mostly deterministic. This cost should also be considered a system cost as opposed to a work *task* cost.

When a device wakes due to its hibernation duration expiring, the device may well discover that its recharge cycle has not been sufficient, and its current store level is below the cost level of even primitive essential communication; in this type of situation, an immediate re-hibernation is the only real option for survival. This case presents a missed communication opportunity; one or both devices need a fallback plan to re-establish message exchange synchronisation.

The recovery plan can take two forms, the first involving immediately revoking the friendship bond, and as such, will need to re-execute the initial bond setup procedure involving controller arbitration to re-establish initial synchronisation.

This option's major disadvantage is the constant dependency of a controller unit situated within the effective radio communication range. There is the desirability given by using direct messaging, that devices can work together and communicate even if they are not within the range of or have any type of contact with a controller device.

Retention of this flexibility is achievable when the devices can provide an internal mechanism of automatically repeating resynchronisation messages. Both the devices can immediately fall

back into a periodic wake and try routine until the two devices eventually successfully achieve a new contact, after which the standard communication protocols can resume.

Mimicking some of the features presenting in the BLE Mesh protocol [34], a device can repeat a message and perform all functions required of a typical message repeater. This facility proves very useful when the distance between a portion of devices and controllers becomes too great. Flood routing allows addressed-and-targeted message broadcasts to every device within the radio communication range. If a device receives the message that is not the destination device, it forwards it one time to every other device it encounters within a predefined period. The message's creation time, embedded in the message payload, gets compared against a predefined ageing value. Once the ageing value has determined the message has become too old, it removes the message from its message queue.

Using this method, even if receiving LPDs are vastly out of range from the sending device, their messages can still get delivered by hopping it over other devices positioned between the sender and the destination devices.

These broadcast messages are far more expensive in terms of time and energy to utilise, but they can still be deterministic in terms of predictability. The general rule to follow is to predict the worst-case cost scenario, and if the *task* comes in under budget, the *Power-Trender* will adjust its calculations and the unused energy left in the store.

Dedicated Wake Channels

It is acknowledged that the current trend towards communication in ultra low power sensor nodes often revolves around setting up and monitoring a low frequency/low data rate out-of-band radio wake-up channel [74]. This type of wake-on-radio strategy allows the devices to

enjoy very low current consumption sleep periods and only wake and incur higher consumption levels when the radio signals target them.

Although this method is a proven method to maximise incoming message collections, it is not predictable in terms of when the energy needs to be made available.

When considering this type of operation alongside the stack, it highlights some major compatibility issues. The operational idea behind the principle involves leaving the radio on all the time in a very low power receive mode. If a message is detected, it can bring the other systems out of hibernation.

Having the radio receiving hardware always energised is only an option if the incoming energy is in surplus; however, this type of receiver's consistent consumption cost would consistently enforce the need to reserve far larger blocks of energy per *task*.

The receiver could be duty-cycled, thus reducing its sustained consumption and allowing a deep hibernation recharge period. This recharge cycle aids in better scheduling of the messages and the ability to determine the delivery frequency. This method is more manageable, as the low power receiver can be energy consumption predicted, planned, and executed, just like the other *tasks*. From the base station's point of view, duty cycling the reception could cost multiple message re-transmissions, as the base station will have to ensure that the device's radio was on a duty period during the transmission. If the base station is unaware of the devices duty rate, then resending the same message until either a reply is received or the probability of duty after multiple messages sendings is high enough for it not to be an issue.

The secondary benefit to this method is that although every device will receive the low power message wake-up, the contents of the message's payload contains the targeted device's address. As such, only the devices addressed as intended recipients will progress to powering up their main radio interfaces in preparation for receiving the actual message.

The usefulness of using this technology to wake the device from hibernation is counterproductive in the sense that the energy management control would have to move to within the base station area, which will create far more communication overhead by the incurred message exchanges needed to pass the status and energy parameters back and forth between all devices.

Employing this solution needs careful consideration towards dealing with the incoming messages scheduled during hibernation periods. The wake-up, receive the message, and re-hibernate out-of-plan is one option. The other solution is to utilise the *Planner Service* to schedule a *task* that will coincide with the messages estimated arrival time.

The stack already has the facilities to keep track of and time *tasks* by tick counting, so the ability to schedule and wake for message reception is feasible. The main primary radio channel is powered down entirely during this message wait period. The responsibility of ensuring that the target radios are in a duty cycle and in a position to receive messages transmitted from a base station is entirely that of the transmitting device.

Initial Comissioning

Implementing the stack within the test environment was relatively straightforward, allowing complete control of the energy input stream experienced by the LPDs in multiple forms and assigning elementary *tasks* that turn on LED lights for different periods.

These assigned rudimentary LED flashing jobs break down into individual *tasks* turning on up to three different LEDs for individually specified lengths of time. These *tasks* are then randomly sent to the devices for execution, and the resulting logging data from each device retrieved and combined for evaluation. The input energy stream was varied for multiple runs

of the same experiments to prove the devices' behaviour is of an expected nature during environmental impacts.

Altogether six units were deployed and monitored in different positions relative to the two different energy streams. Each unit had to make sure they woke for execution only when they had accrued enough charge. This cycle was only achievable by energy scavenging during hibernation, followed by successfully executing and completing the pending *task*.

The communication setup configuration involved a periodic message window spaced 60 seconds apart, and the initial *task* calibration cycles successfully executed during periods of extremely high input energy levels.

Four LED test reference *tasks* are specified and assigned globally recognised *taskID* numbers:

1. LED 1 ON, LED 2 OFF, LED 3 OFF, WAIT 1000mS
2. LED 1 OFF, LED 2 ON, LED 3 ON, WAIT 800mS
3. LED 1 ON, LED 2 OFF, LED 3 ON, WAIT 500mS
4. LED 1 ON, LED 2 ON, LED 3 ON, WAIT 500mS

The retrieved log data makes it possible to evaluate the *Power-Trender's* performance and the settle-down time seen by analysing the feedback loop. It also clearly shows how the group can respond to and rebound from environmental changes randomly introduced during the experimentation periods.

Chapter Summary

All of the stacks services operate together to provide a low-level management protocol that tightly integrates with a radio interface such as the current BLE offerings. The stack integrates within its Protocol API and allows any higher-level application to make use of the features. An implementer needs only to provide the hardware subsystem driver implementation (App-Specific peripheral drivers) and the application Profiles and Services, illustrated below in Figure 117. WIMP can go into a learning mode to gain an initial idea of the energy input vs energy output availabilities. This data is then used to begin the power trending calculations.

116

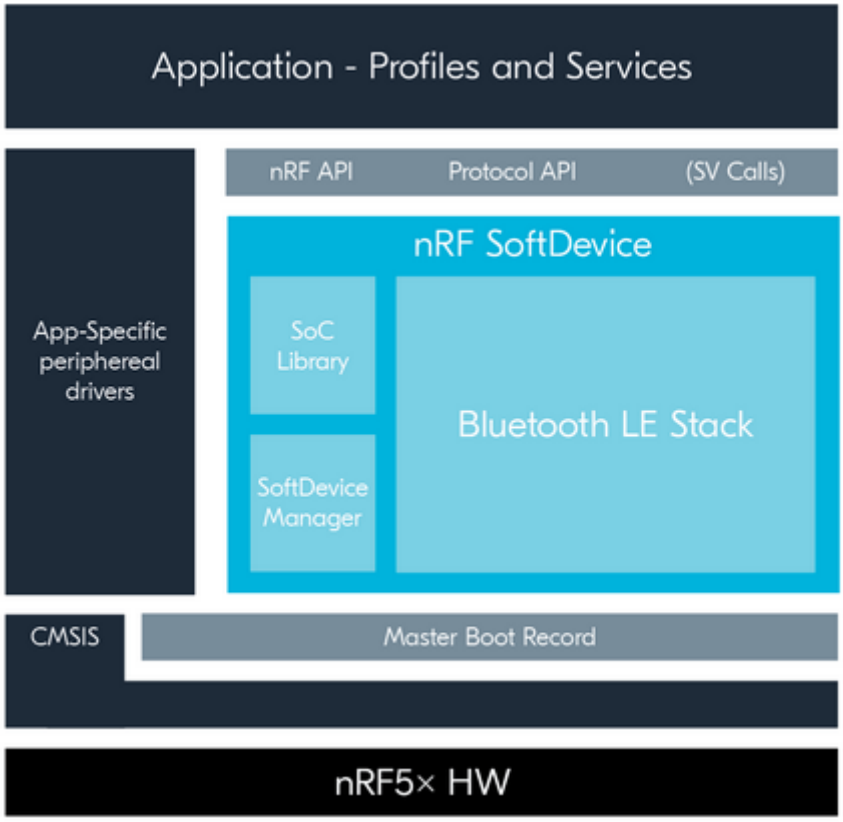


Figure 116: nRF BLE Block Diagram

WIMP also integrates the logging communication service, which shares the *task*, energy usage, and energy charge rates with all the other devices within its group. The *Supervisor* and *Allocator* services use this facility to distribute high-level *tasks* to the application software for execution. This tight integration allows a distributed team-working solution to complete a *task* using multiple hardware units in synchronisation. Each unit will take on individual subtasks, and the meshing nature of the setup will allow them to all track the state of the overall *task's* progress.

A significant change of direction for this project has been upgrading the BLE stack to a Bluetooth version 5 compliant version. This component-level change allows an immediate gain of some valuable features, including:

- The notable increase of sensitivity of the receiving hardware
- Bluetooth long-range mode – this takes advantage of the increased receiver sensitivity and implements new reduced data rate protocols allowing reliable data transmission of multiple kilometres.
- Increased broadcast capacity
- Improved coexistence
- Concurrent multirole technologies
- Greater net throughput
- Improved fault tolerance

This enhancement caused a hardware modification in terms of IC swap out from all current hardware units; however, the gain in the receiver sensitivity allowing harnessing the long-range offerings deemed it worthwhile.

WIMP has been modelled at various stages of its design and has cumulated into a physical software implementation executed on the research platform introduced in chapter 1.

The research implementation of the WIMP stack uses the ‘C‘ programming language. Every module and service’s code is written as per the previous design discussions, with slight modifications that allow statistical and real-time monitoring of the internal algorithms. The test studies' are generally directed towards comparing the WIMP stack's added intelligence capabilities with both the previous analysis’s on fixed and adaptive wake strategies.

The test setup consisted of the RF energy transmitter and solar lamp working together to control the energy streams passing over the eight individual hardware prototyping units, illustrated in Figure 118 below.

117

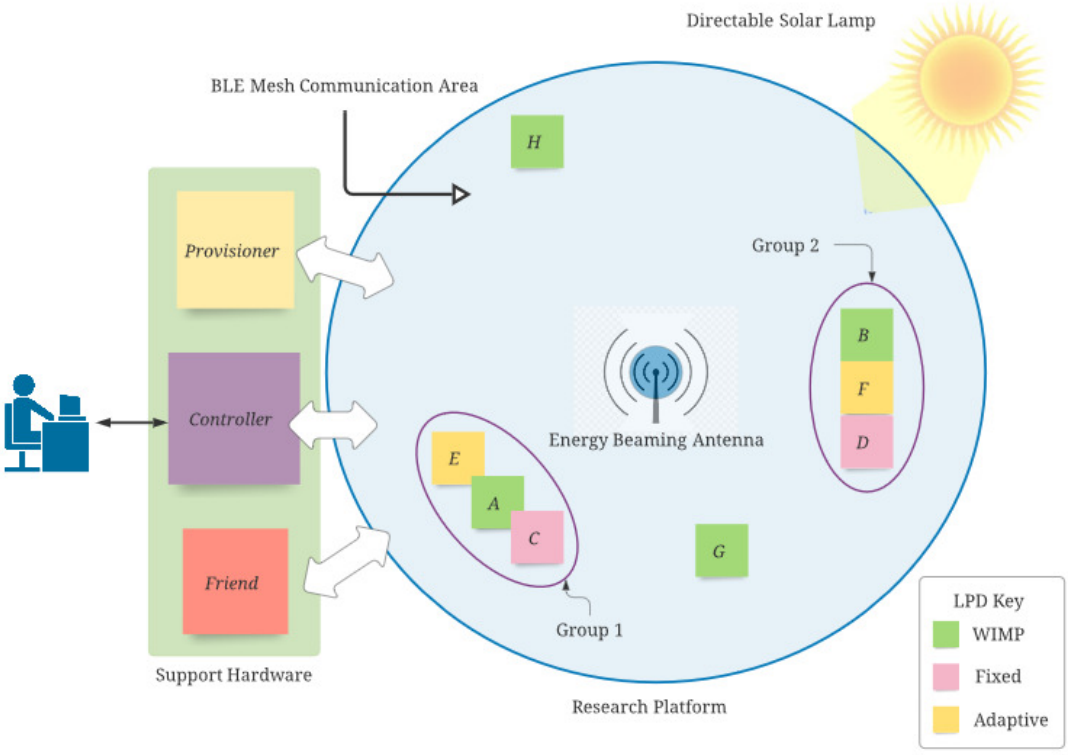


Figure 117: WIMP Test Platform Layout

Of these eight units, two operate using a fixed time wake pattern, two devices uses adaptive wake patterns, and the final four devices boast full WIMP stack implementations.

The four WIMP devices get randomly placed at different positions relative to the primary incoming energy stream sources.

Device positioning involved two adaptive units placed next to two of the WIMP units and the fixed unit placed next to an adaptive unit.

Two other units running the WIMP stack are placed at random positions to aid in data collection and comparisons.

The devices were all given the same six predefined jobs detailed below, and the WIMP devices were allowed to calibrate by themselves. The last WIMP device came preconfigured at design time with *task* consumption predictions instead of self-calibration.

1. WAIT PULSE, LED 1 ON, LED 2 OFF, LED 3 OFF, WAIT **1000mS**, WRITE LOG
2. WAIT PULSE, LED 1 OFF, LED 2 ON, LED 3 ON, WAIT **800mS**, WRITE LOG
3. WAIT PULSE, LED 1 ON, LED 2 OFF, LED 3 ON, WAIT **500mS**, WRITE LOG
4. WAIT PULSE, LED 1 ON, LED 2 ON, LED 3 ON, WAIT **500mS**, WRITE LOG
5. WAIT PULSE, LED 1 ON, LED 2 OFF, LED 3 ON, WAIT **500mS**, WRITE LOG
6. WAIT PULSE, LED 1 ON, LED 2 ON, LED 3 ON, WAIT **500mS**, WRITE LOG

A rudimentary radio protocol uses the research platform's 2.4GHz RF interface to listen for a 4-byte periodic message transmitted from a bespoke external device (Figure 119 below).

118

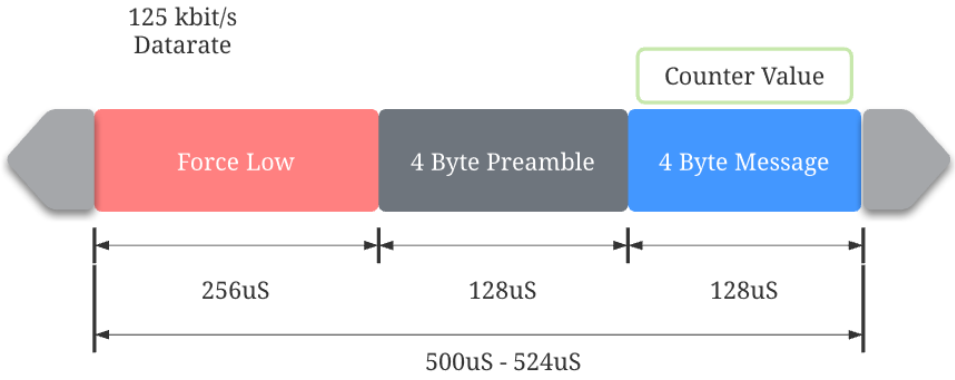


Figure 118: Time Sync RF Message

This message is an incrementing count value (represented as a 32bit unsigned integer number) transmitted every 100mS and facilitates time synchronisation services.

For every *task* the LPDs are expected to execute, they must first turn on their RF receiver interface and wait for the next time sync count transmission to arrive, illustrated below in Figure 120. The message payloads value then provides synchronisation and timing information which get tagged to log messages for later analysis.

119

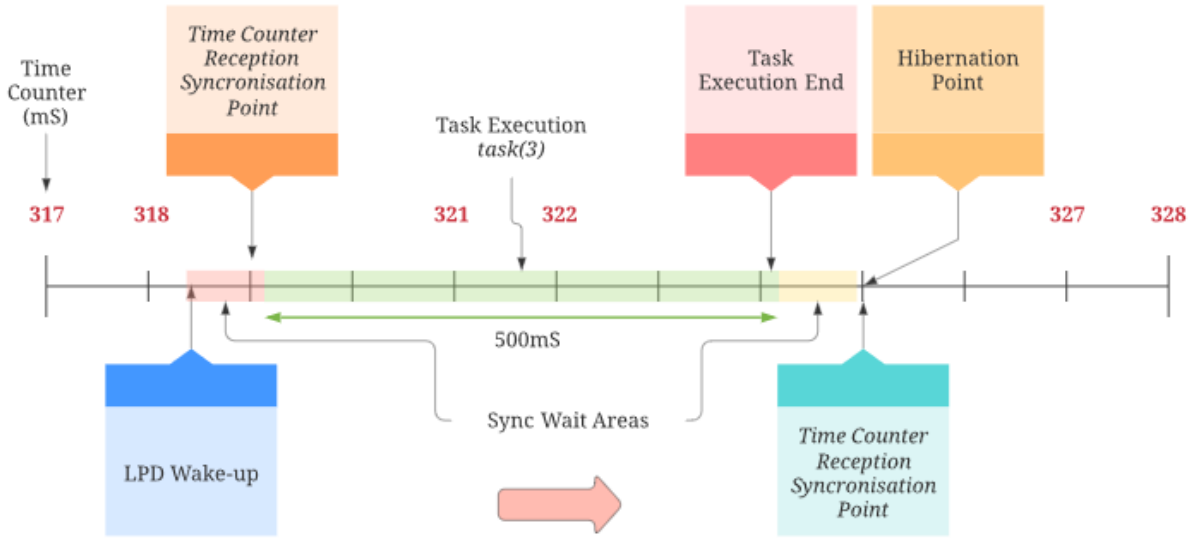


Figure 119: Task Time Sync Points

After the reception of the time sync count message, the devices can disable all radio interfaces and preserve as much energy as possible for *task* execution.

The final part of any of the *tasks*, before hibernation can commence, is to wait for another count pulse to be transmitted. At this point, the devices again must enable their RF interfaces and wait for the external time count message to arrive (as illustrated in Figure 120 above). However, during this period, the unit will time how long it spent waiting since the *task* execution was finished locally, and the external count message arrived wirelessly.

Along with recording the energy store levels at the wakeup, hibernate and job *task* execution points, the *tasks* also involve measuring their incoming charge rate (via the *Tick* function). This measurement requires two consecutive energy store level samples surrounding a known period of inactivity (Figure 121 below illustrates the various measurement points). Although not considered the most accurate representation of incoming useful energy, it serves as a

reference point so the different locations of the devices relative to the energy streams can correlate during data analysis.

120

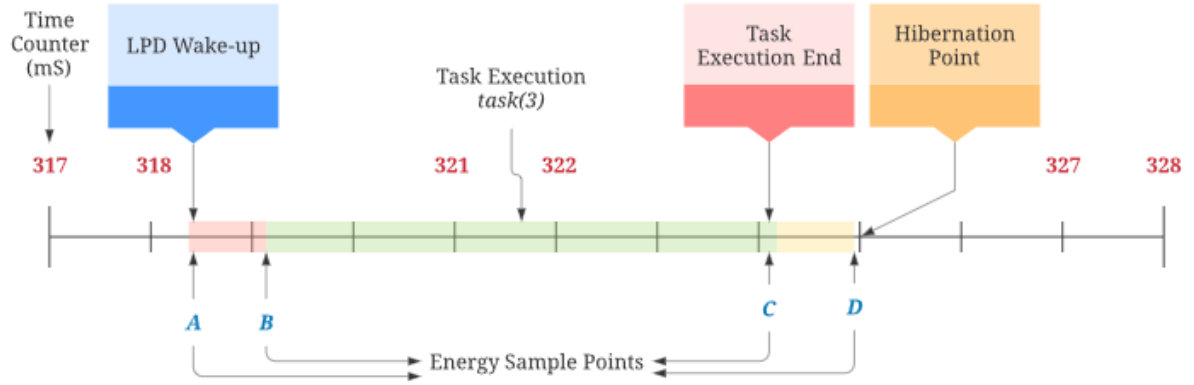


Figure 120: Task Energy Level Sample Points

As to minimise the use of additional components and software libraries, the logging data gets written to the microcontrollers internal flash area. This process is quite an expensive job in terms of both time and energy and needs to be the last operation before hibernation begins. Care ensuring that the upper energy store level threshold gets set correctly is critical. The setpoint should recognise that even during failure, there is a reserve requirement to retain enough energy to record incident reasons; this is solely the cost of keeping the CPU alive until it has signalled its internal flash write routine has finished. Failure in this area could easily corrupt the entire logging contents and void the experiment as MCUs typically have very large flash erase and write page sizes.

The devices all share the same type of 12bit successive approximation ADC modules.

Logging

The logging system records all the critical time points needed for efficiency analysis. The life cycle of a *task* involves the following logging points:

- $\mathbf{T_{s\text{-}mk}}$: Start session count marker (generated inside LPD)
- $\mathbf{T_{s\text{-}lvl\text{-}A}}$: Start Energy store Level value
- $\mathbf{T_{s\text{-}wt}}$: Time since wake event and count sync message received.
- $\mathbf{T_{s\text{-}cnt}}$: Start Time Sync Count message value
- $\mathbf{T_{s\text{-}lvl\text{-}B}}$: Start Energy store Level value
- $\mathbf{P_{tasks}}$: Planned *tasks* after adjustment
- $\mathbf{P_{exe}}$: One or more Task start, Task End timestamps relative to the wake-up point
- $\mathbf{P_{hib}}$: Planned hibernation duration
- $\mathbf{P_{post}}$: Planned *tasks* to execute after hibernation
- $\mathbf{T_{e\text{-}lvl\text{-}C}}$: Start Energy store Level value
- $\mathbf{T_{e\text{-}wt}}$: Time spent waiting for next Time Sync Count message
- $\mathbf{T_{e\text{-}cnt}}$: End Time Sync Count message value
- $\mathbf{T_{e\text{-}lvl\text{-}D}}$: End Energy Store Level value
- $\mathbf{T_{e\text{-}mk}}$: End session marker

All the time values apart from the time waiting for the final count sync message are translatable to delta values referenced from the point of waking.

Figure 122 below details the points in the cycle where the variable measurements get taken.

121

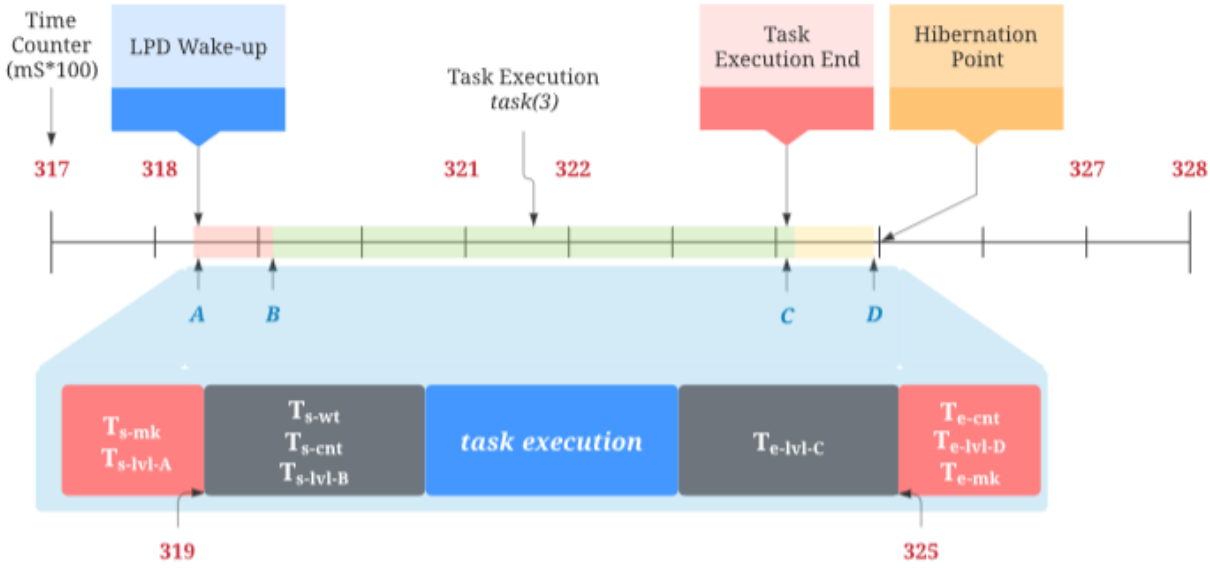


Figure 121: LPD Variable Measurement Points

The incoming charge rate gets measure twice per wake event cycle, once between the wake-up time and the first *Time Sync Count* reception. The second measurement gets taken just after *task* execution has finished and ends just before the final *Time Sync Count* message's reception.

The average charge rate provided by the incoming energy stream gets calculated by dividing the change in concentration over the period by the time interval.

Equation 62

$$S\Delta = \frac{[T_{slvlA} - T_{slvlB}]}{[T_{scnt} - T_{smk}]} \;\&\; E\Delta = \frac{[T_{elvlC} - T_{elvlD}]}{[T_{emk} - T_{ecnt}]}$$

Figure 123 below illustrates that the points used to measure the incoming charge rates are outside the *task* execution area, giving the best chance to make a realistic measurement.

122

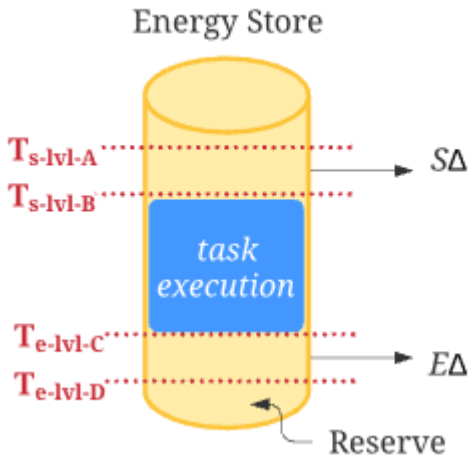


Figure 122: Charge Rate Measurement Areas

During hibernation, the incoming charge rate is measurable using the period between two consecutive *tasks*, from the $\mathbf{T_{e\text{-}mk}}$ point to the following $\mathbf{T_{s\text{-}mk}}$ point and using the delta between $\mathbf{T_{e\text{-}lvl\text{-}D}}$ and $\mathbf{T_{s\text{-}lvl\text{-}A}}$. Energy store charge saturation during hibernation will invalidate this measurement. A basic check for $\mathbf{T_{s\text{-}lvl\text{-}A}}$ being less than the energy stores full value ensures integrity.

Figure 124 below shows a time-correlated Gantt type picture that illustrates the four LPDs running the WIMP stack executing their *tasks* against the global time-synchronised counter. The chart datapoints are derived from the downloaded log entries stored within the LPDs internal flash area.

123

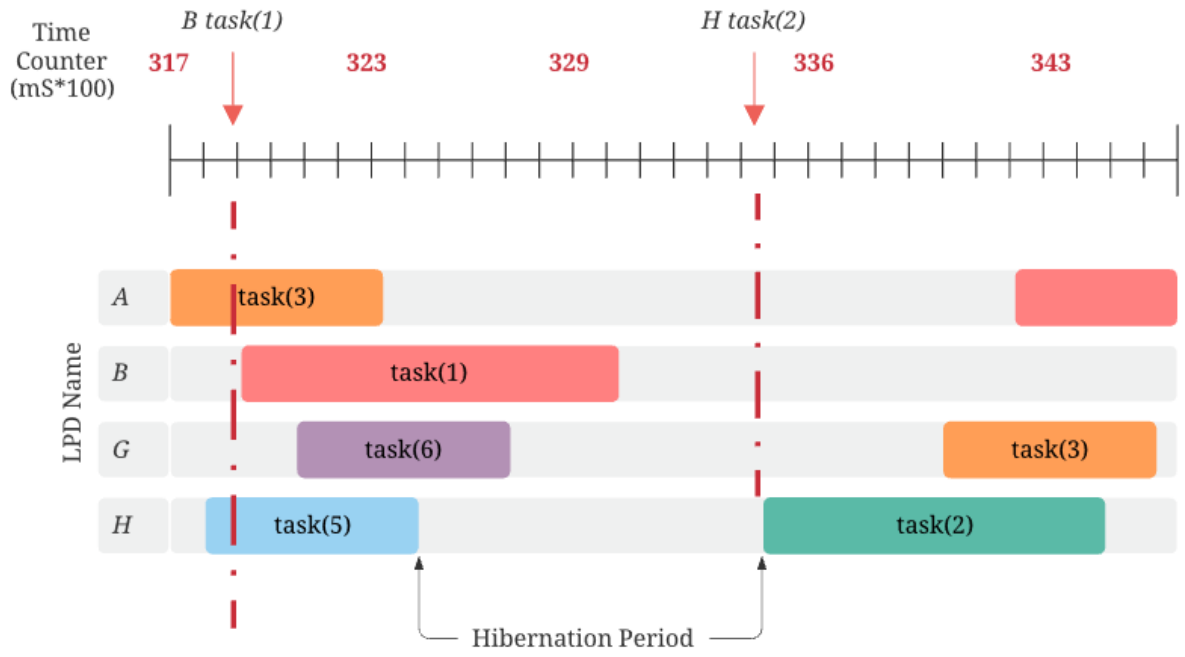


Figure 123: Correlated Execution Patterns

The LPD log entries, shown in Figure 125 below, contain historic execution records with time stamps and level readings. The data requires post-processing to correctly correlate the timepiece differences between the LPDs internal timekeeping and the global Time Sync Counter messages.

The LPDs keep an internal tick counter which gets used to time application software events. A division of this tick also gets fed into the WIMP stack as its periodic *Tick()* call.

To correctly correlate everything together onto the same timeline, an LPD translation function converts the local tick counter into milliseconds. The tick delta between $\mathbf{T_{s\text{-}mk}}$ and $\mathbf{T_{e\text{-}mk}}$ allow the total wake time to be calculated, and the other *task* internal measurement points are derived using similar delta calculations. All these values then get converted into their corresponding time values.

124

```
==================================
ID               : 5411
Time Sync Count  : 345412
Device           : B

Ts-mk            : 21545411
Ts-lvl-A         : 3072
Ts-wt            : 184
Ts-cnt           : 345412
Ts-lvl-B         : 3034
Ptasks           : 3,1,4
Pexe             : 3
Phib             : 426
Ppost            : 1,4
Te-lvl-C         : 1054
Te-wt            : 112
Te-cnt           : 345418
Te-lvl-D         : 999
Te-mk            : 21552441
==================================
ID               : 5412
Time Sync Count  : 345412
Device           : H
```

Figure 124: Downloaded Logs

The adaptive and fixed devices send zero values for the parameters which are not relevant to their implementation; this ensures messages from all devices are equal in size and effort.

Any device experiencing any kind of false-wake must still try and accomplish the *Time Sync Count* message wait before re-hibernation. If even this (dictated by the energy store level) procedure is not possible, then a last-ditch attempt is made to mark its internal log indicating that a false-wake has occurred.

Test Schedule

Ten initial test runs get performed where constant energy streams are permanently available and stable for harvesting. A further ten test runs followed where the energy streams were both relocated and varied in output power.

Each test run executes for 24hours, and the logging incorporated within both the *tasks* themselves and the modifications made to the stack implementation were stored in the device's persistent storage for later retrieval and analysis.

Every device had the same energy store level setup, the same harvesting setup, same thresholds and same internal self-calibration results, the devices themselves are identified as single characters from '*A*' to '*H*.'

When an experiment reaches completion, the devices get collected together, and their logging data retrieved from internal flash storage for continued analysis.

A timeline gets drawn representing the periodic transmission of the count value from the external radio syncronisation device. Each device gets its wake and work periods transcribed to this same timeline, and at each point in time, the energy gets trended against incoming strength vs device expenditure rate.

As when woken, the devices can also keep their own timekeeping. It highlights how the planner's algorithms perform in relation to the decisions other LPDs made, precisely what gets scheduled, and how accurate the schedule predictions are to the actual predictions.

125

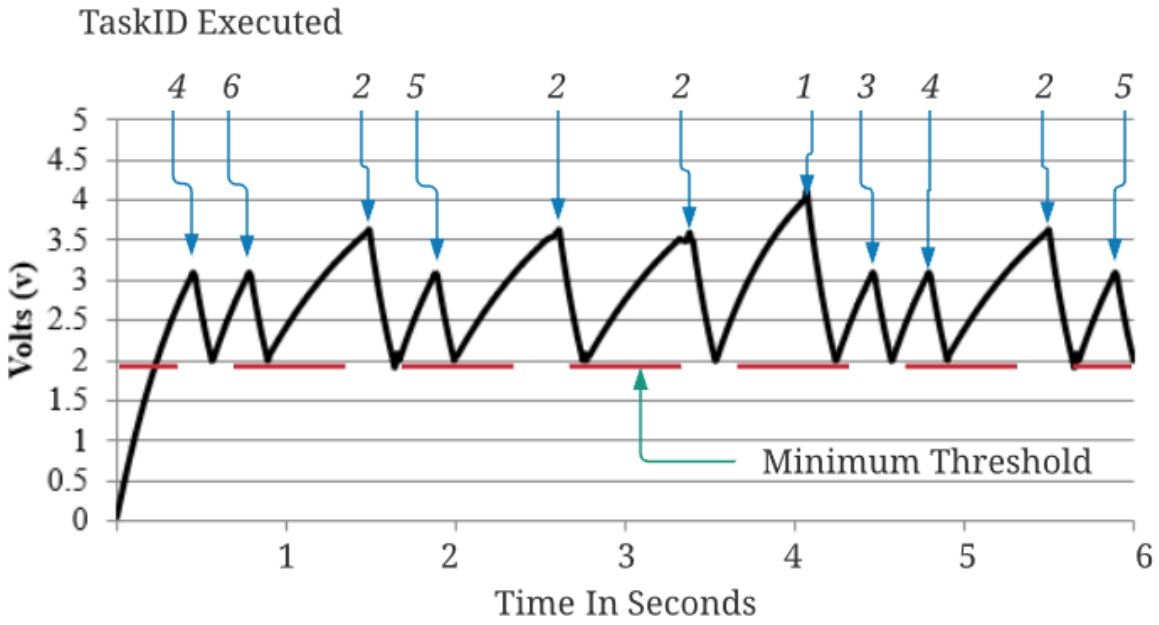


Figure 125: Charge vs Execution

An average random *task* execution run can be seen graphically in Figure 126 above. The graph line shows the charge level and rate of the energy store for a selected LPD. The numbering found above the peaks of this line represents the *task* the unit is starting to execute at that point. The rising slopes represent the charging hibernation periods, and the falling slopes the *task* execution energy consumption rate. The stacks minimum threshold, for this case, is 2V, leaving a good reserve in place for exception mitigation such as false-wakes.

The illustration clarifies that the stack's operation and performance follow the discussed design patterns very tightly and unlocks and delivers the extra efficiency predicted. *Tasks 4* and *6* both involve a 500mS delay, and *task 3* utilises an 800ms delay. The graph clearly illustrates the change in consumption prediction between the *task* types when their execution time arrives. The transition between *task 6* and *2* exemplifies the extra hibernation time needed to complete the longer executing *task* successfully.

The test runs visualised in Figure 126 above shows how the stacks expected operation performs with proper configuration and initialisation values, including successful completion of all self-calibration and *task* calibration procedures. The results, however, show that for the timespan the graph covers, efficiency in terms of maximising *task* execution based on incoming energy allowance is near perfect. The only way to increase output further is to look for additional cost and speed savings within the hardware and application layers and designs. From the point of view of intelligent energy management protocols, no more can be done to realise further worthwhile returns.

It does not prove easy to quantify the efficiency increase presented with this solution, as the operational aspects of the different strategies are so different. Figure 127 below shows a typical run of the LPD using an adaptive protocol; its charge time is similar in ramp, but the need to charge to set levels without underlying knowledge of the change in consumption work *tasks* experience between jobs cripples its maximum possible efficiency output.

126

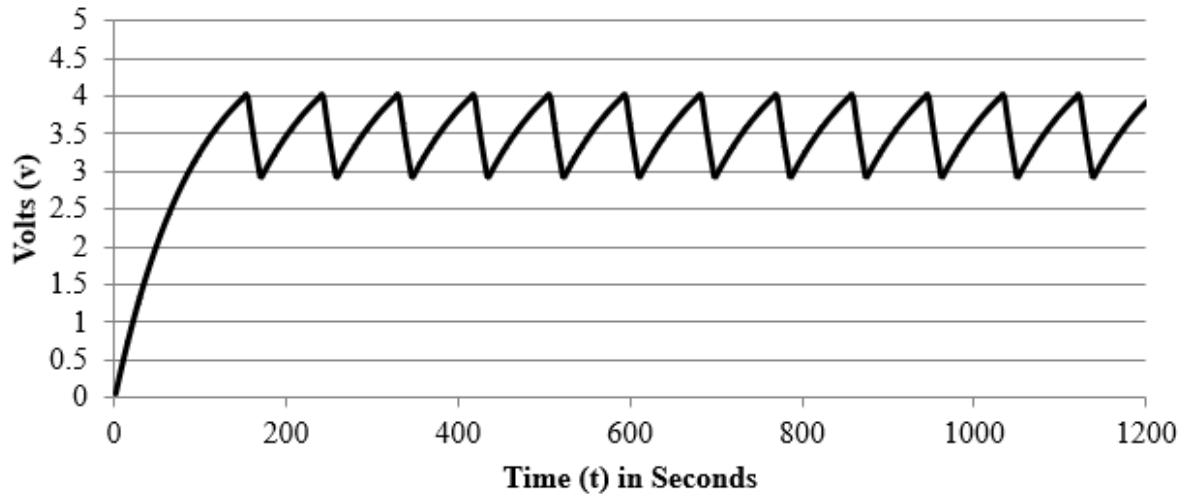


Figure 126: Basic Adaptive Comparison

Two areas of risk where inefficiencies can manifest themselves when using a properly calibrated WIMP strategy are false-wakes and error-correction settle time.

A false-wake is a situation in where a wake event has occurred intending to execute and complete a specific pre-costed *task* fully; however, the expected charge level is not yet available within its energy store. The typical choice made at this point is to immediately recalculate the hibernation time needed to complete the energy store charge and then go back to sleep for that duration.

The in-efficiency accrues when the false-wake procedure's energy cost eats into and uses the energy being harvested and reserved for a *task*.

127

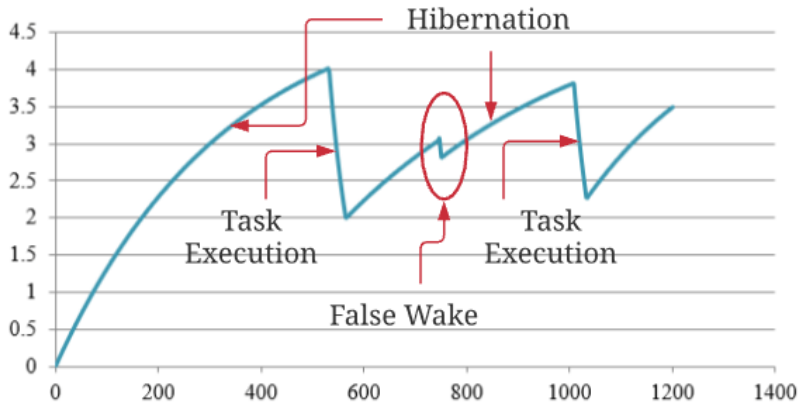


Figure 127: False Wake

Figure 128 above illustrates a false-wake occurrence. The procedure itself will consume both energy and time, both of which will affect the overall efficiency score.

False-wakes effects may be evident for a considerable time after their occurrences. Due to the *Planner* service slightly overcompensating in fear of provoking further consecutive false-wakes, the extended hibernation that follows may well have resulted in a more significant

accumulation of charge available than needed, as illustrated in Figure 129 below. It is visible that the execution of the *task* completed with considerable surplus left in the store. This unspent energy is not lost, simply included in the next *tasks* scheduling plan. The inefficiency, in this case, gets measured by the fact that the previous *task* could have executed a little sooner. The stack considers the unused energy at this point is due to a prediction error that needs smoothing out.

128

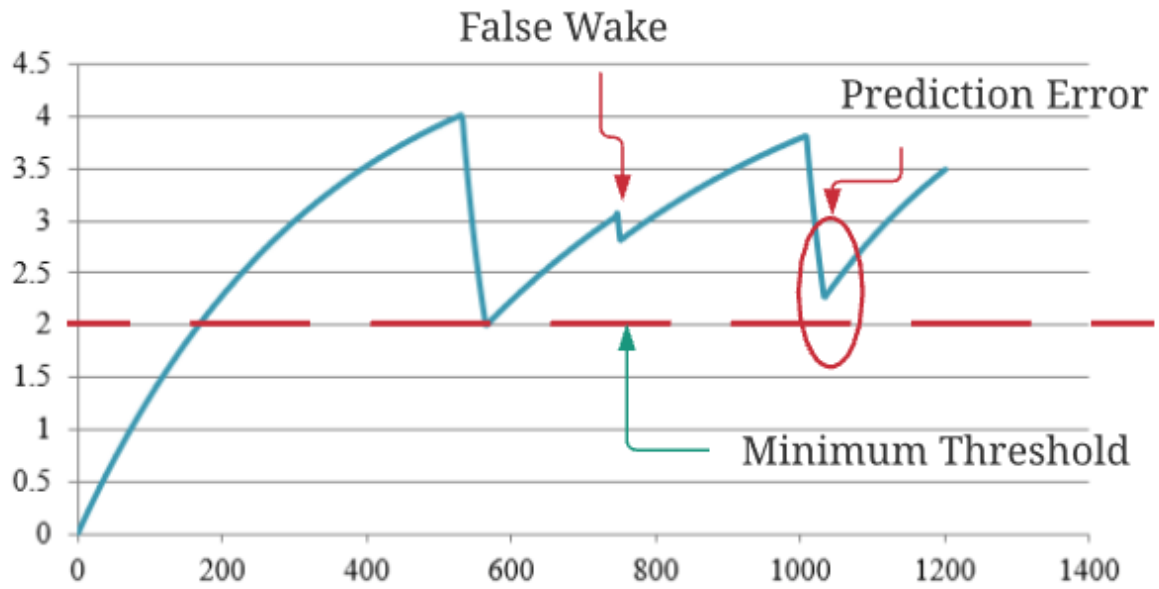


Figure 128: Prediction Error

For these tests, the planning service uses a basic PID (Proportional Integral Derivative) type closed-loop error correction algorithm to consolidate the inaccuracies where *task* prediction values do not align with actual execution values for whatever reason.

The loop configuration treats the *task* execution as the process variable and the time in hibernation as the control variable. The loop uses a mathematical function to control the

process variable via a feedback path which denotes the charge remaining in the store after execution has ceased.

Three coefficients are available to tune the loop's performance, and they all play an effect on the final efficiency score.

1. **Proportional Gain** – The “P” within the PID acronym. A proportional gain is the most fundamental scheme of control. As the process variable deviates further from the target, the control target gets increased. As the process variable approaches the target, the proportional gain applies considerable smaller change values on the control variable.
2. **Integral Gain** – The “I” within the PID acronym. An integral control scheme that gets based on the time the error is present in the system.
3. **Derivative Gain** – The “D” within the PID acronym. A derivative gain will monitor the error and determine the trend of the direction. If the error is decreasing, the derivative gain only has a small impact. However, as the error begins to increase, the derivative gain will start applying more considerable changes to the control variable.

Mathematically, the PID control algorithm used for the first runs is:

Equation 63

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

Where:

u(t)	PID Control Variable
K_p	Proportional Gain
e(t)	Error Value
K_i	Integral Gain
de	Change In Error Value
dt	Change In Time

Running a model over this equation with some generic input test data shows that the algorithm can produce a lovely ringing type settle pattern as shown in Figure 130 below; this will allow the LPD to quickly return into maximum efficiency operating areas when deviance gets encountered.

129

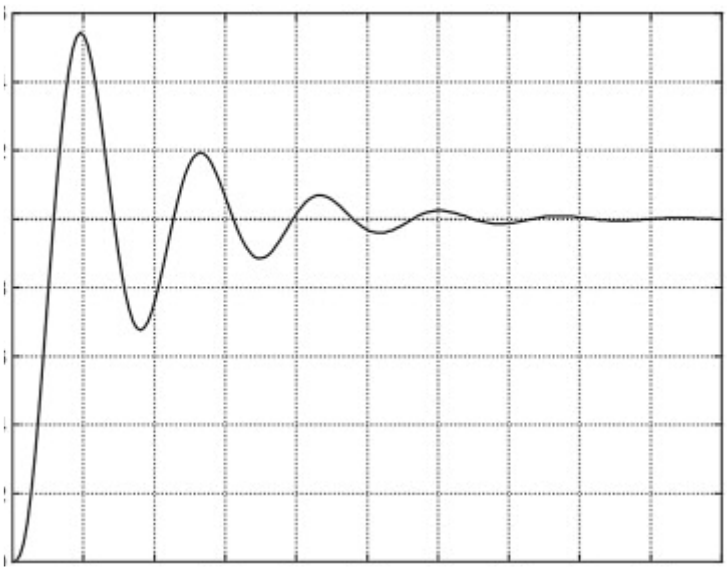


Figure 129: Error Correction Model

The LPD will not be able to replicate this smooth corrective path as its feedback part occurs only upon *task* completion, so the loop responsiveness towards the process is incredibly slow in real terms. The error correction effects in operation are highlighted in Figure 131 below, where the output shows the LPD executing the same *task* multiple times.

130

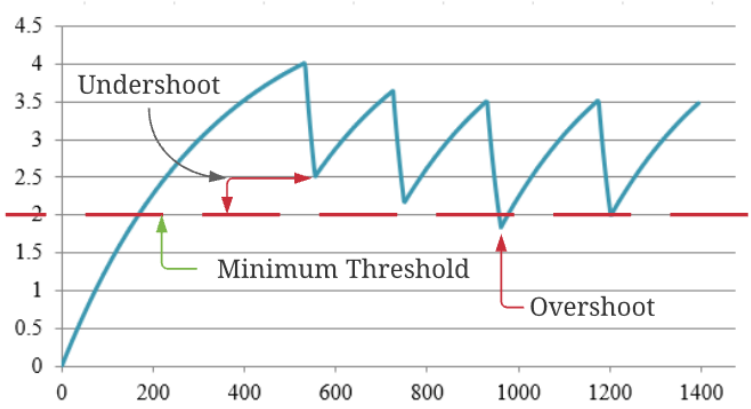


Figure 130: Adaptive Feedback

It can be seen that the correction gets applied at a much larger rate during the first feedback cycle. This tweak then gets followed by smaller change increments which ultimately created an overshoot situation. The test run's final feedback loop detected this breach and introduced a final correction that results in the *task* execution consuming near precisely the energy that got predicted for its execution.

Chapter Summary

The research platform was used to successfully host experiments that allowed implementing of the proposed WIMP software stack.

By aligning the hardware to self-discover the consumption requirements needed to execute a given *task*, it works towards providing a self-correcting charge-to-work operation strategy that allows maximum possible efficiency realisation.

Building on this single task-efficiency realisation, this just-in-time energy charge loop can be adapted to process a stream of *tasks* in sequence, each requiring its own pre-discovered consumption requirement, to truly exploit the idea of time dilation at its most granular levels.

It is clear from the results that once the system has adapted to its incoming energy position, the only inefficiencies introduced from adopting this method come from the error correction settle time and mispredictions.

There are a vast many paths to explore using these types of energy management strategies, including what types of planning algorithms perform best, which error correction algorithms are most efficient. All of this also concerns the actual application being addressed.

However, these experiments have shown that it is very possible to use basic intelligence in a way where real energy gains get realised, which critically sum greater than the overhead introduced in the process.

Chapter 8: Field Test, Oyster Monitoring (Original Content)

Progressing a joint research venture with the biological life sciences department, an exciting and relevant intelligent energy management design has been deployed into service.

The problem stems from trying to understand the reproductive cycles of oysters in order to gain the ability to calculate the best times for fishing them. The idea presented was to monitor the oysters' opening and closing in their usual habitat and use it, along with other datasets of similar measurements, to trend the habits.

A data collection hardware unit has been designed which allows field-effect sensors coupled with magnets to be attached to the oyster shells; readings are then periodically taken, which represent the angle of oyster opening at that point in time.

Various operational challenges hinder this problem, such as the harsh sea environment, the number of oysters needing concurrent monitoring, the fact that a permeant power source is not available, and the length of time the monitoring needs to be active.

Clearly, of interest for this particular research area is energy management which benefits significantly from being introduced into this kind of sensor deployment. To achieve the longevity of monitoring the team were looking for, coupled with the power needed to achieve the actual sensor readings, three critical points needed to be present:

- A large energy store
- The ability to re-charge this energy store when possible
- Precise control of the energy consumption

The unit's design allows simultaneous monitoring of up to sixteen oysters. The sample rate used to capture all of the sixteen monitored oysters' current opening state will be a dynamic variable mostly dependent on the current energy store levels considered alongside incoming

charge levels. However, a ‘requested’ value is accepted, which the unit will always strive to achieve were possible.

The unit is intended to be deployed for extended periods in potentially harsh and hard to access environments (Figure 132 below illustrates the hardened design characteristics). The unit's data only gets retrieved when the data collection size has been considered sufficient or another potentially problematic issue has arisen. For this reason, a sizeable slow-release energy store has been selected in the form of a rechargeable battery. The battery, in this instance, is continuously charged during daylight hours via a solar panel physically attached to the outside of the unit.

131

Figure 131: Environmentally Hardened Unit

The microprocessor selected to operate the unit has full control of powering down as many of its ancillary subsystems as possible to minimise its idle current footprint. It is also able to enter itself into a timed deep-sleep mode when its *tasks* have completed execution. Its analogue-

digital converters can accurately monitor the current charge level of the battery against a fixed reference voltage.

The Unit of Walton

To create an intelligent dynamic energy management algorithm for this problem scenario, the following key aspects need considering:

1. The current charge level of the energy store
2. The incoming charge rate delivered from the solar panel
3. The energy consumption hit obtained from waking up, taking 16 oyster measurement samples and a battery level sample, writing the data to non-volatile storage, and then going back to sleep.
4. The ability to dynamically control the sampling frequency with a priority that reflects the requested sampling rate for optimal data collection

The algorithm deployed to help reach the targets utilised a novel approach to classifying and organising blocks of energy using a unit of measure proposed as a '*Walton.'*

A *Walton* is a measurement quantity which can be defined as "*the minimum amount of energy needed to wholly achieve the highest priority task*." In the Oyster monitoring case, a single *Walton* would equate to the energy needed to wake up, take the sample, write the data to non-volatile storage and go back to sleep. This sampling sequence is defined as the 'main and highest priority *task'* that the unit is designed to, and has to, achieve. Any other *tasks* that have to be accomplished by the unit will have their energy needs measured in multiples or fractions of this now defined single *Walton* unit.

Now the *Walton* has been defined and quantified for this particular solution; the measurement unit gets used to further characterise the other components in the system. Examples of these categorisations include how many *Waltons* the battery can store when fully charged and the number of *Waltons* per hour average obtained from the solar panel. The *task* can also break down into *milli* or *micro Waltons*, for example:

1. Wake up and initialise: 0.15 *Waltons*
2. Take samples: 0.35 *Waltons*
3. Write to non-volatile storage: 0.35 *Waltons*
4. Post-process: 0.10 *Waltons*
5. Sleep: 0.05 *Waltons*

The algorithm that controls the wake cycle and *task* execution cycle can now use this *Waltons* unit to predict both the optimal times to wake up and perform its *tasks*, keeping in mind the operator's requested sampling rate. It is agreed in this particular case that if the battery cannot retain charge to accomplish successful samples at the requested sample rate, then slowing down this sample rate is the primary objective. Obtaining fewer consistently spaced samples is better than random and abrupt, lengthy stoppages altogether, which will result in large gaps in the dataset where the energy in the store has completely depleted.

To accomplish this, the overall input and output *Walton* rates need to be carefully monitored and fed back into the wake/work/sleep loop so these decisions can be made well in advance of entering any type of critical depletion phase.

As of writing, seven units have been deployed into the field and actively monitor just over 100 oysters. The units store the sampled data onto internal SD Card storage and require an operator to visit the individual units at regular intervals to retrieve and download the datasets for their research analysis.

The next phase of this project will add several other subsystems to the package, including a 4G modem and mesh communication, and this also provides a perfect opportunity to put this research into practice in a real-life situation.

The modem will allow the units to automate data delivery, resulting in a truly autonomous product that requires very little maintenance to provide very long term unattended use.

This radio medium also allows us to experiment with cooperative energy management protocols. The idea of cooperation allows us to provide a truly impenetrable data sampling solution with extremely high uptime and MTBF levels (Mean Time Before Failure).

As an example, let us consider just two of the units working together cooperatively. We know that each unit can monitor sixteen oysters, so if we assume that we have thirty-two oyster's setup, we are only interested in measuring sixteen of them in any one period. Here we can place the second unit in extended deep sleep mode with all sensor reading subsystems deactivated. Using the modem for the message exchange, the first unit performs the sampling *tasks* and transmits the data packets to the listening host, and it also uses the internal meshing communication protocols via the modem to transfer its current energy position in *Waltons* to the secondary unit. The second unit only wakes to retrieve and digest this broadcasted energy position information. If the first unit suffers prolonged higher energy expenditure vs energy input over time, the second unit can effectively 'take over' the job of the oyster measuring. Both the units now reconfigure themselves and swap duties. The first unit can now go into extended deep sleep mode and replenish its energy store while listening to the energy position broadcasts now being made from the second unit.

This cooperative mode of operation can significantly impact increasing a sensor networks level of fault tolerance. The inherent redundancy allows us to reduce the risk of any unknown influencing factors that often appear in sensor network measurement situations, such as weather conditions.

Deployment is managed by boat and a team with marine research experience; the current deployment location is in Mersea, Essex, UK (Figure 133 below).

132

Figure 132: Initial Deployment Configuration

The field-effect sensors are attached to the hard oyster shell using an underwater adhesive putty on the top part. On the bottom part, a neodymium magnet gets attached; this is shown in Figure 134 below. As the shell opens and closes, the microcontrollers ADC port can sample the tiny voltage changes, which is then post-processed before being stored.

133

Figure 133: Field-Effect Sensor

Groups of oysters are connected to the same unit and spaced apart when placed back into the habitat, shown below in Figure 135.

134

Figure 134: Oyster Grouping

As shown below in Figure 136, using holed crates dramatically simplifies the sensor attachment and deployment process.

135

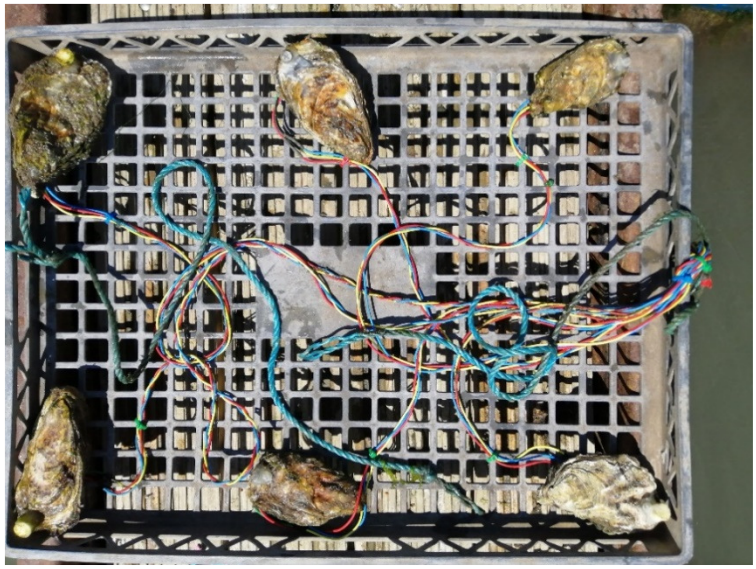


Figure 135: Sensor Submersion

The unit is anchored on a floating platform (shown below in Figure 137) to maximise its solar exposure, and the solar panel is rated above the energy usage, so weather conditions permitting, the units could be considered immortal.

136

Figure 136: Floating Controller Platform

The internal subsystems have five significant parts:

1. Battery (Energy Store - Figure 138)
2. Solar Panel (Energy Harvester - Figure 139)
3. Battery Charge Manager (Figure 138)
4. Automated Sampler PCB (Microchip PIC Based [107] - Figure 138)
5. Microcontroller (Linux based OS- Figure 138)

137

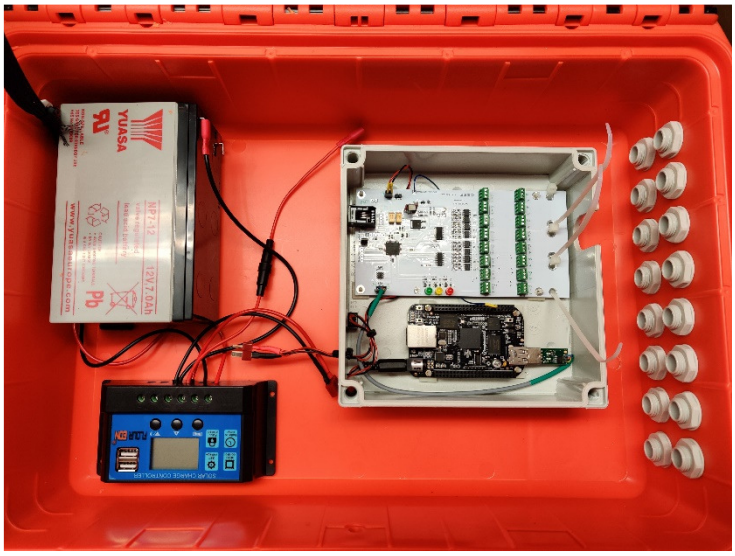


Figure 137: Controller Unit Internals

The size of both the battery and solar panel can be adjusted if needed.

138

Figure 138: Controller Solar Panel

139

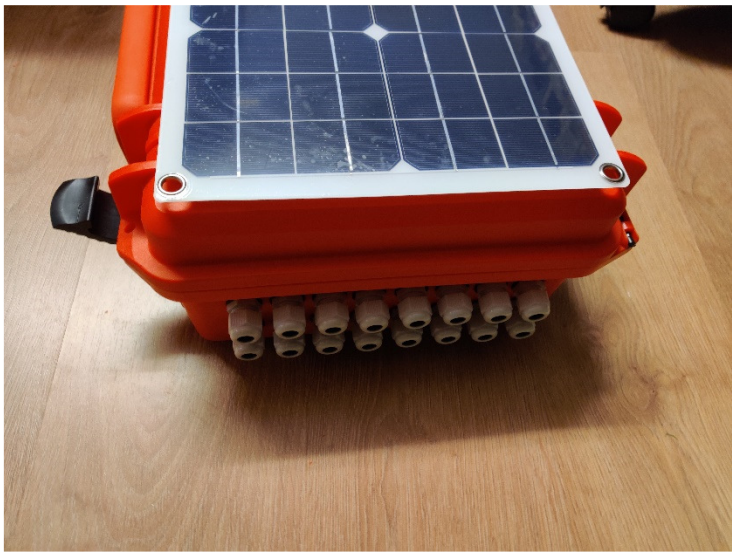


Figure 139: Controller Sensor Entry

The finished unit is sealed in a hardened flight case, all the cable entries are via compression grommets, and a splash-proof enclosure further protects the internal PCBs.

140

Figure 140: Internal Components

141

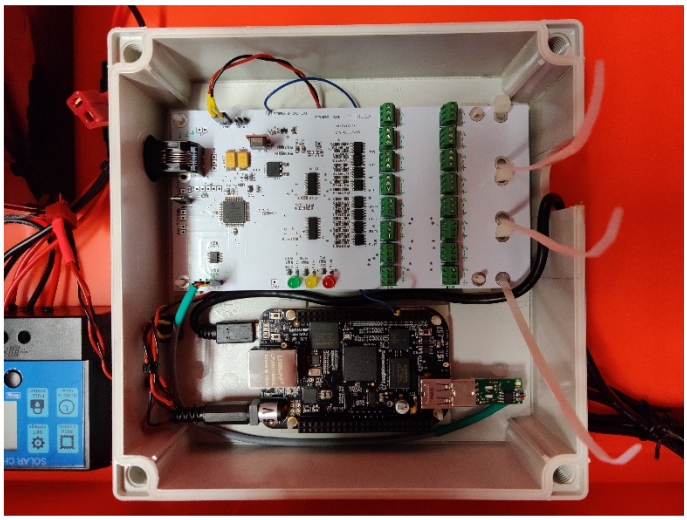


Figure 141: Aquisition PCB and CPU

142

Figure 142: Units Ready For Deployment

Unit Deployment

The units will spend the next two to three months collecting data and monitoring their own performance in being self-sufficient. There is a real-time clock in operation which stamps all of the log entries. The unit's data gets retrieved and analysed for battery charge/discharge logs, which then get cross-referenced with weather reports and temperatures. The syncronised dates and times provide correlation and further understanding of how the algorithms are themselves performing, how scalable the current solution is, and whether any upgrades or modifications will be needed to achieve maximum efficiency.

It would also be beneficial to further reduce the unit's operational energy requirements and increase the features and efficiency of the ambient energy harvesting facilities it currently has available.

Increasing solar panel size, and adding other transducers such as mechanical movement/wave generators to the setup may allow the battery's replacement entirely with supercapacitors. This component swap would allow the realisation of a machine that can live forever.

The battery is currently dictating the unit's life expectancy, as eventually, the efficiency of its charge/use cycle will degenerate to a level of un-usability. At this point, the unit's operation will be unpredictable and ultimately gets decommissioned until a battery replacement can be organised. Obviously, this situation needs to be avoided. The goal will be a unit that can achieve many decades of self-sustained operation in the field, where it continually transmits its data inshore and can maintain its existence solely on the energy it can harvest from its surrounding ambient offerings.

For this particular use case, the unit itself is engineered to a point where it can outlive the life span of all of the oysters it is set up to monitor.

The next iteration of development for this unit is to remove the need for operator attendance completely during its deployment periods.

The unit has realised this goal by utilising Bluetooth connectivity and an LTE modem for direct connection to a mobile network.

Implementing and using the stacks discussed in this thesis has allowed the unit to predict the energy costs and times needed for the sensor readings and implements the messaging capabilities via the mobile network, allowing direct delivery of the unit's datasets.

The unit now passes its energy status message to a monitoring controller connected to the internet. The mobile network talks directly with this LPN and alerts the operators if there are any issues that need addressing. It successfully monitors and adjusts its sample polling time based on the need for regular data measurements and the need to stay operational for as long as possible.

The BLE long-range [109] is also available, allowing the units to implement direct messaging and co-operative working to ensure its longevity is continued. Redundancy can easily be introduced and is limited only by budget restrictions.

Data is scheduled for transmission as quickly as possible based on the flow of incoming energy to outgoing usage, and the software can quickly detect and tag the environmental changes which have the biggest impact on their future survival.

Field Results

As a direct comparison, a previous incarnation of the controller unit that adopted a fixed wake/sleep operation strategy was placed side by side with a unit running the WIMP stack variant. Both units were left to run on three separate occasions in three different lighting conditions, and both units left to run until total exhaustion.

The standard unit failed after 14, 17 and 21 days; the new WIMP stack unit never reached a point of failure whilst testing in any of the runs, the experiment was closed after five weeks.

This field test was an exciting opportunity to put the theories out in the field, solving a real-life problem. The units had an opportunity to test an original theoretical idea and provide vital feedback in real terms regarding the proposed research questions.

The research operation's feedback was that the unit modifications made to the hardware incorporating the WIMP stack enabled their project to complete successfully. Previously, the unit failure rate prohibited progression.

Chapter Summary

The entire concept of intelligent energy management protocols has been working now in harsh real-life environment sensing solutions. The units are showing significant advantages over standard design implementations.

Analysing the oyster units' operational data shows they quickly adapt to their environments and keep nice regular communication intervals during most environmental changes.

The oyster project simplifies the problem quite nicely as it poses no movement possibility to ‘seek’ better energy sources and currently has only solar as its feeding supply.

Thus, it is quite clear how the protocols adapt to weather changes while maintaining nice regular communication intervals.

The cost of establishing a connection with a mobile network, sending a packet, and retrieving any pending messages can vary depending on current signal conditions, and which cell zones are currently operable. This situation proves difficult to predefine a *Walton* unit, so several units are kept as a reserve just in case a re-registration event with the cellular network is needed. This over-planning is a good example though of being able to adapt to unexpected kinds of environmental change.

It is clear from the results that further effort to push the efficiency of basic adaptive energy management routines yields excellent results. A balance is needed, though, ensuring the right amount of intelligence is wrapped around the fundamentally basic decisions the stack has to make.

It is impossible to cater to every situation; it is just as impossible to continually accurately predict something which can take on so many forms and introduce so many dynamic variables.

The art is closely comparable to waking up in the morning, and adjusting one's schedule to counter the unforeseen environmental changes the new day is bringing. Dynamic decision-making is needed; *tasks* get reorganised to fill the available working hours better as long as they retain their prioritisation.

When parallels get drawn, one knows what one wants to do the next day before sleep commences, but its no guarantee of what will *actually* occur. These fundamental realities are dealt with on the fly and usually take no effort. This decision process is transferrable; it does not need over-complication in terms of extensive forward decision algorithms or artificial intelligence, it is just a question of a few measurements and a little ‘dynamics’ in terms of job execution order.

The planning algorithms implemented are easily customisable to best suit the device's needs and complexity. The operating overhead of the stack running in this form is very low indeed, apart from the regular *Tick* requirement, it imposes virtually no interference with its host hardware. This lightweight and ease of implementation are vital if the stack gets deployed on a large scale, and it is large-scale deployment of these kinds of technologies working together where the real energy savings can be measured.

A configuration stage must be understood; however, the *calibrator* service's introduction greatly eases the need to have in-depth measurement procedures that have to be re-initiated every time a design change is made.

It is understood that the use of a battery in this field experiment counters a lot of the original target of this research piece. However, at the time of consideration, it was deemed that the benefits awarded from a real-life test environment and problem-solving situation to test the effectiveness of the protocol were too great to miss and still wholly beneficial.

Chapter 9: Integrating and Benefitting from Mother Nature (Original Content)

Parallels between S^3 Devices and Animate Lifeforms

The Machine That Lives Forever is a concept which describes a self-sufficient electronic device that harvests and manages its energy requirements, adapts to its environment, and survives by drawing parallels with various human and animal life-form models.

A basic life cycle is realised that follows the rest-feed-work process ingrained into every living creature. The critical challenges are finding and storing energy; when enough energy gets accumulated, a work 'task' gets executed, when the energy store is depleted, deep-sleep energy conversation hibernation occurs, allowing for recharge.

This cycle then repeats forever. It is known that if a devices program provides a set of hardcoded rules, these rules get followed without question. Having collections of the same devices all then exhibit the same predictable behaviour and share very similar outcomes until their programs are changed.

This chapter aims to document research based on a stigmergic 'survival by evolution and learning' algorithm, which draws influences from some of nature's phenomenal successes.

Throughout this research, several parallels between the concept of a self-sustaining microcontroller and real-life forms can and do get drawn.

When mother nature gets considered, it shows that life exists everywhere, from the smallest bacterial type through to the highly evolved and intelligent beings that have got to the tops of their food-chains. The more evolved and advanced life forms enjoy a somewhat better and more engaging quality of existence, but it must also be said that every life form shares the same survival and basic existence fundamentals.

When different forms of life are studied to see how they survive or deal with problems, there is always a variance of reaction displayed when the numbers increase. This variance ensures some kind of randomness, which may well result in decisions chosen that have a successful outcome instead of a fail.

The survival rate gets increased simply because *there is* some kind of randomness; in fact, it could be argued that success is directly related to randomness in this way.

It is desirable to retain and pass on these self-inflicted leanings to be better informed the next time a similar decision is required.

A collection of these learnings can grow inside the MCUs resources as it experiences and records the success or failure of some of the decisions it makes; these datasets form an experience set that can be merged and passed on with others through collaborative teaching type strategies.

The MCU code gets extended in various ways to explore introducing seeds of randomness within its operation as a primitive form of artificial learning and intelligence.

The WIMP stack has been further modified to enable basic virtual personality traits to influence management decisions.

Simulations have then indicated the outcomes and trends observed when introducing survival driven random choice type decision making.

The author acknowledges the sheer size and scope of this type of research and does not seek to offer an implementable solution that draws it to a conclusion. However, he feels that it is a natural extension to the previously proposed intelligent management protocol to explore this type of personality-driven survival decision framework, thus fulfilling the philosophical part of this PhD thesis.

At the highest level, several life traits can be identified, which appear to be present in all animate forms of life, these traits get identified as coronaries.

Coronaries

Survival

Col1 Stay alive: one of the primary objectives of the machine should be to stay alive. This fundamental should be at the cost of all other objectives. Strategies should include adjusting sleep times and predicting wakes to coincide with any future time-based energy resources. Work levels should be reduced or suspended completely to match actual or perceived resources. Cautious contingency should be employed.

Balance

Col2 Do an appropriate amount of work: Working too hard will deplete resources. Working too little will result in a failure to prosper. The optimum point lies in analysing incoming arrival versus outgoing energy consumption and implies instantaneous knowledge of any associated time constants. Considerable gains are possible for the correct selection of values.

Hardware wear and increased probability of failure will incur if the LPD continually overexerts; one must keep the expected requirements in mind to achieve correct balance and longevity.

Sustenance

Col3 Sustenance: Any MCU device will necessarily expend energy just in being. The average energy supply available must exceed the minimum energy required to survive, or life becomes impossible. Accumulation of energy to target long-term averages becomes advisable.

Starvation

Col4 Heading towards starvation: If the MCU goes under-voltage due to capacitor depletion (typically less than the device's specified absolute minimum), it can be deemed to have starved as it can no longer metabolise. This slow death should not come without warning, and its prediction should result in increased periods of sleep to conserve energy until more resource is available

Gluttony

Col5 Gluttony: If the capacitor remains in a charged state all of the time, even during active work execution, then the MCU is doing insufficient work for a given resource. It has become bloated, and the awake sleep ratio reduced to redress this.

Over-exertion

Col6 Over-exertion: doing too much work can result in instantaneously dipping below v_min. This burnout can be considered the equivalent of a heart attack. Resources are available, but the device has become corrupt and should be considered dead. The machine's reincarnation becomes difficult due to the energy hump (startup inertia) that needs to be overcome and serves little purpose if vital information has been lost. Overexertion can be avoided by careful time constant analysis.

Birth

Col7 Birth: Being born requires energy. The voltage should be greater than v_min. If the microcontroller sees under voltage, it will consume considerably more power than normal operation due to the MOSFETs' quiescent state. Under-voltage protection can mitigate the problem, but any circuitry will be parasitic. Alternatively, a one-off connection once v_min is exceeded permits birth.

Maturity

Col8 Maturity: A recently born device does not know the nature of any available resource unless hardcoded into its genetics. It may not even know the size of the storage capacitor it carries or the load it consumes, leaving it vulnerable to overexertion or starvation. As it begins to mature, it will learn about its resources and capabilities and any long-term changes.

Prosper

Col9 Prosper: In our context, a prosperous entity is one that achieves the maximum amount of work per unit of time for a given resource. Death would not be prosperous, nor would a poorly designed work strategy such as sleeping too long.

Accumulate Resources

Col10 Accumulate resources: A full accumulator is the best the machine can hope for and can be considered rich. For a capacitor, this means fully charged, i.e.5 time-constants (τ) or more. The energy arrival rate knowledge should govern the depth of any discharge cycle, but a fully charged device comes better placed to deal with the future than one that is partially discharged

or poor. Strategies should accumulate resources during times of plenty and attempt to maintain them.

Rest

Col11 Rest: In the rest or sleep mode, very little energy gets consumed. Long sleep durations are possible and more efficient, but they risk missing useful incoming resources that will not get used efficiently. Checking the incoming resource costs energy and is parasitic, so it needs using with caution. The energy consumed while sleeping is very low indeed, and it is assumed the accumulator has sufficient energy for the device to survive for days.

Awareness of Surroundings

Col12 Awareness of surroundings: The inbuilt accumulator in the form of a capacitor has two distinct time constants; the charging and the discharging. The discharging time constant will consist of the capacitor and a generally fixed load. The charging time constant will have the same capacitor but a time-varying impedance associated with the current source. Accurately determining these two-time constants is key to the efficient utilisation of the resource. A machine with an awareness of the source and load model can be shown to perform better.

Consider the Worst

Col13 Consider the worst: As a device heads towards starvation, has been sleeping for extended periods, but things do not appear to improve, it must begin to consider the worst. In the event of death, any vital information should be conserved for recovery if reincarnated. Writing critical information such as the time constants, sensor readings and critical data to ROM in the last throws of life should be considered, providing future restoration opportunities.

Learn

Col14 Learn: Learning gets considered in several ways: short term such as the instantaneous time constants and long term such as the diurnal of day and night, e.g., for a photovoltaic. Knowing information such as mean and deviation etc., makes for a better-informed entity with a greater chance of survival.

Burn Out

Col15 Burn Out: Failure to correctly manage resources can result in the stored voltage exceeding the maximum rating of the circuit; in the absence of over-voltage protection, the device will experience permanent damage and die

Diurnal and long term cycles

Thus far, we have been concerned with a resource used to charge a capacitor that, when appropriately charged, is discharged through a load. For efficiency, this requires acquired knowledge of charging and discharging time constants. However, it may not be the only cycle that is present. Corollary 1 (Col1) requires that the entity stay alive. With no knowledge of the system, this is difficult. Immediate strategies may include monitoring the rate of energy arrival and basing future decisions on this. Col2 requires appropriate work but should be subject to Col1. Without knowledge of diurnal or long-term patterns, the best available strategy is to accumulate, avoiding Col15 and discharge to a predetermined level, thus obeying Col2. However, if a Col4 event (famine) unexpectedly occurs, an immediate and automated response is to resort to Col10 and sleep for long periods hoping that the Col4 event passes.

If the Col4 event time is known a priori, then contingency can be made for it. Col9 requires the accumulation of resources. By curtailing the Col2 requirement to do useful work and instead of adhering to Col9 until Col4 is achieved. Then when the Col4 event does arrive, the entity is best prepared for a long Col10 period.

Stigmergy

Stigmergy is defined as a *"mechanism of indirect coordination in which the trace left by an action in a medium stimulates subsequent actions"* [127]. It can be considered a fundamental mechanism that helps both self-organisation and provides pathways allowing localised independent actions to stimulate globally coordinated activities.

Survival from the perspective of both organic life-forms and energy dependant self-sufficient LPDs entails a core of basic instinctive features that ultimately must be considered their highest priority goals (tasks).

1. Must be conscious of energy requirements (How much food it needs)
2. Must be conscious of energy usage (How quickly energy gets consumed)
3. Must rest to conserve and process energy (Sleep and repair/recharge)
4. Must replicate to pass on its DNA to evolve and survive

All life forms share these same fundamental requirements, regardless of other influences; these are the most critical factors that must hold true to survive as beings.

How individual species and beings go about meeting these goals are dependent on functional factors such as:

- Basic Instincts (Cognitive functions)
- Inherited instincts (Evolution, genetic inheritance)
- Imitation (Copying success patterns)
- Intelligent Learning (Actively pursuing new success patterns)

OCEAN

Personality traits greatly influence how these functional skills are obtained, used, and nurtured. When considered alongside task requirements, they assist in determining success, leadership

and natural selection. Personality traits are widely researched and categorised under five main headings referred to as OCEAN [128].

1. ***O***penness: willingness to try new things, to be vulnerable, the ability to think outside the box. (Imaginative, Creative, Curious, Perceptive)
2. ***C***onscientiousness: the ability to delay gratification, work within the rules, plan and organise effectively (Persistent, Predictable, Resourceful, Hardworking, Planner)
3. ***E***xtraversion: draw energy or "recharge" from interacting with others (Sociable, Friendly, Outgoing, Assertive)
4. ***A***greeableness: a construct that rests on how one generally interacts with others (Patient, Unselfish, Helpful, Considerate)
5. ***N***euroticism: encompasses one's emotional stability and general temper (Fearful, Timid, Unstable, Wary)

Seven Deadly Sins

Behaviours are further influenced negatively by the introduction of the famous seven deadly sins [129],

- Lust (Over replication)
- Gluttony (Neglecting the feed-work balance)
- Greed (Overindulging the feed-rest balance)
- Sloth (Neglecting the work-rest balance)
- Wrath (Preventing others from achieving balance)
- Envy (Over imitation)
- Pride (Ignoring lessons by others)

The randomness associated with acquiring and maturing a personality is what defines individuality. These individualities enable infinite possibilities of achieving the given tasks. When we couple this with the fact that traits of both parent's personalities are inherited by new-borns, along with potential mutations and changes, it is clear the richness of diversity is a key component towards the survival of the fittest.

Personality Mappings

Personality gets mapped to device functions to gain the biological success achieved in nature and apply it to the hardware realm. Personality traits are defined and used to represent chromosomes and build a DNA chain for an LPD. Each chromosome gets mapped to a cognitive or decision function within the LPD.

The primary cognitive functions are the ability to move in any direction at various speeds. Using radio, LPDs both talk and listen, providing vocal interaction. Energy harvesting based on RF scavenging, solar, thermal, chemical, electromagnetic, and electrostatic gets used (and considered different food types) to collect energy and feed. An energy store in the form of capacitors is considered the food store or stomach.

The task, kept in its simplest form, is to scavenge and find food. Success is measured by how efficiently and repeatable this task is achieved.

A personality algorithm processes, weights, maps, and control the LPD's functions and decisions based on the following (inconclusive list of) categories:

- Willingness to move
- The typical speed of movement
- Maximum possible speed

- Willingness to be a distance from parents
- Willingness to talk to others
- Willingness to listen to parents
- Willingness to listen to others
- Willingness to share information
- Desire for food
- The maximum amount of food before considered full
- The minimum amount of food before a rest period gets forced
- Desire to over-indulge
- Desire to work
- Desire to succeed
- Desire to sleep
- Fear of loneliness, losing contact
- Desire to lead
- Desire to find a partner
- Ability to understand topological surroundings

Life Cycle

Birth

An LPD initially executes a basic skeleton program; it contains only the necessary code to operate its cognitive and sensing functions. Structures are defined, allowing it to independently manipulate its personality and store vital information such as its age, parent's identity, life

success score, friends, and learning memory. LPDs are always aware of their energy levels, and the basic instinct to scavenge and find food is ingrained into their program.

1. A ‘new-born’ LPD is placed into the environment and allowed to charge its energy store for the first time. When appropriate threshold levels have been reached, it wakes from this hibernation and enables its radio, broadcasting a ‘cry’ (Basic instinct)
2. The ‘cry’ alerts two potential ‘parent’ LPDs (the two most successful LPDs from the list of LPDs that reply to it; or the single most successful LPD and its partner who happens to also be in range) who reply to the cry by sending it their own virtual DNA chains.
3. The new-born then creates its own DNA chain by randomly combining the received chains from its parents and permanently storing its parent’s identifications into its memory.
4. The new-born may or may not randomly mutate one of these chromosome entries.

When a child has discovered its parents, its parents also become partners if they are not already.

Learning

A new-born LPD initially learns as much as possible from mimicking its parent’s behaviour. Its parents communicate everything they learn immediately to their siblings. The sibling typically tries to follow their parents, but this is very much determined by their age and personality traits.

Learning involves executing a randomly chosen act and then analysing whether it allowed them to perform their task better. This result is stored in memory using a scoring system against the act’s entry. Acts with scores that grow high are kept in long-term memory and frequently used; acts with scores that get smaller eventually get forgotten.

A sibling LPD can immediately store a learning result, using a low score, received from its parents without performing the *task* itself. Depending on its personality, it may also (as it ages) apply the same learning storage process to information received from other (friend) LPDs. Friends get formed from the learning transfer process itself. The LPD learns something from a stranger; the LPD stores the stranger's identity in its memory and associates the new friend with a score. Being a friend increases the willingness to learn further from that LPD, and as more information gets exchanged from that LPD, the friend score increases, making it 'more of a friend.' Ultimately the top scorer is considered a best friend.

Learning acts consist of one of the following:

- Explore an unoccupied area for a food source
- Make a new friend
- Have a random nap and see if it has benefitted
- Change direction
- Move towards a friend
- Move away from a friend

Communication

Every LPD hears every message sent by every other LPD; however, it only 'listens' to the messages which come from its immediate family, friends, or leaders – and the willingness to listen gets driven by both its personality and the leadership position of the transmitting LPD. Other messages, stranger messages and shouts, are consumed to acquire new friends, information, and new skills based on personality.

The LPD sends messages as it discovers information or food sources by itself or learns new skills; typically, it addresses these messages to its parents and friends (personality driven).

LPDs can also ‘shout out’ a message, which then has more chance of being heard by strangers. This broadcasting feature is used in times of distress, for example, when its energy levels have reached a critical level.

Feeding

The LPD’s sole task is to scavenge, hunt out and consume enough food to survive. How the LPD achieves success in this is resultant on the boundaries described within its personality. As food sources get discovered, the LPD questionably communicates this to other surrounding LPDs, who then use it to determine whether the LPD is currently better positioned to feed than themselves.

Success gets declared upon discovering food supplies, so the LPD's internal success counter increases upon successful discoveries. This internal counter is used to measure the personality's success, which elevates that LPD within the group's hierarchy. This status level increases the chance of becoming the group leader and increases the chance of passing its virtual DNA into a new-born and evolving.

Success counters get reduced due to prolonged periods of being unsuccessful in finding and utilising food sources.

Situation Awareness

The LPDs use an RSSI based location discovery algorithm to identify their position and the position of the LPDs surrounding them. Each LPD maintains a MAC table that records the ID of a transmitting LPD and the RSSI level when the message was received. Periodically every LPD broadcasts its version of this table to every other LPD. This data exchange enables every LPD to build a virtual topology of the network. As LPDs move around and re-broadcast the

new RSSI data, the physical positions of each surrounding LPD are calculated by the LPDs, resulting in creating a geographical map. This mapping gets used to remember feeding locations, locate parents or friends, and explore unoccupied areas.

A leadership structure is assigned based on the success scores of the LPDs. The highest scorer (the most successful LPD) becomes the leader. A leader can command the group to do other specialised tasks, for example, forming geometry patterns, exploring particular areas, or prohibiting various actions. The leader is the only LPD who can communicate with the 'outside world.' If extraneous activities are required of the group, driven by the outside world, this must be agreed upon by communication with the leader.

Intelligence

The LPDs must intelligently control their energy storage and usage. They must keep constant track of the level and usage patterns to avoid situations where they starve themselves. They can learn repeatable feeding patterns allowing them to recharge and interact consistently. When needed, they must rest in a deep-hibernation state, and this may get induced by the fact that they have learned when food sources are scarce or running low on energy themselves. The length of the hibernation is also decided based on these variables.

The LPD's current energy levels determine the depth of the sleep; if they are adequate based on its personality traits, sleeping commences while still listening to other LPD's messages and updating its memory areas. However, if a critically low level is being measured, the LPD must shutdown every available system in an attempt to reduce its energy use to an absolute minimum in order to survive its drought.

Interaction

Interaction occurs when a need has arisen to prevent an LPD from starving; teamwork gets employed to identify the closest source of food for the LPD in need. The LPD then reacts immediately to try and benefit from the source.

Group activities also require interaction. If the master sends a command, then everyone must try and work together to solve the problem. For example, if the command is to form geometry, then the LPDs must use their situation awareness, leadership ladder, and memory to position themselves correctly.

Reproduction

After an LPD has reached a certain age, and based on its willingness to breed, it becomes capable of responding to the cry of a new-born. If it is successful in this process and becomes a parent, it has the opportunity to pass on its DNA to the new-born. The parent retains in its memory the identity of its new sibling alongside the other successful parent's identity. The other parent also becomes its partner. If either parent already has a different partner, this adulterous act causes the original partner to get removed from its memory, and that original partner's success score is lowered, along with it also forgetting the ex-partner.

Being a parent increases the speed of its own learning as it immediately consumes new learning discoveries from both the siblings and the other parent. The creation of a new child also increases the success score of both of the parents.

Personality Algorithms

Any movement or decision the LPD makes results from processing the desired action by the personality algorithm. Every possible command gets stored in a list; these commands are in the simplest form, examples of which are Move, sleep, communicate, listen, learn, etc. The general process is as follows:

- Randomly select a list of 5 to 25 possibly commands from the entire list; this may get weighted by decisions based on current needs (energy, tasks).
- Establish which personality traits affect each of the commands.
- Generate a score by adding (positive trait) or subtracting (negative trait) against each command for every trait it affects.
- Select the command with the highest score.
- Establish the extra parameters needed for the command (speed, direction, duration, destination, LPD).
- Select values for the parameters by again establishing links from the personality traits along with links from any previous learning results
- Execute the command.
- Repeat process when the command has finished.

Social Standings

LPDs will become alive, provisioned, and available at different undetermined times. The devices will need to communicate and integrate into their new surroundings, and a synchronisation procedure allows a communally accepted protocol.

The controller is a dedicated piece of hardware connected to the external world. However, the controlling task can be passed to, or taken on by, an LPD if desired. During such localised

control scenarios, the traditional connection to the outside world gets severed, and the LPD must revert to being entirely self-sufficient.

1) When a provisioned device wakes up, it sends a power level broadcast into the network or group.
2) A controller or message partner device will acknowledge this message with a specific response:

   Response time value of 0: this means that that device will become the master time device. It will assume this responsibility. It must then set up its next wake-up time to a self-determined value and send a message back to the controller, sharing this value.

   Response time value other than 0: this value is how many seconds until the next time the master device will wake up and process its messages.
3) The device will set its wake-up timer to coincide with the next wake-up event of the time master, thus allowing message processing to become synchronised.

After the devices have become time synchronised, a LED blink order and sequence gets established between them, as the devices will all receive the message at approximately the same time. This method is more energy-efficient than the overhead involved of having an onboard real-time clock and associated crystal running on every device.

Enable the Emotion Chip

The LPNs' ability to communicate peer-to-peer opens up the research platform to introduce autonomy and evolution patterns. This enhancement allows firmware extensions so the LPNs can successfully construct a leadership tree and work together to solve problems and exchange relevant information.

A significant portion of this research identifies various biological attributes found in other life-forms that directly impact their survival success. Different decision-making processes based on personality traits provide simple foundations, enabling the nodes to learn from these decisions and remembering how to apply them repeatedly to improve their quality of life.

The learning outcomes can then be passed to other family members or siblings (LPDs) to ensure they also have the best chance of survival. Merging these digital genetics of generations in such a way will allow an evolutional-based improvement approach touching upon the realms of ‘survival of the fittest,’ retaining the genetics best geared for the environment the most substantial chance of success.

Various personality traits (which can be considered chromosomes) are identified, weighted, and implemented within the firmware to influence individual devices' decisions and choices. This combination of these virtual chromosomes provides the LPDs with an artificial DNA sequence. The outcomes of the LPD's choices and decisions will be evaluated and fed back into its personality. Using this feedback method, successful decisions will, in effect, get rewarded, and their personality traits updated to reflect, unsuccessful decisions will oppositely get penalised. They will provide the foundation to be able to avoid negative decision-making.

As new LPDs introduce themselves into the group, they will have a randomised personality and receive the artificial DNA from other devices; the new device can combine two of these DNA sequences with its own, creating a unique and evolved ‘artificial individual.’

Joining the Group

With careful consideration of the security implications, an LPD can automatically assume the provisioner's role for another LPD. This idea takes the provisioning role away from the

provisioner, resulting in the prior knowledge of the types of nodes allowed to join the network having to get synchronised with the LPDs themselves.

However, this does allow segmented groups of devices to continue their work and communications and award alternative devices the ability to introduce stranger devices that may have been purposely deployed or perhaps even lost from a different segmented group.

Discovering Neighbours

For every message received by an LPD, a slightly more in-depth understanding of its direct neighbours and their immediate movements can be made. The message itself gets tagged with an RSSI value representing the signal strength of the received message. Although inherently inaccurate due to RF reflections, environmental conditions and colliding object permeabilities all uniting to detriment its accuracy, introducing simple techniques like including the transmitters power value used during transmission and knowledge on the LPD's own antenna characteristics and calibration points can allow an averaging algorithm to present a relatively consistent and probable answer.

The managed flooding protocol used by the mesh allows each node to dynamically build an internal routing table that lists the network nodes and how many hops a message needs to take to reach it. The hop count gets derived from the reception of multiple versions of the same message arriving from different peers and containing different TTL values.

Deciding Social Position

Once a topology of neighbouring LPDs has been built, knowledge of which LPDs are closest and which LPDs are available to initiate peer-to-peer communications gets continually updated.

To achieve a radio synchronisation between two devices directly, a decision of which device will get promoted to the role of controller denotes who will make the appropriate synchronisation timing decisions.

This requirement also holds when an LPD wishes to send a message to multiple devices all within its area of peer-to-peer communications. In this case, the group must communally agree on which LPD will become the leader.

If a group of devices becomes segmented from the network as a whole, for example, when two groups of deployed mobile sensors all move in different ways, eventually, some kind of network breakdown will occur as a whole. However, there will still be groups of devices available for peer-to-peer messaging. The loss will be the controller and potentially some friend devices.

Social position must also prevail in these types of situations, strengthing the choice of who will take on organising and passing work task messages, if needed. The need for the friend device to propagate the messages gets mitigated using direct peer-to-peer or peer-to-group synchronisation and messaging procedures.

The actual choice of who will take on the leadership role can be subject to various conditions that can range from the node holding the most number of directly contactable peers within its routing tables, the node with the most surplus energy, or the node which has simply survived the longest.

Synchronisation

Synchronisation follows the same procedure previously described for a wake synchronisation event issued by the controller. However, the data representing the scheduled wake time's decision gets transmitted by the new socially selected controller LPD.

The same LPD also has the power, if needed, to send task commands into the network, taking the place of the user control portion of the platform.

Simulation

The BluBot Mesh Controller Application (Figure 144 below) is updated to implement and prove the discussed algorithms within a simulated environment.

The simulation is an object-orientated implementation; the solar lamp (SUN), the energy beaming antenna (ANT), the environment container they sit in and the LPDs are all represented as uncoupled objects offering a standard set of interaction functions.

143

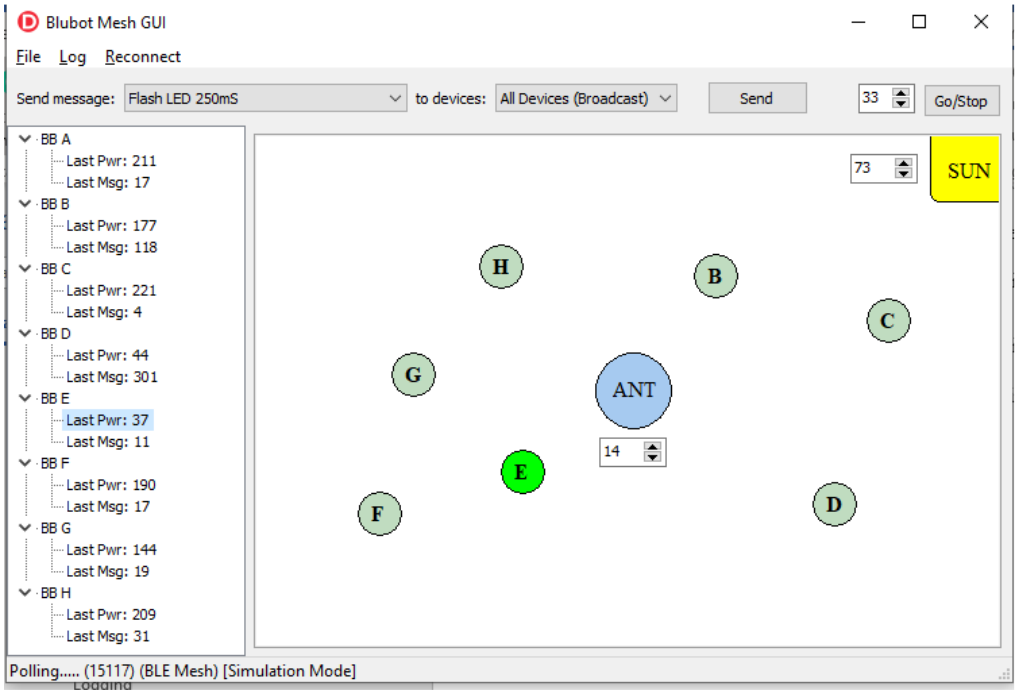


Figure 143: BluBot Mesh Controller Application

The SUN and ANTs power attributes inflict their energy offerings upon every LPD object in the vicinity based on their relative positions and distances. This periodically driven input is collected from the energy providers and delivered to the consumers via a simulation executive.

The executive sits-over, monitors, and progresses the simulation by moving pending messages between LPDs and piping the providers' energy streams into the LPD objects (Figure 145 below). The rate of this sequential LPD update occurs denotes the desired simulation speed.

144

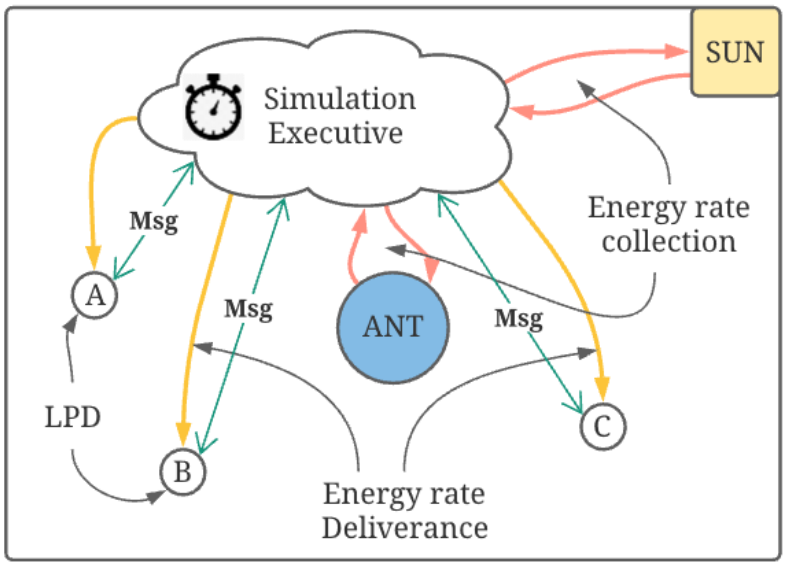


Figure 144: Simulation Executive Overview

The executive's basic flow of duties involves an iteration process triggered periodically by the simulation speed settings. When triggered, the executive will contact every power source in the simulation and request their current power value.

Then the executive continues by contacting every LPD in the simulation and initiates a control message exchange which:

- Computes the distance between the LPD and power source and attenuates the power signal accordingly.
- Updates the LPDs incoming energy amount
- Retrieves any pending messages it has and stores them in its internal broadcast list

- Sends all other pending messages currently stored in the internal broadcast list which have not yet been forwarded

The transferral of messages simulates a controller base store-and-forward setup, the alternative being an immediate transferral of new messages upon arrival to all other LPDs currently not in sleep mode.

Each LPD object requires two main parameters from the executive in order to live:

1. *Energy_In*, calculated and delivered by a decaying propagation type formulae driven by the distance between the energy source and the LPD
2. *Clock_Tick*, which gets derived from the frequency the *Energy_In* variable is updated by the executive

All the other object attributes are concerned with what the device is doing. The *Clock_Tick* member gets incremented every ‘time’ iteration and increments the individual devices' internal timekeeping device.

Energy_In represents the flow of energy delivered to the Device at any point in time. This attribute gets broken down into multiple instances, which each individually represents a different source of incoming energy available for harvesting.

So the parent will infer the *Clock_Tick* members period by passing a value to Energy_In once every second (or as the simulated time ratio value depicts), which in turn invokes the personality model of the individual devices to process its behaviour patterns.

Each LPD object internally maintains an energy store; this is a counter that gets incremented upon energy deliverance and decremented to represent any consumption incurred.

The LPDs themselves are aware they have a task to execute and control their sleep/wake patterns based on the parent's feed time being delivered. The *task* they are assigned comes with

a pre-defined energy prediction; the algorithms must balance the execution of the tasks against feeding and exploring. The LPDs personality registers get preconfigured and then perform adaption as previously described.

Each LPD has a different coefficient map which weight the following attributes around a midpoint of 50

- Desire for food: Controls the balance of the food/work ratio
- Fear of loneliness: Influences the LPDs cognitive functions based on the current positioning of other group members
- Desire to lead: Influences the types of message exchanges that occur and as to whether messages containing jobs and work-tasks get sent or not
- Desire to travel: Influences the LPDs cognitive functions based on how far wandering is tolerated before decisions to return to the starting point get made
- Fear of strangers: Influences if the LPD will actively avoid an approaching unknown LPD
- Desire to talk: Influences the message transmission and reception frequency
- Desire to sleep extra: Influences how much reserve gets kept in the energy store
- Desire to share: Influences if food source position information is to be included in message exchanges or not

The application interface gets used to predefine and tweak any of the personality registers and provide startup conditions (Figure 146 below).

145

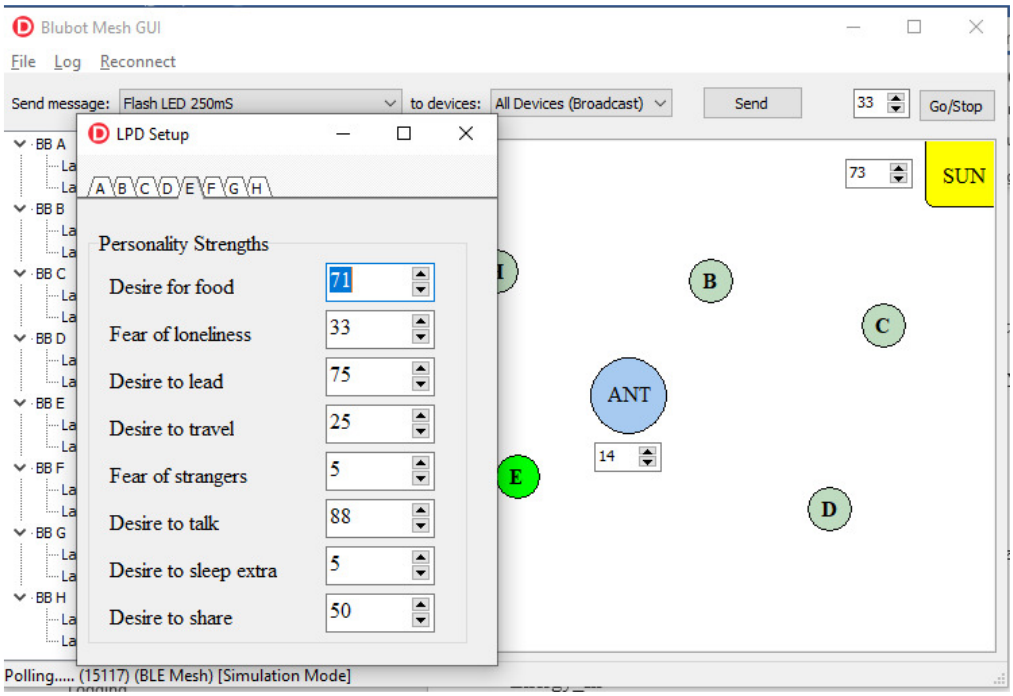


Figure 145: Personality Registers

The devices may or may not be enabled to be mobile, in which case they can control their visible X and Y screen coordinate position during their wake cycles, thus representing movement. Cognitive functions have a consumption cost in energy usage which can be manipulated based on the device's current environmental challenges. This cost is currently preconfigured, and the objects movement speed is constant, one pixel per iteration.

As the LPDs wake, the decisions they make based on using their available time are entirely self-justified. The actual choices made are based on logic ladders and code constructs which use the personality coefficients as weightings to control the decision process flow. The signal and energy flow of the LPD object are illustrated in Figure 147 below.

146

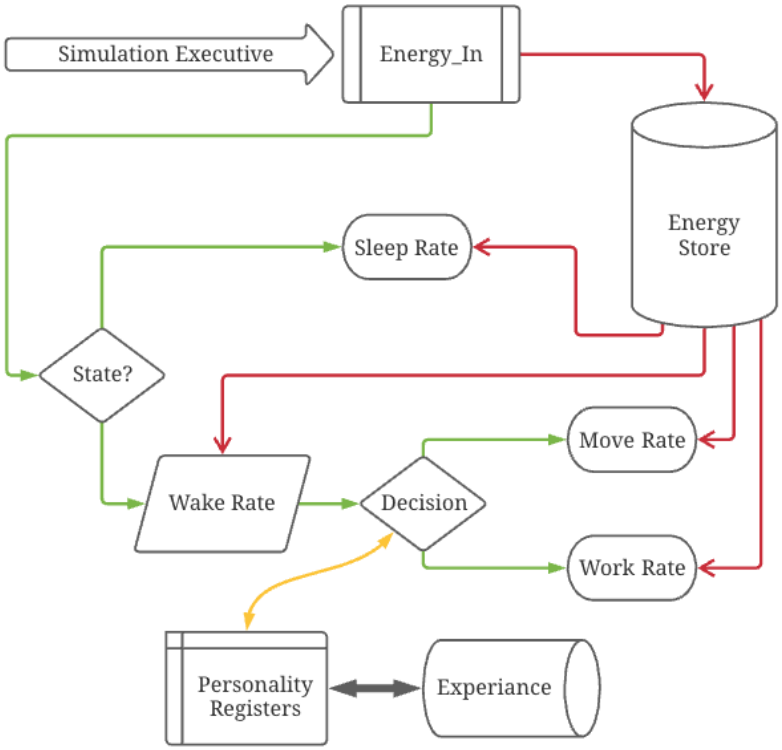


Figure 146: LPD Object Detail

Self Criticism

The LPDs also process the feedback loop, allowing the conscious realisation of the decision's cause-effect outcome. This feedback system can be considered a reward/punish based approach where success is measured in increased or sustained energy income and rewarded by increasing the weighting of making the previous collection of choices again.

Failure, on the other hand, where the energy income measurably diminishes due to previous choices, then their weighting is reduced to promote adopting a different set of choices in the future.

This long-term *decision* outcome resulting from this form of basic self-learning concept get influenced by the recording of weighting logs based on decision collections for success or

failure. These logs can look at individual choices themselves and rate them based on how useful they have been over the LPDs lifetime. Groups of consecutively made decisions, and their success rating also get logged and refined. This data forms an experience database that gets developed and grown over long periods through assimulating the results of the decisions made by the LPD. This database gets repeatedly referred to during decision-making situations, and its contents make further weighting changes to the personality traits to sway influence.

The energy cost of the *task* in hand will be subtracted from the device's energy store every time a part of the job gets executed. There is also an energy cost for the devices operational state; even during hibernation, energy gets consumed. The cost of this energy is simulated using time, and this is derived from the simulation executive. The sleep rate gets subtracted from the energy store during every clock pulse from the executive, and if the device is awake, then the awake rate is deducted.

The combination of the operational costs and *task* execution costs represent the device's total energy expenditure, and the rate of this compared with the rate of incoming energy is what the devices will use to spur their choices of how to utilise their awake time best.

As the personality weightings registers are randomly assigned at birth, followed by coercion and manipulations from the devices parents, it is interesting to note the diversity of operation which results.

Messages

To observe the benefits of sharing learning experiences, a simple message payload structure captures the necessary fields allowing visibility of another LPDs previous feeding success – if that LPD is willing to share it, of course.

The message payload takes the following form:

- Current Position
- Current Power
- Last Power Input Rate
- Closest Friend LPD
- My Leader LPD
- Last Feed Position
- Last Feed Rate
- Last Feed Duration
- Spoken Before?

Depending on the message receiving LPDs personality traits and weightings, the message's contents may or may not influence its preceding decision making.

Messages are generated every ‘multiple of Clock_Tick’ interval and can get set using the application configuration. The application logs all messages from all LPDs to a common output, as shown in Figure 148 below.

147

```
###########
RX >> C @ 18882291
I am C at 44,132
My power is 209
Last input was +2
My friend is A at 63,188
My Leader is A
I last fed at 132,111 +92 in 1433 ticks
###########
RX >> F @ 18882297
I am F at 214,312
My power is 188
Last input was +6
My friend is A at 63,188
My Leader is A
I last fed at 211,301 +131 in 2103 ticks
```

Figure 147: Simulation Message Log

A dump of the current experience database and personality registers is also possible for analysis.

Chapter Summary

The simulation shows results that mimic traits of AI systems; however, the goal is not to require the levels of resources and CPU cycles typically demanded by these types of solutions. The goal is to achieve success in survival with minimalistic amounts of effort and available IQ.

Simulating for the same set period using different personality weightings quickly illustrates the tiny coefficient differences needed to make very dominant generational changes.

Randomness also needs to be introduced into the energy sources power levels to visualise the full dynamic range offered by the different personality trait registers and how it influences feeding habits. However, if we ignore the time durations needed to get to the results, the general conclusion is that the more choices marked as successful are associated with choices made from traits tending towards greed and aggression. Adding more personality traits significantly influences these results, and the weighting adjustments after a period of settlement become relatively small indeed.

Simulation has proven to be the only viable way to get this experiment started due to the need to establish so many different coefficient values that potentially influence the LPDs survival outcome.

Using personality traits to induce decision-making randomness will considerably increase the survival batch's size upon operation conclusion. It ensures that devices will not all reach their demise in the same predictable way and enriches the possibilities of the tasks the LPD may be instructed to do, offering a type of redundancy that provides dependable results from multiple different pathways.

Chapter 10: Conclusions

This research has surrounded the idea of machines or devices, which can survive indefinitely. The driving force behind this being the loss of reliance on the mortal primary fuel cell.

Breaking the problem down into parts within this research has shown the common factor being the energy requirement. The relentless demand for some kind of fuel injection proportional in size to the task at hand. This constant need for fresh fuel supplies can only be managed, mitigated or even changed by *reduction*.

Reduction requires careful understanding and control of *consumption*. If consumption is running wild, then the reduction is impossible.

Reduction is best achieved by *removal* (total reduction), power removal, for example, or load removal. Hibernation, sleeping and idling are all forms of load removal.

The research has focused on different types of strategy in which this load removal attribute gets manipulated. Four different methods (fixed, variable, adaptive and intelligent) have been presented and evaluated.

A method of harnessing wireless energy and solar energy has been introduced to provide a real-world experimentation environment.

Using this energy supply platform, the energy management protocols were put into practice and used to prove that intelligent knowledge and control of the devices energy consumption requirements can result in near-perfect work/sleep ratios which can continually adapt to its changing environmental surroundings.

It was shown that manipulating time by dilations is the answer to maximise useful work task output against incoming energy quality.

So being able to scavenge for and harness our own energy requirements from the environment we live in and have the control and knowledge on how best to use what we have available, we can gain the best possible chance of successfully achieving our assigned tasks and sustain life in the form of existence itself.

Alas, this alone is not enough; being able to eat, work, and sleep is a great start, but to be successful and progress, one must also communicate. Of course, communication must also happen within these same environmental and energy constraints we have already become accustomed to.

A low-energy wireless mesh network gets evaluated, which facilitates inter-device communication, peer-to-peer communication and broadcasting. The interface was encapsulated within the energy management interface allowing hibernation cycles to synchronise with radio reception events, allowing the smallest consumption costs for activation.

So now, with the full skillset available, eating, sleeping, working, talking, the only thing left to do is survive—survival of the fittest.

The research took ideas from natures evolved and proven survival tactics—deployment in large numbers with natural pairings, groupings and randomness.

Associating these attributes as personality traits and using them as pivot points to apply random levels of weighting effectively influencing their long-term survival chances for the better.

Taking this a stage further, a feedback loop gets introduced where the results of the decisions the device is making are analysed and scored. This score then gets fed back into the personality weighting registers and used to mature and tweak the device's characteristic traits.

Storing the result of a particular decision and other influencing variables visible simultaneously allows the creation of a long-term memory store that can be nurtured and developed over a prolonged period of decision-making. This information store then allows the device to refer to its own experience for guidance when confronted with a decision to make.

Finally, taking these learnings and passing them on to close partners and siblings to provide them with a survival advantage of some sorts allows the benefits and successes of learned experiences to be passed across generations of devices, allowing them to evolve into more powerful, more intelligent entities.

The research has presented all of these personality-based improvements within a simulation engine that gets used to test the viability of deployment. It shows that introducing various pivot points used to control a decision flow allows more natural types of group hierarchy to develop, and a leader device quickly emerges. The simulations also show the speed gains associated with introducing new devices into an existing environment in terms of adaption after getting helped along by a parent device sharing its knowledge.

Further Research

This research has delivered new ideas and with it brings further research areas and questions. The conclusions of the experiments and simulations ran throughout this study all show that real gains are available. As always, with electronics, the quick summation of all the tiny parts makes a significant result.

Within the basic implementation of the WIMP stack, the planning algorithm plays a big part in how much forward-planning and multitasking can be performed per wake cycle. Further research is needed here to evaluate various planning and scheduling methods.

This possibility is also very much true for the error correction feedback algorithms found within the *Power-Trender* service. The settling time needed to correct prediction errors can be improved considerably. This area of operation has an enormous impact on efficiency.

The coronary view of existence clearly shows significant promise in providing basic survival intelligence needs, and this potential needs further investigation. The world is full of life forms that have mastered survival skills with very little perceived intelligence, contrary to the AI processor and power-consuming algorithms that repeatedly cause implementors to consider the gains not worth the overhead.

The sole purpose of the platform designed, developed, and implemented within this research is to test new and original ideas and proposals relating to promoting low-power device self-sufficiency. The platform, therefore, is more than capable of continuing research within this arena and can adopt a real implementation of the personality trait engine allowing observations outside of the simulation environment.

References

[1] T. W. Versloot, D. J. Barker and X. O. One, "Optimization of Near-Field Wireless Power Transfer Using Evolutionary Strategies," in *The 8th European Conference on Antennas and Propagation*, Netherlands, 2014.

[2] J. Wang, S. Ho, W. Fu and M. Sun, "Analytical design study of a novel WiTricity charger with lateral and angular misalignments for efficient wireless energy transmission," *Magnetics, IEEE Transactions on,* vol. 47, no. 10, pp. 2616-2619, 2011.

[3] A. Mahmood, A. Ismail, Z. Zaman, H. Fakhar, Z. Najam, M. Hasan and S. Ahmed, "A Comparative Study of Wireless Power Transmission Techniques," *Journal of Basic and Applied Scientific Research,* vol. 4, no. 1, pp. 321-326, 2014.

[4] O. Rönnbäck, "Optimization of Wireless Power," Luleå University of Technology, 2013.

[5] T. Imura and Y. Hori, "Maximizing air gap and efficiency of magnetic resonant coupling for wireless power transfer using equivalent circuit and neumann formula," *Industrial Electronics, IEEE Transactions,* vol. 58, no. 10, pp. 4746-4752, 2011.

[6] V. Jiwariyavej, T. Imura and Y. Hori, "Coupling Coefficients Estimation of Wireless Power Transfer System via Magnetic Resonance Coupling using Information from Either Side of the System," in *International Conference on Broadband and Biomedical Communication*, Australia, 2012.

[7] D. Chaurasia and S. Ahirwar, "An Optimal Parameter Estimation Technique for Wireless Electricity Transmission," *Research India Publications,* vol. 3, no. 1, pp. 1-9, 2013.

[8] V. Marian, C. Vollaire, J. Verdier and B. Allard, "Rectenna circuit topologies for contactless energy," HAL, Eindhoven, 2011.

[9] A. Etinger, M. Pilossof, B. Litvak, D. Hardon, M. Einat and B. Kapilevich, "Characterization of a Schottky Diode Rectenna for Millimeter Wave Power Beaming Using High Power Radiation Sources," in *12th Symposium of Magnetic Measurements and Modeling*, Czestochowa–Siewierz, 2016.

[10] H. S. Khaliq, "A High Gain Six Band Frequency Independent Dual CP Planar Log Periodic Antenna for Ambient RF Energy Harvesting," *2017 Progress in Electromagnetics Research Symposium - Fall,* pp. 3024-3028, 2017.

[11] A. Khemar, "Design and experiments of a dual-band rectenna for ambient RF energy harvesting in urban environments," *Iet Microwaves Antennas & Propagation,* vol. 12, no. 1, pp. 49-55, 2018.

[12] J. Liu and X. Y. Zhang, "Compact Triple-Band Rectifier for Ambient RF Energy Harvesting Application," IEEE, 2018.

[13] S. Mekid, "Energy Harvesting from Ambient Radio Frequency: Is it Worth it?," *Arabian Journal for Science and Engineering,* vol. 42, no. 7, pp. 2673-2683, 2017.

[14] A. Bindra, "Energy Harvesting: A Sustainable Ambient Source to Ultralow Power Devices.," *Ieee Power Electronics Magazine,* vol. 4, no. 4, pp. 4-8, 2017.

[15] S. Lee, “Opportunistic wireless energy harvesting in cognitive radio networks.,” *IEEE Transactions on Wireless Communications,* vol. 12, no. 9, pp. 4788-4799, 2013.

[16] D. L. Andrews and e. Inc., “Energy harvesting materials,” *Hackensack, NJ, World Scientific Pub. Co.,* p. 388.

[17] S. Beeby, “Energy harvesting for autonomous systems.,” Artech House Series Smart Materials, Structures, and Systems.

[18] Ottman, G. K., et al, “Adaptive piezoelectric energy harvesting circuit for wireless remote power supply.,” *IEEE Transactions on power electronics,* vol. 17, no. 5, pp. 669-676, 2002.

[19] N. Muensit and E. Inc, “Energy harvesting with piezoelectric and pyroelectric materials,” Materials science foundations.

[20] M. Z. Chaari, H. Ghariani and M. Lahiani, “Aquire Energy from the Radiation Emitted by the Compact Fluorescent Lights,” vol. 3, no. 3, 2018.

[21] T. Thabet, J. Woods and C. Uk, “An Approach to Calculate the Efficiency for an N-Receiver Wireless Power Transfer System.,” Essex University, Colchester, 2016.

[22] P. Nintanavongsa, U. Muncuk, D. R. Lewis and K. R. Chowdhury, “Design optimization and implementation for RF energy harvesting circuits,” *Emerging and Selected Topics in Circuits and Systems, IEEE Journal,* vol. 2, no. 1, pp. 24-33, 2012.

[23] V. Dyo, T. Ajmal, B. Allen, D. Jazani and I. Ivanov, “Design of a ferrite rod antenna for harvesting energy from medium wave broadcast signals,” *The Journal of Engineering,* vol. 2013, no. 12, pp. 89-96, 2013.

[24] A. Kurs, A. Karalis, R. Moffatt, J. D. Joannopoulos, P. Fisher and M. Soljačić, "Wireless power transfer via strongly coupled magnetic resonances," *Science,* vol. 317, no. 5834, pp. 83-86, 2007.

[25] P. Groap, "Powerharvester Receivers," [Online]. Available: http://www.powercastco.com/test566alpha/wp-content/uploads/2009/03/powerharvester-brochure.pdf. [Accessed 20 08 2018].

[26] P. Group, "P2110B Datasheet," [Online]. Available: http://www.powercastco.com/test566alpha/wp-content/uploads/2009/03/p2110b-datasheet-v12.pdf. [Accessed 23 08 2018].

[27] Tecate Group, "Ultracapacitor & Supercapacitor Frequently Asked Questions," [Online]. Available: https://www.tecategroup.com/ultracapacitors-supercapacitors/ultracapacitor-FAQ.php. [Accessed 06 06 2019].

[28] Pierre Mars, VP of Quality and Applications Engineering, "Coupling a Supercapacitor with a Small Energy Harvesting Source," 24 01 2012. [Online]. Available: https://www.eetimes.com/document.asp?doc_id=1279362#. [Accessed 06 06 2019].

[29] Electronics Tutorials, "RC Charging Circuit," [Online]. Available: https://www.electronics-tutorials.ws/rc/rc_1.html. [Accessed 29 01 2021].

[30] s. (https://physics.stackexchange.com/users/82906/shadi), "What happens to half of the energy in a circuit with a capacitor?," 05 06 2015. [Online]. Available: https://physics.stackexchange.com/q/187825. [Accessed 02 02 2021].

[31] Microchip Incorporated, “Atmel 8-bit AVR Microcontroller with 2/4/8KBytes In-System Programmable Flash,” 08 2013. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2586-AVR-8-bit-Microcontroller-ATtiny25-ATtiny45-ATtiny85_Datasheet-Summary.pdf. [Accessed 02 02 2021].

[32] J. Woods and T. Thabet, “Using Input Impedance to Calculate the Efficiency Numerically of Series-Parallel Magnetic Resonant Wireless Power Transfer Systems,” *Advances in Science, Technology and Engineering Systems,* vol. 3, no. 3, pp. 38-42, 2018.

[33] A. O. Kaka, M. Toycan and S. D. Walker, “Miniaturized stacked implant antenna design at ISM band with biocompatible characteristics,” *COMPEL - The international journal for computation and mathematics in electrical and electronic engineering,* vol. 34, no. 4, pp. 1270-1285, 2015.

[34] Bluetooth SIG, Inc, “Bluetooth Technology Website,” Bluetooth SIG, Inc, 2019. [Online]. Available: https://www.bluetooth.com/bluetooth-technology/solutions/. [Accessed 01 04 2019].

[35] M. M. Marian-Emanuel Ionascu, “Energy Profiling for Different Bluetooth Low Energy Designs,” in *The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, Bucharest, Romania , 2017.

[36] F. J. Ensworth and S. M. Reynolds, “BLE-Backscatter: Ultralow-Power IoT NodesCompatible With Bluetooth 4.0 LowEnergy (BLE) Smartphones and Tablets,”

IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES, vol. 65, no. 9, p. 3360, 2017.

[37] M. Siekkinen, M. Hiienkari, J. K. Nurminen and J. Nieminen, "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," in *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Paris, 2012.

[38] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine,* vol. 43, no. 9, pp. S23-S30, 2005.

[39] J. Lee, M. Dong and Y. Sun, "A preliminary study of low power wireless technologies: ZigBee and Bluetooth Low Energy," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, Auckland, 2015.

[40] N. Baker, "ZigBee and Bluetooth: Strengths and weaknesses for industrial applications," *IEE Computing & Control Engineering,* vol. 16, no. 2, pp. 20-25, 2005.

[41] J. S. Lee, Y. W. Su and C. C. Shen, "A comparative study of wireless protocols: Bluetooth UWB ZigBee and Wi-Fi," in *Proc. Annual Conf. IEEE Industrial Electronics Society (IECON)*, 2007.

[42] E. Georgakakis, S. A. Nikolidakis, D. D. Vergados and C. Douligeris, "An analysis of Bluetooth ZigBee and Bluetooth Low Energy and their use in WBANs," in *Wireless Mobile Communication and Healthcare*, Springer, 2011.

[43] R. Tabish, A. B. Mnaouer, F. Touati and M. Ghaleb, "A comparative analysis of BLE and 6LoWPAN for U-HealthCare," in *Proc. IEEE Gulf Cooperation Council (GCC) Conference and Exhibition*, 2013.

[44] J. R. Lin, T. Talty and O. K. Tonguz, "An empirical performance study of intra-vehicular wireless sensor networks under WiFi and Bluetooth interference," in *Proc. IEEE Global Communications Conf. (GLOBECOM)*, 2013.

[45] K. Shahzad and B. Oelmann, "A comparative study of in-sensor processing vs. raw data transmission using ZigBee BLE and Wi-Fi for data intensive monitoring applications," in *Proc. Int. Symp. Wireless Communication Systems*, 2014.

[46] M. Siekkinen, M. Hiienkari, J. K. Nurminen and J. Nieminen, "How low energy is Bluetooth Low Energy? Comparative measurements with ZigBee/802.15.4," in *IEEE Wireless Communications and Networking Conf. (WCNC)*, 2012.

[47] A. Dementyev, H. Steve, T. Stuart and S. Joshua, "Power consumption analysis of Bluetooth Low Energy ZigBee and ANT sensor nodes in a cyclic sleep scenario," in *Proc. Int. Wireless Symp*, 2013.

[48] C. Gomez, J. Oller and J. Paradells, "Overview and evaluation of Bluetooth Low Energy: An emerging low-power wireless technology," *Sensors,* vol. 12, no. 9, pp. 11734-11753, 2012.

[49] J. Higuera, E. Kartsakli, J. L. Valenzuela, L. Alonso, A. Laya and R. Martinez, "Experimental study of Bluetooth ZigBee and IEEE 802.15.4 technologies on board high-speed trains," in *IEEE Conf. Vehicular Technology (VTC Spring)*, 2012.

[50] K. Mikhaylov, N. Plevritakis and J. Tervonen, “Performance analysis and comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI,” *J. Sens. Actuator Networks,* vol. 2, no. 3, pp. 589-613, 2013.

[51] K. Shahzad and B. Oelmann, “A comparative study of in-sensor processing vs. raw data transmission using ZigBee, BLE and Wi-Fi for data intensive monitoring applications,” in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, Barcelona, 2014.

[52] K. Shahzad, P. Cheng and B. Oelmann, “Architecture exploration for a high-performance and low-power wireless vibration analyzer,” *IEEE Journal on Sensors,* vol. 13, no. 2, pp. 670-682, 2013.

[53] M. Imran, K. Khursheed, N. Lawal, M. O'Nils and N. Ahmad, “Implementation of Wireless Vision Sensor Node for Characterization of Magnetic Particles in Fluids,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, pp. 1634-1643, 2012.

[54] A. Dementyev, S. Hodges, S. Taylor and J. Smith, “Power consumption analysis of Bluetooth Low Energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario,,” in *IEEE International Wireless Symposium*, Beijing, 2013.

[55] H. Cao, V. Leung, C. Chow and H. Chan, “Enabling technologies for wireless body area networks: a survey and outlook,” *IEEE IComM,* vol. 47, no. 12, pp. 84-93, 2009.

[56] J. Lee, Y. Su and C. Shen, “A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi,” *IEEE IECONN,* pp. 46-51, 2007.

[57] Monsoon Solutions Inc, “Low Voltage Power Monitor,” 2012. [Online]. Available: https://www.msoon.com/lvpm-product-documentation. [Accessed 02 02 2021].

[58] Silicon Laboratories, “Bluegiga Bluetooth Smart Software Stack,” 2021. [Online]. Available: https://www.silabs.com/developers/bluegiga-bluetooth-smart-software-stack. [Accessed 7 2 2021].

[59] Texas Instruments Incorporated, “CC2450 Bluetooth Low Energy Wrieless MCU with USB,” 2021. [Online]. Available: https://www.ti.com/product/CC2540. [Accessed 01 02 2021].

[60] Texas Instruments Incorporated, “MSP430F2274 16 MHz MCU with 32KB FLASH, 1 KB SRAM, 10-bit ADC, 2 OpAmp, I2C/SPI/UART,” 2021. [Online]. Available: https://www.ti.com/product/MSP430F2274. [Accessed 01 02 2021].

[61] Texas Instruments Incorporated, “CC2530 Zigbee and IEEE 802.15.4 wireless MCU with 256kB Flash and 8kB RAM,” 2021. [Online]. Available: https://www.ti.com/product/CC2530. [Accessed 01 02 2021].

[62] FieldComm Group, “HART Communication Protocol,” 2021. [Online]. Available: https://www.fieldcommgroup.org/technologies/hart/hart-technology-detail. [Accessed 01 02 2021].

[63] Hart communication Foundation, “HART7 Specification,” HCF, 2007.

[64] T. Lennvall, S. Svensson and F. Hekland, “A comparison of WirelessHART and ZigBee for industrial applications,,” in *IEEE International Workshop on Factory Communication Systems*, Dresden, 2008.

[65] M. D. Yacoub, Wireless Technology Protocols, Boca Raton: CRC Press LLC, 2002.

[66] P. Hatzold, Digitale Kommunikation über Funk, Poing: Franzis Verlag GmbH, 1999.

[67] C. A. Balanis, Antenna Theory Analysis and Design, New Jersey: John Wiley & Sons, Inc, 2005.

[68] E. Mackensen, W. Kuntz and C. Mueller, "Smart wireless autonomous microsystems (SWAMs) for sensor actuator networks," in *SIcon 2004 Conference*, New Orleans, 2004.

[69] ZigBee Alliance, "ZigBee Specification 2005 0534744r05, Version 1.0," ZigBee Alliance, 2005.

[70] Texas Instruments Incorporated, "CC2540F256 2.4-GHz Bluetooth low energy System-on-Chip Datasheet SWRS084F," Texas Instruments, Dallas, 2010.

[71] T. I. Incorporated, "BLE-STACK Bluetooth Low Energy Stack," Texas Instruments, 2021. [Online]. Available: https://www.ti.com/tool/BLE-STACK#descriptionArea. [Accessed 01 02 2021].

[72] T. M. Wendt and L. M. Reindl, "Wake-Up Methods to Extend Battery Life Time of Wireless Sensor Nodes," in *IEEE Instrumentation and Measurement Technology Conference*, Victoria, 2008.

[73] J. A. Gutiérrez, E. H. Gallaway and R. L. Barret Jr., "Low-Rate Wireless Personal Area Networks," IEEE Press,, New York, 2003.

[74] M. S. Durante and S. Mahlknecht, “An Ultra Low Power Wakeup Receiver for Wireless Sensor Nodes,” in *2009 Third International Conference on Sensor Technologies and Applications*, Athens, 2009.

[75] G. Lu, D. De, M. Xu, W.-Z. Song and J. Cao, “TelosW: Enabling ultra-low power wake-on sensor network,” in *2010 Seventh International Conference on Networked Sensing Systems (INSS)*, Kassel, 2010.

[76] P. Dutta, M. Feldmeier, J. Paradiso and D. Culler, “Energy metering for free: Augmenting switching regulators for real-time monitoring,” in *SPOTS'08 IPSN Conference Proceedings*, 2008.

[77] Maxim Integrated, “MAX1724 1.5µA IQ, Step-Up DC-DC Converters in TSOT and µDFN,” 26 09 2017. [Online]. Available: 1.5µA IQ, Step-Up DC-DC Converters in TSOT and µDFN. [Accessed 02 02 2021].

[78] “Z-Wave, Safer, smarter homes start with Z-Wave,” 2021. [Online]. Available: https://www.z-wave.com/learn. [Accessed 02 02 2021].

[79] S. J. Danbatta and A. Varol, “Comparison of Zigbee, Z-Wave, Wi-Fi, and Bluetooth Wireless Technologies Used in Home Automation,” in *7th International Symposium on Digital Forensics and Security (ISDFS)*, Barcelos, 2019.

[80] Zigbee Alliance, “Project Connected Home Over IP,” 2021. [Online]. Available: https://www.connectedhomeip.com/. [Accessed 02 02 2021].

[81] Thread Group, “Thread, Built for IoT,” 2021. [Online]. Available: https://www.threadgroup.org/What-is-Thread/Thread-Benefits. [Accessed 01 02 2021].

[82] E. Mackensen and M. Lai, “Bluetooth Low Energy (BLE) based wireless sensors,” in *SENSORS, 2012 IEEE*, Taipei, 2012.

[83] E. Mackensen and T. M. Wendt, “Energy-Harvesting-basierte Energieversorgungen für drahtlose Sensor-Systeme,” in *Elektronik energy harvesting congress 2012*, München, 2012.

[84] Z. Feng, L. Mo and M. Li, “Analysis of low energy consumption wireless sensor with BLE,” in *2015 IEEE SENSORS*, Busan, 2015.

[85] P. Zenker, S. Krug, M. Binhack and J. Seitz, “Evaluation of BLE Mesh capabilities: A case study based on CSRMesh,” in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, 2016.

[86] Bluetooth SIG, “Introducing Bluetooth Mesh Networking,” Bluetooth SIG, 2019. [Online]. Available: https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking/. [Accessed 03 06 2019].

[87] A. Chiumento, B. Reynders, Y. Murillo and S. Pollin, “Building a connected BLE mesh: A network inference study,” in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Barcelona, 2018.

[88] H.-S. Kim, J. Lee and J. W. Jang, “BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices,” in *2015 3rd International Conference on Future Internet of Things and Cloud*, Rome, 2015.

[89] Y. Murillo, B. Reynders, A. Chiumento, S. Malik, P. Crombez and S. Pollin, “Bluetooth now or low energy: Should BLE mesh become a flooding or connection oriented network?,” in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Montreal, 2017.

[90] C.-M. Yu and Y.-B. Yu, “Reconfigurable Algorithm for Bluetooth Sensor Networks,” *IEEE Sensors Journal,* vol. 14, no. 10, pp. 3506-3507, 2014.

[91] Z. Guo-Sheng, L. Qun, W. Hui-Qiang and W. Jian, “A new topology formation algorithm for Bluetooth scatternet,” in *Second International Conference on Embedded Software and Systems (ICESS'05)*, Xian, 2005.

[92] L. Kajdocsi, J. Kovács and C. R. Pozna, “A great potential for using mesh networks in indoor navigation,” in *2016 IEEE 14th International Symposium on Intelligent Systems and Informatics (SISY)*, Subotica, 2016.

[93] Y. Yun, J. Lee, D. An, S. Kim and Y. Kim, “Performance Comparison of Indoor Positioning Schemes Exploiting Wi-Fi APs and BLE Beacons,” in *2018 5th NAFOSTED Conference on Information and Computer Science (NICS)*, Ho Chi Minh City, 2018.

[94] M. Ji, J. Kim, J. Jeon and Y. Cho, “Analysis of positioning accuracy corresponding to the number of BLE beacons in indoor positioning system,” in *2015 17th International Conference on Advanced Communication Technology (ICACT)*, Seoul, 2015.

[95] R. Twohig, D. Newell and M. Duffy, “Effect of energy management circuitry on optimum energy harvesting source configuration for small form-factor autonomous sensing applications,” *Journal of Industrial Information Integration,* vol. 11, no. 1, pp. 1-10, 2018.

[96] “Power management system for a 2.5 W remote sensor powered by a sediment microbial fuel cell,” *Journal of Power Sources,* vol. 196, no. 3, pp. 1171-1177, 2011.

[97] S. Chung-Yang and T. Nan-Chyuan, “Human powered MEMS-based energy harvest devices,” *Applied Energy,* vol. 93, no. 1, pp. 390-403, 2012.

[98] “Improved Energy Management System for LowVoltage, Low-Power Energy Harvesting Sources,” *Journal of Physics: Conference Series,* vol. 773, 2016.

[99] Ranvijay, R. S. Yadav, A. Kumar and S. Agrawal, “Energy Management for Energy Harvesting Real Time System with Dynamic Voltage Scaling,” in *Communications in Computer and Information Science*, Berlin, 2011.

[100] A. R. M. Khairudin and H. Salleh, “Multisource Energy Harvesting Circuit for Low Power Application”.

[101] D. Squires and F. Huff, “Energy Harvesting for Low-Power Sensor Systems,” Renesas Electronics America Inc, 2015.

[102] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang and G. Zussman, "Ultra-Low-Power Energy-Harvesting Active Networked Tags (EnHANTs)," Columbia University, New York.

[103] R. Moghe, D. Divan and F. Lambert, "Powering low-cost utility sensors using energy harvesting," in *2011 14th European Conference on Power Electronics and Applications*, Birmingham, 2011.

[104] A. M. Zungeru, L.-M. Ang, S. R. S. Prabaharan and K. P. Seng, "Radio Frequency Energy Harvesting and Management for Wireless Sensor Networks," *Green Mobile Devices and Networks: Energy Optimization and Scavenging Techniques,* pp. 341-368, 2011.

[105] A. Kansal, J. Hsu, S. Zahedi and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems,* vol. 6, no. 4, 2007.

[106] Nordic Semiconductor, "nRF52832 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR," Nordic Semiconductor, [Online]. Available: http://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832. [Accessed 9 March 2018].

[107] Microchip Technology Inc, "Microchip," Microchip Technology Inc, 1998-2021. [Online]. Available: https://www.microchip.com/. [Accessed 02 02 2021].

[108] Texas Instruments, "BQ25570 Ultra Low Power Harvester Power Management IC with Boost Charger, and Nanopower Buck Converter," 6 March 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/bq25570.pdf. [Accessed 9 March 2018].

[109] Bluetooth SIG, Inc., "Radio Versions | Bluetooth Technology Website," Bluetooth SIG, Inc., 2018. [Online]. Available: https://www.bluetooth.com/bluetooth-technology/radio-versions. [Accessed 11 March 2018].

[110] M. Walton and J. Woods, "The Machine that Lives: Blubot," *COJ Elec Communicat.,* vol. 1, no. 5, p. 6, 2020.

[111] Texas Instruments, "DRV8836 1.5A Low Voltage Stepper or Single/Dual Brushed DC Motor Driver," Texas Instruments Incorporated, 2017. [Online]. Available: http://www.ti.com/product/DRV8836. [Accessed 9 March 2018].

[112] Texas Instruments, "Ultra Low Power Management IC, Boost Charger Nanopowered Buck Converter Evaluation Module," Texas Instruments Incorporated, 2017. [Online]. Available: http://www.ti.com/tool/BQ25570EVM-206. [Accessed 04 22 2018].

[113] C. A. Balanis, Antenna Theory Analysis and Design, Noida: Wiley, 2017.

[114] Tektronix Inc., "TTR500 Series Vector Network Analyzer," Tektronix Inc., 2021. [Online]. Available: https://uk.tek.com/vna/ttr500. [Accessed 03 02 2021].

[115] Ofcom, "Short Range Devices Information Sheet," 20 July 2010. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0021/115653/Draft-IR-2030.pdf. [Accessed 28 November 2018].

[116] ETSI, “ETSI,” 2010. [Online]. Available: https://www.etsi.org/deliver/etsi_en/300300_300399/30033001/01.07.01_60/en_30033001v010701p.pdf. [Accessed 28 November 2018].

[117] E. Ali, Z. Yahaya, N. &. Nallagownden and M. Perumal & Zakariya, “A Novel Rectifying Circuit for Microwave Power Harvesting System,” *International Journal of RF and Microwave Computer-Aided Engineering.,* vol. 10.1002, no. mmce.21083, p. 27, Febuary 2017.

[118] Broadcom Inc, “Broadcom Inc,” Broadcom Inc, 2005-2021. [Online]. Available: https://www.broadcom.com/. [Accessed 25 02 2021].

[119] University of Illinois, “Dielectric Constants of Various Materials,” 01 05 2018. [Online]. Available: web.hep.uiuc.edu/home/serrede/P435/Lecture_Notes/Dielectric_Constants.pdf. [Accessed 02 02 2021].

[120] SEGGER Microcontroller GmbH , “Embedded Studio — A Complete All-In-One Solution,” SEGGER Microcontroller GmbH , 2019. [Online]. Available: https://www.segger.com/products/development-tools/embedded-studio/. [Accessed 2019 06 01].

[121] SEGGER Microcontroller GmbH , “J-Link Debug Probes,” SEGGER Microcontroller GmbH , 2019. [Online]. Available: https://www.segger.com/products/debug-probes/j-link/. [Accessed 21 05 2019].

[122] Zephyr Project, “Bluetooth Mesh - Zephyr Project,” 10 08 2018. [Online]. Available: https://zephyrproject.org/. [Accessed 20 02 2021].

[123] G. Zhai, Y. Zhou and Y. Xuerong, “A Tolerance Design Method for Electronic Circuits Based on Performance Degradation,” *Quality and Reliability Engineering International,* vol. 31, no. 4, pp. 635-643, 2015.

[124] S. Banerjee and et al, Capacitor to Supercapacitor, Springer: Springer Nature Switzerland AG 2020, 2020.

[125] W. Kester and A. Devices, “Taking the Mystery out of the Infamous Formula, "SNR = 6.02N + 1.76dB," and Why You Should Care,” Analog Devices, Inc, 2009.

[126] Maxim Integrated, “The ABCs of Analog to Digital Converters: How ADC Errors Affect System Performance,” 2020. [Online]. Available: https://www.maximintegrated.com/en/design/technical-documents/tutorials/7/748.html. [Accessed 18 02 2021].

[127] F. Heylighen , “Stigmergy as a Universal Coordination Mechanism: components, varieties and applications,” Vrije Universiteit Brussel, Brussel, 2015.

[128] S. Rothmann and E. P. Coetzer, “The big five personality dimensions and job performance,” *SA Journal of Industrial Psychology,* vol. 29, no. 1, p. a88, 2003.

[129] S. Tucker, The Virtues and Vices in the Arts: A Sourcebook., Cascade, 2015.

[130] M. Siekkinen, M. Hiienkari, J. Nurminen and J. Nieminen, “How low energy is Bluetooth low energy? Comparative measurements with ZigBee,” in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, Paris, 2012.