Design considerations and algorithms for low-cost, low-power satellite communications in the Internet of Remote Things

Garrett Kinman

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

April 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Masters of Electrical Engineering

Abstract

There has been tremendous growth within the Internet of Things (IoT) to provide ever more data at ever lower cost. However, much of this growth has been limited to the connected regions of the world, where technologies such as WiFi and cellular are widely available. Much of the remote regions of the Earth have been left with little connectivity, often expensive and power-hungry. The past few years have seen the launch of several independent, low-cost Low-Earth Orbit satellite IoT providers, whose services have enabled low-cost, low-power direct-to-satellite communications. In this project, the key goal was to be able to transmit data off of low-cost, low-power environmental sensors to be placed in remote regions of the Arctic. To accomplish this, the satellite IoT service by Swarm Technologies was selected, and then, an integrative approach was taken to integrate the Swarm satellite modem with a microcontroller. However, communications with these young, independent satellite services—such as Swarm—are intermittent and stochastic, and frequent transmission attempts drastically reduce battery life. To improve the power performance of direct-to-satellite communications, multiple algorithms are presented to

allow for more infrequent transmission attempts and increased transmission success rates. These include algorithms for general system operation and satellite packet construction, as well as an online learning direct-to-satellite packet scheduling algorithm. Finally, an energy model is presented for evaluating and comparing energy performance of the system and scheduling algorithm.

Abrégé

L'internet des objets (IdO) a connu une croissance fulgurante pour fournir toujours plus de données à un coût toujours plus bas. Cependant, une grande partie de cette croissance a été limitée aux régions connectées du monde, où les technologies telles que le WiFi et le cellulaire sont largement disponibles. Une grande partie des régions reculées de la Terre n'ont bénéficié que d'une connectivité limitée, souvent coûteuse et gourmande en énergie. Ces dernières années ont vu le lancement de plusieurs fournisseurs indépendants de services IoT par satellite en orbite terrestre basse, dont les services ont permis des communications directes par satellite à faible coût et à faible consommation d'énergie. Dans ce projet, l'objectif principal était de pouvoir transmettre des données à partir de capteurs environnementaux peu coûteux et peu puissants placés dans des régions éloignées de l'Arctique. Pour ce faire, le service IoT par satellite de Swarm Technologies a été sélectionné, puis une approche intégrative a été adoptée pour intégrer le modem satellite Swarm à un microcontrôleur. Cependant, les communications avec ces jeunes services satellitaires indépendants, tels que Swarm, sont intermittentes et stochastiques, et les

tentatives de transmission fréquentes réduisent considérablement la durée de vie de la batterie. Afin d'améliorer les performances énergétiques des communications directes par satellite, de nombreux algorithmes sont présentés pour permettre des tentatives de transmission moins fréquentes et des taux de réussite plus élevés. Il s'agit notamment d'algorithmes pour le fonctionnement général du système et la construction de paquets par satellite, ainsi que d'un algorithme d'apprentissage en ligne pour l'ordonnancement des paquets directs par satellite. Enfin, un modèle énergétique est présenté pour évaluer et comparer les performances énergétiques du système et de l'algorithme d'ordonnancement.

Acknowledgements

The author of this thesis would like to acknowledge first and foremost Dr. Zeljko Zilic for his supervision, advice, and guidance. Dr. Zilic gave major help in both introducing the author to this project via Dave Purnell, and in giving advice on thesis expectations, timelines, and so forth. Additionally, he hest spent considerable effort in helping to review and edit this thesis. Secondly, the author would like to acknowledge Dave Purnell, the head of the project, for providing all the resources necessary for the design and testing of the hardware and subsequent authorship of this thesis. Finally, the author would like to acknowledge Juan Morency Trudel, a fellow MSc student under Dr. Zilic, who helped to review printed circuit board (PCB) designs and provided excellent feedback for them, as he has much more experience with PCB design.

Contribution of Authors

The author, Garrett Kinman, is responsible for all of the written content of this thesis, as well as all of the simulation code that created the experimental results obtained. The research that went into this thesis does depend upon several open-source libraries for interfacing with the Swarm satellite modem and satellite pass prediction. In addition, several utility libraries were used for simulating the proposed online learning algorithm. These are referenced throughout the text and listed in the appendix.

Contents

List of Figures

List of Tables

List of Acronyms

ASCII	American Standard Code for Information Interchange.
GNSS	Global Navigation Satellite System.
GNSS-R	GNSS reflectometry.
GPS	Global Positioning System.
IoRT	Internet of Remote Things.
IoT	Internet of Things.
JSON	JavaScript Object Notation.
LEO	Low Earth Orbit.
LPWAN	Low-Power Wide Area Network.
MCU	Microcontroller.
NMEA	National Marine Electronics Association.
PCB	Printed Circuit Board.
RF	Radio Frequency.
UART	Universal Asynchronous Receiver-Transmitter.

USB Universal Serial Bus.

Chapter 1

Introduction

1.1 Preamble

An ongoing challenge in the study of sea level dynamics has been that of data. While existing models can produce predictions of sea levels based on factors such as global warming, tidal patterns, and post-glacial rebound, these models ultimately rely on water level measurements [2, 3]. To improve and validate these models, low-cost sensors offer a means of scaling to produce relatively high spatial resolution data, which means sensors need to be placed frequently along coastlines [2, 4]. In some regions of the world, this is easier than in others.

If a sensor is in or near a city, the site is likely to be easy to access and have an abundance of connectivity options. One could use all manner of standard IoT protocols from cellular to LoRa ("long range"), or one could simply log all the data to an SD card to be collected

manually. In more remote regions, however, many of those options disappear. There may not be cell service or other typical IoT communications available, and it may be too far to feasibly collect the data manually. In many of the most remote regions of the globe, the standard has been to use satellite communications, but these are often costly and require a lot of power, two factors often at odds with the scalability needed for high spatial resolution [5, 6]. This is exactly the problem faced by the project this thesis addresses.

There is already a prototype that uses the multipath interference of global navigation satellite system (GNSS) signals to measure water levels at low cost, a technique known as GNSS reflectometry (GNSS-R) [2]. This solution is meant to be affordable and scalable, however, it has the same limitations as much of the currently existing domain of IoT: it can produce data, but it is unable to uplink that data from remote locations to where that data is needed [2]. This limitation of IoT has spawned a subdomain dedicated to solving the issues of bringing IoT to the remote corners of the globe, the Internet of Remote Things (IoRT) [5].

While connectivity options are scarcer in IoRT than for typical IoT, there still remain a few options. These range from low-power wide area networks (LPWANs) to low-power cellular network protocols to geostationary and low-Earth orbit satellites [7]. Additionally, there have been efforts into unmanned aerial vehicles supporting the Internet of Remote Things [7, 8]. Amongst these, however, satellite options are the only proven options for truly global coverage [7, 9, 10]. Due to inherent limitations of geostationary satellite

communications, LEO satellites remain the most practical for the remotest of regions where terrestrial infrastructure does not reach [5, 6].

Even within the realm of LEO satellite communications, there are several further relevant ways to subdivide: 1) by communication directness, 2) by satellite company type, and 3) by LEO orbit configuration. Regarding communication directness, this is in terms of whether individual devices or sensors communicate directly to satellite (known fittingly as "direct-to-satellite") or indirectly via some sort of local network (often an LPWAN) centered around a satellite gateway [7, 9]. Then, regarding satellite company type, there are those services provided by established satellite companies from the pre-IoT era—who often offer satellite internet and satellite phone coverage as well—and those services provided by independent LEO satellite companies [7, 9]. These independent LEO satellite services are often younger, more geared towards IoT, and use smaller CubeSats, which are a class of small and modular picosatellites [7]. Because of these often differing purposes, independent services often use different LEO satellite configurations; because IoT can tolerate intermittent connectivity better than satellite phones can, their satellites can use primarily polar orbits to provide global (but intermittent) coverage [9]. In contrast, the traditional LEO satellite providers often have their satellites arranged to provide continuous or near-continuous coverage [9].

Using more intermittent, opportunistic LEO satellite constellations for IoRT has the primary benefit of requiring fewer satellites, thus reducing cost [9]. These cost savings can be felt at the user level, but so can the intermittent connectivity, which requires data

buffering and knowledge of when satellite passes occur [9]. This in and of itself presents both challenge and opportunity: some algorithm(s) must be devised to buffer data and transmit at appropriate times, but a cleverly devised algorithm provides potential for energy savings [9]. To the best of this author's knowledge, only one previous paper (by Huang et al.) has examined this problem and proposed an online learning algorithm, where online learning here refers to machine learning methods that can learn sample-by-sample in the field, as opposed to offline in batched datasets [10, 11].

This previous work examines a gateway for indirect-to-satellite communications, where it is unknown when new data will be received by the gateway, and thus the authors pose their problem as a queue scheduling problem [10]. They then propose an online learning algorithm based on Lyupanov optimization, which is a common approach for similar problems such as the allocation of resources and green scheduling in cloud computing [10, 12]. This previous work, however, is for an indirect-to-satellite gateway receiving data from a local network (e.g., an LPWAN) at unpredictable times. Because this thesis ultimately designs for direct-to-satellite communications with a known data production rate, a novel algorithm is presented. Thus, to the best of this author's knowledge, this thesis is the first to introduce an online learning algorithm for scheduling of packets for intermittent direct-to-satellite communications. The algorithm presented in this thesis is rather derived from reinforcement learning, specifically Monte Carlo learning and the k-armed bandit problem.

At a high level, the goal of this thesis became to research and design a scheme for the

aforementioned low-cost water level sensors to be able to operate and transmit their data from remote locations. Because of the relative youth of the selected independent LEO satellite service provider, Swarm Technologies, a highly integrative approach is taken to complete sensor design from requirements, to protocol comparison and selection, to hardware design, and finally to software and algorithm design. In doing this, this thesis aims to highlight key design considerations for creating a low-cost, low-power communications scheme for an Internet of Remote Things device. The key novel contributions of this thesis are the algorithms, particularly the online learning direct-to-satellite scheduling algorithm and associated energy model.

1.2 Contribution to Knowledge

- An online learning direct-to-satellite scheduling algorithm based on modified forms of Monte Carlo learning and the k-armed bandit problem. The algorithm uses a form of Monte Carlo learning to estimate the probability of successful transmission for a given set of satellite pass characteristics (maximum elevation angle and pass duration). The algorithm also balances exploration and exploitation using a form of softmax exploration, inspired from the k-armed bandit problem. To the best of this author's knowledge, this is the first such online learning algorithm for efficiently scheduling packets for intermittent direct-to-satellite communications.

- Routines for operating the Swarm M138 modem, efficiently serializing data to binary, determining candidate satellite passes, and applying the online learning direct-to-satellite scheduling algorithm.
- An energy model for the Swarm M138 modem to quantify average power under varying conditions and performance metrics.
- Simulation results demonstrating the performance and behavior of the above online learning direct-to-satellite scheduling algorithm.
- Practical information for integrating the Swarm M138 modem into a complete system. At the time of implementation and of writing this thesis, Swarm's services are quite young, and there are few practical resources on using the devices.

Chapter 2

Background

2.1 Transmission Technologies

This thesis primarily focuses on satellite IoT, but it is valuable to understand other existing communication protocols that could be used in an IoRT context and to justify the functional value provided by independent satellite IoT services. We examine three categories of technologies: 1) cellular technologies, 2) Low-Power Wide Area Networks (LPWANs), and 3) satellite-based technologies. These technologies are evaluated based on the defined requirements and other relevant factors.

2.1.1 Cellular Technologies

Cellular technologies are the first category considered for IoRT. They rely on a prebuilt infrastructure of cell towers distributed across the landscape. A device communicates with the nearest tower if it is within a sufficient range to receive a signal. Modern cell towers typically support high data rates but have limited availability in remote areas. Generally, cellular services are operated by large, established companies, ensuring long-term service reliability. We explore two possible cellular standards: LTE-M and NB-IoT, both developed as part of the same set of standards and designed explicitly for IoT and machine-to-machine communications [13].

2.1.2 Cellular Technologies

The first category of prospective technologies for IoRT is cellular. In brief, cellular communications rely on a prebuilt infrastructure of cell towers spread throughout the landscape. A device communicates with the nearest tower if a tower is within sufficient range to receive a signal. In general, modern cell towers support relatively high data rates, but their availability in remote regions is very limited. From a practical perspective, cell towers are also generally operated by large, established companies, meaning the presence of service post-installation is more likely to be assured long-term. Within the category of cellular, two possible standards will be examined here, LTE-M and NB-IoT. The reason for the selection of these two is because they were both developed as part of the same set of

standards, with LTE-M and NB-IoT having been developed as subsets of the regular LTE standard [13]. There are, however, other cellular standards that exist, but these are two expressly designed with IoT and machine-to-machine communications in mind [13].

LTE-M

First is LTE-M, which is the higher data rate standard amongst the two. Functionally, LTE-M acts like an LPWAN (as will be discussed in the next subsection), except using regular cell towers like traditional cellular protocols such as LTE. This protocol allows communication within a few km of a cell tower [14]. LTE-M uses a peak power consumption during transmission of about 1.4 W, and it achieves up to 1 Mbps upload speeds [13, 15]. In terms of practical considerations, it requires a modem, SIM card, and data plan for each device. A modem can cost about CAD 40, a SIM can cost CAD 1.35, and data can cost CAD 0.14 to 0.68 per MB. In total, LTE-M is fairly well-suited for many IoT applications in terms of cost and power consumption, although its coverage is dependent on cellular infrastructure.

NB-IoT

NB-IoT, like LTE-M, also acts as an LPWAN based on regular cell towers. Functionally, NB-IoT is very similar to LTE-M, with similar costs (for both hardware and data) and similar power consumption [13, 15]. This is largely because NB-IoT, like LTE-M, also operates on

a subset of LTE standards and can often use the same hardware, SIM cards, and data plans [13]. The key difference is NB-IoT has lower data rates around 100 kbps, or about 10 times lower than LTE-M [15]. Additionally, NB-IoT enables communications within 10 km of a cell tower in rural and remote areas [14]. Thus, for most environmental sensors in the Internet of Remote Things, it is anticipated that NB-IoT is the more sensible cellular technology, save for those applications that may have a particular need for higher data rates.

2.1.3 LPWANs

Low-Power Wide-Area Networks (or LPWANs for short) are precisely what the name suggests: low-power networks that can cover a wide geographic area [16]. These networks are designed primarily for IoT, where high data rates are unneeded, low power consumption is needed, and prices are to be kept as low as possible. Within the category of LPWAN technologies (with the exception of the aforementioned cellular LPWANs, LTE-M and NB-IoT), three of the most popular are examined here: 1) SigFox, 2) LoRa, and 3) LoRaWAN. While this list is not exhaustive of all LPWAN options, they are amongst the most common, and they are representative of the overall capabilities of LPWANs.

SigFox

SigFox, amongst the non-cellular LPWANs, is the most similar to the cellular LPWANs, LTE-M and NB-IoT. The key difference is SigFox uses special SigFox gateways instead of

cell towers [17, 18]. These gateways function similarly, except that they exclusively offer low-cost (CAD 15 to 30 per transceiver), low-data rate (100 to 600 bps), and low-power (0.5 to 4 W uplink) connectivity [18, 19]. Because of this, SigFox is generally able to provide service to devices within tens of kilometers of a base station [17, 20]. A downside of this, however, is it is dependent on existing infrastructure, but is unable to use the more widely used cellular infrastructure unlike the cellular LPWANs.

However, SigFox makes up for this with the ability to self-host SigFox base stations [17]. SigFox's rules for this depend on the country. For example, permission from SigFox Canada is required to self-host a SigFox gateway within Canada. Hosting one's own SigFox gateways, however, would introduce a few practical constraints into any data transmission schemes. First is that each gateway creates a local network, so each gateway still needs another means to communicate with the SigFox backend servers such as cellular or ethernet [17]. Also, because each gateway is designed to service many devices in a region, each one is higher power (e.g., 2.3 to 7.5 W). Additionally, to offset the monetary and energy costs of hosting a SigFox gateway, one would want to cluster sensors more heavily. While this particular point is not necessarily a positive or a negative, it is a factor in any environmental sensing system design nonetheless.

Overall, self-hosting a SigFox gateway seems a relatively promising option for any application in which multiple sensors clustered within a radius of tens of kilometers is desired. However, outside of technical performance, there is another factor to consider for

SigFox. At the time of writing this thesis, there was recent news of the possibility of the company SigFox going bankrupt [21]. It is unclear whether this will impact the long-term reliability of SigFox's services, but it may be important to consider for anyone investing in sensors that rely on SigFox's continued service for potentially years to come.

LoRa

The next LPWAN technology, LoRa ("Long-Range"), does not have the same limitations as SigFox. LoRa is a proprietary radio technology that is on the physical layer of the OSI model [14,22], and it allows for sending packets long-distance over license-free RF bands [14]. What this means is long ranges up to tens of kilometers, low data rates (50 bps to 300 kbps uplink), and low power consumption about equivalent to that of SigFox [14, 16, 20]. Unlike SigFox, LoRa is not a packaged service. One must set up one's own base stations with some other method for uploading data from the local cluster to the cloud. Once a cluster is established, however, there is no recurring service fee for using LoRa, nor is there a single company to fear going bankrupt, as there is a wide availability of LoRa-capable products produced by numerous vendors.

LoRaWAN

Lastly amongst the LPWANs is LoRaWAN, which can best be described as "LoRa meets SigFox". LoRaWAN is a higher-level networking protocol that uses LoRa under the hood [14].

This means it carries many of the same functional properties as LoRa regarding data rates, range, and power consumption. Where it differs from LoRa is there are many existing LoRaWAN gateways on the market, which translate LoRa packets to IP packets, which must then be sent via another technology (e.g., cellular) to the cloud [14]. Some of these LoRaWAN gateways can cost on the order of CAD 74, and their power consumption is generally higher than for simple LoRaWAN nodes, although the exact power depends on how the gateway is connected to the outside world. All considered, a self-hosted LoRaWAN gateway is largely equivalent to a self-hosted SigFox gateway, but with the dependence on a 3rd-party service removed.

2.1.4 Satellite Technologies

Compared to cellular and LPWAN options, satellite connectivity functions fundamentally differently. Whereas cellular and even some of the LPWAN options depend on existing terrestrial infrastructure—and where self-hosted LPWANs require creating one's own terrestrial infrastructure—satellite options have global or near-global coverage without the need for ground-based infrastructure nearby to the sensors. The greatest advantage of this is it places little to no constraints on sensor location, meaning satellite communications can reach into the most remote of regions [5, 6]. Within the category of satellite communications considered in this thesis, however, there are two key subcategories of satellite service: geostationary and low-earth orbit (LEO). Across these two subcategories,

this section will examine four commercial satellite IoT services: 1) Inmarsat M2M, 2) Iridium SBD, 3) Astrocast, and 4) Swarm.

Inmarsat M2M

The first satellite IoT service, Inmarsat M2M ("Machine-to-Machine"), operates several geostationary satellites. This grants it near-global—although not fully global—coverage. Most notable is the polar regions have little to no coverage under Inmarsat M2M (or any geostationary satellite), which is an immediate deal breaker for any project located outside those bounds [6]. For sensor locations within the geostationary coverage regions, the functional specifications are as follows. The Inmarsat terminals (which communicate with the satellites) are relatively large devices that can cost on the order of CAD 2,000 to 4,000 and require on the order of 19 W when transmitting. Part of the reason for the high power consumption is the terminals have to communicate with geostationary satellites, which are much farther than LEO satellites. Data plans cost about CAD 200 for 20 MB of data. In total, Inmarsat M2M does not seem like it was designed for low-power IoT in mind, and it is likely more suitable for more traditional satellite data purposes than modern low-cost, low-power IoT.

Iridium SBD

Iridium SBD ("Short-Burst Data"), unlike Inmarsat M2M, is powered by a constellation of 66 LEO satellites in polar and non-polar orbits, meaning global coverage [23, 24]. Because Iridium's satellite constellation also offers traditional satellite phone service, it is designed to offer continuous or near-continuous service [9]. Iridium's SBD service, however, is intended more for IoT and IoT-type applications, and as such has smaller and cheaper satellite modems (ranging from CAD 454 to 600, depending on version). Iridium is capable of transmitting about 1 kbps using a maximum power consumption of about 1.6 W [23]. Also because Iridium SBD is designed for IoT with smaller amounts of data, the data plans offer smaller portions of data, at about CAD 50 per month for 30 kB.

Astrocast

Unlike Iridium and Inmarsat, which have been in the satellite communications industry for a while, Astrocast is a newer entrant. Astrocast's service, like Iridium SBD, uses a constellation of LEO satellites to provide connectivity for IoT. Astrocast operates a constellation of picosatellites in both equatorial and sun-synchronous polar orbits. These orbits, like Iridium, grant Astrocast global coverage, but Astrocast's coverage is more intermittent than that of Iridium, as is typical for independent satellite IoT providers [9]. Further, the company and service were designed from the ground up for low-power, low-cost IoT. From a functional perspective, this IoT-oriented design can be seen in its

specifications. Astrocast's ground nodes cost about CAD 67 per device and have a peak transmission power of less than 0.35 W. The Astrocast data plans cost about CAD 15 per device for 60 kB. Finally, the ground nodes have a universal asynchronous receiver-transmitter (UART) serial interface for easy integration with a microcontroller.

While Astrocast's coverage is technically global due to their LEO satellite orbits, they do not yet offer service to all regions. As a young company, they have not yet secured the rights to use their frequency bands in North America. When reaching out to the company, the author of this thesis was told they did not anticipate having service in North America for another 18 months (which would mean mid to late 2023). If in Europe—or elsewhere after they have secured appropriate regional rights—Astrocast would be an excellent option for environmental sensors in the IoRT. It is purely because of their unavailability in North America that Astrocast was not chosen for the project in this thesis.

Swarm Technologies

Last amongst the satellite communication providers examined in this thesis is Swarm. Swarm offers an extremely similar service as Astrocast, as it is also an independent satellite IoT provider [9]. Swarm operates a constellation of polar and near-polar picosatellites that offer intermittent communication [9]. They are likewise a very young company, however Swarm is based in the United States, and they currently offer service to North America [25]. The Swarm modems cost about CAD 121 per device with volume discounts down to about CAD

80. The data plans cost about CAD 82 per device per year for 750 192-byte packets per month. This is about 1 packet per hour. Additionally, data plans can be stacked, up to 4 plans per device. During actual transmission, Swarm reports a data rate of about 1 kbps and a maximum power consumption of about 3 W. Like for Astrocast, each Swarm modem has a UART interface for integration with a microcontroller. This is rather comparable to Astrocast's ground nodes, and it is likewise designed from the ground up to provide low-cost, low-power satellite IoT.

2.2 Reinforcement Learning

Reinforcement learning is a branch of machine learning that deals with sequential decision-making [26, 27]. Where reinforcement learning differs from other fields of machine learning is that there is no supervisor with correct answers; rather, there is only a reward signal to reward good outcomes and punish bad outcomes [26].

Within a reinforcement learning problem, there exists some agent and an environment. The agent is able to make some observations of the environment and use that to perform some action, which impacts the environment [26]. The agent's perception of the environment is represented as the state, a member of some state space S. The actions the agent can select from are members of some action space A. Depending on the state, action, and mechanics of the environment, a reward R is granted. The goal of reinforcement learning, thus, is to train an agent how to select an action, given a state, so as to maximize cumulative reward

over sequential decisions [26, 27]. This model for action selection is represented by the policy π, which is learned with experience.

2.2.1 Monte Carlo Learning

To accomplish learning, there exist numerous algorithms and techniques, but among the simplest and most interpretable is Monte Carlo learning. This technique essentially keeps a running average of the cumulative reward received by the agent whenever a given state was visited [26, 27]. That is, if a state was ever visited during an episode (a complete sequence of states and actions with a clear beginning and end), it records what the final cumulative reward turned out to be at the end of the episode.

To state this more mathematically, given an environment with states S, actions A, and rewards R, the goal of Monte Carlo learning is to find an optimal policy π^* that maximizes the expected return. In Monte Carlo learning, the agent interacts with the environment repeatedly by following a policy π. This generates an episode $E = \{(s_0, a_0, r_1), (s_1, a_1, r_2), \ldots, (s_T, a_T, r_{T+1})\}$, where $s_t \in S$, $a_t \in A$, and $r_t \in R$ [26, 27].

The main idea behind Monte Carlo learning is to estimate the value function by averaging the returns observed after visiting a state. The value function is given by:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s], \tag{2.1}$$

where G_t is the return at time step t defined as the sum of discounted rewards:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.2}$$

In the first-visit Monte Carlo method, the value function $V(s)$ is estimated as the average of the returns following the first visit to state s in each episode. The algorithm can be summarized as follows:

1. Initialize $V(s)$ for all $s \in S$ and a counter $N(s)$ for each state s.

2. Generate an episode E following the policy π.

3. For each state s visited in the episode, update the value function and counter:

   - Calculate the return G_t after the first visit to s.
   - Update the counter $N(s) = N(s) + 1$.
   - Update the value function $V(s) = V(s) + \frac{1}{N(s)}(G_t - V(s))$.

4. Repeat steps 2-3 for a desired number of episodes or until convergence.

2.2.2 k-Armed Bandit Problem

A particular sub-problem within reinforcement learning is that of the k-armed bandit. Where most reinforcement learning problems involve sequences of several states and several actions, the k-armed bandit problem involves just one state, one set of k possible actions, and no sequential decision-making [26, 27]. Rather, the goal of the k-armed bandit problem is to

optimize for which action to take, given that the expected reward for each action is unknown. Thus, algorithms tailored for the k-armed bandit problem frequently focus on exploration in early epochs to discover the value of the action and then exploitation in later epochs to benefit from that knowledge for close-to-optimal behavior [26, 27].

The k-armed bandit problem can be formally defined as follows:

- An agent interacts with a set of k actions, where each action i has an unknown reward distribution R_i.
- At each time step $t = 1, 2, \ldots, T$, the agent selects one of the k actions, denoted by the action $a_t \in \{1, 2, \ldots, k\}$.
- After choosing an action, the agent receives a reward $r_t \sim R_{a_t}$, sampled from the reward distribution of the chosen action.
- The agent's objective is to maximize the total reward over T time steps, i.e., $\sum_{t=1}^{T} r_t$.

Softmax (also known as Boltzmann) exploration is one method to address the exploration–exploitation trade-off [28]. In this approach, action probabilities are determined using a softmax function, which converts the action-value estimates into a probability distribution. Given a set of action-value estimates $Q(a)$ for each action a and a temperature parameter $\tau > 0$, the probability of selecting action a is given by:

$$P(a) = \frac{\exp\left(\frac{Q(a)}{\tau}\right)}{\sum_{j=1}^{k} \exp\left(\frac{Q(j)}{\tau}\right)} \tag{2.3}$$

where $Q(a)$ is the estimated value of action a, k is the total number of actions, and τ is a temperature parameter that controls the level of exploration. A high temperature results in more exploration, with the actions having similar probabilities, whereas a low temperature leads to more exploitation, where the action with the highest estimated value has a significantly higher probability of being chosen [28]. When $\tau \to 0$, the softmax exploration becomes a pure greedy strategy, always choosing the action with the highest estimated value. Conversely, when $\tau \to \infty$, the agent selects actions uniformly at random.

In the context of the k-armed bandit problem, the agent updates the action-value estimates based on the observed rewards, and the softmax exploration strategy guides the agent in balancing exploration and exploitation to maximize the cumulative reward.

2.3 Summary

This chapter has laid out the key pieces of background information on various protocols applicable for IoRT, and it has presented the formulations of Monte Carlo learning and the k-armed bandit problem. It is upon this background knowledge that the rest of the project for this thesis was built upon. The background on Monte Carlo learning and the k-armed bandit problem also form the basis of the custom algorithm presented later in this thesis. The online learning algorithm that will be presented is based upon a custom modification of the Monte Carlo learning algorithm combined with a modified formulation of the k-armed bandit problem.

Chapter 3

Methodology

3.1 Research Approach

The challenge of determining the best method for transmitting data from sensors in remote Arctic regions requires a specific research approach (shown in Figure 3.1). First, it is essential to define ”best” in the context of data transmission from these remote Arctic locations, leading to the establishment of high-level requirements for such a transmission system.

With these requirements outlined, various transmission protocol options and schemes are considered. Transmission protocols include existing technologies like cellular, Low-Power Wide Area Network (LPWAN), and satellite data transmission. Transmission schemes refer to high-level strategies that utilize one or more protocols to create an overall data transmission solution.

A comprehensive comparison of these transmission protocols and schemes is conducted, examining their advantages, disadvantages, and relevant factors. This analysis leads to the selection of Swarm satellite IoT service as the focus of this thesis. Due to the novelty of Swarm's services and satellite IoT services in general, much of the remaining work involves starting from scratch.

Next, the specific requirements for a satellite IoT system are established, factoring in Swarm modem specifications and constraints. This step helps solidify the Swarm modem and satellite requirements, paving the way for the design and implementation phases. Some of these specifications and characteristics directly influence the final designs and algorithms presented.

During the design and implementation stages, an iterative process is followed. Initially, a third-party commercial breakout board for the chosen Swarm modem is tested. However, due to its unsuitability for production, a first prototype PCB is designed and tested, integrating the Swarm modem with a Raspberry Pi Zero. This prototype ensures the basic operation of the modem.

A second prototype PCB is then designed and tested, integrating a lower-power microcontroller with the Swarm modem and relevant sensing components. While not fully production-ready, the second prototype successfully demonstrates MCU integration with the Swarm modem and data transmission, fulfilling the thesis objective of designing a low-cost, low-power transmission scheme for the Internet of Remote Things. This

prototype also incorporates a software implementation of an algorithm for data transmission and power conservation. Furthermore, an online learning algorithm is introduced, allowing each device to adapt to its specific location and environmental conditions for optimal satellite communication.

In addition to verifying the boards and microcontrollers, it is crucial to simulate and demonstrate the value of the online learning direct-to-satellite packet scheduling algorithm. Simulated results are presented to assist in designing a field-ready algorithm, taking into account the factors that affect the success of data transmission, such as satellite pass quality. Additionally, a simple mathematical model of the system's power consumption is provided, considering the stochastic nature of data transmission success and its impact on power usage. This model allows for both optimistic and pessimistic evaluations of power consumption under varying conditions, serving as a rough tool for assessing the average power of the system.

3.2 High-Level Requirements

Given the objective of creating low-power, low-cost sensors for remote areas like the Canadian Arctic, two primary conceptual requirements emerge: 1) the system must be low-power, and 2) the system must be low-cost.

For the specific sensor developed in this project, existing water-level sensors are quite expensive (e.g., USD 1,000 to 10,000) and demand substantial infrastructure for setup [2].

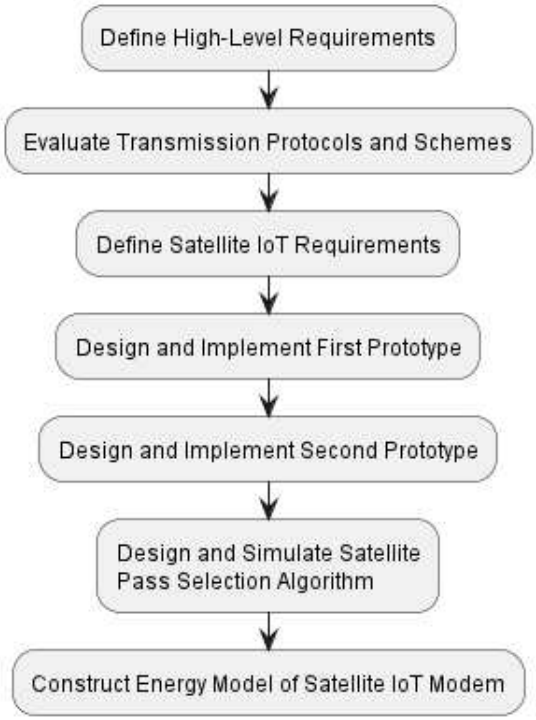

Figure 3.1: High-level research, design, implementation, and testing approach.

Consequently, gathering water level data at high spatial resolution becomes less financially feasible, especially in remote regions such as the Canadian Arctic. A key goal of this project is to create affordable sensors that can be widely installed across the Arctic to provide high-resolution sea level rise data.

The initial prototype for this project, for instance, uses a Raspberry Pi Zero (without wireless data transmission), resulting in approximately 2 W of power consumption. This high power consumption necessitates a large 2.4-kWh lead-acid battery, increasing costs in two ways: 1) the immediate material cost, and 2) the substantial expense of transporting large batteries to remote locations. A lower-power system can reduce battery requirements,

saving on both immediate costs and installation expenses. A labor-intensive and travel-intensive device will significantly increase costs, even more so than for regular IoT. Once a sensor is placed in the Arctic (or any other remote location), return trips for maintenance become prohibitively expensive.

Considering the factors mentioned above, three high-level requirements are defined:

1. The sensors must operate year-round (including the Arctic winter) on an airplane-transportable battery without manual intervention for power. The battery must have a sufficiently small capacity to be transported by plane to the site location without exceptional cost or inconvenience.

2. The sensors must be as low-cost per sensor as possible, both in terms of immediate hardware cost and fieldwork and setup cost. Furthermore, the monthly operating expenses per device must be minimized.

3. The sensors must run year-round without in-person maintenance, although remote maintenance (e.g., a remote ”reset” functionality) is acceptable.

These requirements naturally lead to the conclusion that the sensors must be low-power to capitalize on the price advantage and transportability of smaller batteries, as well as reliably autonomous. To achieve this, this thesis will focus primarily on reducing costs and promoting longevity by lowering average power, thereby enabling smaller batteries and more affordable sensor setup. This thesis will also emphasize interpretability in algorithm design

to enhance trust in autonomous operation.

A final note, however, is environmental constraints such as operating temperatures and humidity are not considered in this thesis. The project in this thesis did not result in a production-ready sensor hardened to the conditions of the Arctic. The scope of the project in this thesis was limited to reducing the energy associated with data transmission.

3.3 Comparison of Options

3.3.1 Comparison of Transmission Protocols and Standards

There are many details about the reviewed protocols and standards to compare. Thus, side-by-side comparisons of all the key considerations of each service have been put into Table 3.1.

From Table 3.1 (and the previous in-depth review), some protocols and services emerge as strong candidates for low-cost, low-power IoRT, while others can be immediately discarded. First, SigFox can be removed since self-hosting a SigFox gateway provides minimal additional benefits compared to LoRa or LoRaWAN while increasing business risk due to third-party dependence.

Inmarsat M2M can also be discarded, as it is too high-power and expensive for low-cost, low-power IoRT. Even when using a terminal as a gateway for indirect-to-satellite communications with multiple sensors connected via LoRa or LoRaWAN, the terminal, data,

Name	Coverage	Upfront Cost	Operating Cost	Power
LTE-M	Heavily limited in remote locations	Tens of CAD per module + SIM	CAD 0.14 to 0.68 per MB	∼1.4 W max TX
NB-IoT	Heavily limited in remote locations	Tens of CAD per module + SIM	CAD 0.14 to 0.68 per MB	∼1.4 W max TX
SigFox	Tens of km radius from gateway	Low tens of CAD per transceiver + tens of CAD per gateway	Depends on country (usually CAD 10 per year or less)	∼0.5 to 4 W max TX, <10 W for gateway
LoRa	Tens of km radius from gateway	Low tens of CAD per transceiver	None	∼0.5 to 4 W max TX
LoRaWAN	Tens of km radius from gateway	Under CAD 100 per gateway	None	<10 W for gateway
Inmarsat M2M	Global, except polar regions	CAD 2,000 to 4,000 per terminal	CAD 200 for 20 MB	∼19 W max TX
Iridium SBD	Global	CAD 454 to 600 per modem	CAD 50 for 30 kB	∼1.6 W max TX
Astrocast	Global	CAD 67 per modem	CAD 15 for 60 kB	<0.35 W max TX
Swarm	Global	CAD 80 to 121 per modem	CAD 82 per modem per year	∼3 W max TX

Table 3.1: Comparison of all the evaluated transmission protocols and standards.

and power supply costs are difficult to offset. Iridium SBD, Astrocast, and Swarm are better suited for the use case pursued in this thesis. Moreover, polar coverage is required for this particular project.

Astrocast and Swarm both stand out as exceptional choices for IoRT. These independent LEO satellite IoT providers were recently developed with low-cost, low-power IoT in mind. Furthermore, they are both sufficiently low-power and low-cost to enable one modem/ground node per sensor for direct-to-satellite communications. This simplification reduces points of failure and engineering effort. Additionally, it streamlines the algorithmic problem of optimizing transmission scheduling compared to indirect-to-satellite, as seen in previous work [10].

Among the remaining options, Iridium SBD is marginal. It could potentially work in some systems, but it is outperformed by Astrocast and Swarm for the low-cost, low-power IoRT market. On the other hand, LTE-M and NB-IoT depend on the specific project. For some projects, cell coverage may be adequate, or sensor locations may be flexible enough to make cellular feasible. This capability is further enhanced when considering the integration of such a system with LoRa or LoRaWAN. This type of heterogeneous transmission scheme is discussed in greater detail in the following section.

3.3.2 Comparison of Transmission Schemes

From the previous section examining key aspects of various protocols, two overall ”schemes” emerge: 1) a heterogeneous transmission scheme where LoRa or LoRaWAN serve as hubs or range extenders for an LTE-M or NB-IoT gateway, and 2) a homogeneous transmission scheme where each sensor has an Astrocast or Swarm modem and can communicate directly with LEO picosatellites. These two high-level schemes will be referred to as ”hub-and-spokes” and ”direct-to-satellite,” respectively.

The hub-and-spokes scheme, as illustrated in Figure 3.2, employs an LTE-M or NB-IoT gateway as the hub, with LoRa- or LoRaWAN-equipped sensors as the spokes. Alternatively, a satellite gateway could be used for indirect-to-satellite communications instead of cellular. This scheme extends the limited range of cellular service and capitalizes on its higher data rates. One gateway, using the relatively high 1 Mbps uplink rate of LTE-M, for example, could likely service several sensors. This setup could create a clustered pattern of sensors in remote areas where gateways are placed in areas with cell service, and the spokes extend an additional 15–20 km.

However, the hub-and-spokes scheme has several significant downsides. First, it is heavily limited in coverage. For some use cases, site location flexibility may make this limitation acceptable. For many use cases, though, placing sensors in locations where cellular + LoRa cannot reach may be necessary, and a solution involving chained LoRa repeaters might become too complex and expensive to remain in the domain of low-cost, low-power IoRT.

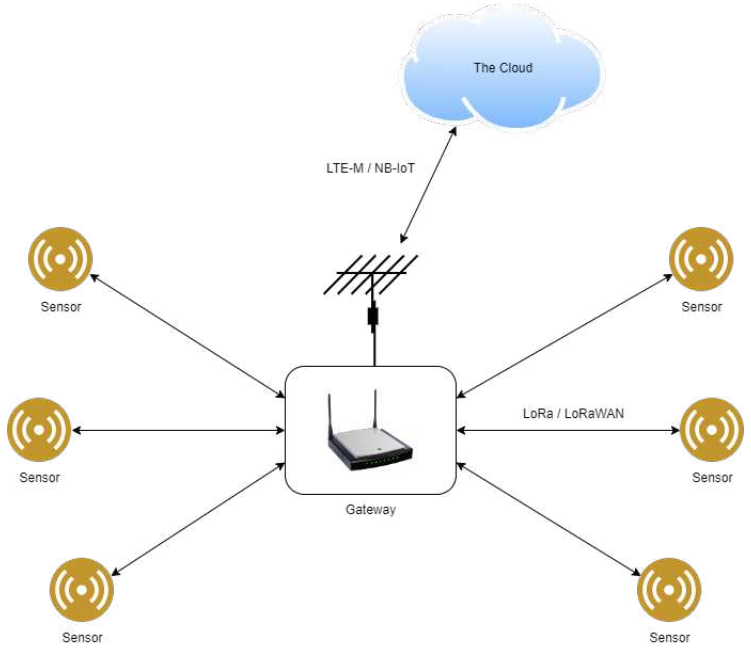

Figure 3.2: Diagram of a hub-and-spokes transmission scheme.

This point highlights the other salient drawback of the hub-and-spokes scheme: it is inherently more complex to design and implement. It employs two separate protocols or standards and creates a middleman gateway. This means designing, implementing, and configuring both sensors with LoRa or LoRaWAN and gateways. The spokes and gateways would have their own hardware, software, and power supplies, introducing more possible points of failure and increasing the engineering costs. Moreover, gateways would likely use significantly more power, necessitating the creation of larger (and thus more expensive) hubs.

On the other hand, the satellite IoT scheme addresses many of these issues. It offers truly global coverage, allowing sensors to be placed almost anywhere on Earth without dependence on cell service. As a result, no hub-and-spokes scheme is necessary for reaching

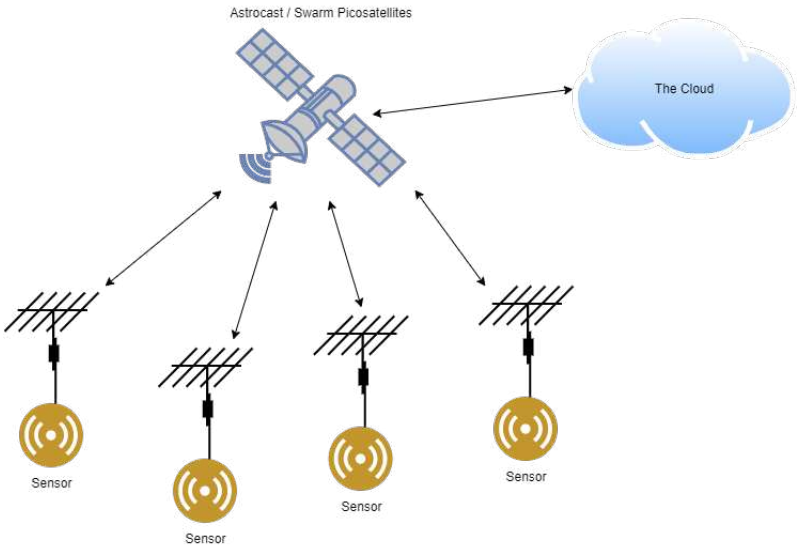

Figure 3.3: Diagram of direct-to-satellite transmission scheme.

remote areas; it becomes practical to make each sensor its own gateway for direct-to-satellite communications. This leads to several related benefits.

With each sensor as an independent device, only one standard sensor device needs to be designed. This means a single set of hardware, software, and power supply, resulting in fewer points of failure. In contrast to the hub-and-spokes scheme, a gateway failure in the direct-to-satellite scheme does not entail automatic failure for the whole cluster. Additionally, no high-powered hub with a larger power supply is needed. All these factors mean the direct-to-satellite scheme enjoys the significant advantage of simplified engineering and operation. However, this comes at the expense of needing extra hardware and data plans for every single sensor. This is a manageable downside, however, as the new satellite IoT providers such as Astrocast and Swarm are affordable enough (in both hardware and data plans) to

be considered low-cost, low-power IoRT.

With all these points in mind for both schemes, the satellite IoT scheme was chosen for this project, as the hub-and-spokes scheme was too limiting in the Canadian Arctic, where many of the sensors produced in this project will ultimately be placed. Also, as stated earlier, Swarm was chosen as the satellite IoT service for this project, as Astrocast was unavailable in North America at the time. The rest of this methodology section will then focus on the Swarm-based satellite IoT scheme, from lower-level constraints and requirements, to design and implementation, to the testing methodology.

3.4 Satellite IoT Requirements

With Swarm chosen as the means of communication for the low-cost, low-power IoRT in this thesis, it is now possible to examine more in-depth requirements and constraints. To begin, here is a brief overview of how exactly data is transmitted to the cloud via Swarm's service. Swarm operates a number of LEO satellites. Each one beams down packets towards the Earth's surface, announcing the satellite's presence overhead, for any modems listening. Any modems that receive one of these packets will then attempt to transmit any queued packets and listen for acknowledgements from the satellite. If successful, the satellite will move on in its orbit until it reaches a ground station operated by Swarm. It is here it will transmit down all the packets it has received from various modems. The ground station will receive the modems' packets, organized by device ID, and upload them to Swarm's data

servers. Here, the packets are available to the user either via a web portal or by an API.

Of all of this, the part that is relevant to this thesis is the integration and operation of the modem itself with the sensor. This encompasses both hardware matters (e.g., PCB design) and software (e.g., packet bundling and packet scheduling). To design both the hardware and software to use these modems first requires some examination into their key constraints, specifications, and requirements.

3.4.1 Swarm Modem Operating Specifications

To note first is that the Swarm modem referred to in this thesis is the M138 modem, the Swarm modem that is available at the time of writing this thesis and the modem with which this project was designed. With this established, the operating specifications of the modem will be evaluated first.

The Swarm modem has four total operating states, all of which consume power: 1) Sleep Mode, 2) GPS Acquisition Mode, 3) Receive Mode, and 4) Transmit Mode. First, when the modem is powered on, it enters GPS acquisition mode, wherein it acquires a GPS fix so it can determine the time and location. This typically lasts for about 30 seconds. The modem will also re-enter GPS Acquisition Mode every 4 hours or every time the modem enters Sleep Mode and is awoken.

Once a GPS fix has been acquired, the modem will enter Receive Mode, wherein it listens for a packet from any satellites that may be passing overhead. This mode lasts until either

Power Supply	Mode	Typical Current	Typical Power
3.3 V	Transmit	$850mA$	$2.8W$
	GPS Acquisition	$45mA$	$150mW$
	Receive	$26mA$	$86mW$
	Sleep	$<80\mu A$*	$<260\mu W$*
5.0 V	Transmit	$550mA$	$2.8W$
	GPS Acquisition	$45mA$	$230mW$
	Receive	$26mA$	$130mW$
	Sleep	$<110\mu A$*	$<550\mu W$*

Table 3.2: DC power characteristics in the four modes of operation, for both 3.3 and 5.0 V.

a packet is received from a satellite (at which point it enters Transmit Mode), the modem is instructed to enter Sleep Mode, or enough time elapses that the modem automatically re-enters GPS Acquisition Mode.

If a packet is indeed received from a satellite, the modem will then enter Transmit Mode, wherein it will attempt to transmit any queued packets up to the satellite and listen for an acknowledgement. If this is successful, it will return to Receive Mode as normal, unless put into Sleep Mode. Sleep Mode, as the name suggests, is a much lower power idle mode for the modem, in which it uses at least 2 to 3 orders of magnitude less power than the other modes.

As is typical for communications, the greatest power consumption occurs during Transmit Mode. This can be seen in Table 3.2, where Transmit Mode consumes by far the most power. Note, however, that the values for Sleep Mode (marked with an asterisk) are maximum rather than typical, as Swarm does not report a typical value in the data sheet; this is perhaps

because it is too low or too inconsistent to be reliably measured. Also note that the modem can be supplied with either a 3.3 V or a 5.0 V supply, and this selection does indeed have a slight impact on typical power consumption in GPS Acquisition and Receive Modes.

One important observation from Table 3.2 is that using a 5.0 V supply seems to result in slightly higher power consumption in Receive and GPS Acquisition Modes, and possibly in Sleep Mode as well, even though it consumes the same power in Transmit Mode. In contrast, Figure 3.4 shows Swarm's predictions for the battery life of a 10,000 mAh, 3.7 V LiPo battery under varying transmit attempt frequencies. It becomes clear that the transmit attempt frequency is a large factor influencing average modem power consumption. However, increasing the transmit attempt frequency also leads to more frequent entry into GPS Acquisition Mode and Receive Mode. To determine the relative importance of these modes for overall power consumption, additional information is needed.

One such piece of information is that Swarm reports a 192-byte (the max length) packet takes about 3.7 seconds and 12.24 joules of energy to transmit.

The information and calculations presented above provide some insight into the energy required by the Transmit Mode but do not offer much clarity on the energy consumption of GPS Acquisition Mode and Receive Mode during a typical transmission cycle. In fact, one of the primary aims of this thesis is to quantify an energy model for communication using a Swarm modem and to develop an algorithm for operating the modem from the host device to reduce power consumption. However, constructing these models requires more information,

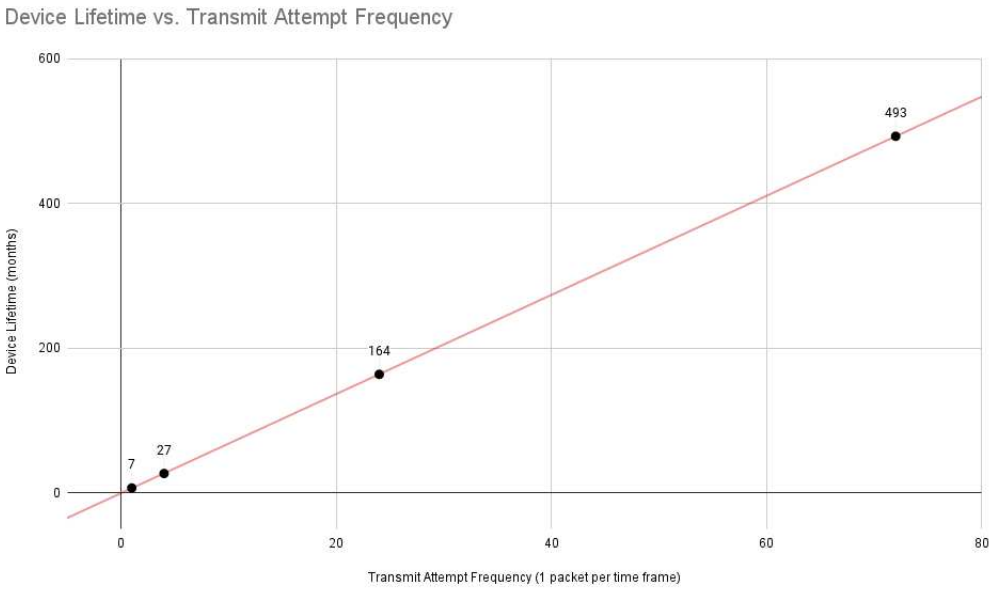

Figure 3.4: Swarm's reported battery lifetime for a 10,000 mAh, 3.7 V battery.

which will be covered later in this thesis.

Nevertheless, several notable ideas for energy savings are emerging:

1. Keeping the modem in Sleep Mode as much as possible is beneficial. While the impact of using a 3.3 V or 5.0 V supply on average power in Sleep Mode is unclear, it is evident that Sleep Mode consumes at least 2 to 3 orders of magnitude less power than other modes.

2. The total process of exiting Sleep Mode, entering intermediary states, and transmitting data is the most energy-consuming part of modem operation.

3. Failing to transmit wastes significant energy compared to sleeping. If the modem

wakes, it must acquire a GPS fix (consuming much more power than Sleep Mode) and then wait in Receive Mode for a signal from a satellite. If no packet is received from the satellite, all the energy consumed in those two modes is wasted. Therefore, it is advisable to attempt transmission only when there is a high probability of successful communication with a satellite.

The above points will be expanded upon in the following sections, serving as the basis for a more comprehensive energy-saving transmission scheme.

3.4.2 Swarm Satellite Transmission Functional Specifications

Understanding how to minimize power consumption during transmission requires a grasp of how satellites, transmission, and data plans function. With a finite number of satellites, a satellite is not always overhead at any given location and time. Additionally, the angle and environmental conditions (e.g., background RF noise) can vary. Data plans also limit how much each device is permitted to send, and frequent transmissions consume more energy. All these factors impact transmission approaches.

Swarm provides an online satellite pass checker, which, given a set of Earth coordinates, offers a list of upcoming satellite passes, their times, durations, and maximum elevation angles. Figure 3.5 shows a visualization of these satellite passes. Qualitatively, max elevation angles observed in Montreal, Quebec, Canada ranged between 15 and 85 degrees, with pass durations typically between 10 minutes and an hour.

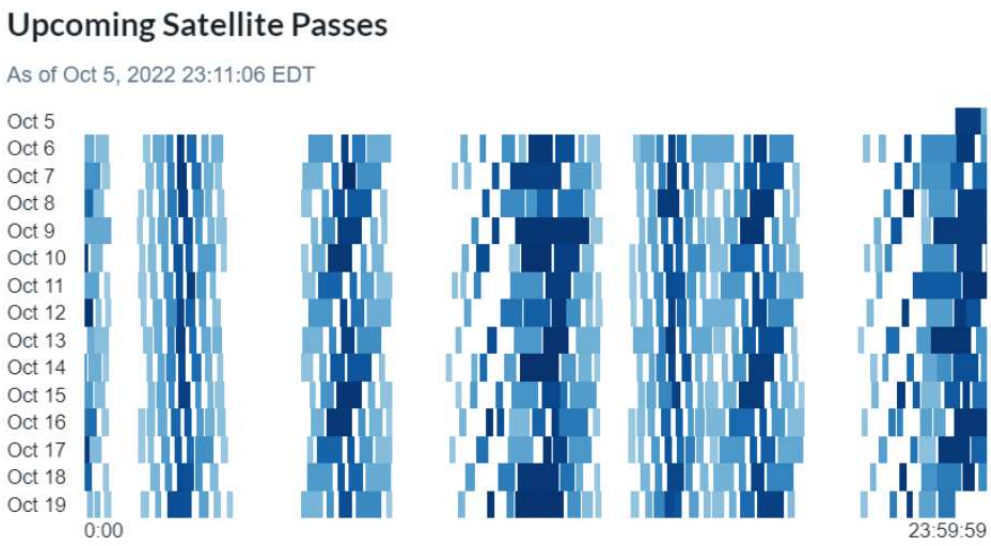

Figure 3.5: Sample view of online Swarm satellite pass predictor (for Montreal, Quebec, Canada). Darker shades of blue correspond to higher maximum elevations (in degrees) above the horizon.

However, satellite transmission success can be stochastic, meaning that even during a satellite pass, the modem may not always succeed in a transmission attempt. Factors impacting this include satellite pass "quality", RF background noise, environmental conditions, antenna setup, and many others.

Satellite pass quality is related to the pass predictor's information on duration and maximum elevation angle. Swarm does not provide guidance on the factors leading to successful transmission, so this thesis aims to develop an empirical model for quantifying a "good" pass, where a "good" pass is relatively likely to result in successful transmission. For example, for a sensor that has an average success rate of 90%, a good pass that is likely to result in successful transmission may mean a satellite pass that has a 90% or higher

Background Noise Intensity (dBm)	Quality (for Transmission)
-90 and higher	Bad (unlikely to work)
-93 and lower	Marginal
-97 and lower	OK
-100 and lower	Good
-105 and lower	Great

Table 3.3: Background noise intensity required for likelihood of transmission.

probability of being successful. In contrast, another sensor with a lower average success rate of 30%, a good pass that is relatively likely to result in successful transmission may mean a mere 30% or higher probability.

Swarm provides guidance on the second factor, RF background noise, stating that a background noise intensity of -93 dBm or lower is likely required for successful transmission. Note that Swarm does not quantify "likely" in this context, only stating qualitatively that transmission is unlikely to work at any higher than -93 dBm. As a result of this factor, transmission is more reliable outside of cities. Table 3.3 shows Swarm's qualitative guidance on what levels of RF background noise are considered good.

Physical constraints such as satellite passes and RF background noise are not the only factors affecting transmission to satellites. Swarm's data plans also impose constraints. Unlike other transmission protocols, Swarm charges a set amount of USD 60 (about CAD 82) per year per data plan, with up to four data plans stackable onto a single modem. Each data plan allows up to 750 packets per month, where each packet can be up to 192 bytes, for a theoretical maximum of about 150 kB. Of these, a maximum of 60 packets can be

downlink, i.e., transmitted from the satellite to the modem. For reference, 750 packets per month correspond to about 25 packets per day or approximately one packet per hour.

Considering these constraints, several transmission strategies can be explored. If one needs measurements at a higher temporal resolution than one hour (e.g., every 15 minutes), they can either bundle measurements or transmit more frequently. Transmitting more frequently, however, reduces the expected battery life of a system and incurs additional monetary cost.

The impact on power consumption and battery life is not only due to the transmission itself but also because each cycle of waking from sleep, acquiring GPS, listening for a packet from a satellite, and transmitting consumes extra power. Furthermore, environmental variables introduce uncertainty regarding how long the modem will need to be awake before successfully transmitting, or if it can transmit at all. This thesis will further examine this question. In the meantime, data bundling appears to be a promising power-saving strategy.

3.4.3 Swarm Modem Hardware/Software Interface

All transmission planning strategies are based on the idea that the modem can be controlled, i.e., it can be instructed to enter different modes or to transmit a specific packet of data at a certain time. They also depend on the supply voltage provided to the modem. These factors rely on the physical and software interface for interacting with the Swarm M138 modem.

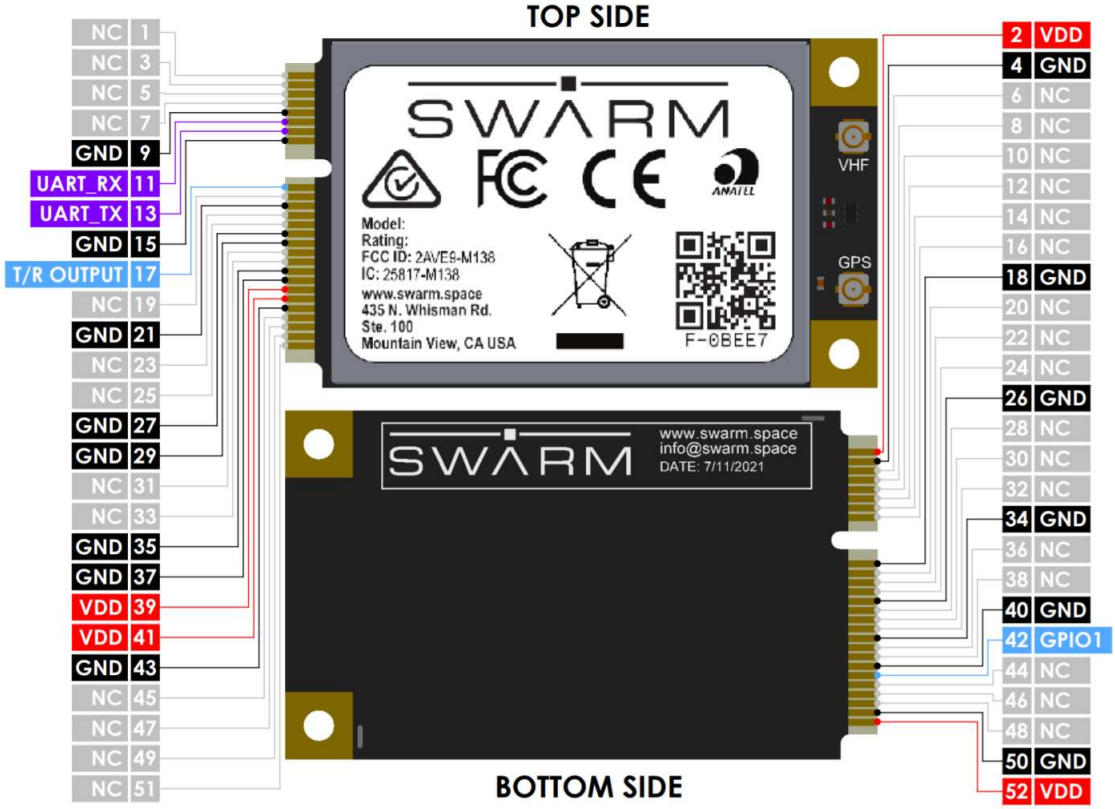

Figure 3.6: Diagram of Swarm M138 modem signaling on the mPCI-e connector [1].

Regarding the physical interface, the modem connects with an mPCI-e connector, a standard connector for wireless modems in embedded devices. However, while the physical connector is mPCI-e, the signaling is not standard mPCI-e. The Swarm modem pinout is shown in Figure 3.6. A brief description of each Swarm modem signal is provided in Table 3.4.

The VDD signal, acting as the modem's power supply, has strict requirements. Due to the modem's sensitivity to RF background noise (as little as -90 dBm can disrupt communications), Swarm recommends a minimum amount of decoupling capacitance

Signal Name	Description
VDD	Power supply; can be 5.0 V at up to 600 mA or 3.3 V at up to 1000 mA
GND	Ground
UART_RX	3.3 V serial receive
UART_TX	3.3 V serial transmit
T/R OUTPUT	Transmit/receive indicator; HIGH when transmitting, LOW when receiving
GPIO1	Software-configurable GPIO

Table 3.4: Swarm modem signal descriptions.

placed close to the VDD pins on the modem. This eliminates as much of the high-frequency components from the otherwise DC signal as possible because these small AC components affect modem operation. Higher AC components within the otherwise DC VDD signal were observed to result in increased measured RF background noise (by up to 10 dB higher). The modem data sheet contains details on the decoupling and feed-through capacitance that Swarm recommends.

As shown in Table 3.4, the primary means of communication with the modem is via UART communications, with two optional signals, T/R OUTPUT and GPIO1, also available. In terms of UART communications, the Swarm modem has a pre-defined command and message set formatted as NMEA sentences. Some take the form of direct commands, some are command responses, and some are unsolicited messages. These commands and messages begin with a prefix consisting of a '$' symbol, followed by a few alphanumeric characters, and then a space. The prefix defines the command or message category, while further details within the message determine the command's function or the message's content. The full details of the Swarm modem software interface can be found in the data sheet, but a

Prefix	Description
$CS	Configuration settings
$DT	Date/time status
$FV	Firmware version read
$GJ	GPS jamming/spoofing indication
$GN	Geospatial information
$GP	GPIO1 control/status
$GS	GPS fix quality
$MM	Messages received management
$MT	Messages to transmit management
$PO	Power off
$PW	Power status
$RD	Receive data unsolicited message
$RS	Restart device
$RT	Receive test
$SL	Sleep mode
$M138	Modem status unsolicited message
$TD	Transmit data

Table 3.5: List of Swarm modem command/message prefixes.

summarized list of the prefixes is shown in Table 3.5.

The list presented here demonstrates the modem's capabilities and how it can be controlled. While the full details are beyond the scope of this document and can be found in the data sheet, some of these commands and messages are particularly useful for managing a low-power operation of the modem.

The $SL prefix allows the modem to be commanded to sleep for a specific number of seconds or until a specified date and time. This is useful for power-saving operations, as it enables the modem to sleep until the next calculated optimal transmission time. This approach depends on determining "a good time to transmit" and "a good pass" as previously

described. By performing offline pass predictions and identifying good passes likely to result in transmission, the modem can be set to sleep until the next suitable pass.

The special $M138 prefix announces asynchronously when a GPS fix has been acquired. This is valuable because it informs the host device when the modem switches from GPS Acquisition Mode to Receive Mode. If the modem already has queued packets for transmission, it will automatically enter Transmit Mode upon receiving a packet from a satellite. Otherwise, the host must wait for the correct $M138 message before issuing any new $TD commands.

The $RT prefix enables configuring periodic reports on the intensity of RF background noise (in dBm), with reports sent asynchronously at a specified period (in seconds). This has two advantages. First, it alerts the host if attempting transmission is not worthwhile due to high RF background noise, which can vary by up to 20 dB in the same location. Second, it may serve as a useful additional parameter for good packet scheduling, as higher RF background noise may require better satellite passes. This point will be explored further later in the thesis.

The $TD prefix queues data for transmission. Data can be sent in two primary formats: 1) an ASCII string or 2) binary data. Depending on the use case, sending ASCII strings might be advantageous, but sending raw binary data is generally more efficient for numeric data. It should be noted that raw binary data must be represented in hexadecimal using ASCII characters for UART, but the modem reinterprets these back into binary.

Byte-efficient data encoding is crucial for two reasons. First, each packet has a limited number of bytes available (192). For instance, a JSON string format, although common, wastes bytes when representing numerical values. Second, more measurements can fit into each packet with efficient data encoding, allowing for more bundling and fewer transmissions for the same amount of data.

3.5 Design

Translating high-level strategies into a practical device involves hardware, software, and algorithmic design. The hardware design necessitates creating a custom PCB to interface the modem with the host device. This was done in three stages: 1) testing a 3rd-party Swarm modem breakout board, 2) designing a first prototype PCB to integrate the Swarm modem with a Raspberry Pi Zero, and 3) designing a final prototype PCB to integrate the Swarm modem with a microcontroller (in this project, a Raspberry Pi Pico) and the relevant components for GNSS-R water level sensing.

The software design was primarily accomplished in two stages: 1) a small Python program for testing the 3rd-party breakout board and the first PCB prototype, and 2) leveraging an existing open-source library for the Swarm modem to perform higher-level tasks and algorithms on a microcontroller on the final prototype PCB. Lastly, the algorithmic design was informed by background research and practical experience testing data transmission with the Swarm modems.

3.5.1 Hardware Design

Understanding the modem's unusual signaling on the mPCI-e connector made it clear that a custom PCB would be necessary for an IoRT sensor, such as the one designed in this project. However, an off-the-shelf Swarm breakout board offered by SparkFun was used for early testing of transmit capabilities and came with an antenna ground plane, which was also utilized for testing the final prototype PCB.

Figure 3.7 presents the schematic for the final prototype PCB design produced in this project. The left-hand side displays spaces for four GPS receivers and four GPS antenna connectors, which are the project-specific sensing components for GNSS-R water level sensing. The remaining two-thirds of the schematic on the right is largely generalizeable to other projects that would use the Swarm M138 modem, including an mPCI-e connector, decoupling and feed-through capacitors, and headers for the microcontroller. Figure 3.8 presents the final prototype PCB layout, and Figure 3.9 shows the physical PCB with microcontroller and Swarm modem attached.

A few of the key design decisions here include the microcontroller selection, microcontroller integration, and decoupling and feed-through capacitors. The Raspberry Pi Pico was chosen for this project because GNSS-R water level sensing requires relatively intensive processing to be performed in-situ for each measurement. The dual-core processor on board allows for easy separation of duties, with one core producing measurements and the other managing the Swarm modem.

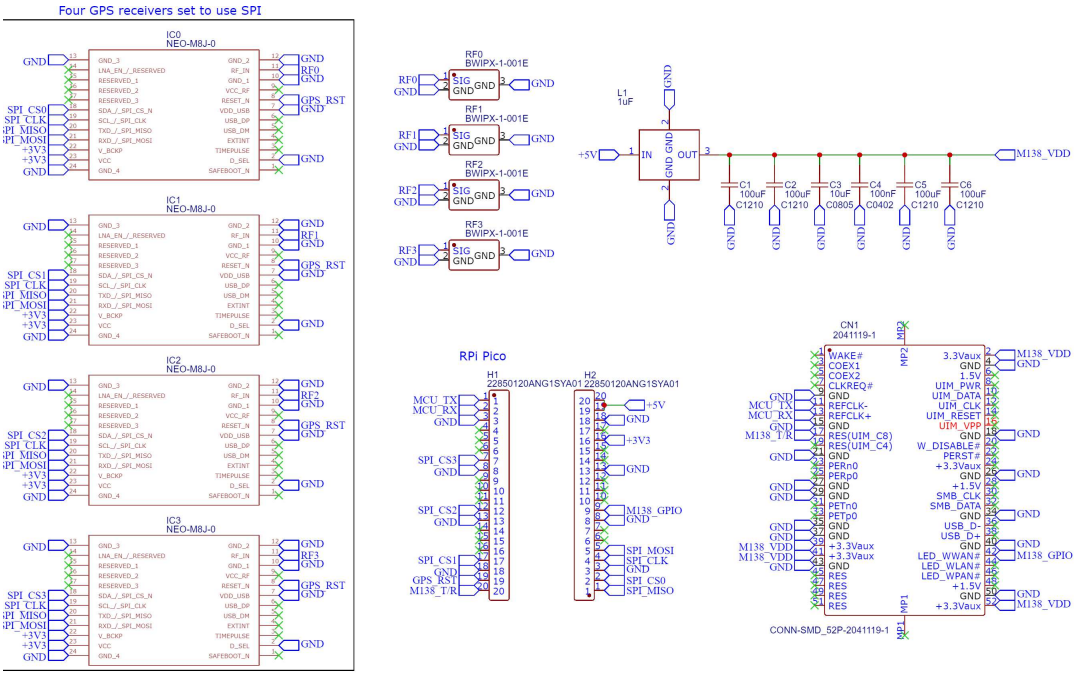

Figure 3.7: Schematic of the final prototype PCB, integrating modem and MCU.

The next key design decision in the hardware involved placing female headers on the PCB, allowing a Raspberry Pi Pico development board (with male headers) to be easily slotted in. This choice was made primarily because development boards are already available, relatively low-cost (under CAD 10, depending on the retailer), and significantly reduce design time for the sensors (especially for a prototype).

The final key design consideration in the hardware pertained to the decoupling and feed-through capacitors. The recommended capacitance was incorporated into the PCB, placed as close to the VDD pins on the modem as possible. Additionally, space for two extra 100-µF capacitors was added, although they were not assembled by the PCB manufacturer. These extra capacitors were included as a contingency in case more capacitance was needed, since

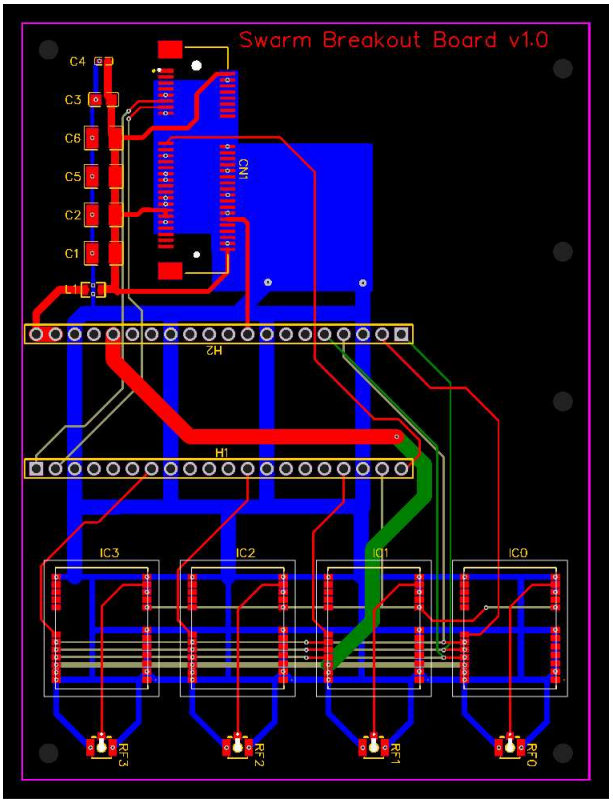

Figure 3.8: PCB layout of the final prototype PCB, integrating modem and MCU.

Swarm strongly emphasizes the importance of these capacitors for removing unwanted high-frequency components and provides the recommended capacitance as a minimum. In the context of this project, these extra capacitors were ultimately not required.

3.5.2 Software Design

With the final PCB prototype, a comprehensive understanding of the Swarm modem's specifications and operation, and several high-level strategies for low-power transmission in place, the software design process could commence. Although the low-level software design

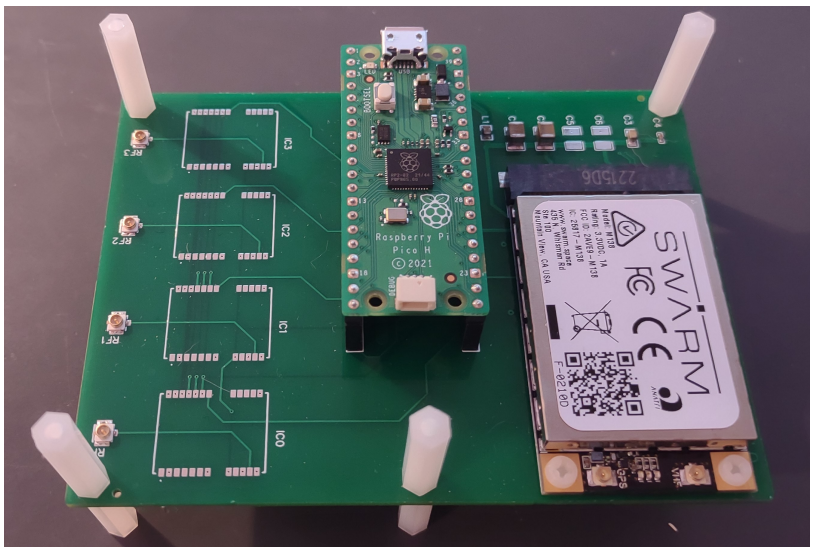

Figure 3.9: Image of final prototype PCB with Pico and Swarm modem (GPS receivers unattached).

relies on UART communications and Swarm's custom command set, this aspect was greatly simplified with the help of an open-source Arduino library by SparkFun for operating the modem. This library internally manages all the low-level UART tasks and exposes a higher-level set of types and methods. The remaining higher-level logic was built on top of this library.

The high-level view of the two main processes can be seen in Figure 3.10. Note that while this design was intended for the dual-core Raspberry Pi Pico, the processes could also be implemented concurrently rather than in parallel. The only significant change would be idling the processes (without pre-emption) instead of entering the physical cores into low-power states.

The process on the left of Figure 3.10 is simplified for the purposes of this thesis to

Name	Type	Bits
Water level	Floating-point	32 bits (4 bytes)
Error	Floating-point	32 bits (4 bytes)
Roughness	Floating-point	32 bits (4 bytes)
Minutes since Jan. 1st, 1970	Positive integer	28 bits (<4 bytes)
Status	Positive integer	4 bits (<1 byte)
Total		**128 bits (16 bytes)**

Table 3.6: Format for each datum within the software.

focus on the data transmission aspect: generating data and placing it into a circular queue. Although a circular queue introduces the possibility of dropping data, it guarantees finite memory usage, which is crucial for a long-running sensor in a remote location. The Swarm modem's internal transmission queue can also drop packets; by default, packets time out after 48 hours, but this value can be configured. Consequently, the circular data queue on the microcontroller was designed to accommodate 48 hours' worth of data.

Efficient Packet Data Bundling

Also note in Figure 3.10 that there are a few implicit sub-algorithms being utilized. First is a routine for bundling data into packets efficiently. This requires specifying a format of representing data and a format of bundling them together. Because the ultimate goal is to minimize power consumption, this means minimizing the number of transmissions to send a given number of data points, which means maximizing the number of data points per 192-byte packet, which means minimizing the number of bytes per data point. Table 3.6 shows the format of data chosen for this particular project.

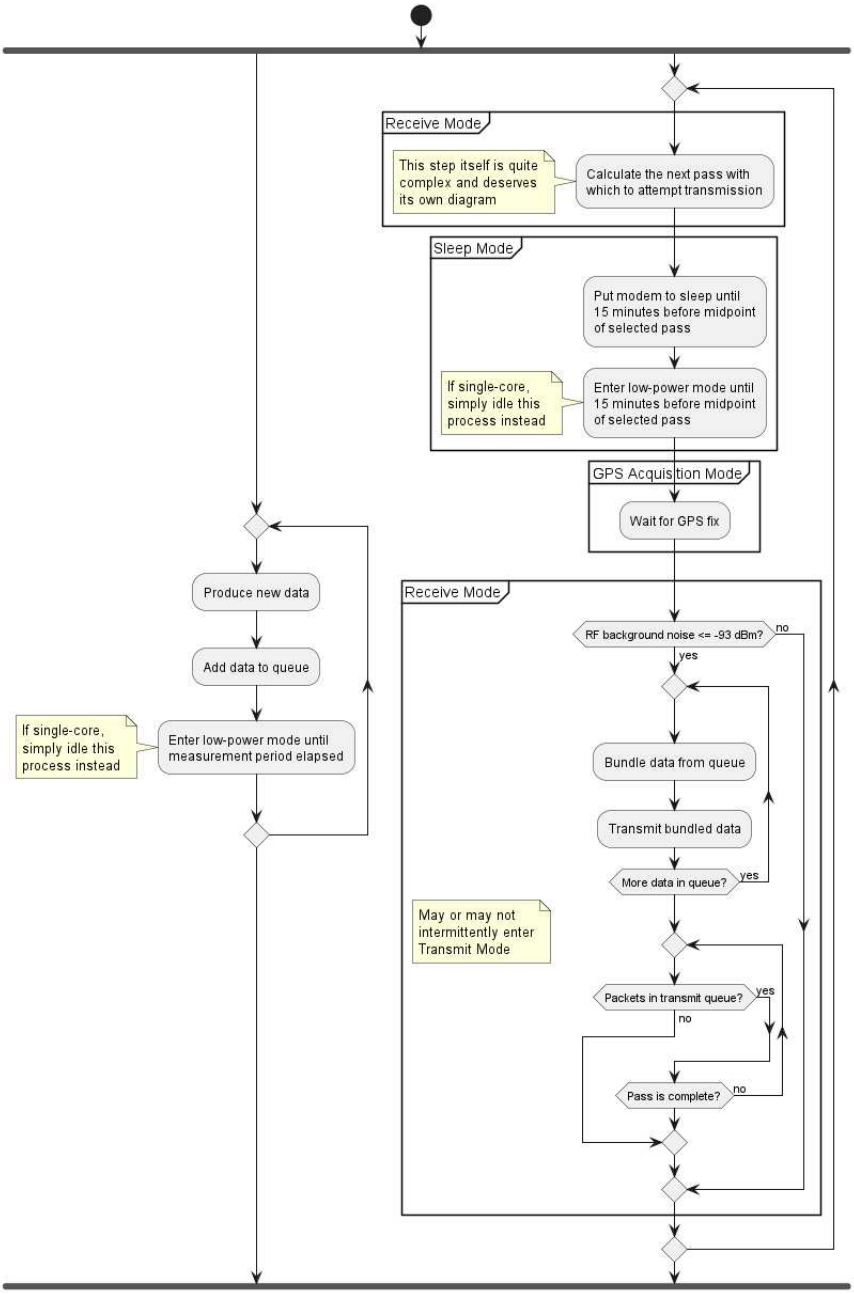

Figure 3.10: Activity diagram for the host device with two processes.

Note that each datum includes a timestamp, represented as minutes since January 1st, 1970. This is similar to Unix time, which is defined as seconds since January 1st, 1970 [29]. The rationale for this choice is that, in this project, temporal precision down to the second is both unnecessary and unattainable due to the nature of the GNSS-R calculations. By using minutes instead of seconds, it saves a few extra bits where the status code can be incorporated using bit manipulation. As a reference, 28 bits can represent time up to the year 2480 CE in this format, while leaving 4 bits to represent up to 16 unique status codes. These 4 bits can be placed as the 4 most-significant bits of the 32-bit value representing the timestamp, as the 28 least-significant bits are more than sufficient for the timestamp representation.

This format allows each datum to fit within 16 bytes, which means each packet will be optimally utilized with 12 data points per packet. Once a bit-minimizing, power-saving data format is selected, it is possible to create an array of data points using C++ (since it uses the Arduino framework) and convert it into raw bytes to send to the modem over UART for transmission.

Beyond data representation and bundling, the second implicit routine within the high-level processes shown in Figure 3.10 is good satellite pass selection. While the algorithm for predicting satellite passes is fairly straightforward from the user's perspective, thanks to an open-source SGP4 satellite pass prediction Arduino library, determining what satellite passes are "good" depends on environmental conditions, setup details, and empirical observations.

Due to the complexity of this aspect, the following subsection is devoted to the topic.

3.5.3 Online Learning Direct-to-Satellite Packet Scheduling

During the development and testing of this project, it was observed that transmitting to satellites can be unreliable. Transmission performance was found to vary significantly due to minor changes in equipment setup or environmental factors. For example, it was observed that heavily overcast days led to RF background noise levels too high for successful transmission (i.e., higher than -93 dBm). Additionally, an unshielded microcontroller within 10 to 20 cm of the antenna could increase measured RF background noise by 5 to 10 dB. Even slight adjustments in the antenna's orientation towards or away from a distant cell tower could impact the RF background noise by several dB. Considering these and potentially countless other factors, creating a generalizable "good" pass model would likely be challenging and require substantial time, resources, and diversity of hardware.

Moreover, previous work on indirect-to-satellite scheduling has demonstrated the success of online learning strategies [10]. Simulations conducted by Huang et al. showed an online-learning algorithm based on Lyapunov optimization theory outperforming a greedy baseline scheduling algorithm in terms of backlog reduction [10]. Specifically, under simulated conditions, Huang et al. reported a 21-point improvement in the percentage of queues with zero backlogged data after a given number of time steps, which the authors

state corresponds with a greater per-byte energy efficiency [10].

Given these findings, it was decided that the best approach for selecting good satellite passes would involve each individual sensor learning for itself, adapting to its specific site conditions and hardware setup through online learning. Numerous potential approaches could be considered for such a learning algorithm. For instance, the previously mentioned work [10] on indirect-to-satellite scheduling addressed the problem as a Lyapunov optimization problem for network queuing, aiming to avoid assumptions about data availability. However, in this direct-to-satellite application, where the frequency of new data generation is known, it is unnecessary to treat it as a network queuing optimization problem. Therefore, a novel approach is employed.

Algorithmic Problem Statement

For the purposes of a sensor that will be placed in a remote location far from human access for potentially years at a time, a simple and interpretable model is preferable. This is so that it can more easily be trusted to perform as expected [30]. To achieve this, a relatively simple algorithm inspired from reinforcement learning has been devised. Before getting to this algorithm, however, it is important to clearly state its goals. Essentially, the goal is for the algorithm to be able to learn an estimate of the probability of successful transmission, given three input variables: 1) the satellite pass duration (in minutes), 2) the maximum elevation angle of the satellite pass (in degrees), and 3) the RF background noise (in dBm).

To begin building the algorithm, some notation and some theory is required. To borrow the notation of reinforcement learning, it can be said the state space S is the set of all possible input variable combinations, and the action space A is the set of all possible actions [26]. In this case, A consists of all possible candidate satellite passes with corresponding pass characteristics $s \in S$. Note that A and S represent the spaces of all possible satellite passes and their corresponding pass characteristics, and the set of actions and corresponding states available at any given time step is a function solely of upcoming satellite passes.

Further borrowing from reinforcement learning notation, it could then be stated that the value function V is the mapping of a given state $s \in S$ to the estimated probability of successful transmission for the satellite pass characteristics represented by s. Lastly, the policy π represents the conditional probability of choosing a particular action $a \in A$ given a state $s \in S$. That is to say the policy is the mechanism for choosing satellite passes.

$$\pi(a|s) = P(A_t = a|S_t = s) \tag{3.1}$$

In Equation 3.1, note that A_t represents the action at time step t, and S_t represents the state at time step t.

Regarding V, a natural objective is thus to approximate it with experience. That is, as the system runs and has successes and failures transmitting with different states $s \in S$, the idea is it will converge closer to the true probabilities of successful transmission for a given state, i.e., the value function V [26, 27].

Modifications to Monte Carlo Learning

Monte Carlo learning is a promising, simple, and interpretable approach to approximating the value function for this problem. In a traditional reinforcement learning problem, Monte Carlo learning considers the value of a state as the average return at the end of an episode [26, 27]. In the context of direct-to-satellite packet scheduling, episodes have a length of one, meaning there is no sequential decision-making, which simplifies the problem. If the reward is set to 1 for a successful transmission and 0 for an unsuccessful one, the value function can be represented by the average rate of successful transmission for a given state.

However, Monte Carlo learning requires a discrete state space, while the state space for this problem is continuous. Discretizing the state space can address this issue. Using the Swarm pass checker, it is evident that satellite passes are mostly between 15 and 90 degrees, with durations ranging from 10 to 60 minutes. Moreover, although RF noise is continuous, modems only report whole numbers, such as -95 dBm. Considering only integer values within the range of -93 dBm (the highest noise level Swarm reports successful transmissions) to -106 dBm (the lowest noise level measured in this project) provides a naturally discretized state space. Table 3.7 demonstrates the chosen discretization of the state space, with 5 buckets for each state variable. This bucketing scheme results in 125 unique combinations, representing the discretized state space. It is important to note that the bucketing scheme is arbitrary, and optimizing it is beyond the scope of this thesis.

The next challenge is determining the policy π. Once a good approximation of the true

Bucket Number	Max Elevation Angle (°)	Pass Duration (minutes)	RF Background Noise (dBm)
1	15 to 30	10 to 20	-93 to -95
2	31 to 45	21 to 30	-96 to -98
3	46 to 60	31 to 40	-99 to -101
4	61 to 75	41 to 50	-102 to -104
5	76 to 90	51 and higher	-105 and lower

Table 3.7: State space bucketing for each state variable.

value function is obtained, the policy could be to exploit and only select the most promising satellite passes. However, the system will initially have no knowledge of what constitutes a good satellite pass, so exploration of passes with varying characteristics is necessary. This presents the classic exploration-exploitation problem in reinforcement learning [26, 27]. The typical approach is to explore early on and gradually shift to exploitation over time.

Modifications to the k-Armed Bandit Problem

The problem of determining the policy for packet scheduling shares similarities with the k-armed bandit problem, where an agent plays the same one-step episode repeatedly. In each game, the agent has a selection of options that may yield varying stochastic rewards. The agent's goal is to learn which actions provide the greatest expected reward so as to maximize the total expected reward over time [26, 27]. A challenge arises if the agent is too greedy early on, as it may not explore sufficiently; conversely, if the agent never becomes greedy, it may fail to exploit when enough information is known. One approach to this problem is softmax (Boltzmann) exploration, which uses the softmax function to calculate a set of

probabilities corresponding to each possible action [28].

The problem concerning this thesis, however, differs from the k-armed bandit problem in two significant ways:

1. The set of actions available to the agent varies in each episode.

2. Maximizing probability of successful transmission is not the sole objective

Regarding the first point, the agent faces a different selection of satellite passes in each episode, each with unique pass characteristics and timings. This can be resolved by using the modified Monte Carlo learning methods described earlier, which allow for tracking the estimated reward of each action.

Regarding the second point, this is because a good pass in an hour is not the same as an equally good pass in 24 hours. This is because of a few factors: data does eventually need to be transmitted, the circular queue holding data has a finite capacity, and the Swarm modem will drop packets from its transmission queue after a specified timeout (by default, 48 hours). Thus, to further borrow from reinforcement learning, a configurable discount factor λ can be applied to the expected reward of satellite passes. However, unlike the discount factor used in traditional reinforcement learning to discount future rewards over multiple time steps, the meaning behind the discount factor used for this problem is to give slight preference to earlier satellite passes. The purpose of this is to prevent long backlogs in the queue caused by waiting too long for satellite passes.

Temporal Bounds for Packet Scheduling

This does bring another issue: timing. Under one data plan, each modem can transmit at most one packet per hour to remain within budget for monthly packets. Additionally, packets will expire within the modem's transmission queue after 48 hours by default. Finally, there are effectively infinite satellite passes to consider as possible pass options. Clearly, some rules are needed for determining the interval of consideration for packet scheduling. Such rules are shown in Figure 3.11.

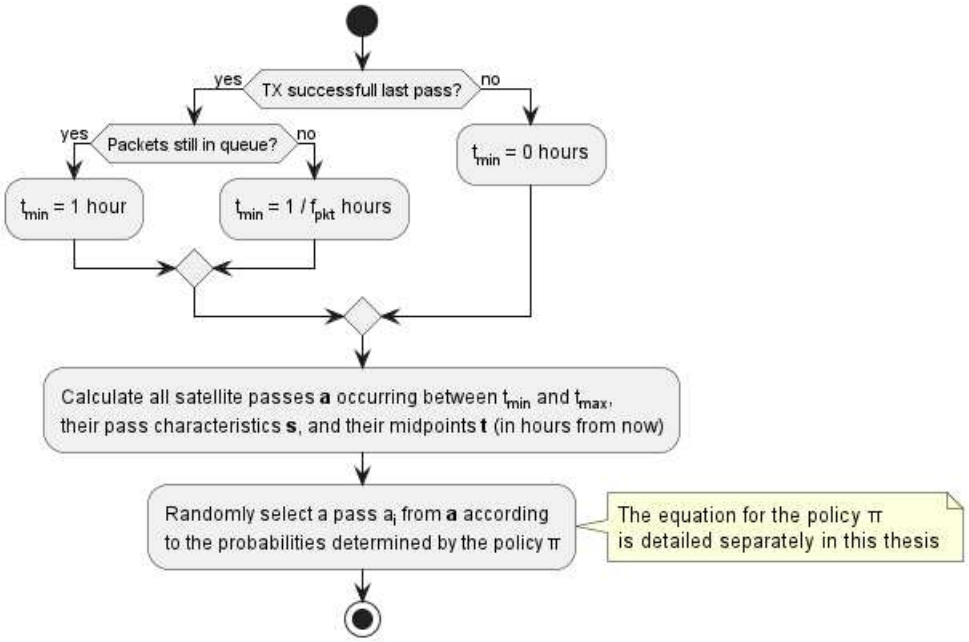

Figure 3.11: Routine for selecting the next satellite pass with which to attempt transmission.

To fully follow Figure 3.11, a bit of notation should be established. Let t_{min} and t_{max} be the minimum and maximum amount of time (in hours) from the present moment for

a satellite pass, respectively. Let $\mathbf{a}$ be a vector representing the set of all satellite passes occurring between t_{min} and t_{max} hours from the present, and let a_i be the i-th element of $\mathbf{a}$ chronologically. Similarly, let $\mathbf{s}$ be a vector representing the set of all corresponding states of all satellite passes in $\mathbf{a}$, and let s_i be the i-th element of $\mathbf{s}$ chronologically. Finally, let $\mathbf{t}$ be a vector representing the set of all corresponding midpoint times (in hours from the present) of all satellite passes in $\mathbf{a}$, and let t_i be the i-th element of $\mathbf{t}$ chronologically. Note that $\mathbf{a}$ is encoded simply as a chronological indexing of upcoming candidate passes, meaning that selecting the i-th element of $\mathbf{a}$, a_i, represents the action of selecting the i-th next satellite pass.

Algorithmic Formulation

Also, let r_{data} be the rate at which data points are generated (in data points per hour), let $bundlesize$ be the number of data points that comprise a full bundle (as described in the previous section), and let r_{pkt} be the rate of full packet bundling, i.e., how many times per hour a full bundle of new data will be produced.

$$r_{pkt} = \frac{r_{data}}{bundlesize} \tag{3.2}$$

Finally, let $\text{softmax}(\mathbf{z})$ be the vectorized softmax function and let $\text{softmax}(\mathbf{z})_i$ be the softmax function for the i-th element of a vector $\mathbf{z}$, and let $v(s)$ be the value function for a given state (i.e., pass characteristics) s. Like with softmax, also let $v(\mathbf{s})$ represent the

vectorized value function. With these variables, it is now possible to construct a policy π:

$$\pi(a_i|s_i) = \text{softmax}(\lambda^{\mathbf{t} \ominus t_{min}} \odot v(\mathbf{s}))_i \tag{3.3}$$

First, note that the symbols $\ominus$ and $\odot$ denote element-wise subtraction and multiplication, respectively, as they operate on the vectors $\mathbf{t}$ and $\mathbf{s}$. With this in mind, Equation 3.3 essentially states that the probability of selecting a satellite pass a_i from the interval between t_{min} and t_{max} hours from the present is equal to the softmax of the estimated probability of transmission success for that pass, multiplied by a discount factor based on its future occurrence. This formulation results in higher probabilities for passes with better quality and closer temporal proximity to the present (i.e., occurring less far in the future), while still allowing for exploration of passes predicted to be less successful. This approach enables Monte Carlo learning to refine the value function estimates for each state over time. Prioritizing passes that occur sooner also minimizes data loss due to full queues or dropped packets.

The modified Monte Carlo learning designed for this problem acts as follows:

1. Initialize $V(s)$ to one for all buckets $s \in S$ and a counter $N(s)$ to zero for each state s.

2. Generate a list of candidate satellite passes $\mathbf{a}$ and their corresponding bucketed pass characteristics $\mathbf{s}$ and pass midpoint times $\mathbf{t}$.

3. Select an action a_i from $\mathbf{a}$ by following the policy π in Equation 3.3.

4. For the corresponding state s_i from **s** visited in the episode, update the value function and counter:

   - Calculate the return G_t. If transmission was successful, $G_t = 1$. Else, $G_t = 0$.
   - Update the counter $N(s) = N(s) + 1$.
   - Update the value function $V(s) = V(s) + \frac{1}{N(s)}(G_t - V(s))$.

5. Repeat steps 2-4.

3.5.4 Uplink Transmission Energy Model

At this point in the thesis, various insights on system power consumption have been presented, as well as designs and algorithms that employ multiple energy-saving strategies. Next to introduce is a unified uplink transmission energy model to estimate the average power of a practical system, using the power consumption values of the modem in different modes, as previously discussed in this thesis.

In terms of the stochastic nature of transmission, two main questions need to be addressed: whether a transmission will succeed during a given pass; and, if successful, how long the modem will be in Receive Mode before it can transmit.

To begin building the model, let t_{SL} be the empirical (either measured or simulated) mean time the modem is in Sleep Mode, t_{GPS} be the mean time the modem is in GPS Acquisition Mode, and t_{RX} be the mean time the modem is in Receive Mode before transmission is

successful. With this and some known typical modem power consumption numbers—P_{SL}, P_{GPS}, and P_{RX}—total energy usage in these modes over a single transmit attempt cycle, $E_{attempt}$, can be calculated. For the sake of simplicity, the value of E_{TX} (the energy required to transmit a full packet) provided by Swarm for a full 192-byte packet will be used.

$$E_{attempt} = P_{SL}t_{SL} + P_{GPS}t_{GPS} + P_{RX}t_{RX} + E_{TX}N_{pkt} \tag{3.4}$$

In the above equation, note that N_{pkt} represents the number of packets transmitted in a given pass. Depending on satellite pass selection and/or previous transmission attempt successes or failures, N_{pkt} may be 1 or larger. Or, in the case of an unsuccessful attempt, N_{pkt} may be 0. Using this, an expression for N_{pkt}, when N_{pkt} is not 0, may be derived.

$$N_{pkt} = \frac{r_{pkt}}{p_{success}r_{attempt}} \tag{3.5}$$

Note that $p_{success}$ is the probability of transmission success, $r_{attempt}$ is the mean transmission attempt rate, and r_{pkt} is the frequency at which new, fully bundled packets are generated. Because of the earlier-stated restriction on t_{min} based on r_{pkt} from Figure 3.11, $r_{attempt}$ is guaranteed to be less than or equal to r_{pkt}, meaning N_{pkt} is guaranteed to be 1 or greater. The possibility of more than 1 packet per transmission attempt is because it is assumed that successful transmission of 1 packet will usually entail successful transmission of all queued packets.

In Equation 3.4, also note that, while P_{SL}, P_{GPS}, t_{GPS}, P_{RX}, and E_{TX} (at least for full packets) are functionally constant, t_{SL} and t_{RX} are variable as well. Here, t_{SL} represents the mean time the modem is in Sleep Mode before making a transmission attempt, and thus can be represented approximately as follows:

$$t_{SL} = \frac{1}{r_{attempt}} \tag{3.6}$$

Meanwhile, t_{RX} depends primarily on how long the modem waits until either it receives a packet from a satellite and begins transmission or the pass is over. For a successful transmission attempt, the most optimistic assumption may be that it is able to transmit almost immediately after exiting GPS Acquisition Mode. Meanwhile, the pessimistic assumption for a successful attempt may be that it only transmits at the very end of the satellite pass. Mathematically, this is also easier to formulate, as the most pessimistic case would be the modem reaching the end of a given satellite pass in Receive Mode, only to not have transmitted. In terms of t_{RX}, this case and successful transmission at the very end of the pass would be approximately equal. To note is all three of these cases depend on the mean pass duration, which can be denoted as t_{pass}.

Using the previously stated Equation 3.4 for $E_{attempt}$, it can take two forms depending on if the transmission attempt was a success or failure. If it is a success, it can be expressed as follows in Equation 3.7, where ϵ_{pass} represents the proportion of a satellite pass the modem idles in Receive Mode before it receives a packet from the satellite and is able to transmit.

For pessimistic and optimistic models, ϵ_{pass} can be treated as either 1 or 0, as these serve as the upper and lower bounds of possible time in Receive Mode for a given satellite pass.

$$E_{success} = P_{SL}\frac{1}{r_{attempt}} + P_{GPS}t_{GPS} + \epsilon_{pass}P_{RX}t_{pass} + E_{TX}\frac{r_{pkt}}{p_{success}r_{attempt}} \tag{3.7}$$

Likewise, if the attempt is a failure, the model can be represented as follows in Equation 3.8. Note that there is no ϵ_{pass} value and no N_{pkt}, as the system will have to wait out a full pass without transmitting any packets.

$$E_{fail} = P_{SL}\frac{1}{r_{attempt}} + P_{GPS}t_{GPS} + P_{RX}t_{pass} \tag{3.8}$$

The above two equations can be combined for a more complete model:

$$E_{attempt} = p_{success}E_{success} + (1 - p_{success})E_{fail} \tag{3.9}$$

This then simplifies down into the following expression:

$$E_{attempt} = \frac{P_{SL}}{r_{attempt}} + P_{GPS}t_{GPS} + p_{success}(\epsilon_{pass}P_{RX}t_{pass} + \frac{E_{TX}r_{pkt}}{p_{success}r_{attempt}}) + (1 - p_{success})P_{RX}t_{pass} \tag{3.10}$$

In the above equation, P_{SL}, P_{GPS}, and P_{RX} are all constants and given by Swarm. The exact values only depend on if one's modem is integrated using 3.3 V or 5 V power supply.

The calculated results for this project will be given using 5 V power supply values, as that is what was used. Similar to these, t_{GPS} is rather constant, and Swarm reports to be about 30 seconds typically. Also note that $r_{attempt}$ will depend on site conditions, project requirements, and packet scheduling algorithm. Even a non-learning algorithm will need some method of determining which satellite passes to attempt, after all. Similarly, $p_{success}$ and t_{pass} depend heavily on site conditions and packet scheduling algorithm. Lastly, r_{pkt} depends on project requirements and data serialization scheme.

3.6 Summary

This chapter provided details on the implementation of a custom PCB and selection and integration of a microcontroller with the Swarm M138 modem. It also detailed several routines for efficiently serializing data for and managing overall operation of the Swarm M138 modem. This section also presented the algorithmic problem statement for the proposed online learning direct-to-satellite scheduling algorithm, and it formulated the algorithm itself. Finally, this section derived an energy model that may be used in evaluating expected average power of a practical system using the Swarm M138 modem. From here, the logical next step is to simulate the proposed learning algorithm and to evaluate those simulated results according to the derived energy model.

Chapter 4

Experimental Results

A key goal of this thesis was to document the process of integrating the Swarm modem onto a custom PCB with a microcontroller, and to interface with the modem with software in order to minimize power consumption. Much of the testing was done as simple verification testing, e.g., using the PCB, microcontroller, and modem to see if they worked together. There are, however, two key results beyond simple verification: 1) simulated testing of the online learning direct-to-satellite packet scheduling algorithm, and 2) an energy model of average modem power consumption.

4.1 Experimental Methodology

While some of the testing involved in the project in this thesis is simply to verify that the hardware and software works as expected, there is more complexity in testing two key aspects.

First is some form of testing to demonstrate the value of the online learning direct-to-satellite packet scheduling algorithm. The second is the application of the previously-derived energy model (Equation 3.10) to demonstrate what the average power of a whole operational system might look like.

4.1.1 Simulation of Online Learning Algorithm

The ultimate test for the online learning direct-to-satellite packet scheduling algorithm would involve setting up multiple sensors under varying conditions in the field and monitoring their transmission success rates over weeks or months. However, this method is time-consuming, expensive, and requires more advanced, production-ready hardware, which is beyond the scope of this thesis. Instead, the algorithm is tested using a simulated environment and simulated data as a proof of concept, an approach also employed in previous work on indirect-to-satellite scheduling [10].

Simulations are employed primarily because they can demonstrate the algorithm's ability to learn an underlying unknown pattern about satellite pass quality. Additionally, simulations offer a quicker, more cost-effective, and easier way to verify the algorithm's potential compared to long-running field tests involving hardware. Simulations can also be used to tune the discount factor λ hyperparameter.

To create a simulation for the algorithm, two main components are required: 1) virtual transmitters with an underlying transmission success probability model, and 2) randomly

Preference Model	Max Elevation Angle Preference	Pass Duration Preference	RF Background Noise Preference
1	High angles	Long durations	Low noise
2	Mid to high angles	Mid to long durations	Low to mid noise
3	Low to high angles	Short to long durations	Low to high noise

Table 4.1: Conceptual preference models for the virtual transmitters.

generated satellite pass characteristics and RF noise data.

Three conceptual preference models were developed and translated into mathematical models, as shown in Table 4.1. Note that, in the table, the preferences refer to the conditions required for a high likelihood of success. For example, the first preference mode requires high angles, long durations, and low noise in order to have a high likelihood of success. The aim of these preference models was to investigate how different transmission difficulties would impact the algorithm's performance, such as its ability to perform well with a preference model requiring more stringent conditions for successful transmission.

To create a mathematical model of each virtual transmitter, a function is declared that takes in the three variables above as inputs and outputs a probability from 0 to 1 of transmission success. The functions themselves are constructed by simply multiplying three stretched-and-shifted sigmoid curves together, one for each variable preference. For example, the desired sigmoid to represent a preference for high angles would be one shifted and stretched so as to produce a value close to 1 for high angles (e.g., 70 degrees and higher) but a value close to 0 for low angles (e.g., 30 degrees and lower). The source code

Preference Model	k_θ	θ_0	k_d	d_0	k_γ	γ_0
1	0.5	70	0.5	35	-1	-102
2	0.5	50	0.5	20	-1	-99
3	0.5	30	0.5	10	-1	-96

Table 4.2: Constants for the three preference models.

for this can be accessed from the appendix, or the equation form in Equation 4.1.

$$P(success) = \sigma(k_\theta(\theta - \theta_0)) \times \sigma(k_d(d - d_0)) \times \sigma(k_\gamma(\gamma - \gamma_0)) \tag{4.1}$$

Note that $\sigma(x)$ represents the sigmoid function, θ represents the max elevation angle, d represents the pass duration, and γ represents the RF background noise. Also note that k_θ, k_d, k_γ, θ_0, d_0, and γ_0 represent configurable stretching and shifting constants to represent the different conceptual preference models. The different values chosen for these constants to create the three preference models are shown in Table 4.2.

The intuition behind this mathematical representation of these preference models is not to create a true-to-life model of transmission success, but to create simple underlying transmission success probability models for the algorithm to learn. If the algorithm can learn value function estimates for these, it ought to be able to learn real-world value function estimates.

With the virtual transmitters, simulated satellite passes with randomly generated pass characteristics can be fed to them. This is done with agents, which each have a preference model and a value function approximator. Each iteration, each agent is given a random

RF background noise value and a vector **a** of satellite passes, their corresponding random midpoint times **t**, and their corresponding randomly generated pass characteristics **s** (except each s_i from **s** also includes the RF noise value). The randomly generated pass characteristics are drawn from a uniform distribution, and the RF background noise values are drawn from three differing distributions for three different experiments:

1. Uniform across all buckets (-107 to -93 dBm).
2. Uniform within one bucket (-107 to -105 dBm).
3. Constant (-106 dBm).

It is possible a given sensor may experience the full range of possible RF noise intensities, but it is also possible a given sensor in a remote location will only often see a narrow sub-range of possible RF noise intensities. Thus, these three noise configurations were selected to observe how the algorithm performs under these differing RF noise distributions.

Each agent then calculates the probabilities of selecting each satellite pass from the discretized states, the agents' value function approximators, and when the pass midpoints occur in time. These probabilities are calculated according to the policy π. Then, satellite passes are chosen randomly for each agent according to the calculated probabilities. With the satellite passes chosen, the satellite pass characteristics and the RF background noise values are fed into the agents' respective preference models to produce probabilities of transmission success. Finally, transmission successes are randomly determined according to the agents'

preference model outputs, the agents' value function approximators are updated, and the whole process repeats.

4.2 Simulated Online Learning Results

Figures 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6 present the simulated results of the online learning direct-to-satellite packet scheduling algorithm discussed earlier. Although these specific results may not accurately represent real field conditions, the simulation's main objectives were to 1) demonstrate the algorithm's ability to learn an unknown pattern behind transmission success probabilities, and 2) highlight key factors affecting the algorithm's performance in response to essential variables.

The most noticeable outcome is the algorithm's response to overall transmission "difficulty." When transmission is either relatively hard or easy, the algorithm has limited potential for significant improvements in the success rate. For instance, when there are few good passes available, the algorithm often needs to choose between a mediocre and a bad pass. This scenario is observed in the first preference model's lower (but still notable) improvement.

Another important observation is the simulated response to different discount factors. In the first preference model, which had a lower likelihood of successful transmission, varying discount factors had little impact on $r_{attempt}$, as the penalty for waiting for a decent pass outweighed the cost of other poorer options. However, for the other two preference

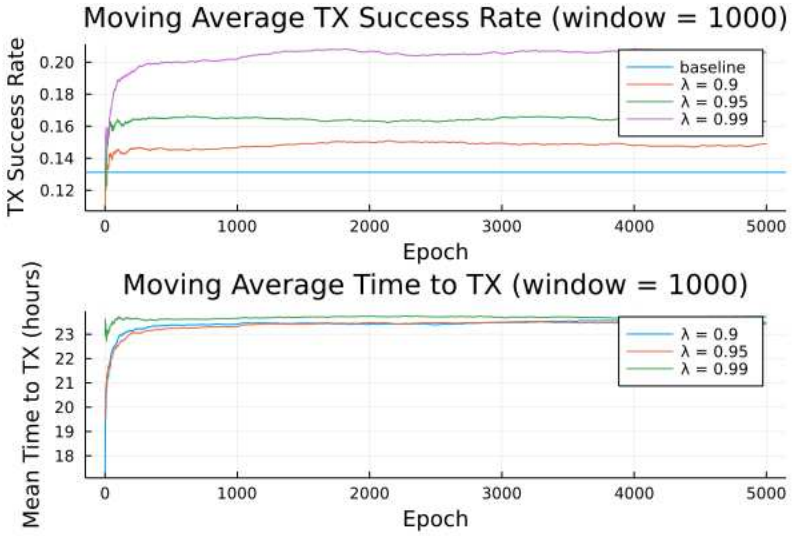

Figure 4.1: Simulated results for preference model 1 and random noise within 1 bucket.

models, discount factors closer to 1 resulted in significantly higher average times to attempt transmission. One potential limitation of these simulations might be the distribution of satellite pass characteristics. Real satellite passes are periodic and RF noise is usually consistent for a given location, while simulations assume a uniform distribution of all characteristics. To maintain a higher $r_{attempt}$, using a lower value of t_{max} or conducting field tests for periodicity may yield improved results.

Regarding noise distribution, Figures 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6 also compare results for two noise models: one with RF noise values drawn randomly and uniformly from all possible values, and another with RF noise values drawn randomly and uniformly from a single bucket. The second model was simulated because most remote sites are expected to have relatively consistent RF background noise levels. The main effects of this constraint are

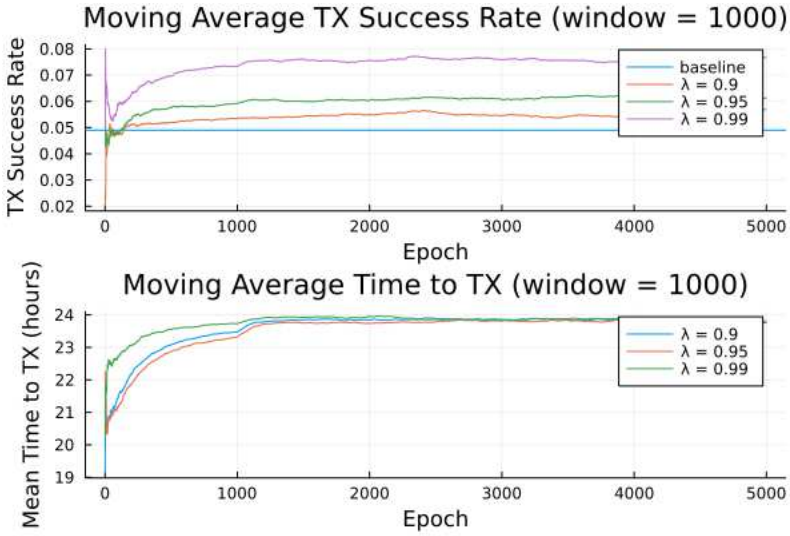

Figure 4.2: Simulated results for preference model 1 and random noise across all buckets.

that the algorithm consistently learned faster and converged to higher success rates when limited to the same bucket of RF noise values. For example, the algorithm learned in 1000 epochs for preference model 2 and fully random noise what it learned in under 300 epochs for same-bucket noise. For reference, 1000 epochs amount to about 42 days at one attempt per hour or nearly 3 years at one attempt per 24 hours, while 300 epochs correspond to about 13 days at one attempt per hour or just under a year at one attempt per 24 hours. Nevertheless, the algorithm demonstrates its capability to learn.

4.3 Simulated Power Consumption Results

Because the ultimate purpose of the packet scheduling algorithm is to reduce average power, it is valuable to demonstrate what sorts of energy savings some of the simulated results above

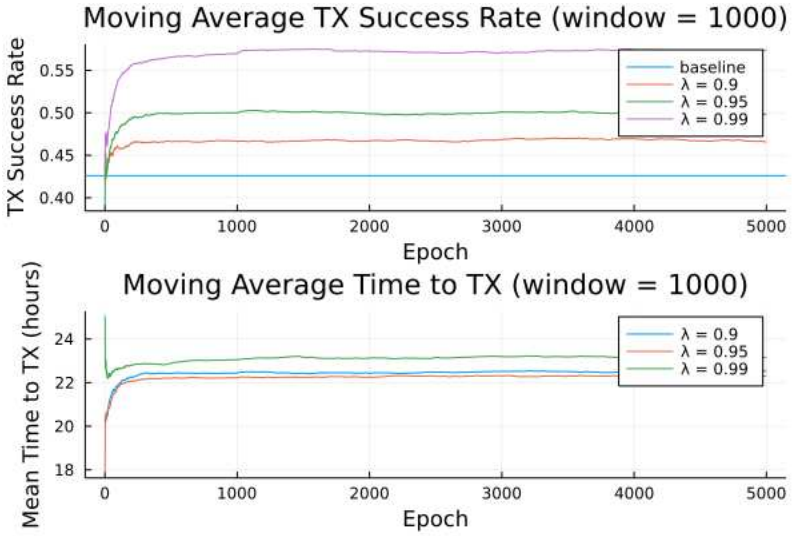

Figure 4.3: Simulated results for preference model 2 and random noise within 1 bucket.

Constant	Value
P_{SL}	$550\mu W$
P_{GPS}	$230mW$
t_{GPS}	$30s$
P_{RX}	$130mW$
E_{TX}	$12.24J$
r_{pkt}	$\frac{1}{3}hr^{-1}$

Table 4.3: Sample constants used for transmission attempt energy model.

would bring. Using the previously derived Equation 3.10 for $E_{attempt}$, some values for the variables can be inserted to produce sample results.

Constants used within Equation 3.10 for sample results are shown in Table 4.3. Note that some of these may depend on exact hardware setup and use case, but they are all constant with respect to the packet scheduling algorithm. This is in contrast to the terms that are variable with respect to the algorithm, which are detailed next.

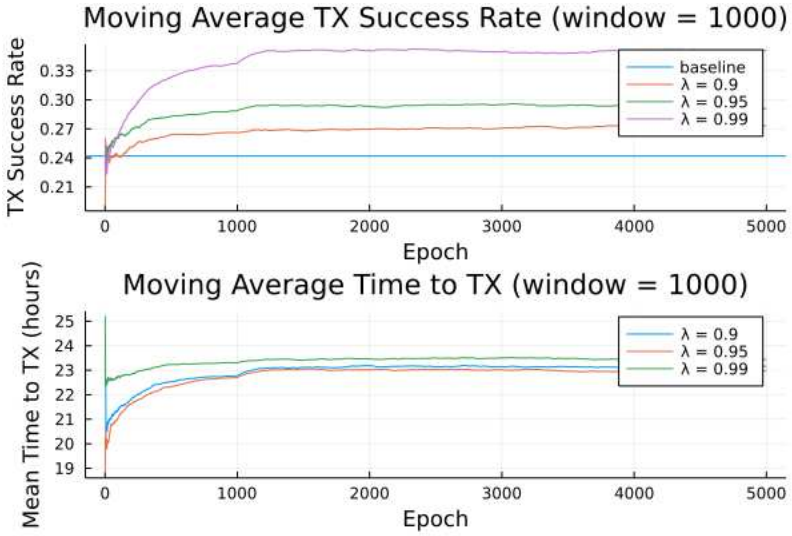

Figure 4.4: Simulated results for preference model 2 and random noise across all buckets.

In addition to the above constants, the ranges of the variables in Equation 3.10 are shown in Table 4.4. Note that "pessimistic" refers to the boundary of the interval that results in higher average power, per the equation for $E_{attempt}$, and "optimistic" likewise refers to the boundary of the interval that results in lower average power. For example, a lower $r_{attempt}$ means less frequent transmission attempts. In terms of power consumption, this is good, but too high of an average time to transmit may result in lost data. Also note the difference between average power and the value for $E_{attempt}$ given by Equation 3.10; while a low $r_{attempt}$ will, all else held equal, result in a higher $E_{attempt}$, it will result in lower average power, as shown by Equation 4.2 below. The term in the denominator is the average time elapsed during a complete cycle.

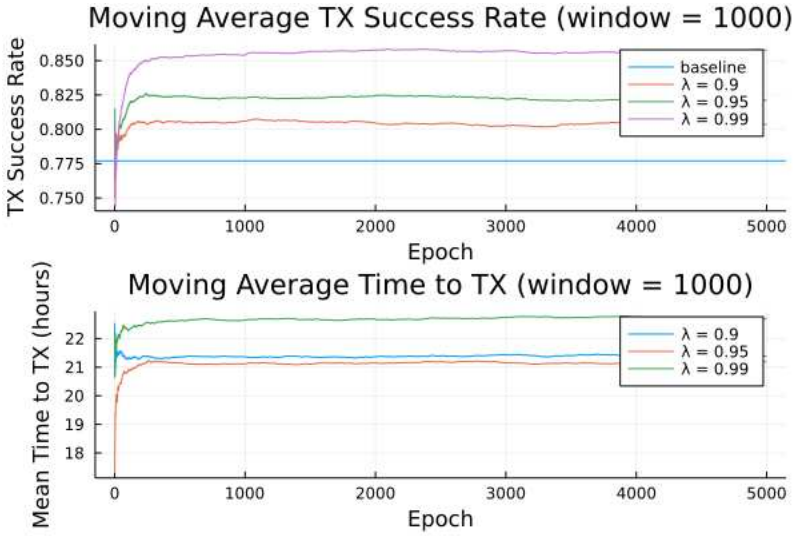

Figure 4.5: Simulated results for preference model 3 and random noise within 1 bucket.

Variable	Pessimistic Value	Optimistic Value
$r_{attempt}$	$1hr^{-1}$	$\frac{1}{48}hr^{-1}$
$p_{success}$	0.0	1.0
ϵ_{pass}	1.0	0.0
t_{pass}	$60min$	$10min$

Table 4.4: Sample variable ranges for transmission attempt energy model.

$$P_{avg} = \frac{E_{attempt}}{\frac{1}{r_{attempt}} + t_{GPS} + p_{success}\epsilon_{pass}t_{pass} + (1 - p_{success})t_{pass}} \tag{4.2}$$

The results of these four variables on average modem power consumption are shown in Figure 4.7. Most important to observe from the plotted results is that the two dominant factors in determining average power are average success rate and average time between attempts. What does not appear in the plotted results, however, is that these two variables are not independent of one another; rather, higher success rates lead to lower attempt

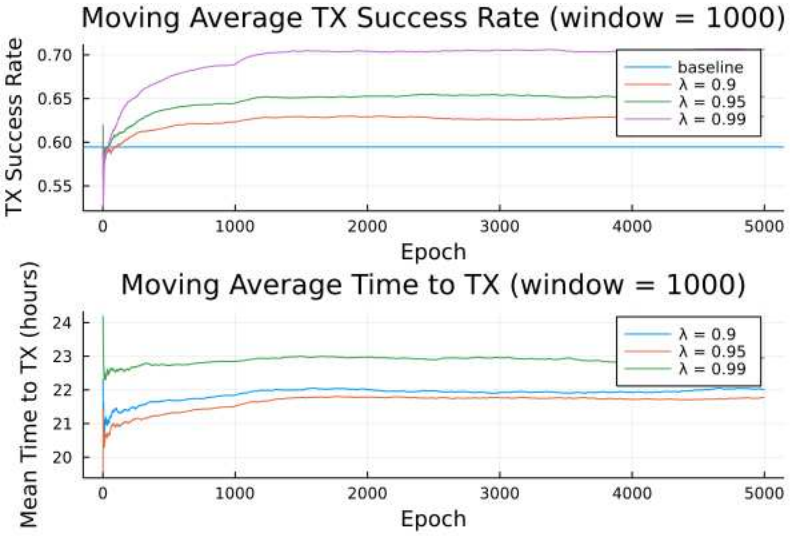

Figure 4.6: Simulated results for preference model 3 and random noise across all buckets.

frequencies, as detailed earlier in this thesis.

To give some perspective on the potential impact of improved success rates, then, see Table 4.5. Note that the $p_{success}$ and $r_{attempt}$ values are taken from the simulated results, and ϵ_{pass} and t_{pass} are taken as 0.5 and 25 minutes, respectively. For each preference model, three results are shown: 1) a baseline based on taking the earliest available passes and no scheduling, 2) another baseline based on the same average $r_{attempt}$ as the simulated results but no scheduling, and 3) the test case with scheduling and simulated average $r_{attempt}$.

For the simulated baseline values, however, the average time to attempt is represented as the first available satellite pass, modeled simply by Equation 4.3 below, which represents the algorithm for determining t_{min} from earlier in this thesis, where successful transmission attempts lead to waiting a minimum of $\frac{1}{r_{pkt}}$ hours, while unsuccessful attempts lead to no

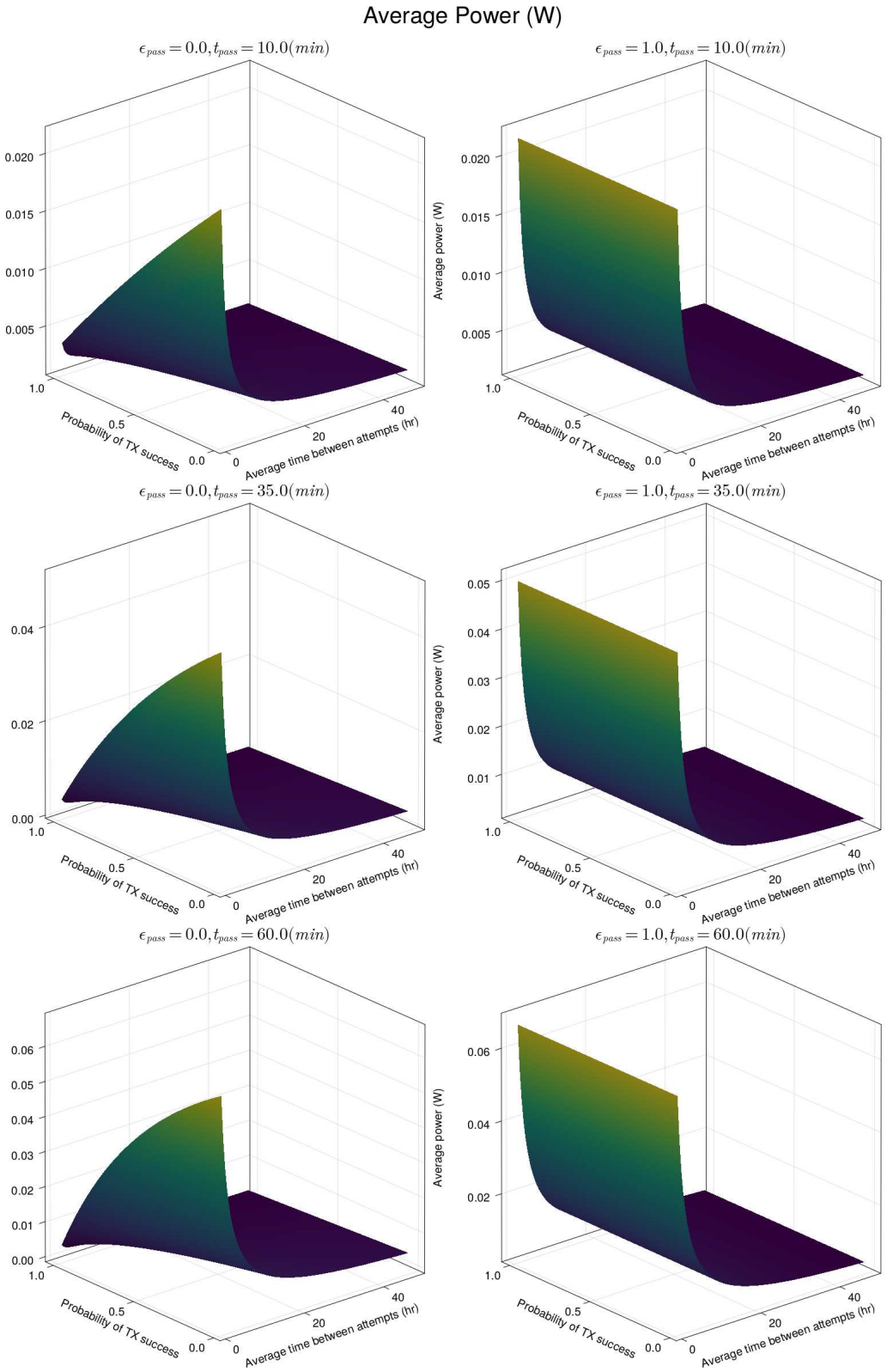

Figure 4.7: Average power of the Swarm modem under various variable values.

Preference Model	Success Rate	Attempt Frequency	Average Power	Required Battery Capacity
	0.13	$2.564hr^{-1}$	$67.54mW$	$592.1Wh$
	0.13	$\frac{1}{24}hr^{-1}$	$3.810mW$	$33.40Wh$
1	0.20	$\frac{1}{24}hr^{-1}$	$3.735mW$	$32.74Wh$
	0.42	$0.7937hr^{-1}$	$29.31mW$	$256.9Wh$
	0.42	$\frac{1}{23}hr^{-1}$	$3.575mW$	$31.34Wh$
2	0.57	$\frac{1}{23}hr^{-1}$	$3.405mW$	$29.85Wh$
	0.78	$0.4274hr^{-1}$	$14.95mW$	$131.1Wh$
	0.78	$\frac{1}{22}hr^{-1}$	$3.234mW$	$28.35Wh$
3	0.85	$\frac{1}{22}hr^{-1}$	$3.151mW$	$27.62Wh$

Table 4.5: Sample power and battery savings from simulated packet scheduling for a year of operation.

minimum wait.

$$t_{SL,baseline} = \frac{p_{success}}{r_{pkt}} \Rightarrow r_{attempt,baseline} = \frac{r_{pkt}}{p_{success}} \tag{4.3}$$

The simulated results show the potential the online learning direct-to-satellite packet scheduling algorithm has in reducing average power and battery requirements. Note, however, these are from simulated data, and this is for the modem power consumption alone. Microcontroller and sensing equipment power are not included. Also note that the dominant source of the power savings come from a lower average attempt frequency, although the improved success rate of the scheduling algorithm does introduce not-insignificant power saving. Additionally, a direct-to-satellite packet scheduling algorithm can be considered as an enabler for lowering the attempt frequency, as it

provides a built-in mechanism for selecting future passes.

4.4 Summary

This chapter detailed how the proposed online learning direct-to-satellite scheduling algorithm was simulated. These simulations were not designed to be true-to-life, rather to evaluate the proposed algorithm's ability to learn and to evaluate characteristics of its behavior. Further, the proposed algorithm was evaluated for average power under the derived energy model for a Swarm modem-based system. These simulated results and the implications they hold for future work are discussed in the following chapter.

Chapter 5

Discussion and Future Work

There are several key takeaways from the development of low-cost, low-power satellite-powered communications in the IoRT. Many of these relate to a few core notions that designing practical devices of this nature is highly integrated, finicky, and there are many variables for which to account. Because of the core challenge of low-cost, low-power IoRT, work must be done at every level. A custom PCB needs to be designed with care for decoupling capacitance, hardware design must be such that the antenna is shielded from noise produced by the microcontroller itself, and data must be serialized, bundled, scheduled, and transmitted efficiently to save precious bits and joules.

As such, the primary goal of this thesis was to present a number of practical considerations for designing and implementing a satellite-powered Internet of Remote Things device, from lower level hardware considerations to higher level algorithmic design. Therein lies the source

of this thesis's primary limitations, however. Because this thesis was intended with a higher level, integrative perspective, it does not produce a lot of production-ready designs, nor does it take too hard a look into any single component or subsystem.

5.1 Physical and Hardware Limitations

For example, one factor that was discovered in verification testing is the sensitivity of the antenna to noise. It is so sensitive, in fact, that it was observed that simply placing the PCB with the microcontroller underneath the ground plane tended to reduce RF noise by several dB. Further, minor changes in the exact positioning of the PCB and microcontroller under the ground plane could vary the measured noise by as much as 3 dB. Clearly, the antenna is capable of picking up RF interference from an unshielded device in the immediate vicinity (e.g., 10 to 20 cm). Factors such as this illustrate the importance of a finalized, production-ready housing and ground plane. It was beyond the scope of this thesis and the expertise of this author to enter the realm of mechanical housing and antenna ground plane design. Nonetheless, these factors evidently impact the RF characteristics of the device, and can impact overall ability to transmit. In fact, it was precisely this that played a role in the design of a online learning direct-to-satellite packet scheduling algorithm; because this thesis could not possibly account for the range of possible housing, ground plane, and even power supply designs, as well as varied site conditions and lines of sight, there could be no single ultimate "good" pass model.

5.2 Limitations of the Online Learning Algorithm

The online learning direct-to-satellite packet scheduling algorithm does also have its limitations. Key amongst these is that it currently relies on simulated results. While the results achieved in this thesis do demonstrate the ability of the algorithm to learn, and they do indeed demonstrate the impacts of several variables on the performance of the algorithm, they are nonetheless simulated results. They give no real-world example of exactly how the algorithm performs on real data in the field. There are two natural steps that could be taken in future endeavors to improve on this: 1) the simulations could be made to more closely reflect real-world data, or 2) multiple sensors could be placed out in the field in different locations for several weeks or months.

Between these two options, the first one is the natural first step for the same reason simulations were chosen in the first step: it is much cheaper and faster to simulate than to set up multiple devices for potentially months at a time. Additionally, because the risk of poorer-than-desired results leading to tweaking the algorithm and starting anew is so high, it makes much more sense to do as much tweaking in simulations as possible and only to test on hardware when confident in good results.

5.2.1 Potential Simulation Improvements

The most apparent way to improve the simulations would be to generate a more representative RF noise distribution and to use actual Swarm satellite passes. Regarding

the former, one could set up a few devices in reasonably accessible locations and have them simply log RF background noise (in dBm) over several days, preferably under differing weather conditions. One could then retrieve the devices and perform a statistical analysis on the recorded RF noise data with which to generate more realistic distributions. Regarding the latter, there are regular versions of the same type of satellite pass prediction library as used on the microcontrollers. Where the microcontrollers use an open-source SGP4 Arduino library, there are existing open-source libraries meant for regular desktop use, such as in Python. One could use one of these libraries along with randomized coordinates to make satellite pass predictions. This would solve the issue of the simulations using fully randomly generated pass characteristics, and it would instead likely improve $r_{attempt}$. This is because the periodic nature of satellite passes means decent to good passes likely occur relatively regularly, rather than randomly and uniformly distributed over the interval (t_{min}, t_{max}) as in the current simulations.

5.2.2 Tradeoff Between Low Power and Learning Rate

This does relate to another key weakness of the satellite pass algorithm in its current state, at least on the simulated data: its low average $r_{attempt}$ values. This is very important to improve, as it can be the difference between a real sensor taking 3 years to learn and a month to learn. While, as stated above, this very well might be obviated with a more true-to-life simulation, other strategies such as further limiting t_{min} may be necessary.

A downside to this, of course, is likely smaller power savings, as more frequent transmissions requires significantly more average power. That is to say there exists a tradeoff between having a high learning rate and achieving low average power. A key point in any practical implementation would be balancing learning rate and transmission attempt frequency, as a slow learning rate may take too long to improve (and thus become maximally efficient), but a high learning rate may require much more average power. Perhaps further research could examine the possibility of adaptive learning rates. One possibility for polar regions would be increasing the transmission attempt frequency (e.g., by decreasing the discount factor λ) during the summer when solar power is abundant, and lowering the transmission attempt frequency during the polar winters when solar power is absent.

5.2.3 Potential Online Learning Algorithm Improvements

There is also another limitation related to the packet scheduling algorithm: only one algorithm was designed and tested. While this algorithm did prove able to learn, and while it does achieve a desired property of interpretability, it does have its weaknesses. First and foremost is the bucketing mechanism to discretize an otherwise continuous state space. The bucketing significantly reduces the resolution of the algorithm, as, for example, 30 degrees max elevation angle is considered to be the same bucket as 15 degrees, but not the same bucket as 31 degrees. This not only reduces the resolution, but also likely the accuracy of

the value function approximation, as opposite ends of each bucket are considered equivalent.

One future improvement could be to fuzzify the discretized states such as in fuzzy logic [31]. The simplest approach to this could be to construct a linear interpolation between each bucketed value. For example, 31 degrees max elevation angle would be considered to be roughly half into the first bucket and half into the second bucket. This would necessitate reformulating the Monte Carlo learning and softmax exploration to account for partial membership in the discretized states, but it would likely help to alleviate the resolution and accuracy issues of the algorithm presented in this thesis. One could also further embrace the fuzzification and go in the direction of a more standard fuzzy inference or fuzzy control system [31, 32].

5.3 Other Potential Optimizations

Beyond the packet scheduling algorithm and simulations, however, there are a few other possible future directions. First is that, while the data binarization and bundling scheme is rather efficient (for this project, it allows up to 12 data points per packet, at only 16 bytes per datum), improvements are certainly possible. These, however, would likely involve more advanced techniques and may not yield huge savings in energy. Nonetheless, the following are two ideas: 1) data compression, and 2) floating-point truncation. For data compression, there are many potential encoding schemes that could likely save a few bits or

bytes. This thesis anticipates they would be most helpful if they allow fitting in an extra datum (or more) per packet. Regarding floating-point truncation, this is highly project-dependent. In some use cases, it may be acceptable to lose the precision afforded by 32-bit floating-point numbers by truncating the mantissa and/or lose the range afforded by 32 bits by truncating the exponential portion. Alternatively, there are other floating-point binary number representations designed for high precision over prescribed intervals, which may be an option for certain applications [33].

Another area of future direction would be further work into lowering maintenance of these devices. While the devices for this project were designed generally to be low-maintenance, there are a few areas that could receive more attention to improve this. The most immediate way to do this would be simple testing. Leave a device (or perhaps several) running for several weeks or months in order to detect any potential memory leaks or memory bugs that may have inadvertently entered the embedded software. Another would be to implement a remote reset procedure whereby the modem receives a downlink packet from a satellite, triggering the system to restart the modem and/or the microcontroller. This would be controlled by using a few bits as a status code in each datum or each packet.

Overall, this thesis serves merely as a stepping stone into the deeper reaches of low-cost, low-power direct-to-satellite communications for IoRT. There is an abundance of areas left to touch upon such as physical enclosures and weatherproofing, and also many areas that can be tweaked or optimized further yet. As independent satellite IoT services such as Swarm

and Astrocast mature from their current infancy, more research, more designs, and more commercial products are likely to surface.

Chapter 6

Conclusion

The research in this thesis began with a question: What is the optimal data transmission method for low-cost, low-power sensors in the Arctic? This inquiry led to the analysis of transmission options, the selection of LEO satellite IoT services, specifically Swarm Technologies, and the examination of Swarm satellites and modem requirements. A custom PCB was designed to integrate the modem with a microcontroller, adhering to the modem's strict design requirements.

Successful data transmission proved to be stochastic, highlighting the need for an online learning algorithm to identify "good" satellite passes and avoid energy waste. A custom algorithm, inspired by reinforcement learning, was developed for direct-to-satellite communications. Simulated results demonstrated the algorithm's learning capabilities and potential for significant power savings. Future work includes refining simulations,

improving the algorithm, and conducting large-scale, long-term field testing.

In conclusion, this thesis addresses the challenge of transmitting data from remote sensors in the Arctic and offers an integrative approach to designing a comprehensive system. As LEO satellite IoT technology matures, further research and products are expected to emerge, simplifying the design process for similar systems.

Bibliography

[1] Swarm Technologies, *Swarm M138 Modem Product Manual*, 2022. Revision 1.41.

[2] D. J. Purnell, N. Gomez, W. Minarik, D. Porter, and G. Langston, "Precise water level measurements using low-cost GNSS antenna arrays," *Earth Surface Dynamics*, vol. 9, no. 3, pp. 673–685, 2021.

[3] M. E. Tamisiea, C. W. Hughes, S. D. P. Williams, and R. M. Bingley, "Sea level: measuring the bounding surfaces of the ocean," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 372, 2014.

[4] X. Fang, "Improving data quality for low-cost environmental sensors." August 2018.

[5] M. D. Sanctis, E. Cianca, G. Araniti, I. Bisio, and R. Prasad, "Satellite Communications Supporting Internet of Remote Things," *IEEE Internet of Things Journal*, vol. 3, pp. 113–123, 2016.

[6] D. Palma and R. Birkeland, "Enabling the Internet of Arctic Things With Freely-Drifting Small-Satellite Swarms," *IEEE Access*, vol. 6, pp. 71435–71443, 2018.

[7] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A Survey on Technologies, Standards and Open Challenges in Satellite IoT," *IEEE Communications Surveys & Tutorials*, vol. 23, pp. 1693–1720, 2021.

[8] Z. Jia, M. Sheng, J. Li, D. T. Niyato, and Z. Han, "LEO-Satellite-Assisted UAV: Joint Trajectory and Data Collection for Internet of Remote Things in 6G Aerial Access Networks," *IEEE Internet of Things Journal*, vol. 8, pp. 9814–9826, 2021.

[9] J. A. Fraire, S. C. Umaña, and N. Accettura, "Direct-To-Satellite IoT - A Survey of the State of the Art and Future Research Perspectives - Backhauling the IoT Through LEO Satellites," in *ADHOC-NOW*, 2019.

[10] H. Huang, S. Guo, W. Liang, and K. Wang, "Online Green Data Gathering from Geo-Distributed IoT Networks via LEO Satellites," *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2018.

[11] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online Learning: A Comprehensive Survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[12] Y. Qi, L. Pan, and S. Liu, "A Lyapunov optimization-based online scheduling algorithm for service provisioning in cloud computing," *Future Generation Computer Systems*, vol. 134, pp. 40–52, 2022.

[13] M. Lauridsen, I. Z. Kovács, P. E. Mogensen, M. Sørensen, and S. Holst, "Coverage and Capacity Analysis of LTE-M and NB-IoT in a Rural Area," *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, 2016.

[14] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "IoT Connectivity Technologies and Applications: A Survey," *IEEE Access*, vol. 8, pp. 67646–67673, 2020.

[15] A. Sorensen, H. Wang, M. J. Remy, N. Kjettrup, R. B. Sørensen, J. J. Nielsen, P. Popovski, and G. C. Madueño, "Modeling and Experimental Validation for Battery Lifetime Estimation in NB-IoT and LTE-M," *IEEE Internet of Things Journal*, vol. 9, pp. 9804–9819, 2022.

[16] A. F. Khalifeh, K. Aldahdouh, K. A. Darabkh, and W. T. Al-Sit, "A Survey of 5G Emerging Wireless Technologies Featuring LoRaWAN, Sigfox, NB-IoT and LTE-M," *2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, pp. 561–566, 2019.

[17] G. Dahl and E. Eskång, "Researching possibilities for autonomous operation of a Sigfox radio base station north of the Arctic circle," 2018.

[18] A. Lavric, A.-I. Petrariu, and V. Popa, "Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions," *IEEE Access*, vol. 7, pp. 35816–35825, 2019.

[19] C. Gomez, J. C. Veras, R. V. Ferré, L. Casals, and J. P. Aspas, "A Sigfox Energy Consumption Model," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[20] S. Y. Mane, "LPWANs – Overview, Market Scenario and Performance Analysis of Lora, Sigfox Using NB-Fi Range Calculator," *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pp. 1–4, 2021.

[21] I. Lunden, "Sigfox, the French IOT startup that had raised more than $300M, files for bankruptcy protection as it seeks a buyer," Jan 2022.

[22] S. Wielandt and B. Dafflon, "Minimizing Power Consumption in Networks of Environmental Sensor Arrays using TDD LoRa and Delta Encoding," *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pp. 318–323, 2021.

[23] C. Gomez, S. M. Darroudi, H. Naranjo, and J. Paradells, "On the Energy Performance of Iridium Satellite IoT Technology," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[24] L. David and A. Zaman, "Simulating Iridium Satellite Coverage for CubeSats in Low Earth Orbit," 2018.

[25] J. Foust, "SpaceX to acquire Swarm Technologies," Aug 2021.

[26] D. Silver, "Lectures on Reinforcement Learning." URL: `https://www.davidsilver.uk/teaching/`, 2015.

[27] R. Sutton and A. Barto, *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series, MIT Press, 2018.

[28] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.

[29] H. Oe, M. Matsushita, and K. Inoue, "A practical approach to the year 2038 problem for 32-bit embedded systems," in *Proc. AsiaBSDCon 2020*, 2020.

[30] R. Marcinkevics and J. E. Vogt, "Interpretability and Explainability: A Machine Learning Zoo Mini-tour," *ArXiv*, vol. abs/2012.01805, 2020.

[31] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, vol. 4. Prentice Hall New Jersey, 1995.

[32] Y. Wu, B. Zhang, J. Lu, and K. Du, "Fuzzy Logic and Neuro-fuzzy Systems: A Systematic Introduction," *International Journal of Artificial Intelligence and Expert Systems*, vol. 2, no. 2, pp. 47–80, 2011.

[33] J. Lu, C. Fang, M. Xu, J. Lin, and Z. Wang, "Evaluations on Deep Neural Networks Training Using Posit Number System," *IEEE Transactions on Computers*, vol. 70, pp. 174–187, 2021.

Appendix A

Simulation Code Availability

The source code for simulating the online learning direct-to-satellite packet scheduling algorithm and calculating average power are available here: https://github.com/garrettkinman/Self-Learning-Satellite-Pass-Selection.

Appendix B

Software Libraries Used

Several open-source software libraries were used in the implementation and simulations for this thesis:

- SparkFun Swarm Satellite Arduino Library:

  https://github.com/sparkfun/SparkFun_Swarm_Satellite_Arduino_Library

- SparkFun SGP4 Arduino Library:

  https://github.com/sparkfun/SparkFun_SGP4_Arduino_Library

- NNlib.jl: https://github.com/FluxML/NNlib.jl

- Plots.jl: https://github.com/JuliaPlots/Plots.jl

- ProgressLogging.jl: https://github.com/JuliaLogging/ProgressLogging.jl

- StatsBase.jl: https://github.com/JuliaStats/StatsBase.jl

- Makie.jl: https://github.com/MakieOrg/Makie.jl