12-2016

An Analysis of Bluetooth Low Energy in the Context of Intermittently Powered Devices

Steven Lain Hearndon
Clemson University

An Analysis Of Bluetooth Low Energy In The Context Of Intermittently Powered Devices

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master's of Science
Computer Science

---

by
Steven Lain Hearndon
December 2016

Accepted by:
Dr. Jacob Sorber, Committee Chair
Dr. Brian Malloy
Dr. Brian Dean

Abstract

Bluetooth Low Energy (BLE) has become a prominent low-power wireless solution for portable, battery-powered devices, potentially allowing them to run for several years, even off of a simple coin cell. But batteries must still be replaced. The emergence of batteryless devices is gaining momentum due to their ability to run for decades with no maintenance. The design of BLE relies on exact timing, which usually means a constant power source. The question, then, is how well will BLE function when used in batteryless sensors that run on harvested energy and therefore lose power frequently.

In this paper, I evaluate the suitability of using BLE in the context of these Intermittently Powered Devices by analyzing the energy requirements of the three main BLE events: an advertisement, the connection establishment, and the periodic connection event. I then apply the results in an evaluation of BLE on a Peripheral powered by harvested solar energy and compare and contrast connectionless broadcasting and connection-oriented operation. The results show that batteryless BLE devices are not limited to connectionless operation as convention suggests, and that connected devices have the potential for better performance overall. Based on these findings, I describe a modified BLE protocol that would allow for sustainable connection-oriented operation to make it a more effective wireless standard for IPDs.

Dedication

To my wife, Erin, for your tremendous support during the pursuit of my master's degree, especially during my work on this thesis. I couldn't have done it without your encouragement. I love you so much, and I'm glad that I'll have more time for you soon.

To my kids, Keegan, Gavin, and Eastyn, for your patience, having to do without Daddy so many times while I studied and worked on projects. I can start making it up to you now.

To my parents, Marty and Suzanne, for all of your support and, especially, for providing a home for us while I've been in school. I'll never forget it.

To God and my Savior, Jesus Christ, for calling and equipping me, and for Your peace and Your provision, time and time again. My life is Yours.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Bluetooth Special Interest Group states that Bluetooth Low Energy, or BLE, "was built for the Internet of Things[2]." Their design assumes these "Things" will have access to continuous power from batteries, which means billions of devices powered by billions of batteries with lifetimes of less than five years. This does not make for a sustainable system. Energy harvesting technologies allow for devices to be deployed for decades without maintenance. Is BLE a suitable wireless standard for these devices? Or could BLE serve as the foundation for a better protocol, one that truly meets the needs of low energy devices?

The vision of the Internet of Things is a world where everything is connected, providing us with data about our surroundings at any given time. Essentially, such embedded computing comes down to systems that collect data and systems that act on that data. Most of these collectors have the simple task of taking readings from one or more electronic sensors and reporting their values to other devices in the network. As such, these sensors need wireless communication and the protocols to govern it. BLE's prominence makes it the de facto choice for this role.

These sensors also need power, and the constraints of batteries are creating an

emerging interest in batteryless devices[5] that run on free energy from the environment by harvesting, for instance, solar, kinetic, or thermal energy, and storing it in capacitors. They have a much longer lifespan than batteries, require no maintenance, and don't have the same environmental concerns. But their inconsistent power source means these devices die frequently, which is why they are often referred to as Intermittently Powered Devices, or IPDs. This presents new challenges to hardware and software designers, but overcoming these challenges leads to many benefits.

Imagine sensors embedded in the concrete of a bridge that harvest kinetic energy from the bridge's movement, gathering and reporting data on the structural integrity of the bridge as well as usage statistics. Imagine moisture and sunlight sensors spread across a large farm that can provide farmers with exactly what their crops are experiencing down to small regions, and even drive intelligent watering systems. Imagine health sensors with a form factor similar to a sticker or bandage that can be applied to patients to collect data while they're in the hospital. These sensors would be much less bulky than the wired systems currently used in hospitals, and when the patient checks out, these stickers could simply be thrown away.

All of these scenarios require wireless communication, and while BLE may be a strong contender, it has its limitations. Would the 2.4 GHz signal even work through concrete? Is the limited range of BLE impractical for the farm network? Are the timing requirements of BLE too strict for IPDs that frequently lose power and, as such, lose track of time? And are the energy requirements for BLE too great for devices that run on harvested energy?

BLE can work on energy harvesting devices, and there are a few sites online such as [4] and [5] that give instructions on how to do so. The common convention among them is to set the device up as a "beacon" that simply broadcasts data to any other device in range due to the extra energy it takes to establish a connection.

While broadcasting may be a good choice in some contexts, such as publicly available data, is it the only viable option?

In this thesis I analyze and compare the power requirements of connectionless broadcasting versus connection-oriented communication to determine the advantages and disadvantages of both methods in the context of IPDs. I begin in Chapter 2 by providing some background for the BLE protocol and IPDs and give the motivation for using them together. In Chapter 3, I analyze the energy requirements for the different BLE events and show how these events affect the operational energy costs of the two methods. Chapter 4 extends this analysis by comparing the two methods on a BLE Peripheral running on harvested solar energy. In Chapter 5, I discuss the implications of these findings and how they should influence batteryless BLE use within the current constraints of the protocol, and then I suggest some ways that BLE could be better adapted to IPDs by loosening some of those constraints. Chapter 6 lays out some of the related work in this area. And finally I summarize the conclusions in Chapter 7.

Chapter 2

Background and Motivation

In this section I describe the relevant details of the Bluetooth Low Energy protocol stack followed by a high-level overview of intermittently powered devices and the motivation behind them. I then describe how the constraints of these systems could affect their use of BLE.

2.1 Bluetooth Low Energy

Introduced in 2010, the purpose of Bluetooth Low Energy (BLE, sometimes marketed as Bluetooth *Smart*), was "to design a radio standard with the lowest possible power consumption, specifically optimized for low cost, low bandwidth, low power, and low complexity[23]." While BLE is part of the Bluetooth protocol as of Bluetooth 4.0, the two technologies have little in common, other than their frequency band.

BLE has grown rapidly compared to other wireless standards. In 2014, BLE accounted for 85% of the wireless radios used in commercial devices[3], with major players such as Apple and Samsung promoting its success. It has a leaner protocol

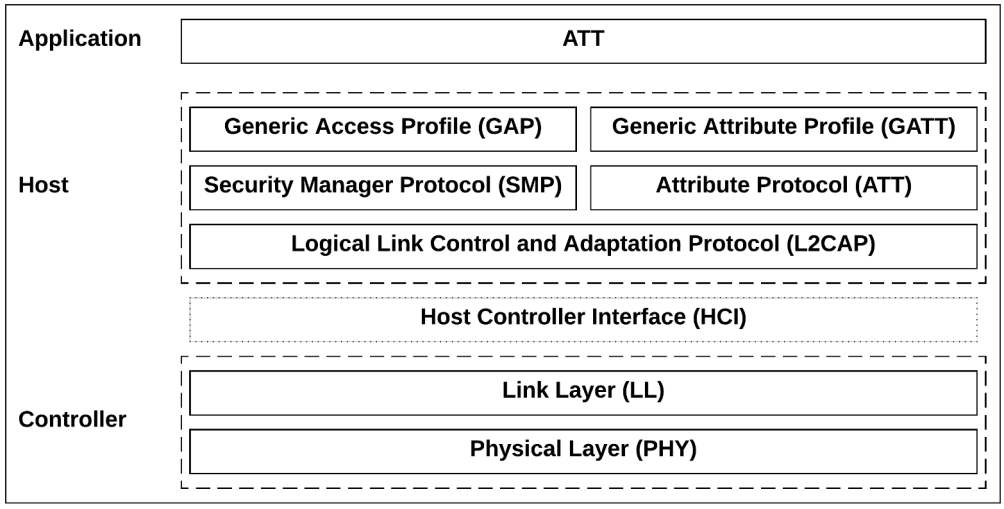


Figure 2.1: The BLE Protocol Stack

stack than classic Bluetooth, there are no licensing costs, and there are no fees to access the spec. As such, it is becoming the de facto standard for wireless communication among small-scale devices.

2.1.1 The BLE Protocol Stack

The lightweight BLE protocol stack is the main contributor to its energy efficiency, and one of the keys to its success. While it has the same number of layers as classic Bluetooth, their implementation is much simpler. The protocol stack is divided into 3 parts: the Controller, the Host, and the Application. What follows is a brief description of the lower 2 parts and the layers they contain. Much of the details come from [23]. Refer to figure 2.1 for a visual depiction of the protocol stack.

The Controller: This group includes the lowest layers in the protocol stack, and handles the actual exchange of data between BLE devices. The functionality of the Controller, and particularly of the Link Layer, is the most relevant to the scope of

this paper.

The Physical Layer handles the actual radio communications in the 2.4 GHz ISM (Industrial, Scientific, and Medical) band. It divides this band into 40 channels, 37 of which are used for connection data while the other 3 are used for advertising. BLE uses a very simple frequency hopping technique to minimize the effects of radio interference potentially present in that band, which could include WiFi and classic Bluetooth.

The hops to a new channel on each connection event use the following formula:

$$\text{new\_channel} = (\text{current\_channel} + \text{hop\_increment}) \mod 37$$

The hop increment is set by the master to a random value between 5 and 16. This guarantees that each channel will be used exactly once every cycle since 37 is a prime number.

The Link Layer interacts directly with the physical layer and is the only hard real-time constrained layer in the stack, since it handles all of the strict timing requirements of the protocol. As such, it is usually implemented as a blend of custom hardware and software and separated from the other layers by a standard interface. As far as I know, each manufacturer's implementation of the Link Layer is a unique, closed system, and no open-source implementation currently exists.

Several functions of the Link Layer are typically implemented in hardware, such as random number generation and AES encryption. The software side manages such things as the roles and the link state of the radio. A device can operate in one of these roles:

- **Advertiser** - simply broadcasts advertising packets
- **Scanner** - regularly listens for advertising packets

- **Master** - initiates a connection and then manages it
- **Slave** - accepts a connection request and then follows the master's timing

Other than implementation constraints, there is nothing in the protocol to prevent a device from taking any of these roles, and even more than one at a time.

A slave can send out 4 types of advertising packets, defined by these 3 properties: *Connectable*, *Scannable*, and *Directed*. **Connectable** means that a scanner can initiate a connection after it receives the advertising packet. **Scannable** means a scanner can send a scan request after receiving a packet. And **Directed** means it is targeting a specific scanner. A directed advertising packet cannot carry a payload (user data), while an undirected can. Directed packets, therefore, can only invite a connection, and so they are always connectable. The following table shows the 4 types of advertising packets based on these three types.

For the scanner role, the BLE specification defines two types of procedures: *passive scanning* and *active scanning*. **Passive scanning** means the scanner is simply listening to broadcasted advertising packets, without the advertiser ever knowing whether the packets were received or not. **Active scanning**, on the other hand, allows the scanner to transmit a Scan Request packet after receiving an advertising packet. This Scan Request will, in turn, trigger the advertiser to respond with the aptly named Scan Response, which allows for additional data to be sent by the ad-

Advertising Packet Type	Connectable	Scannable	Directed	Generic Access Profile Name
ADV_IND	Yes	No	Yes	Connectable Undirected Advertising
ADV_DIRECT_IND	Yes	No	Yes	Connectable Directed Advertising
ADV_NONCONN_IND	No	No	No	Non-connectable Undirected Advertising
ADV_SCAN_IND	No	Yes	No	Scannable Undirected Advertising

Table 2.1: Advertising Packet Types

vertiser. This method, however, ***does not*** allow the scanner to send any data to the advertiser, as the specification dictates a fixed payload for the Scan Request packet.

The Host Controller Interface: The Bluetooth specification allows for the physical separation of the Host and Controller on different chips. This often makes sense for larger devices to have a dedicated processor for the Controller, due to its hard real-time requirements and access to the physical layer. The Host Controller Interface enables communication between the Host and Controller over a serial connection. It is defined as a set of commands and events that are shared between the 2 parts, along with several *transports* for actual communication (e.g. UART, USB, SDIO, etc.). If both parts are on the same chip (known as a System on a Chip, or SoC), this piece is not used.

The Host: This group consists of the following layers:

The Logical Link Control and Adaptation Protocol serves as the multiplexer for the upper layers, encapsulating their packets into the standard BLE packet format. It also handles fragmentation and recombination, breaking large messages up into chunks that fit the maximum payload size for transmission and then recombining them on the receiving end.

The Attribute Protocol handles the client/server aspects of the BLE protocol. A client issues a request to a server to read or write an attribute, and the server responds with the requested value or permission to write.

The Security Manager, as its name implies, handles the security aspects of a connection (e.g. initial key exchange and encrypted transmission) and hides the public Bluetooth Address, if required, to avoid device tracking.

The Generic Attribute Profile builds on the Attribute Protocol by adding

a hierarchy and data abstraction that defines how data is organized and exchanged between applications. It basically provides a standardized structure for applications of a common type to exchange information. For instance, there are official profiles for heart rate and body temperature, making it easier for developers to create applications that work with off-the-shelf sensors.

The Generic Access Profile provides the framework for devices to discover other devices, broadcast data, establish secure connections, and take care of other low level operations in a consistent way. Specifically, this piece defines the *roles* a device can take in the network, the *modes* it can operate under within those roles, and the *procedures* available in each mode.

The four roles a device can adopt:

- The **Broadcaster** role corresponds to the Link Layer advertiser role, and is used when devices are simply sending out advertising packets periodically with data, such as a thermometer broadcasting the temperature. This data is freely available to any device listening, and devices in this role cannot *receive* any data.
- The **Observer** role corresponds to the Link Layer scanner role, and is used by applications to gather the data sent out by Broadcasters.
- The **Central** role corresponds to the Link Layer master role, and allows a device to initiate and establish connections with multiple peripherals. The BLE protocol is asymmetric in that the computing and power requirements for this role are larger than the Peripheral role, especially if it is managing multiple devices.
- The **Peripheral** role corresponds to the Link Layer slave role. This role is first responsible for advertising its presence until a Central connects with it, and then with maintaining the timing required for regular communication with the

Central. While it's processing and memory requirements are minimal, its strict timing needs *assume a constant source of power*.

The modes are the states that a device can switch to within a role to perform a particular procedure. The procedures are the sequences of actions the device can take in a mode to accomplish a task, such as broadcasting, observing, or establishing a connection.

2.1.2 Connectionless vs Connection-Oriented

A device acting as a Broadcaster communicates unidirectionally entirely through its advertising packets to whatever is listening. It does this by sending the same message three times in succession, once on each advertising channel. Between each transmission it also briefly receives on the same channel, listening either for a Scan Request from an Observer if it is sending scannable advertising packets, or for a connection request from a Central if it is connectable. Furthermore, if it receives a Scan Request at that time, it responds with a Scan Response, another transmission on that channel.

One advantage of the Broadcaster role is that the device is fully in control of the process. In particular, it can control how often it advertises, its transmission power, and on some devices, how many channels it advertises on. As for how often it advertises, according to the protocol it does this at a fixed rate defined by the ***Advertising Interval***, which ranges from 20 ms to 10.24 s. Shorter gaps between broadcasts increase the likelihood that a scanner will pick them up, since it is scanning at some fixed ***Scan Interval*** and receiving for the length of the ***Scan Window*** (which is less than or equal to the scan interval). Of course, since the device can start and stop advertising at any time, technically it can broadcast at any interval it

chooses, but a Scanner would need to be scanning frequently to ensure it receives the broadcast.

A device acting as a Peripheral must still start with advertising, just like a Broadcaster, but in this case it operates in discoverable mode. Once a Central detects it, the Central can initiate a connection. This event involves several exchanges to establish the connection, after which there is communication at fixed intervals *determined by the Central* in its ***Connection Interval***, which can range from 7.5 ms to 4 s. At each interval there is a connection event *initiated by the Central* in which it sends a message to the Peripheral, which returns a response. This may trigger several messages back and forth depending on if there is more information to exchange. Once the exchange is done, the Peripheral can sleep until the next connection event.

The connection parameters also include the ***Slave Latency*** and ***Connection Supervision Timeout***. The first is an integer value from 0 to 499 that defines how many times the Peripheral may skip a connection event before the Central disconnects. While this may seem promising for batteryless devices, it is limited by the second value, which is the maximum amount of time between two received packets before the Central considers the connection lost. This value is a multiple of 10 ms and ranges from 100 ms to 32 s. According to the Bluetooth specification, this value should be larger than $(1 + SlaveLatency) * ConnectionInterval$. Yet the Slave Latency is required to be less than or equal to $((SupervisionTimeout/ConnectionInterval) - 1)$. These rules are in conflict with one another based on the time ranges they must also follow. For example, if the Connection Interval were set to its maximum value of 4 s, and the Slave Latency was set to its maximum value of 499, the Supervision Timeout, according to the formula, would be 2,000 seconds, which is over half an hour. Unfortunately, the time constraint defined by the Supervision Timeout takes

precedence.

2.2 Intermittently Powered Devices

Computational devices continue to get smaller and smaller, enabling sensors to be embedded in a variety of everyday "things". The common assumption remains, though, that these devices will have access to continuous power, meaning these devices are usually equipped with batteries.

Intermittently Powered Devices (IPDs), as their name implies, are devices that do not have access to a continuous source of power. Instead, they run entirely on harvested energy from sources in their environment, such as solar, kinetic, or thermal. While these devices may store some of the energy in capacitors, it is intentionally limited to keep the physical size of the device small. As such, these devices frequently lose power altogether, causing code execution to be halted prematurely. They then have to restart from the beginning, often losing all stored variables in the process.

So why not just use batteries? For one, all batteries have a limited lifespan, regardless of their usage, typically on the order of 3 to 5 years. One recent article predicts that 6.4 billion devices will be connected in 2016[6]. Who is going to change all of those batteries when they die? The need for a device to manage recharging and signal the need for a replacement battery also adds cost and complexity. Plus, the disposal of all of those batteries creates a problem for the environment.

Devices that run on harvested energy, on the other hand, could last for decades without any maintenance. And the nature of the components on these devices make them easier to recycle and cheaper to produce. But these benefits come with tradeoffs.

- Programmers must be extremely conscious of energy usage in their designs.
- Short, large bursts of needed energy are problematic and hard to plan for.

- Devices die a lot, so programs often fail to complete or even make progress.
- Because they die a lot, it is very difficult to keep track of time, something that is very important in network communication.

Some research has been done in this area to suggest ways that these devices could overcome each of these drawbacks[14, 20, 15]. But to our knowledge, no one has addressed the issues of using BLE with IPDs.

2.3 BLE and IPDs

Both transmission and reception are "expensive" events in terms of energy. BLE achieves its low energy consumption primarily by managing small data exchanges at fixed intervals, thus allowing the radio to sleep between these connection events. But these strict timing events pose a problem for IPDs that have no guarantee of continual operation.

Every connection event requires a short, large burst of energy for the radio communication, and any one of them could expend the small storage of energy before it is replenished by the harvester. And if the device does die, it will lose its connection and need to reestablish it when it regains operation. It is certainly conceivable that an IPD could get stuck in an infinite loop of starting up, advertising, connecting to a Central, and then dying before it can ever send any data.

With that in mind, it would seem that operating in the Broadcaster role is the obvious choice. And in fact, this is the recommended method for batteryless devices. But there are several tradeoffs to connectionless BLE:

- Broadcasting only utilizes 3 of the 40 channels, which increases the risk of interference, especially if there are multiple peripherals broadcasting in the same space, as they all must use the same 3 channels.

- The default operation is to broadcast on all 3 channels in series, which requires 3 transmission and reception events. It is possible, according to the Bluetooth specification, to broadcast on just 1 or 2 channels, but this increases the risk of not being received by a nearby scanner.
- The BLE protocol does not allow for bidirectional communication with this method, so it is impossible for the Central to pass any instructions to the Peripheral. This may not be always be necessary, but at the same time there could be several use cases where it would be beneficial for the Central to direct the Peripheral, such as having it change how it reports its values, or change its reporting interval.
- This method does not allow for secure broadcasts in and of itself, although since the broadcast data is user created, it could easily be encrypted data as long as the Observer can decrypt it. But this fails to take advantage of the security features of BLE that are "baked in". Is security in this context an issue? While it may not be important to secure transmissions of soil moisture levels, many would argue that body area network data such as heart rate is sensitive and should certainly be secured.

Connection-oriented BLE has many advantages over connectionless, such as secure, bidirectional communication. But perhaps the biggest advantage is that it has the potential for a lower amortized energy cost, depending on how long it can maintain a connection.

Chapter 3

BLE Energy Analysis

In order to better understand how these devices would behave in different scenarios, I first determined the energy cost of the various BLE events on the Peripheral. While the actual energy consumption of BLE is affected by several factors, including how much data needs to be exchanged, the vast majority of the energy usage comes from the radio transmissions and receptions characteristic of the three main BLE events: advertising, establishing a connection, and the subsequent periodic connection events.

3.1 Energy Profiling Setup

To evaluate the energy consumption of these events, I setup a test rig seen in Figure 3.1 that included a Teensy 3.2[10] microcontroller with a 72 MHz ARM Cortex processor, a Sparkfun Current Sensor breakout board[8], and a BLE Nano[1] from RedBearLab which features the Nordic nRF51822 SoC[7]. The power for the Nano was supplied by the Teensy, which was powered by a USB connection. The supply line passed through the current sensor, which uses an INA169 chip and a

shunt resistor to measure the current on the positive power rail. The current sensor was connected to an analog input pin on the Teensy, which used its analog-to-digital converter (ADC) to measure a variable voltage coming from the sensor and translate that value to current using Ohm's law.

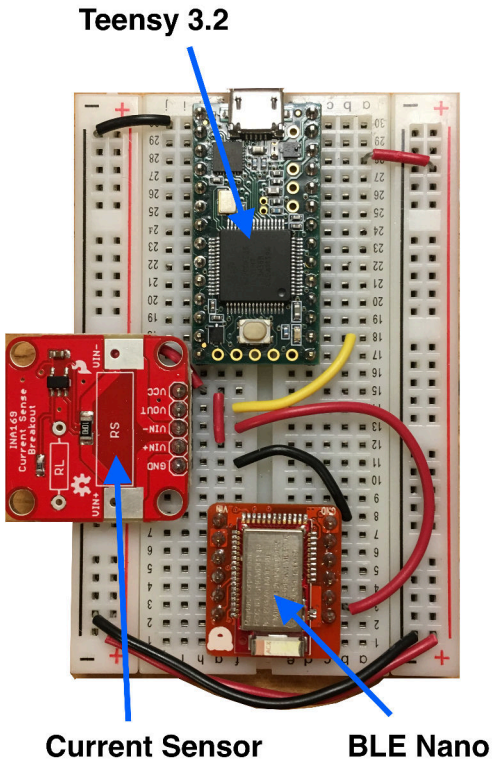


Figure 3.1: Current Testing Rig

The Nano was configured as a Peripheral to send connectable, undirected advertising packets every 500 ms, with the transmit power set to 0 dBm. Two of the pins on the Nano were connected to two of the pins on the Teensy, and the Nano was programmed to make one pin high when advertising and the other high when connected. This was so the Teensy could be aware of the three states (Advertising, Connected, and Idle) and mark the data accordingly to make finding the states in the data easier.

I first programmed the Nano to do nothing but sleep and recorded the current draw with the Teensy. The value was determined to be less than 0.6 mA, so I set the Teensy to only output values above 1 mA. This was helpful since the Teensy took current readings every 75 μs, most of which would be between the events I was interested in. Filtering out the periods of sleep made it much easier to process the data and partition it into observable events.

I then programmed the Nano to send connectable advertising packets. I used a program on my laptop to monitor the advertisement, initiate a connection, and then terminate the connection. The Teensy took current readings in Amps during these events and transmitted them over a serial connection to my computer along with the state (adv or conn) and the microseconds elapsed since the Teensy started up. This data was recorded in a CSV file for analysis.

3.2 Energy Profiling Results

I first analyzed this data by calculating the time difference between each row and the previous row. From this I was able to confirm that most readings happened approximately every 75 μs, and could also easily see the gaps in the data where I filtered out power consumed while sleeping. Partitioning the data by events made it easy to confirm that advertising packets were sent out approximately every 500 ms, as the Nano was programmed to do. It also showed the gap between connection events, which started out relatively small at around 11 ms since the Connection Interval was set by my laptop acting as the Central (to $\approx$ 15 ms) and was therefore out of my control. I did, however, program the Nano to *request* that the Connection Interval be set between 500 ms and 1 s. It is up to the manufacturers' implementations and the Central's logic to determine how that request is handled, if handled at all. From

the data I could see that after a brief time the Central complied with the smallest interval requested and began connecting every 500 ms after that.

I then used a tool to add a colored bar to the adjacent cell of each row with a length that represented the ratio of the current of that row to the maximum current recorded. This allowed me to see a basic representation of the energy consumed for each event. With that I was able to observe that all of the advertisements showed the same pattern, as did the connections with data and the connections without. A unique pattern was formed when the device established a connection, which was also clearly identifiable since the recorded state for each row transitioned from Advertising to Connected during this window. I extracted representative data for each event and graphed them, the results of which can be seen in Figures 3.2, 3.5, 3.3, and 3.4.

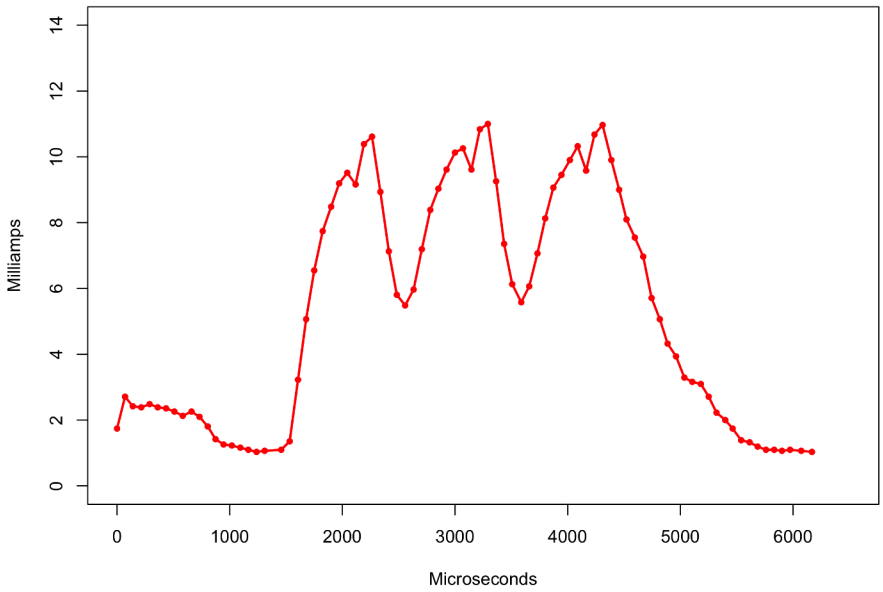


Figure 3.2: Advertising

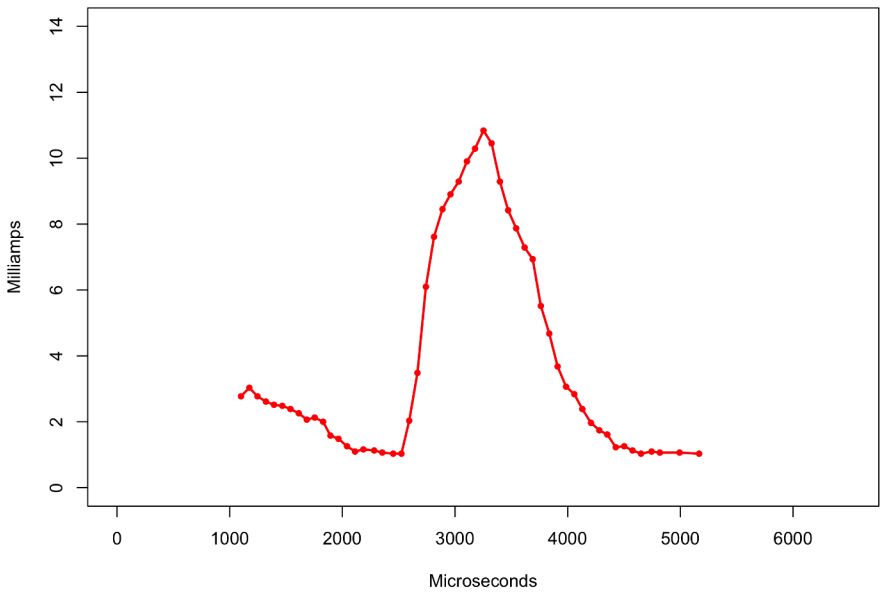


Figure 3.3: Connection - No Update

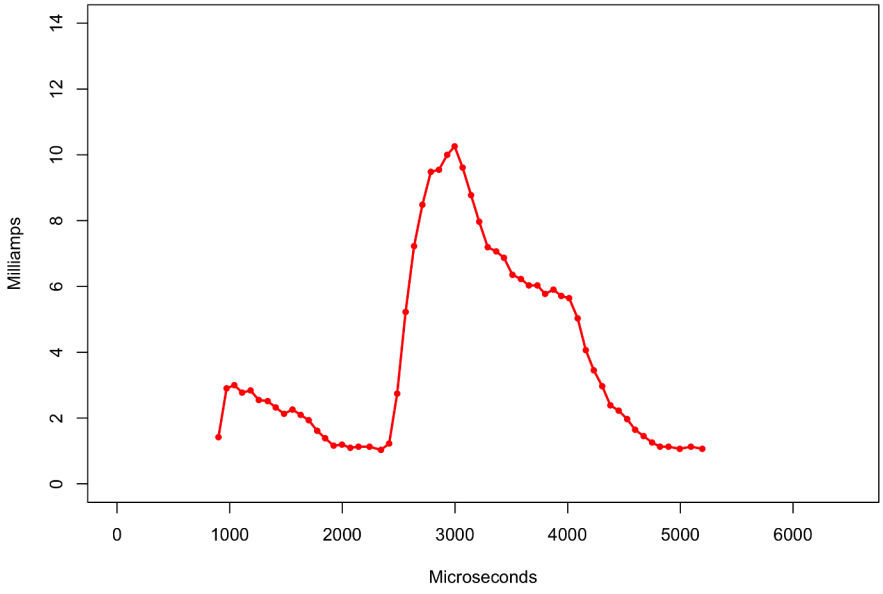


Figure 3.4: Connection - With Update

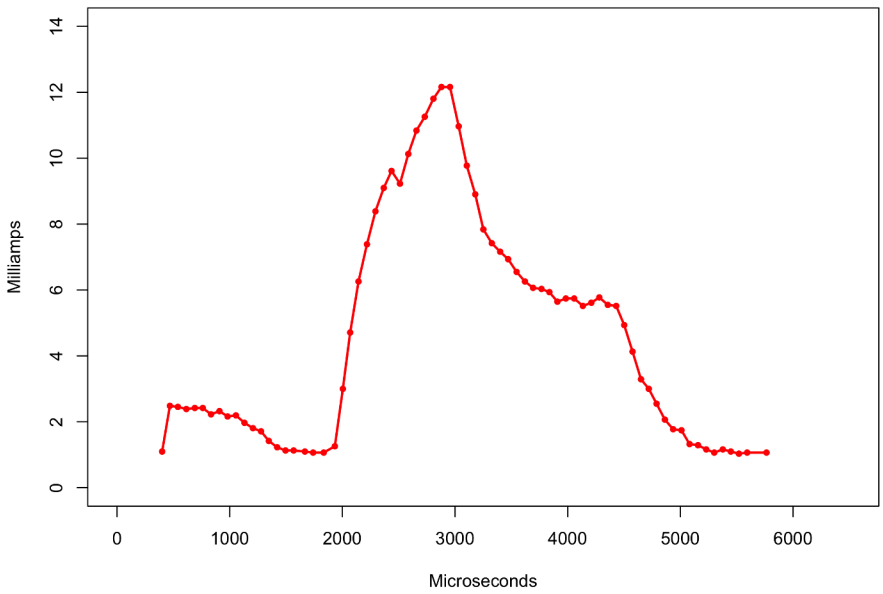


Figure 3.5: Establishing Connection

In Figure 3.2 we can see that Advertising is a costly operation due to the fact that the Peripheral both transmits and listens three times in a row, each on a different channel. This can be seen in the three peaks in the graph. The protocol makes it possible to advertise on only one or two channels (although not all manufacturers implement this feature), but that decreases the chance that the packet will be received by a scanner, which changes to a different advertising channel every Scan Interval. The developer could conceivably have the Peripheral try a different channel at every advertisement, but this is not as efficient since the energy cost of transmitting on three channels is still less than twice the cost of transmitting on only one[17]. Instead it might be better to simply increase the Advertising Interval.

Figures 3.3 and 3.4 show the energy consumption of a single connection event, initiated by the Central, which happens every Connection Interval. I intentionally programmed the Nano to update a sequence number every second, and the Central

subscribes to updates on that value. Yet, based on the connection parameters I programmed the Peripheral to request, the Central still connects with the Peripheral every 500 ms. The difference between the two graphs, then, is that in the first the Central is "checking in" with the Peripheral, but since there is no new information, nothing gets exchanged. In the second, as soon as the Central checks in the Peripheral pushes a notification of the change along with the new value. This explains the extra energy consumption shown after the main peak.

The surprising discovery was that, according to Figure 3.5, the cost of establishing a connection appeared smaller than expected since it has been portrayed as the most expensive operation. To further investigate, I wrote a Python script to run a simple calculation of the approximate energy consumption in Joules for each operation. I did this by treating each measurement as though the current C shown was constant for the time interval T it represents, determined by the difference between its *elapsed time* and the one before it. Then, since $Joules = Voltage \times Current \times Time$, and I knew that the test rig supplied the Nano with 3.3 V, my Python script calculated the total Joules J for the event as the sum of the Joules for all N records in the event.

$$J = \sum_{i=1}^{N} 3.3 \times C_i \times T_i$$

The results of these calculations are shown in Table 3.1.

The next step was to determine if the results from these individual samples were consistent with a larger data set. More extensive Python scripts were used to record the data from longer sessions with multiple connects and disconnects, automatically partition the data on every gap bigger than 2 ms, and calculate the energy statistics for the entire session. The results of one of these sessions is shown in Table 3.2

BLE Event	Total Time	Energy Used
Advertisement	6167 μs	107.232 μJ
Connection Establishment	5364 μs	80.750 μJ
Connection with No Update	4066 μs	48.819 μJ
Connection with Update	4294 μs	55.320 μJ

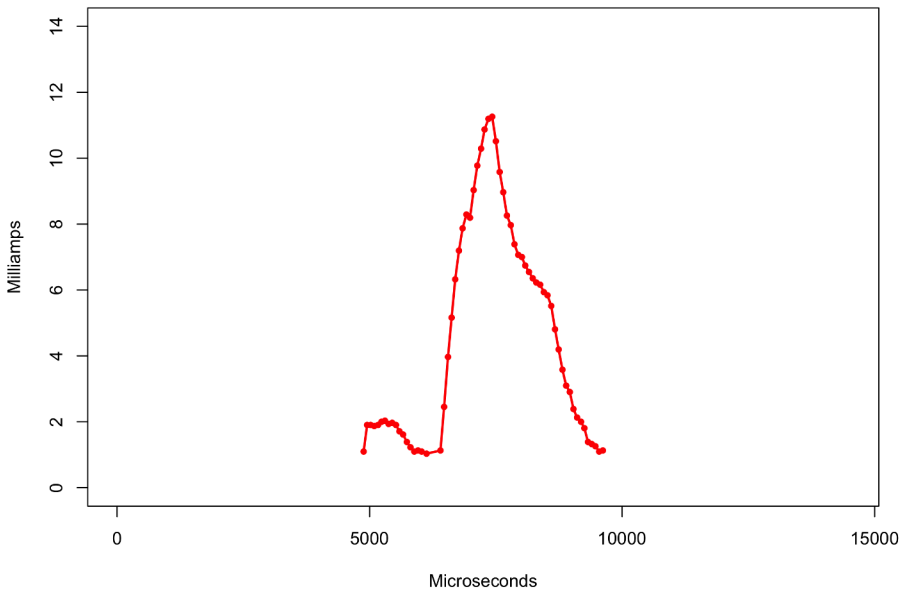
Table 3.1: Energy Cost for BLE Sample Events

From this table we can see that the results for advertisement and connection events are fairly consistent with the first samples selected. The cost to establish a connection, however, varies quite a bit, in that the minimum and maximum values are both almost two standard deviations from the mean. The graphs of the data from these two events are shown in Figure 3.6. Here we see that the maximum cost event shows that the device was on its third advertisement when it connected, and then had a connection event immediately after, while the minimum event shows that it connected on the first advertisement. An expanded view of this event in Figure 3.7 shows that it also had its first connection event within about 5 ms of establishing the connection.

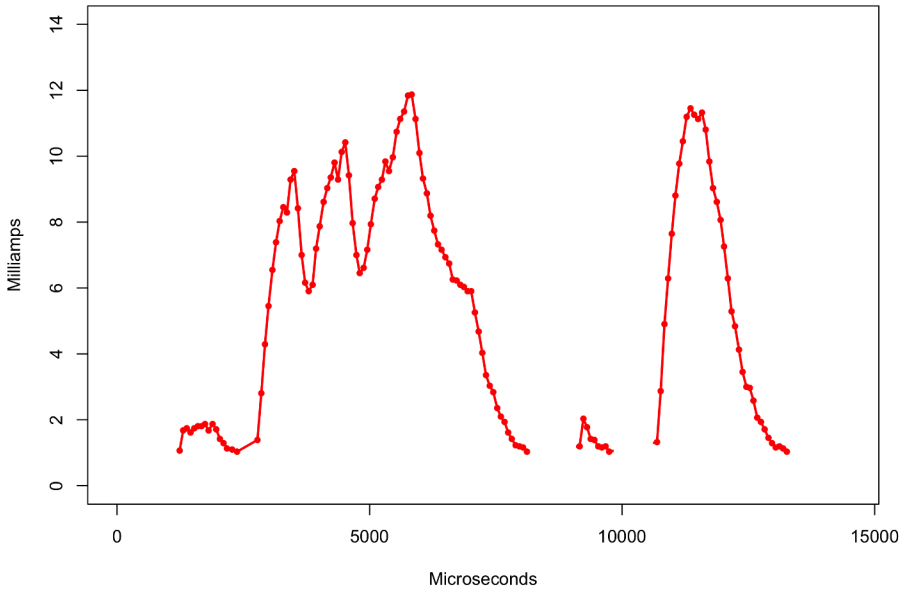
As I stated previously, the Connection Interval was outside of my control for these tests, and the Central (my laptop) demanded to make contact with the Periph-

BLE Event	Sample Count	Mean	Median	Min	Max	Var	StdDev
Advertisement	150	106.067	104.467	102.455	123.686	27.964	5.288
Connection Establishment	12	127.655	130.434	70.650	191.351	1269.373	35.628
Connection	328	41.662	40.194	32.789	93.377	43.222	6.574

Table 3.2: Energy Cost Statistics for BLE Session



(a) Minimum CE



(b) Maximum CE

Figure 3.6: The Minimum and Maximum Connection Establishment Events

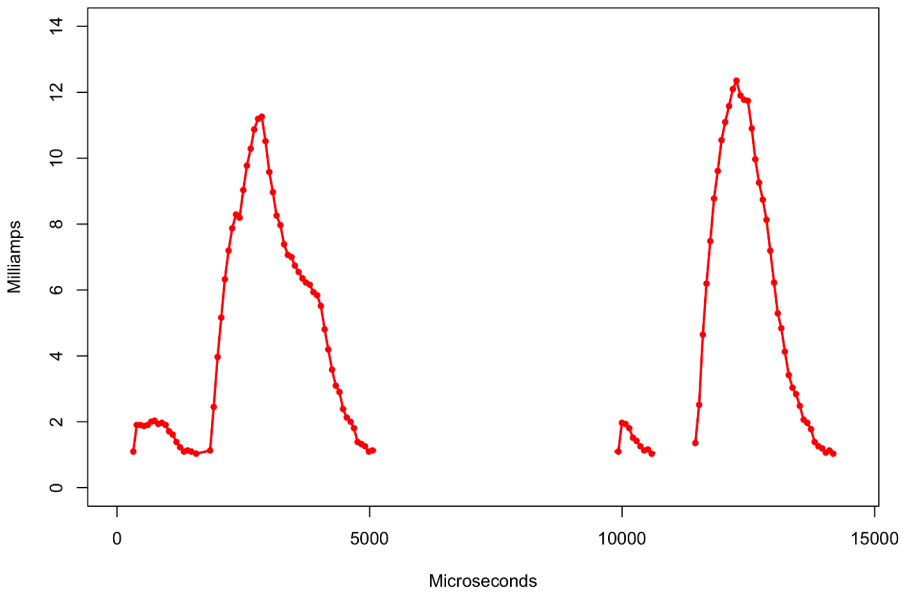


Figure 3.7: The Minimum CE and What Followed

eral every 15 ms for the first part of the connection, until it gave in to the Peripheral's request for less frequent contact and agreed to its ***minimum*** interval. This took approximately 330 ms in the first dataset, which shows it took 22 Connection Events before the Central made the change. It also appears that the Central's timing of this somehow starts before a connection is even established, which is why in some cases it makes that first connection within just a few milliseconds after establishing the connection.

This explains how establishing a connection may seem like an expensive endeavor, but I point out first that even in the worst case it is still less than twice the cost of the average advertisement, and that this connection cost always includes some portion of the cost of the last advertisement, which would have happened anyway. Furthermore, if I had more control of the Central, I could have set the Connection Interval to 500 ms to match the Peripheral's advertising schedule. I argue then that

the connection cost would be more in line with the average.

This indicates that establishing a connection is feasible in batteryless situations, requiring just a small amount of additional energy. Connection-oriented communication allows for bidirectional and secure communication that is more reliable. It could also benefit energy usage. This premise was the motivation for the experiments in the next chapter.

Chapter 4

Harvested Energy Experiments

A full analysis of Bluetooth Low Energy in the various contexts of Intermittently Powered Devices is too broad of a scope for this thesis due to the many possible scenarios and harvesting techniques. With that in mind I make certain assumptions for the purpose of this evaluation. First, I focus only on the Peripheral's performance while powered by harvested energy, allowing for a Central that is fully and continually powered, which is a very common scenario. Second, since a BLE radio has a minimum current required for even a single transmission, I assume a harvesting methodology and scenarios that accommodate that requirement. Third, because of its common use and readily available supplies, I focus exclusively on solar energy for the harvesting platform.

The energy profiles analyzed in the first experiments showed that the cost of establishing a connection was comparable to an advertisement, and that the ongoing cost of connection events is more efficient than broadcasting. With that in mind, the purpose of the following experiments is to compare and contrast how a BLE device running on solar energy performs in connectionless mode and connection-oriented mode.

4.1 Harvested Energy Setup

To conduct the solar experiments, I used another device from Red Bear Lab called the Duo that has both Bluetooth Low Energy and WiFi built in. I set up the Duo as the Central and programmed it to log all advertising and connected activity from the Nano, which I setup as the Peripheral. The Duo could identify the Nano and initiate a connection, or it could simply act as an Observer, depending on a flag that I could set remotely. This allowed me to compare connectionless operation against connection-oriented operation without changing the Nano.

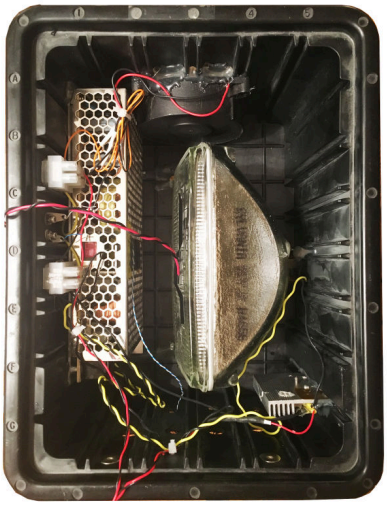


Figure 4.1: The Light Box

The Duo also controlled the light-box seen in Figure 4.1, a tool created in our lab to mimic solar energy. This box houses a vehicle headlamp and power supply that can be controlled by a microcontroller to produce light at different levels to power a solar panel. While certainly not a perfect analogue to solar radiation, it allowed for testing the two modes consistently under various conditions.

The solar power supply for the Nano was a simple setup that included a 22 mm $\times$ 35 mm solar panel, three 100 μF capacitors, and a Sparkfun Energy Harvester breakout board, which features the LTC3588 Piezoelectric Energy Harvester from Linear Technologies[9]. The energy harvester has a buck converter that allows a charge to build up in the three capacitors to around 4 V before discharging down to just over 2.5 V. This gives the device the opportunity to get some work done, rather than getting just enough power to boot, only to die again.

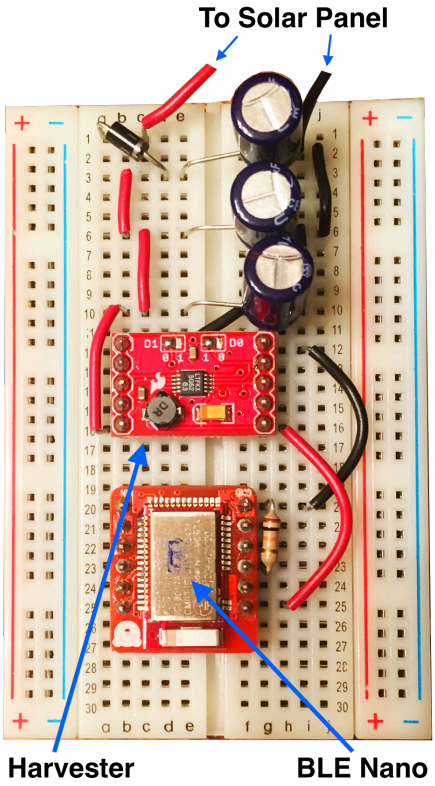


Figure 4.2: BLE Nano with Energy Harvesting Power Supply

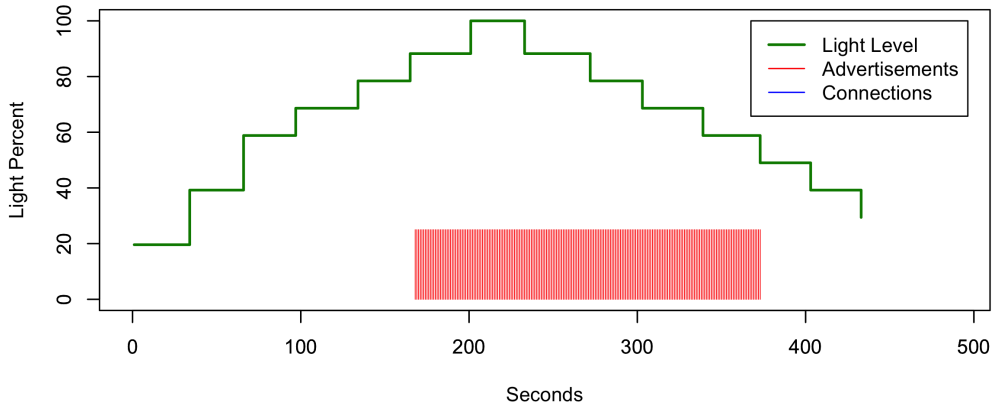
The experiments began with testing the Nano at different light levels, starting from nothing and gradually working the level up until the device powered on. The buck converter kept the device from powering up until there was sufficient energy to broadcast an advertisement, so the light level had to get to around 85% before the device would even respond. Once broadcasting, one of the first things observed was that the Central did not receive all of the advertisement packets from the Peripheral. This is common in most scenarios, but in this case the Peripheral was set to broadcast every 500 ms while the Central scanned continuously. While this could have been a hardware issue, decreasing the interval to 250 ms increased the number of packets that were received, though still not to 100%. This demonstrates the fact that there is no guarantee an Observer will receive a Broadcaster's packets, even in ideal conditions.

I wrote a Python script to send light level commands to the Central at set time intervals to create different solar "scenarios" while another Python script recorded the data from the Central. The two most revealing scenarios are shown in the results. All tests were done with the same Nano running the same code. The only difference between them was whether the Central was set to initiate a connection or not.

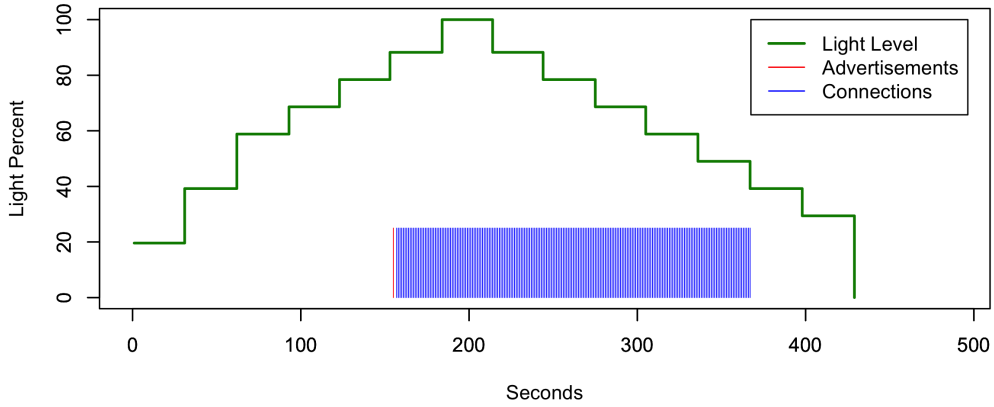
4.2 Harvested Energy Results

Based on the energy profile data, the expectation was that the connection-oriented mode would out perform the connectionless as the light level gradually got lower. At first, this wasn't the case, which didn't match the data from earlier. Further investigation showed that the Central was not following the Connection Interval specified, seemingly due to an issue with the library. Using the energy profiler developed earlier, I was able to analyze the data from connection events and calculate that the Central was requiring a Connection Interval of ***30 ms***. This highlights one of the issues that will be mentioned in the Discussion chapter. Since the Nano was only advertising at 250 ms, this was not a fair comparison. Since the Central's Connection Interval could not be changed, the Peripheral's Advertisement Interval was changed to match. The following graphs show the data gathered after that change.

Figure 4.3 shows a simple scenario where the light level is increased in stages and then decreased in matching stages. The level was to be held steady for 30 seconds for each stage, but timing issues allowed for some stages to be a few seconds longer. Even still, Figures 4.3a and 4.3b show that both connectionless and connection-oriented modes performed similarly under the same conditions. This alone defies the conventional wisdom that batteryless BLE devices should use connectionless broadcasting due to their energy constraints, but the following scenario makes



(a) Broadcast



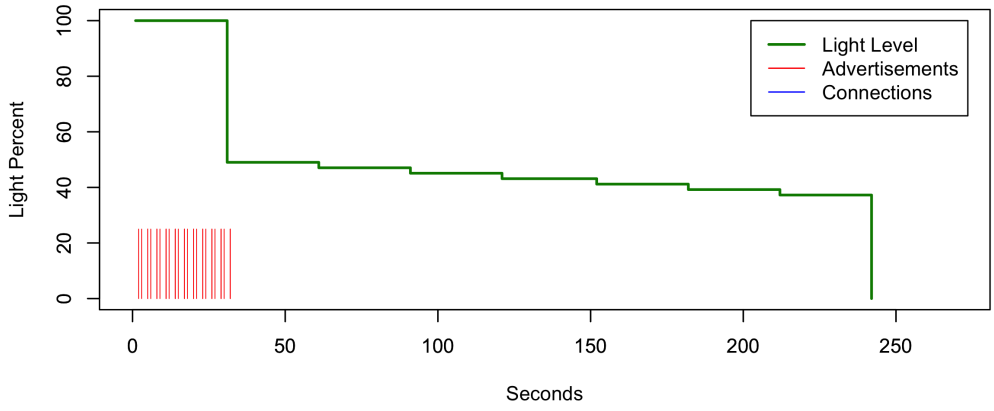
(b) Connected

Figure 4.3: Energy Harvesting Test 1 - Increasing Then Decreasing

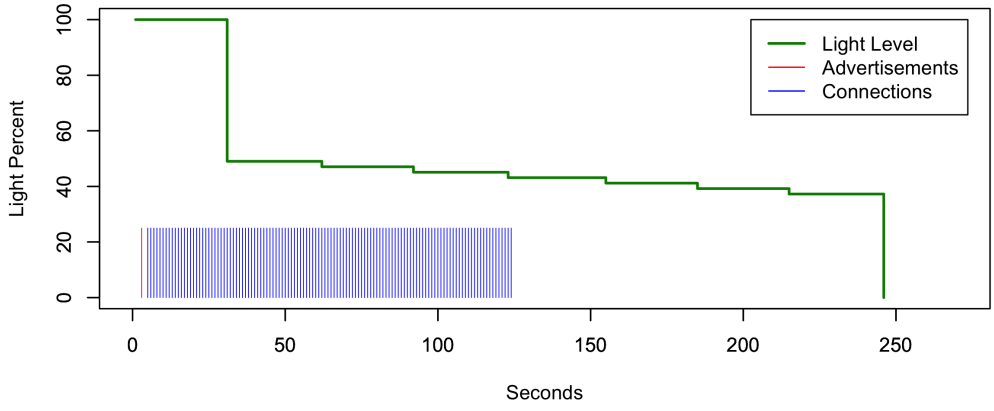
this even more clear.

Figure 4.4 shows the results of both modes when given plenty of light to get started, followed by more gradual steps down from a mid-level position, decreasing by 5% of the total every 30 seconds. As you can see from Figure 4.4a, the Broadcaster dies almost immediately after the light level is decreased to the first stage, at around 50%. The connected Nano in Figure 4.4b, on the other hand, continues for another

90 seconds as the level continues to decrease, even while connecting every 30 ms. This test clearly shows a scenario where a connected BLE device would out perform a broadcasting one, potentially "buying time" to allow for the possibility that the sunlight would increase again so it could continue to report, such as what happens on a cloudy day.



(a) Broadcast



(b) Connected

Figure 4.4: Energy Harvesting Test 2 - Energy Burst Then Trailing

Chapter 5

Discussion and Future Work

The main purpose of this paper is to explore the suitability of Bluetooth Low Energy in the context of Intermittently Powered Devices. As such, the experiments were conducted within the constraints of the current protocol. The results showed that the use of BLE on devices powered by harvested energy is viable in both connectionless and connection-oriented modes, and that the latter has several advantages over the former. And while the constraints of the BLE protocol and current implementations limit the options, there are still things that can be done within these constraints to make BLE more energy aware.

5.1 Following Protocol

As the results showed, there are benefits to establishing a connection between the Central and the Peripheral, both in terms of the effectiveness of the communication and the energy cost. Even still, there are scenarios where that is still not feasible, such as when energy is trickling in and there are little to no reserves. In those cases it's likely that once a device stored enough energy to send an advertisement, it would

die immediately afterward, leaving no time to make a connection. Or if a connection could be made, the added cost, however small, would be enough to shut down the device.

With this in mind I suggest a hybrid approach using several techniques. First, as long as the data does not need to be secured, send the data in both the advertisement packets and the connected packets. There is nothing in the protocol to prevent this. I even used this technique with my experiments. The data has to be handled differently in the two modes, though. Say, for instance, the developer would like the Central to receive updates from a Peripheral every second when possible. During advertising, the device should attempt to send several packets per second to have a better chance that one of them is received. The packets would be marked with a sequence number that gets incremented every second, and then if the Central gets a duplicate, it could ignore it. Once connected, updates are handled automatically by the protocol. This sequence number would require access to nonvolatile memory, such as FRAM, which is currently available to certain microcontrollers.

Second, give the Central the ability to ***adaptively*** connect with the Peripheral based on some simple factors. For instance, it could initiate a connection once it observes three sequential advertisements from the Peripheral, surmising that if it has had enough energy to continue those costly advertisements, it might be able to afford to connect and maintain a connection. The Peripheral could help with this by sending a sequence number in the advertising packet which naturally gets reset if the Peripheral dies. It could also send data concerning the rate at which it's harvesting energy, if it's capable of monitoring it, and any other data that could help the Central make a good decision to initiate a connection. This introduces the concept of a third mode: connectionless, connection-oriented, and ***connection-preferred***.

Also, recent technologies from the research on IPDs could help make a BLE

Peripheral more effective within the current protocol. One such technology is federated energy[14], which separates the harvested energy storage into smaller, isolated stores with capacitors. This approach allows it to make the device it's powering aware of the levels of energy it has, and even assign energy stores to particular tasks, like radio transmission. For a BLE Peripheral, this could be used to indicate whether it has enough energy to establish a connection and maintain it for enough connection events to make it worthwhile.

This last technique and others like it would work even better if the Connection Interval, and other connection parameters, could be set by the Peripheral rather than the Central, which leads to some suggestions for changes to the BLE protocol.

5.2 Future Work: Beyond the Protocol

While BLE is workable for IPDs, what would a modified version of BLE that better fits this context look like? What follows are some thought experiments on how BLE could be modified to better suit the unique needs of IPDs.

5.2.1 An Unlocked Link Layer

While the details of the Link Layer are clearly laid out in the BLE specification from the Bluetooth SIG, as of this writing all actual implementations of the Link Layer are closed systems. One of the reasons given for this is that the timing requirements for the Link Layer are so tight that most manufactures implement the Link Layer as a blend of hardware and software. It's also possible that the various implementations of the Link Layer are closely guarded secrets of their manufacturers, since BLE is a commercial product and such secrets give them a competitive advantage. Either way, most of the things that researchers might like to experiment with are locked down

and only exposed through incomplete and poorly supported libraries.

What the research community needs is a fully open-sourced BLE implementation with a fully hackable Link Layer. Such a thing is not beyond the scope of a research lab, as the BLE protocol is freely available and, unlike traditional Bluetooth, fairly simple to understand. This would require a special System on a Chip that could handle the Physical Layer and some of the tougher aspects of the Link Layer, like AES encryption. But all other aspects of the Link Layer would be fully exposed and changeable, such as expanding the time limits imposed on the Supervision Timeout (instead of 30 seconds, what about 30 *minutes*?).

This would also allow the use of a tool like CusTARD[15], a device that measures voltage decay on a capacitor used as a time keeper to infer how long a device has been unpowered. What if the Peripheral was able to record a "snapshot" of the current connection state and then recall it when the device resumes power? With CusTARD, then, it could make a good approximation of the Central's current time. Assuming the Slave Latency and Supervision Timeout were set high enough and the device had not been out of the connection too long, this information could help it "jump back in" to the connection with the Central.

All of these things and more could be possible if there was a fully open-source BLE platform for research.

5.2.2 A More Relational Model

The biggest drawback to the current BLE protocol in terms of use with IPDs is its traditional view of the relationship between devices. As mentioned in the Background chapter, a device operating in the GAP role of Central is also operating in the Link Layer role of Master, while a device with the GAP role of Peripheral is using the

Link Layer role of Slave. And in the BLE protocol, the Slave must obey its Master.

This dynamic may make sense in many current BLE deployments, such as when the Master is a user's smartphone and the Slave is a lowly heart rate sensor powered by a small but effective coin cell battery. In that case, it seems reasonable for the Central to declare that it needs updates every second for the data to be effective. In such a case the Peripheral also has the resources to obey.

But in scenarios involving IPDs, such requirements are too overbearing and unnecessary. For our farm example from the introduction, the Centrals in the field are taking readings from many, many Peripherals scattered throughout the field. As long as they're getting some data from most of them most of the time, they can provide useful data further up the chain. In cases like that, it makes much more sense to transfer more control to the Peripheral, especially if it's running entirely off of harvested energy.

So, for example, suppose the Peripheral is responsible for initiating a connection event rather than the Central. That puts a little more of the energy burden on the Central to scan for the Peripheral's contact and, in most cases, only respond with a quick acknowledgement of receipt. Furthermore, suppose the Peripheral could either specify it's connection parameters, or the two could define them collaboratively. These things would allow the Peripheral to miss many scheduled connection events in a row, for much longer than the current 30 second limit. And the Central, which could have a more robust power supply, could help the Peripheral by gradually increasing its scan window on either side of the scheduled connection time depending on how long they've been without contact.

It's clear that the current BLE protocol was developed for continually powered systems. IPDs, on the other hand, require a remodel of the protocol. Such a remodel could truly make BLE the standard of choice for all Internet of Things scenarios.

Chapter 6

Related Work

While no other work has focused on Bluetooth Low Energy in the context of Intermittently Powered Devices as of this writing, many of the pieces of this study are related to work done previously.

In [22], the authors create and evaluate a batteryless implantable glucose monitor that uses BLE to transmit the data collected, but they use RF as the energy source and supply power to the device wirelessly whenever readings are taken. Similarly, the authors of [11] design a platform to support implantable neural sensing devices in rats for neuroscience research which are also batteryless and use BLE for wireless communication. They also supply the devices with continuous power from RF power transmitted by coils nearby while testing the rats. This thesis evaluates BLE devices with unpredictable and unreliable power sources.

The authors of this Texas Instruments publication [16] describe specific methods for measuring the power consumption of BLE during its various events. The authors of [13] give an overview and evaluation of BLE by describing its functionality and analyzing power consumption to predict battery life based on the settings of the connection parameters, such as the Connection Interval. The authors of [17], on the

other hand, focus their study on the energy needed for device discovery, particularly on the energy impact of advertising on the Peripheral and the energy impact of scanning on the Central. While these papers show the energy models for connected and unconnected operation individually, none of them compare the two as this thesis does.

There are many papers that provide a general evaluation of BLE's performance and compare it to other platforms. This paper [18] evaluates its general energy requirements and data throughput. This paper [12] focuses on modeling the BLE discovery process using simulation, looking particularly at the probability of discovery under various settings and circumstances as well as the expected discovery latency. And in [21], the authors compare the energy needs of BLE to that of Zigbee, and also look at how interference on the same channel affects both, ending with suggestions for modifications to the BLE protocol that could help alleviate some of these issues. All of these papers assume continuous power.

The authors of [19] provide yet another analysis of BLE in terms of its capabilities and energy requirements. Additionally, though, at the end of their paper they have a section titled "Energy harvesting based BLE sensor systems", where they speculate on BLE's ability to run entirely on harvested energy given their calculations for the various BLE events. They conclude from calculations alone that current energy harvesters, based on their listed output range, are capable of supplying the energy needs of BLE. They say nothing, though, of the power variance from these harvesters and how that would affect the BLE devices they're powering. And they don't test BLE on an actual device powered by harvested energy, as this thesis does.

Finally, in [24], the authors discuss how the many different constrained devices that are to be connected to the Internet have no standard way of doing so. They go on to propose that as BLE becomes more and more prominent, it could provide a generic access model for these devices. Research like this recognizes BLE's potential

as a widespread technology, and how it could be leveraged to provide connectivity to even the smallest sensor. The work in this thesis supports this argument, provided the protocol could be adapted to better handle the constraints of such devices.

Chapter 7

Conclusion

In this thesis, I evaluated the energy requirements of BLE's three discrete events, an advertisement, a connection establishment, and a connected exchange. Based on these findings I also tested connectionless versus connection-oriented operation of BLE on an Intermittently Powered Device running exclusively on harvested solar energy by testing it under simulated solar conditions.

The results of these experiments demonstrate that the conventional approach of using the connectionless Broadcaster role for batteryless conditions is not always the best course of action. In fact, this research indicates that connection-oriented operation may be easily achievable if the Peripheral is able to broadcast, and that doing so allows for more consistent, sustainable, and versatile communication than broadcast mode. This leads to a better approach with a new mode, ***connection-preferred***. This mode would allow a Central to gather data from a Peripheral while it is broadcasting and only initiate a connection if it infers from past performance that it would likely succeed. Such an approach could benefit from the best of both worlds.

From there, I made suggestions for future work that included developing an

open-source BLE hardware and software platform with a fully hackable Link Layer and utilizing newly developed technologies for energy supply awareness and unpowered time tracking to make a BLE Peripheral more adaptive to the current protocol requirements. I also discussed how the BLE protocol could be adapted to better accommodate the unique needs of IPDs, such as loosening the timing requirements and giving more control to the Peripheral to set the parameters for the connection with the Central. These changes could make BLE an ideally suited wireless standard for the Internet of Things.

References

[1] BLE Nano. http://redbearlab.com/blenano/.

[2] Bluetooth Low Energy — Bluetooth Technology Website. https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy.

[3] Bluetooth Smart's Rise From Obscurity to Mainstream — EE Times. http://www.eetimes.com/author.asp?section_id=36&doc_id=1321690.

[4] Energy Harvesting and Bluetooth® Low Energy — DigiKey. http://www.digikey.com/en/articles/techzone/2015/nov/energy-harvesting-and-bluetooth-low-energy-designing-a-batteryless-beacon.

[5] Energy-harvesting devices replace batteries in IoT sensors — Core & Code.

[6] Gartner Says 6.4 Billion Connected. http://www.gartner.com/newsroom/id/3165317.

[7] nRF51822 / Bluetooth low energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR. https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822.

[8] SparkFun Current Sensor Breakout - INA169 - SEN-12040 - SparkFun Electronics. https://www.sparkfun.com/products/12040.

[9] SparkFun Energy Harvester Breakout - LTC3588 - BOB-09946 - SparkFun Electronics. https://www.sparkfun.com/products/9946.

[10] Teensy USB Development Board. https://www.pjrc.com/teensy/index.html.

[11] Chih-Wei Chang and Jin-Chern Chiou. A Wireless and Batteryless Microsystem with Implantable Grid Electrode/3-Dimensional Probe Array for ECoG and Extracellular Neural Recording in Rats. *Sensors*, 13(4):4624–4639, April 2013.

[12] Keuchul Cho, Woojin Park, Moonki Hong, Gisu Park, Wooseong Cho, Jihoon Seo, and Kijun Han. Analysis of Latency Performance of Bluetooth Low Energy (BLE) Networks. *Sensors*, 15(1):59–78, December 2014.

[13] Carles Gomez, Joaquim Oller, and Josep Paradells. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors*, 12(12):11734–11753, August 2012.

[14] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. Tragedy of the Coulombs: Federating Energy Storage for Tiny, Intermittently-Powered Sensors. pages 5–16. ACM Press, 2015.

[15] Josiah Hester, Nicole Tobias, Amir Rahmati, Lanny Sitanayah, Daniel Holcomb, Kevin Fu, Wayne P. Burleson, and Jacob Sorber. Persistent Clocks for Batteryless Sensing Devices. *ACM Transactions on Embedded Computing Systems*, 15(4):1–28, August 2016.

[16] Sandeep Kamath and Joakim Lindh. Measuring bluetooth low energy power consumption. *Texas instruments application note AN092, Dallas*, 2010.

[17] Jia Liu, Canfeng Chen, Yan Ma, and Ying Xu. Energy analysis of device discovery for bluetooth low energy. In *Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th*, pages 1–5. IEEE, 2013.

[18] E. Mackensen, M. Lai, and T. M. Wendt. Performance analysis of an Bluetooth Low Energy sensor system. In *2012 IEEE 1st International Symposium on Wireless Systems (IDAACS-SWS)*, pages 62–66, September 2012.

[19] Elke Mackensen, Matthias Lai, and Thomas M. Wendt. Bluetooth low energy (ble) based wireless sensors. In *Sensors, 2012 IEEE*, pages 1–4. IEEE, 2012.

[20] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: System support for long-running computation on RFID-scale devices. *Acm Sigplan Notices*, 47(4):159–170, 2012.

[21] Matti Siekkinen, Markus Hiienkari, Jukka K. Nurminen, and Johanna Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 232–237. IEEE, 2012.

[22] Sérgio Silva, Hugo Martins, António Valente, and Salviano Soares. A Bluetooth Approach to Diabetes Sensing on Ambient Assisted Living systems. *Procedia Computer Science*, 14:181–188, 2012.

[23] Kevin Townsend, Robert Davidson, Akiba, and Carles Cufí. *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. O'Reilly, Sebastopol, CA, 2014. OCLC: 870896083.

[24] Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta. The Internet of Things Has a Gateway Problem. pages 27–32. ACM Press, 2015.