Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2014

Project Title:	**A Wireless Low Energy Ambulatory Electroencephalogram**
Student:	**Thomas Alexander Morrison**
CID:	**00642176**
Course:	**EIE4**
Project Supervisor:	**James Mardell**
Second Marker:	**Dr. Pants Georgiou**

Abstract

Certain neurological medical disorders require continuous monitoring to fully understand and diagnose. Examples of medical interest include epilepsy, syncope, multiple sclerosis, migraines, strokes, Parkinson's and Alzheimer's disease. Electroencephalography (EEG) is the recording of electrical activity along the scale, resulting from ionic current flows within the neurons of the brain and is useful for both diagnostic and monitoring such aforementioned conditions. Monitoring brain activity can help physicians understand certain characteristics, triggers, and the severity of the disorder. It may be possible to gauge the regions of the brain where the condition is originating and if the patient is a suitable candidate for treatment.

However, symptoms from neurological disorders often appear sporadically and with little to no warning. Seizures vary from minutes to years apart, and sometimes are not realised or detected without proper equipment. Hospitalising patients for long periods of time is a costly option, and in such circumstances the patient could remain in hospital indefinitely. Such circumstances lend themselves to an ambulatory system, where an outpatient can be monitored continuously without discomfort or hospitalisation, improving quality of life while decreasing costs.

With the emergence of low power wireless technologies coupled with portable devices such as tablets and phones, it is a natural technological step to bring care and monitoring outside of the hospital. Through leveraging low energy radio capable platforms, i.e. smartphones and tablets, in the context of an Ambulatory Electroencephalogram (AEEG), it is possible to empower the patient to inexpensively take health care into their own home and out of the hospital. This project looks at maximising transmission of EEG signals over the emerging wireless technology, Bluetooth Low Energy (BLE). Further, while this project is targeted at EEG signals, there is no reason why this research and technology cannot be applied to other signals and systems, examples including glucose monitoring, electrocardiography and spirometers.

Contents

1 Acknowledgements

The opporunity is taken here to express gratitude to all those who have directly contributed towards the project.

Firstly, the project surperivsors, James Mardell, Chen Guangwei and Esther Rodriguez-Villegas from the Circuit and Systems deparmental group, for the opportunity to undertake this piece of work along with my second marker for his efforts in reviewing this work.

Cambridge Silicon Radio (CSR) provided hardware and technical support in the spirit of academia. Particular acknowledgements go to employees Adam Hill, Martin Spikings, Mark Wade, Neil Stewart and Simon. CSR have requested that this project report or relevant parts of it be made available to them.

Guan Yang and Jacob Rosenthal from New York, United States, for pubically making available code to drive nRF8001 radio. Imperial College student Stelius Ioakim for sharing experience of printed circuit board (PCB) assembly.

Dr. Nissim Zur, CEO of Vitelix Limited is an expert in low power wireless technologies, and has conversed over many aspects of the CSR1010 chip used. Further, he has also taken an interest in this project's work in regards to maximising the speed, and has requested the results be shared with him.

Special thanks go towards Mike Harbour and Victor Boddy for their efforts in printed circuit board manufacture and assembly. Countless hours were spent in the lab pushing the department's PCB fabrication facilities outside specifiction and assembling the boards.

And finally, my girlfriend for her dainty and dexterious hands, along with her patience which was invaluable during cricuit assembly.

2 Introduction

2.1 Problem Landscape and Motivation

Neurological disorders and their sequelae are estimated at present to affect upto one seventh of the world's population, a figure which currently stands at 1 billion. With the ever increasing life expectency and decreasing (relative) fertility rates, the age demographic has shifted towards an ageing population. This has caused a growth in neurological disorders such as Parkinson's, multiple sclrosisi, Alzheimers and other dementias. From the 2012 report published by the World Health Organisation (WHO)[11], the societal costs of dementia for the United Kingdom totals £23 billion, a figure that matches the costs of cancer (£12 billion), heart disease (£8 billion) and stroke (£5 billion). Similarly a 2010 Swedish paper published that the costs of dementia (50 billion SEK) was higher than that of depression, stokres and alcohol abuse (32.5, 12.5, 21-30 billion SEK respectively)[10]. With neurological disorders constituting to approximately 12

EEGs are used extensively for detecting, characterising and monitoring brain activity related to the afromentioned dieseases. Examples applications include diagnosing patients regularly loosing conciousness with syncopy; measuring the effectiveness of medication for patients currently diagnosed with epilepsy[5]; deciding whether the patient is a candidate for removal of the cortex part associated with the disorder[12]. Unfortunately many neurlogical disorder symptons occur sporadically with little to no inidication of an impending event, such as in the cause of epilepsy a seizure. In some circumstances the patient could remain in hospital indefinitely, however symptons (e.g. seizures) can be seconds to years apart, and sometimes are not realised or detected without proper equipment. The diagnosis, monitoring and treating many of these diseases is currently a costly procedure due to the resource requirements (staff, time and equipment). Common practices include hospitalising and monitoring patients between 24 hours and a week. Such circumstances lend themselves to an ambulatory system, where an outpatient can be monitored continuously without discomfort or hospitalisation, thus improving quality of life and social welfare (e.g. no sick days off work).

Since the arrival of BLE in smartphones in 2012, the vast majority of smart phones are "Bluetooth Smart Ready" (incorporating BLE technology). This is a relatively new technology capable of (very) low power communication for applications requiring a low data rate. This low power consumption means devices can have long lifetimes. While other low power wireless options exist such as ANT or ZigBee, BLE is the only popular radio technology being manufactured into smart phones and tablet devices. In comparison, (classic) Bluetooth power consumption is typically 1 to 2 orders of magnitude higher than BLE.

Through leveraging widely popular and familiar smart phone devices with this new technology, in the context of an AEEG, it is possible to empower the patient to inexpensively move their health care out of the hospital and into the home. By coupling cheap low power sensors and radios, with powerful ubiquitous consumer technology, it is possible not only to cheaply and efficiently monitor patients, improve the quality of life for patients but also help physicians further their understanding of neurological disorders. Further, when combined with data-mining or so called big-data analytics novel insights may reveal new understanding and ultimately better forms of treatmeant.

2.2 Existing Technologies and Products

Popular EEG products on the market that have already shown integration with consumer electronics as well as research include Emotiv Systems' EEG headset. The device retails at $750 (approximately £450 at the time of writing), sporting 14 saline sensors, gyroscopes for positioning and a lithium battery capable of deliverying upto 12 hours of continuous use.

Figure 1: Emotiv System's EEG Neuroheadset

NeuroSky have released EEG products, the MindWave and MindWave mobile. The foremost operates a proprietary radio system in the 2.4 GHz band, at a datarate of 250 kbit/s, a baudrate of 57,600 and 10 meter range. It is also noted that there is a 5

Figure 2: NeuroSky MindWave Mobile EEG Headset

Away from specifically EEG, the consumer fitness sector is being targeted quite strongly, and many devices already exist that utilise lower power technologies to act as gateways for real-time data logging. For example, there already exists a competitive market between heartbeat monitors, cadence monitors and pedometers. These 'activity trackers' use low power electronics and radios to log user's activities and update the user in real time with activity information through the user's phone or smart watch. Popular products on the market at the time of writing include the Fitbit, Fuelband and Jawbone, which all make use of the BLE technology to connect to smartphones.

2.3 Project End Goals

Currently, the Imperial College Circuit and System's group has a wired EEG measurement device. The wired connection between the EEG sensors and a fixed computer serverly restricts patient's mobility and hence are impracticable for long periods of use. This project will explore using BLE technology for electrocengraphy, and build a prototype system capabale of interfacing with an analouge front to transmit EEG data to a portable device such as a tablet or smart phone. The project will explore the maximum throughput of such a device, the energy requirements and attempt to quantify the overall suitability of BLE to EEG montoring .

Idealistic requirements of the project include

- Running time of atleast 12 hours
- Channel resolution of 8 bits (albeit number of channels undefined)
- A weight of less than 7 grams
- 10 meter range
- BLE wireless technology
- Ability to communicate with a smart phone or tablet
- Real time update of signals

2.4 Structure

The project is organised as the following. An overview of EEG signals and BLE technology is presented, giving background where it is believed revelant and of value to the rest of the project. It should be noted that these sections are not a exhaustive description of EEG signals nor BLE. The report will then move onto preliminary resarch, where the technology is evaluated and the components are discussed in detail (including risks) and a selection made for the prototype. A desired specifications is then presented from the components chosen. The implementation section documents and discusses building, optimising and any issues found while developing the prototype. The results section lists the project findings, followed by a thorough evaluation of these results. A projec wide conclusion can then be found where BLE's suitability for the application is discussed along with the limitations of the current system, and suggested future work for project extension or further interesting reseearch as a

result of this project. Lastly, a section is reserved for final remarks for comments that don't formally fit elsewhere into the report.

3 Theory and Technology

3.1 Electroencephalography

Different freqs

3.2 Bluetooth Low Energy

The original Bluetooth, hereforth refered to a Bluetooth Classic (BTC), was intially concieved as the solution to wired communication over short distances (typically less than 100m). The original specification had an air over-the-air rate of 1MBps, though this has increased to around 3 MBps in the latest version of BTC. Similarly, BLE has an over-the-air rate of 1MBps. Despite the odd realisation that BLE, a much newer technology, has the same over the air rate of last the first incarnation of BTC, the maximum theoretical throughput of BTC is 700kBps, compared to less than 250kBps for BLE - roughly one third of the maximum throughput BTC was capable of (the latest version of BTC brings the disparity to one ninth). While intuatively it may seem that BLE is a less efficient technology, BLE can be orders of magnitudes more efficient than BTC in particular use cases.

Applications where BLE excels in are ones where communication between two devices is only required intermitently, and the volume of information sent is small. An example would be a thermometer in a greenhouse connected by radio to a visual display unit inside the home. Temperature changes at a rate slow enough that it is only necessary to check the temperature every 10 minutes. Once every 10 minutes the radio thermometer device can wake up take a measurement, send a notification of a measurement then return to a deep sleep. BLE does this much better than BTC, taking only a few milliseconds to connect. BTC takes between a few hundred milliseconds to several seconds to reconnect. While both millisecond orders of magnitude and second orders of magnitude are small when compared to an order of magnitude of minutes, over time it adds up to a significant amount, and BLE devices can last many years of a small, single coin cell. BLE is excellent for applications which involve small episodic transmission of data. In the scenario described a BTC system would have a lifetime of approximately 100 days from a typical 3v lithium cell. Off the same cell, a BLE system would have a lifetime of many years. In fact, in this scenario the BLE system lifetime can be extended further as BLE can support connectionless communication, whereby it simply wakes up and transmits the thermometer state to any device that's listening without the need for acknowledgement.

The reason the reconnection times are much faster for BLE is as far as the communication devices are concerned, they never disconnected. Rather in the BLE protocol, the devices agree to meet a specified periods known as connection interval (CI). The devices are free to perform any operations in the mean time, though typically enter a state of hibernation. In many scenarios between two radios, one device will be much more power conscious. In the example above, the battery powered device in the greenhouse would typically be the power conscious device while the visual disuplay unit inside the home will likely be powered from the grid, and have no concern as to its power consupmption. In these situations, the power conscious device is defined to be the slave and the remaining device to be the master.

The slave device also has the ability to skip connection intervals. That is, the slave to skip a

upto a predetermined number of connection intervals, known as the slave latency. While the master must always check back to see if the slave has sent anything, the slave, if it has nothing to send, doesn't need to wake up. For example, if the slave latency is 120, and the connection interval is 1000ms, then the slave is not obliged to communicate with the master for upto 2 minutes, despite the master having to check every 1000ms. If the slave is not able to make contact with the master after 2 minutes, then the master will begin counting the number of times the slave has missed the obliged connection period (that is the CI multiplied by the time slave latency). If this value reaches above a certain threshold (a typically recommened value of 6), the master will consider the slave disconnected, and be required to go through a connection process again[UNLESS USING BROADCASTING TO SEND DATA]. Continuing with the scenario, the slave will be considered disconnected if it hasn't made contact with the master after 12 minutes.

Another contribution to reduced power over BTC is the reduction in the number of channels used in communicaton. BTC, BLE along with other wireless technologies that operate in the 2.4GHz ISM band such as a WiFi and ZigBee all make use of spread spectrum techniques to achieve a sufficient level of noise immunity. That is, the available bandwidth is split (spread) into smaller channels and radios communicate between one another using a pre-dertmined channel hop sequence. As the number of channels decreases, the channel band size grows requiring less accurate and complex modulation hardware, hence decreasing power consumption. BTC originally used 79 channels, and BLE reduces this to 39.

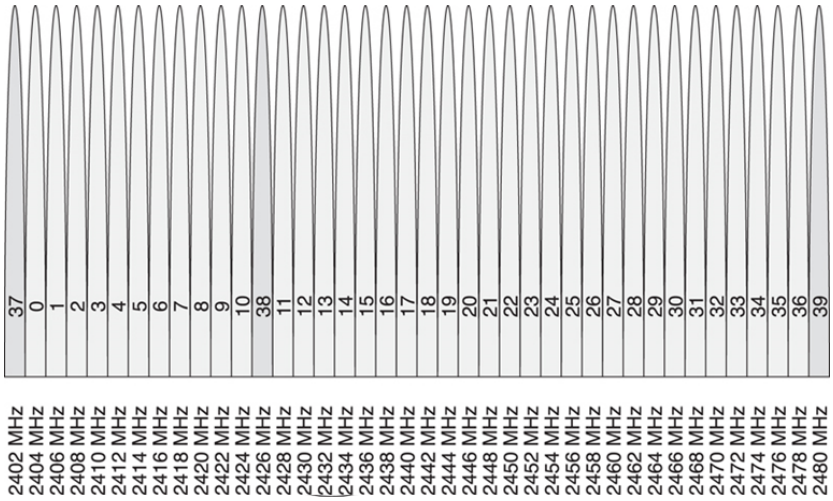


Figure 3: BLE channels (advertisement channels render darker)

The number of channels dedicated to advertising has also decreased, meaing less time is spent searching for discoverable devices. The advertisement channels have been specifically chosen not to interfere with the common WiFi channels. Finally, the radio characteristics are very dependent on temperature. Complex mechanisms are used to compensate and recalibrate on-the-fly radio parameters. Due to the episodic nature of BLE the radio does not come under such thermal extremes. All these design changes have positive hardware ramifications. The reduced compelxity of BLE means reduced hardware requirements (notably memory), in turn reducing the leakage current.

BTC was designed with the idea that it would be used to do many common jobs, and hence particular configurations were built into it. In BTC, these configurations are known as profiles. Exam-

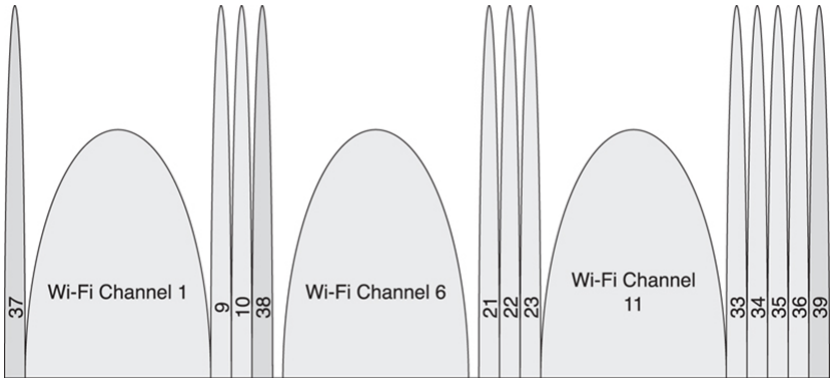


Figure 4: BLE channels with WiFi channels overlaid

ple profiles include the audio distribution profile (A2DP), which is used in by many Bluetooth product manufacturers to allow a device, such as a phone to interact with an audio system, such as in a car. Another example would be the serial port profile (SPP), meant to emulate the highly popular and robust RS-232 serial standard for data transfer (recall that BTC was concieved as a solution to wires). This is all built into what is known as the Bluetooth stack – a software framework that interacts between the physical layer and the application layer[1].

BLE also makes use of this paradigm but is often superficially depicted as BTC operating at lower speeds and power consumption. It is not currently compatible with BTC and there are no plans for it to be. Like BTC, the BLE architecture has 3 over-arching parts: Application, Host and Controller. The controller, simply put, is the radio and related hardware controllers and the application the use case, which could be a cadence monitor, thermometer or even an electroencephalogram. It is the host controller interface (HCI), commonly known as the "stack" that provides the necessary software to enable the application layer to communicate with the radio (see Figure 5).

In an attempt to be economical with time and space components of the stack deemed irrelevant will not be discussed further here. For example a description of the security manager is not relevant as this project is not concerned with sending data over encrypted links. Similarly, the Logical Link Control and Adaption Protocol, while used extenivesly in all radio communication will not be discussed in detail as it doesn't provide any insight into maximising throughput or minimising power.

In BTC profiles were diverse and large enough to warrant chip designers releasing tailored chips to perform well for a specific profile, i.e. a chip supporting A2DP may contain coder/decoder (CODEC) hardware for real time audio streaming. In BLE the profile framework is far lighter. BLE's profiles are all built ontop of the Generic Attribute Profile (GATT), which in turn is built upon the Attribute Protocol (ATT), a protocol optimised to run on BLE devices. Attributes are an umbrella term, being the atomic unit of data communicated between BLE devices. Profiles are hierarchical constructions of attributes, in the top down order of profile, service, characteristic and descriptors, as shown in Figure 6. A BLE device implements atleast one profile, Generic Access Profile (GAP) along one more which is typically the purpose of the device. GAP contains important information such the the device name and prefered connection properties. This second profile can be one of the standard profiles as defined by the SIG group or a bespoke profile for the application, such as the

[1]Depending on what level of the stack one is working, master can take the names central or client, while slave can take peripheral or server.

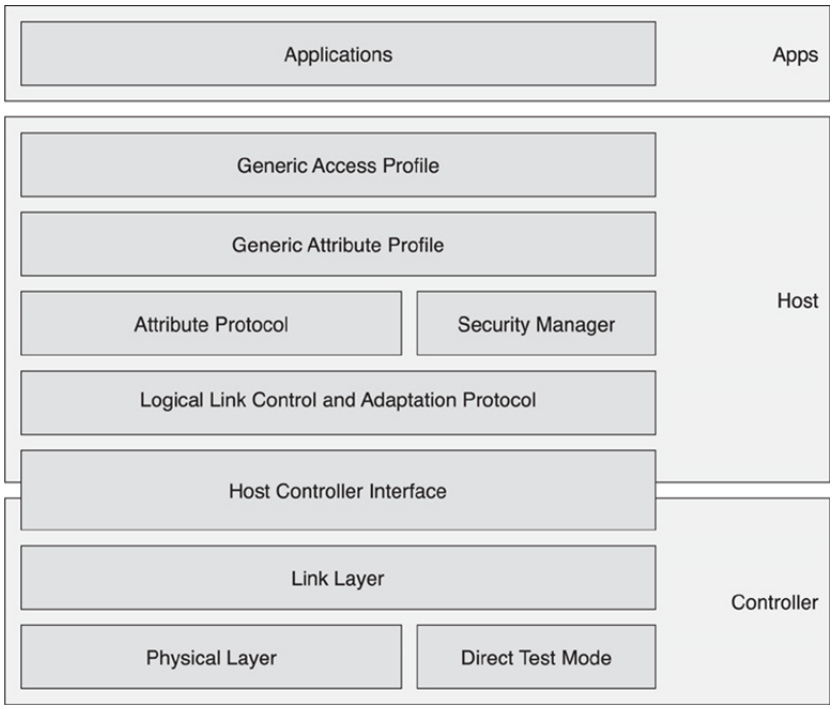


Figure 5: BLE Architecture

case for a EEG. Popular, SIG defined examples of BLE profiles include the heart rate profile (HRP), health thermometer profile (HTP) and even a glucose profile (GLP) with room for many more to be incorporated into the core BLE Special Interests Group (SIG) defined GATT specifications.

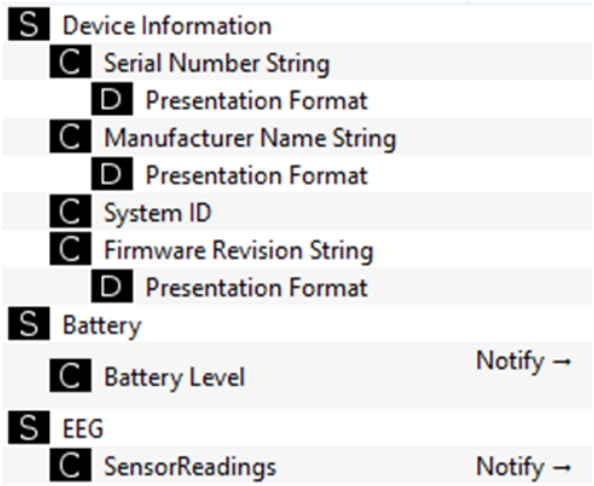


Figure 6: Example GATT profile consisting of 12 attribute - 3 services, 6 characteristics and 3 descriptors.

Attributes represent information about state. In the case of the greenhouse, the state would include the measured temperature. A suitable profile name which encapsulates the state is ”thermometer”, which contains the services ”device information”, ”battery service”, and ”thermometer”. As in figure 6, the greenhouse thermometer service may contain the same device information and battery services, but replace the ”EEG” service with the thermometer service, and the ”SensorReadings” service with temperature. The other profile, GAP, will contain the attributes which define how BLE unit

discover and establish connection to one another. This includes the device name, perhaps "greenhouse thermometer", along with the preferred slave connection properties, e.g. a connection interval of 4 seconds, with a slave latency of 150 (10 minute window). All this information is contained within the GATT database, and shared as needed to other BLE units.

When communicating attributes, there are four data operations available. Read and write typically require one device to access the others characteristic. In the case of the greenhouse, the house device would request to read the thermometer. Notification and indications differ to read and write, in that the device(s) after information subscribes to the changes of state of the characteristics. For example, when greenhouse slave device wakes up, if the thermometer measurement changed, it will send a notification or indication alert the master device inside the house. The former methods can be thought of as synchronous means of communication while the latter asynchronous. Notifications differ from indications in that indications require a application level acknowledgment. That is, the indication is bubbled up to the user code, which then either accepts or rejects the indications. Notifications are acknowledge near the bottom of the BLE stack, verifying correct receipt and message integrity. Therefore notifications are suitable for higher throughput applications. As shown in figure 6 shows notifications are operation configured for battery and EEG sensor readings. In this example, whenever the battery level changes, any devices subscribed to the characteristic battery level will receive a notification.

In BTC the network topology used pico-nets, whereby one device has one master, but could be a master of another device. BLE operates a simpler network topology, star, whereby a device can either be a master or slave, but not both. The master has the responsibility of organising itself between the slaves. If one slave requires large amounts of bandwidth, it my impact the quality of service encountered by the other devices.

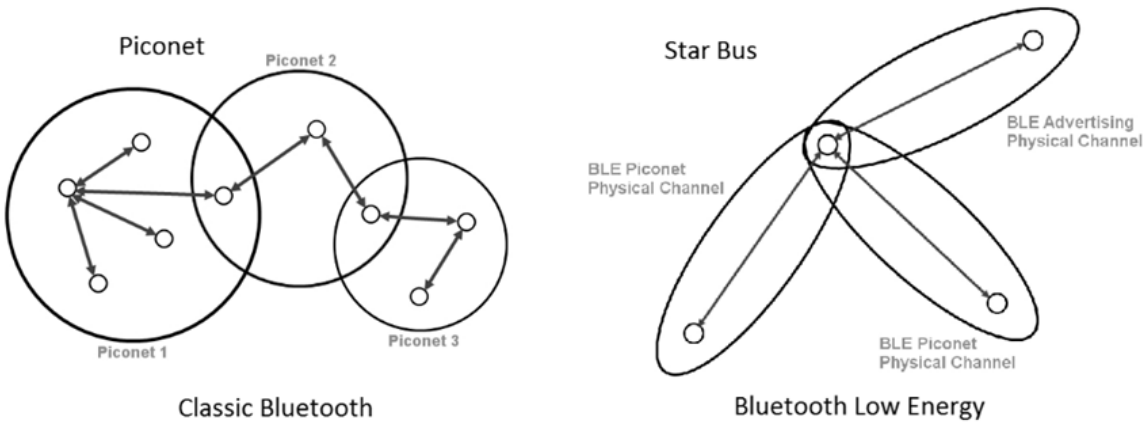


Figure 7: Bluetooth Network Topologies

Both BLE and BTC devices move through generic system states. The same abstract view can be applied to both technologies and is shown in. Note that depending on the role of the device, the state moves right (master) or left (slave) from standby. It may appearing confusing to have another state for scanning which can only move into the standby state, but this is useful for searching and discovering devices with no commitment to connecting. Such a use case might be suitable for devices that intermittently broadcast small amounts of information. Assuming that the master BLE device is

in the initiating state, searching for a connectable device, and at the same time the BLE slave is in the advertising stage, periodically broadcasting information using advertisement packets; the two devices will find one another and may initiate a connection request.

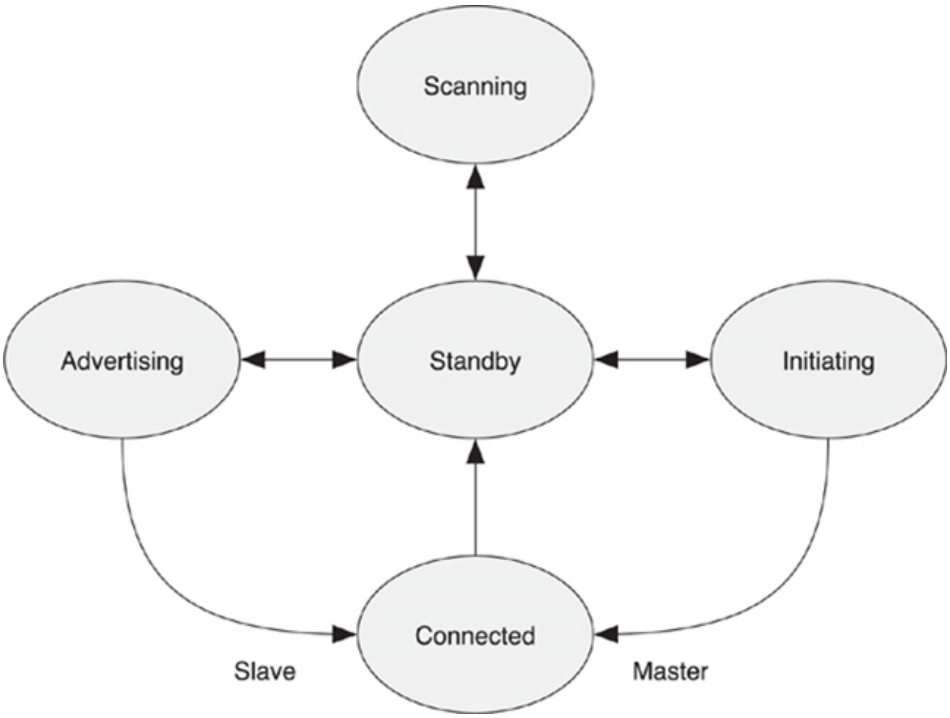


Figure 8: Device State Transition Diagram

BLE is principally composed of two types of packets, advertisement (Figure 9) and data (Figure 10). Both packet types vary in length dependent on the payload but share a 32 bit advertising access address field, a 24 bit Cyclic Redundancy Check (CRC) field, an 8 bit header and an 8 bit length field which defines the Packet Data Unit (PDU) size (the last 2 field are often collectively called the header field). Advertisement PDUs range from 0 to 29 bytes (232bits), meaning the total packet size can vary between 72 and 320 bits. The over-the-air data rate is 1Mbit/s, meaning advertisement packets are transmitted at a rate between 72 and 320μs (a single bit is transmitted every 1μs). In addition to the access code and CRC fields, data packets consists of an 8 bit preamble and a PDU varying between 0 and 27 bytes (4 of these bytes are reserved for encryption). Hence, the packet length varies from 80bits to 296 bits (328 including encryption).

Figure 11 shows a connection between two devices being initiated. The first packet shown is is an indirect advertisement packet, available to all listening BLE devices. The second packet is a connection request from the master device, communicating in the payload its address, the address to establish a connection with, and random access address, a connection interval length, the hop sequence, channel map, the connection timetout, slave latency and other things. Here the advertisement packets are rendered in green, the data in (predominantly) yellow (the magenta also represent a type of data packet)

- The channel map bit pattern corresponds to a contiguous stream of 37 ones, corresponding to all 37 channels being operational (the master has no reason to prevent transmission).

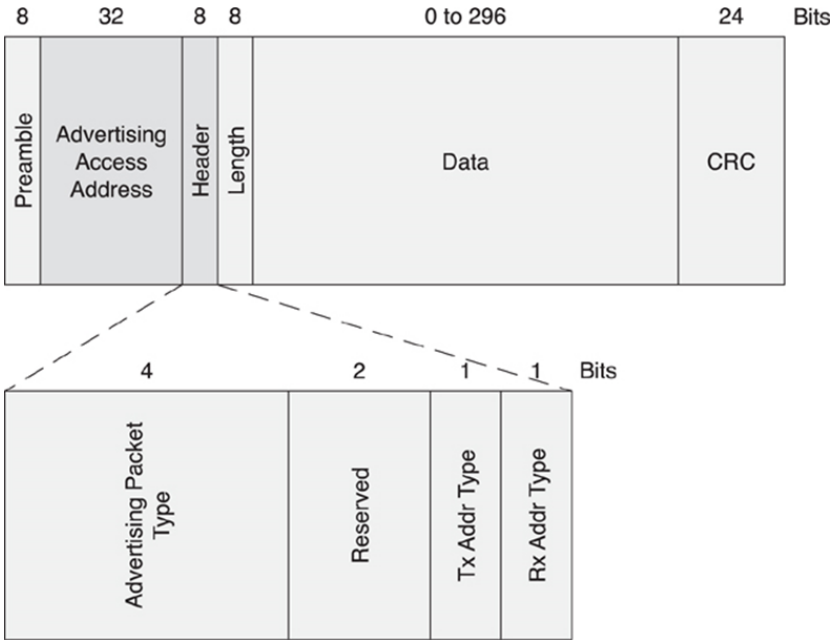


Figure 9: BLE Advertisement packet

Img advertisment

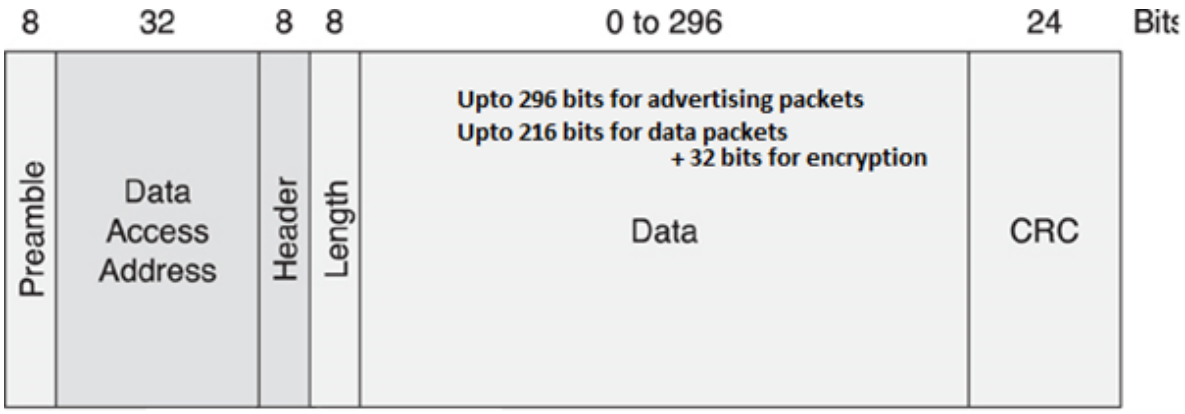


Figure 10: BLE Data packet

- The hop byte indicates between each connection event (CE) how many channels the device pair will increment. From packet 242 onwards, the channel map increments every CI (2 packets/30ms).
- The access address is a randomly chosen number (by the master) which acts as a identifier for a connection between two devices.
- The interval of 0x18 (24 decimal) represents the CI length in units of 1.25ms. Therefore 0x18 represents 30ms between subsequent CE. Between each master to slave directional data packet (every 2 packets), the total sum of the time is 30000μs

It is highlighted that (at least initally) the slave interval is 0, meaning the slave should wake up every CE. In packets numbers 243 and 244, the slave and master respectively have nothing to send, and simply acknowledge on another with an empty PDU. The protocol realises flow control through the the lazy acknowledgment of Sequenc Number (SN) and Next Expected Sequence Number (NESN)

bits, embedded into the header of every data channel.

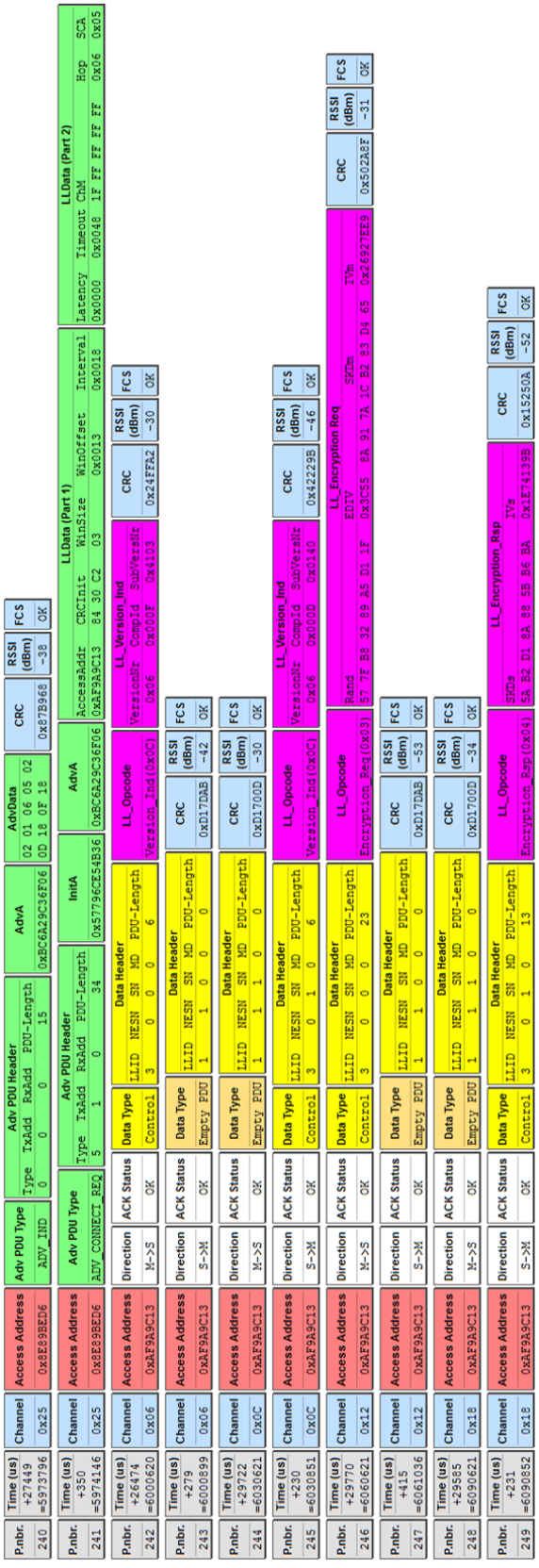


Figure 11: BLE connection initialisation

While explained for completeness, advertisement and packets used in the initial establishment of a connection are of no concern to this project since they do not contribute to data throughput and once a connection has been established, do not contribute to the power consumption of the device. Hence, their contribution is removed from measurements as it is assumed the long term amortised cost is zero. The (incomplete) connection initialisation process shown in Figure 11 is known as "Just Works" pairing.

As previously mentioned, to achieve high throughputs notifications are used. To achieve the maximum throughput, it is desirable to send notifications as fast as possible. However, data cannot be sent out without flow control, and all notification packets must first be received then acknowledged before the next packet is sent. When a packet is sent, there is a mandatory 150μs inter-frame period. For the maximum throughput the connection receiving device would just acknowledge the data in an 80 bit acknowledgment response. Another 150μs frame would occur between this responding device and a transmitting device, bringing the total time up to:

$$296\mu s + 150\mu s + 80\mu s + 150\mu s = 676\mu s$$

This is shown graphically in Figure 12 (encryption is enabled). The maximum duty cycle is obtained when encryption is used

$$\frac{328\mu s + 80\mu s}{708\mu s} \approx 55.6\%$$

which is relatively low when compared radio technology duty cycles, e.g. BTC. The 150μs inter frame period exists to prevents the silicon from heating to excessively, preventing power consuming hardware being required to recalibrate the radio.

From the figure it is possible to calculate the upper bound for the number of packets that can be transmitted per second is

$$\frac{1000000\mu s}{676\mu s} \approx 1479.29\ packetes/second$$

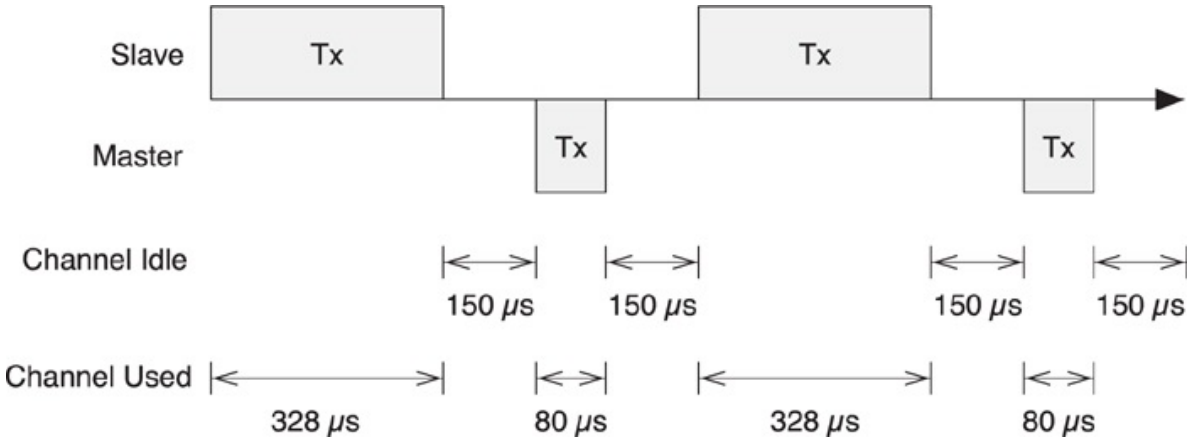
Of the 37 bytes (296 bits) sent for a notification packet, 27 of them are known as the data payload. Even if encryption is not used, 4 bytes of the 27 are required for the L2CAP header use. The remaining 23 bytes are quotes as the available application bytes.

$$1479.29\frac{packets}{s} \times 23 \times 8bits \approx 270kbp/s$$

While a lot of literate quoteCITATION PLEASE] this as the maximum usable data rate of this 23 bytes 3 bytes are required to identify the command type (notification) through an op-code (1 bytes) and which attribute it belongs to through an attribute handle (2 bytes). This means that only 20 bytes of the original 27 reserved for the application are usable, reducing the upper bound for throughput to

$$1479.29\frac{packets}{s} \times 20 \times 8bits \approx 236.7kbp/s$$

Here forth, the usable payload of a "notification" or "data packet" is 20 bytes. When aiming for either low power and/or high efficiency, it is important fill the usable payload with as much data as possible, as every notification packet sent has a fixed cost of 80 bits associated with it. TALK ABOUT PCK EFFICIENCY



centering

Figure 12: BLE's packet flow for maximum throughput

The BLE specification defines no maximum

The BLE protocol was not designed for high throughput applications, but rather for low latency episodic communication devices transferring small data payloads of representing device state. The energy per bit is low compared other popular technologies, and should not be used as the bottom-line metric for performance. Rather, a holistic metric of the whole system energy consumption for a specific use case will form the basis of the performance metric in this report.

4 Preliminary Research

This chapter deals with evaluating the BLE devices to choose a device most suitable for a maximising throughput and EEG signals for an AEEG system. This chapter goes on to evaluate and select AEEG ancillary components required for the prototype.

4.1 Radio Evaluation

There are a number of competing companies int the BLE consumer space. The most popular chip manufacturers at the time of evaluation areTexas Instruments (TI), Nordic Semiconductor (NS) and CSR, although other manufacturers exist, they normally design combo-wireless technology chips, e.g. Broadcomm, which only offers WiFi, BTC and BLE combined system on chip (SoC). These chips are not suitable for this project due to their complexity, packaging and intend application - the typical use case of such combo-chips are in high powered devices such as tablets, smart phones and notebooks.

TI provided an inexpensive BLE packet sniffer which was used extensively throughout this project.

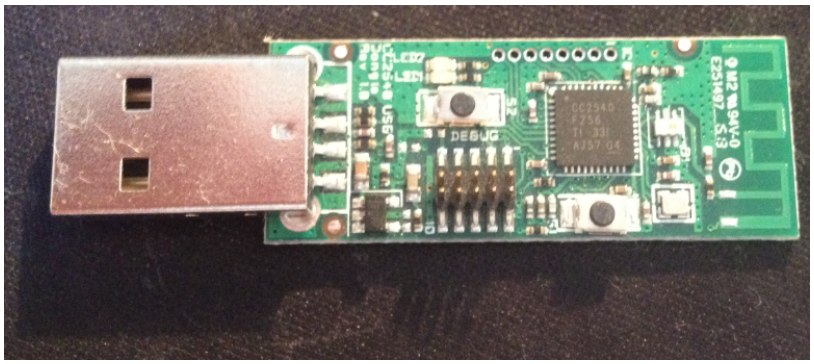


Figure 13: TI's BLE packet sniffer

As discussed in the preivous sectino, to achieve the highest throughput values, the BLE communication method used is notifications . Notification data packets can carry a usable payload of upto 20 bytes

After sourcing

4.1.1 nRF8001

The first radio tested was NS's nRF8001, selected due to the data sheet claiming it was the lowest power consuming device. The reason for this was because the chip only contained a controller and host layer. The application layer must be provided by an external micro controller, which is a large amount of effort to write. Fortunately a hobbyist open source project[7] was available that provided the necessary code to get limited connectivity up and running. Figure **??** shows the prototyping setup used to measure the performance of the nrf8001 microchip.

Using a shunt resistor, the measured peak current was approximately 14.5mA, which correlates with the figure found in the data sheet (14.6mA). This was at a

The ATMega328 consumes approximately XmA, however for the remainder. As the device was hardware limited to only 1 packet per connection event further development was abandoned.

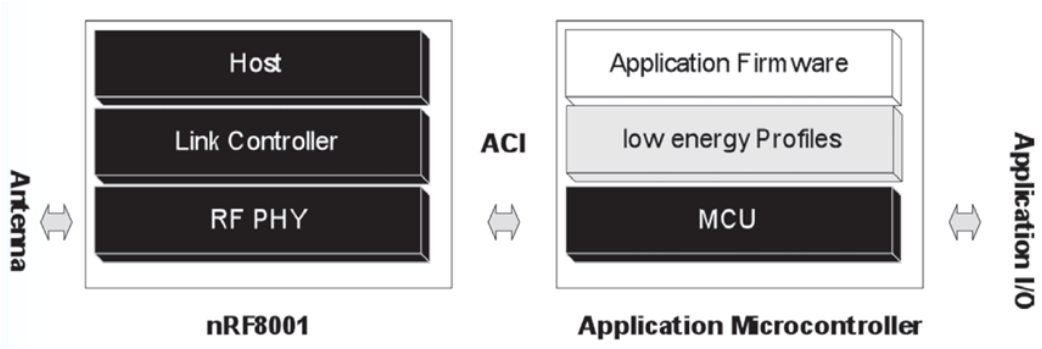


Figure 14: nRF8001 application block diagram

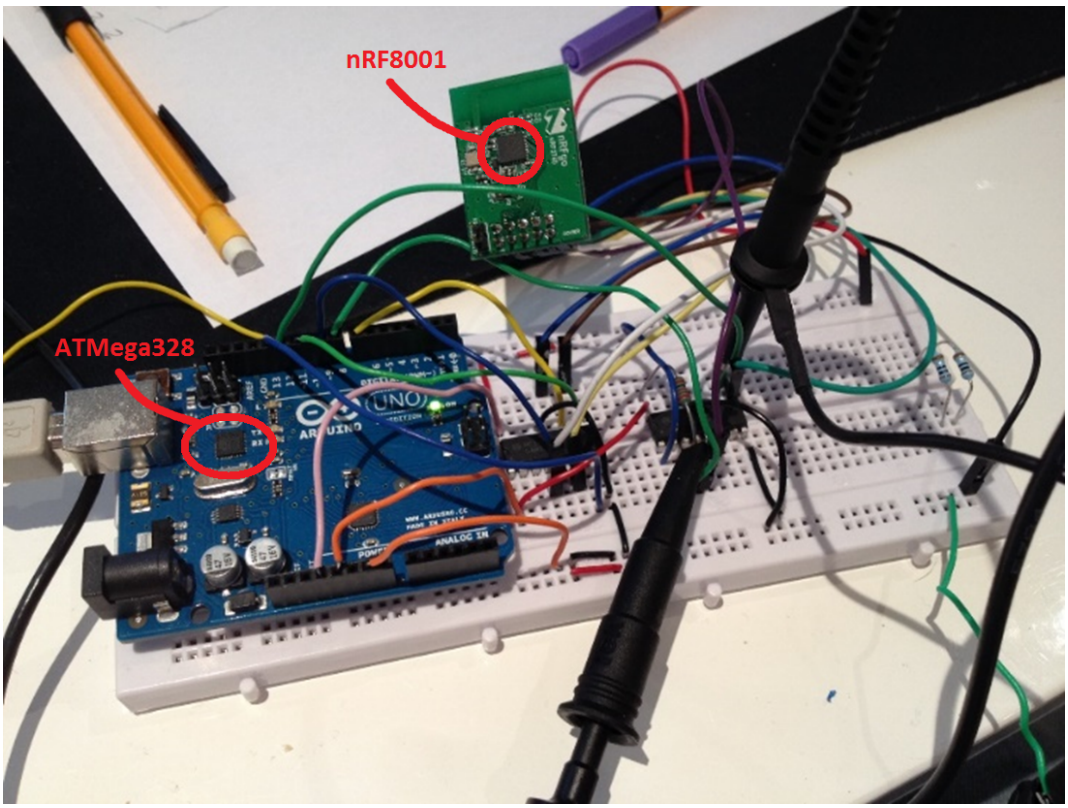


Figure 15: Arduino Uno development board acting hosting the application layer for the nRF8001 radio. Level shifters were required for interfacing with the low-paper chip

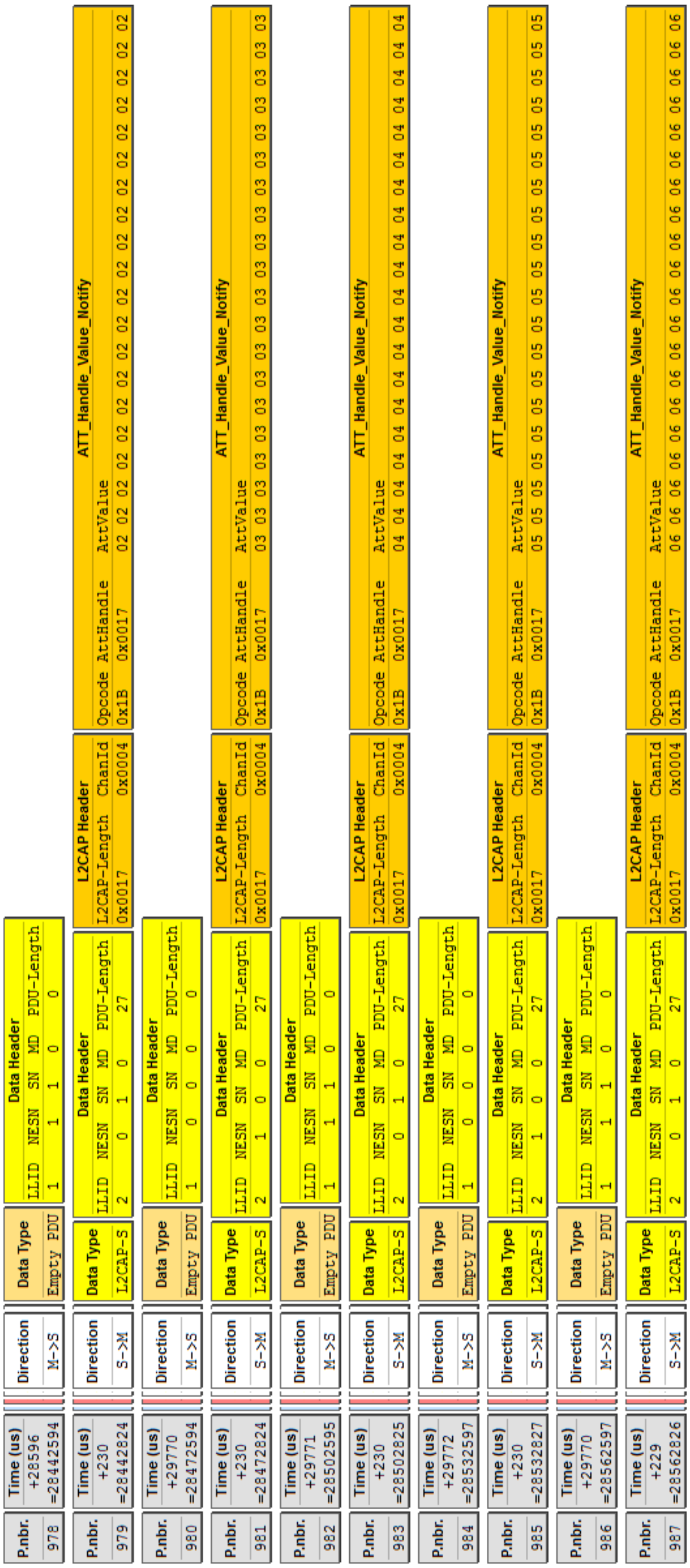


Figure 16: Packet-sniffed connection between Apple iPhone 4s and nrf8001. Notification rate 1 packet every 30ms. Some fields removed due to page space constraints

Initially, the device was configured to send a large amount of notification within a single {CE and at the lowest CI. The master device was an Apple iPhone 4S, and it was found the total throughput was 5328 bit/s or 2.3

As the device can only send 1 packet per CI, to achieve the highest throughput, the device must be run at the smallest connection interval of 7.5ms. At this interval the upper bound for throughput becomes

$$\frac{1000}{7.5} \times 20\, bytes \approx 2666\, byte/s$$

With 16 channels at an 8 bit resolution, the highest frequency measurable (according to Nyquist condition) is

$$\frac{12666\, byte/s}{16} \times \frac{1}{2} \approx 83Hz$$

A maximum support frequency running a solo channel is approximately 1328 Hz.

4.1.2 nRF51822

Another NS product tested was the nRF51822. Information on the suppport packets per connection event was also not present in the datasheet[9]. This device was found capable of receiving up to 6 packets per second, and this was verified by an engineer[8]. While a marked improvement over the nRF8001, this meant that the device was only capable of operating at 60

4.1.3 CSR1010

Through direct contact with CSR engineers, it was reported a particular single mode BLE radio device, the CSR1010, was measured reaching 250kbps

> *. . . the theoretical maximum throughput for a single BLE connection using a much bandwidth as possible is around 300kbps at the air interface. The usable application data rate is somewhat less, but 80kbps should be possible (Using ATT packets with a 23-byte MTU should provide approximately 200kbps application data rate, if my maths is correct).*
>
> *However, in practise you are limited by what other activities each end of the link is doing. In particular smart-phones (as one end of the BLE link) will likely severely restrict the BLE throughput to account for other Bluetooth activities (such as eSCO links) and co-existence with WiFi etc. But if you controller both ends of the link (e.g. single-mode BLE devices at both ends) then it should be easier to get close to the maximum theoretical rate.* [sic]

[sic]

In the first paragraph the engineer is treating the full 27 bytes in the data payload as the application data. This number should of come from

$$\frac{1000000\mu s}{676\mu s} \times 27 < 320kbps$$

320 to 300 is a generous approximation, it is believed that the engineer actually rounded a smaller number, 305, which was calculated by including an unecessary 32 bit message integreity code (MIC), used only when encryption is enabled:

$$\frac{1000000\mu s}{708\mu s} \times 27 \approx 305kbps \approx 300kbps$$

Many literature pieces and training materials on BLE make this assumption, including those on the SIG website [2] [1].

In the second paragraph the engineer is highlighting some important facets of BLE devices, and that is although one end of the link is determined by the lower of the two device's thorughputs. The engineer cites smart phones as such devices that restrict the link capability. This was seen in the nRF8001 - the iPhone 4S could not handle a connection interval less than 30ms. Combined with that the nRF8001 stack couldn't transmit more than one packet per connection interval on average, the maximum throughput achievable by this device was cut by 4.

When asked how to show the math behind the numbers achieved

> *While testing our uEnergy silicon (CSR1010) at HCI level I can achieve 250kbps throughput using a dedicated test application (i.e. data generated on-chip). HCI packets are 27 bytes, so to get my earlier number I simply did (20/27) * 250 to get an application-level data rate (27 bytes minus 4 bytes L2CAP header minus 3 bytes ATT header, leaving 20 bytes for application in each packet).*

I haven't tested raw throughput at the application level, but as our ATT protocol runs on stack alongside the HCI/controlller level, it should not see any additional processing overheads. [sic]

The HCI is the component between the host and controller layers shown in Figure 5. The engineer is saying that data throughput should no be affected by any high-level stack activity (any stack congestion, if existing, shouldn't affect throughput). Although the engineer has given no commet on latency (data will take time to propagate up the stack).

CSR supplied a development kit consisting of provided a development board (CNS10004V5D) and Universal Serial Bus (USB) dongle. The CSR1010 is part of CSR 's new μEnergy product line, for which CSR have developed their own compiler and integrated development environment (IDE) known as xIDE. For initial throughput testing, an example health thermometer application was used. From exploring and understanding the main features of the operating system abstraction layer (OSAL), it was immediately possible to increase the number of notification packets sent per connection interval from 1 to 8.

The standard mechanism provided in the example, is for every connection interval a confirmation of the first acknowledge is raised in the application layer. This is used in the program as a point of reference for application layer timers, so schedule the next wake up time, etc. This mechanism provides visibility at the CE level, but not at the notification packet level - if many notifications are sent and acknowledged, this method doesn't provide any information that the CE may have room for more notifications. This is a problem when large throughputs are desired, as it appears the buffer queue is 8 packets in size. When more than 8 are added to the queue, they are ignored, causing data loss. Hence, through the naive method, whereby notifications are placed into the queue before or after the start of the CE, upto a maximum of 8 notifications are correctly sent in order. While it is possible to continously flood the stack (buffer queues) with more notification requests, without knowing if the stack is capable of dealing with them, data may be lost. This mechanism is visualised in Figure 17.

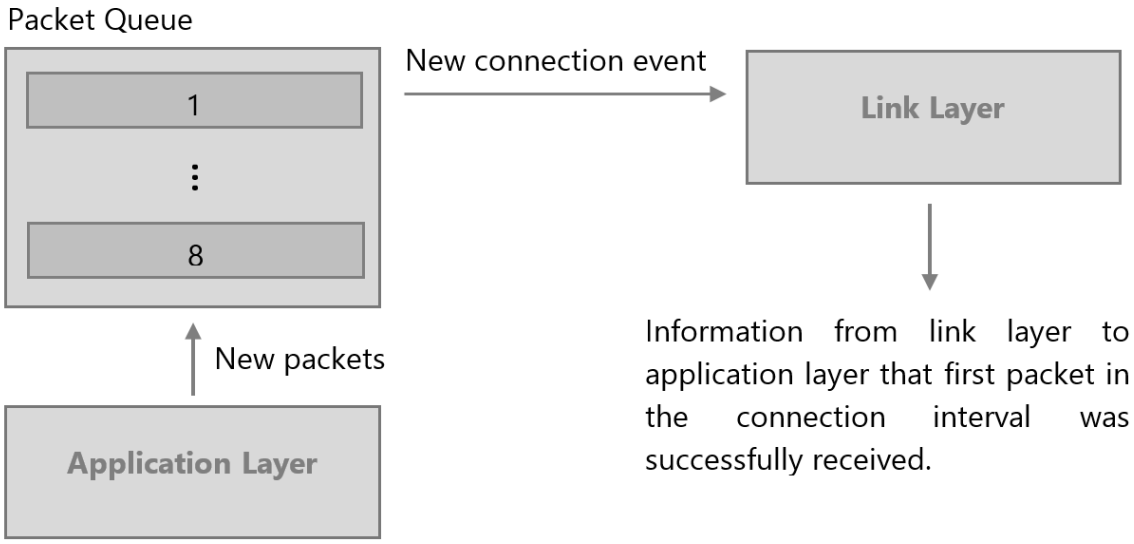


Figure 17: Notification of the first packet

An experiment to test throughput was setup whereby each connection event sends 8 notifications per connection event with the same payload, bar one byte which increments with the connection event. The output can be seen in Figure 18.

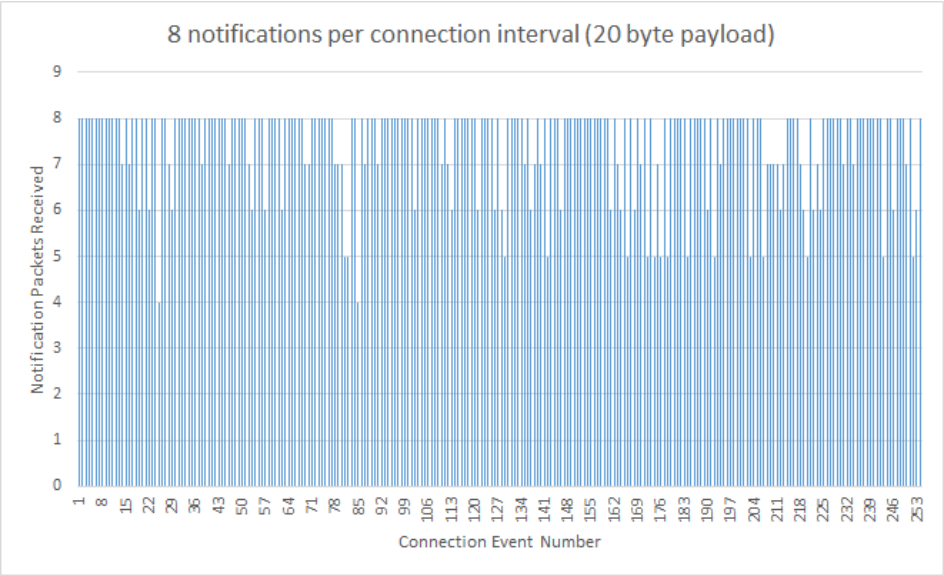


Figure 18: Queuing of 8 packets being lost

From the experiment above, the conclusion was that the link layer stack became flooded with notification requests. Had the requests been accepted into the stack, then they would of been transmitted. If they failed to arrive at their destination, then the flow would of caused a negative acknowledgment, and the packet would of been retransmitted. Provided the notification made it through the link layer stack, it should appear in the output.

After greater exploration of the software system framework, the application programming interface (API) and software mechanisms, it's possible use the message pump to signal whenever a packet is acknowledged. Upon receiving this signal, the device can then dispatch a subsequent packet. Figure 19 shows such a mechanism.

The experiment was repeated again, this time sending a new notification every time the previous one was acknowledged using the message pump signal. The results are shown in Figure 20

A capture of the notification data was taken over a period of approximately 46 seconds. Figure 21 shows part of the capture.

In the 46.060 second period a total of 55286 notification packets were captured, bringing the average throughput to

$$\frac{55286}{46.060} = 1200.30395137\ packets/s$$

For the 7.5ms CI used, the number of CE is 133.333 and the number of packets per connection interval is

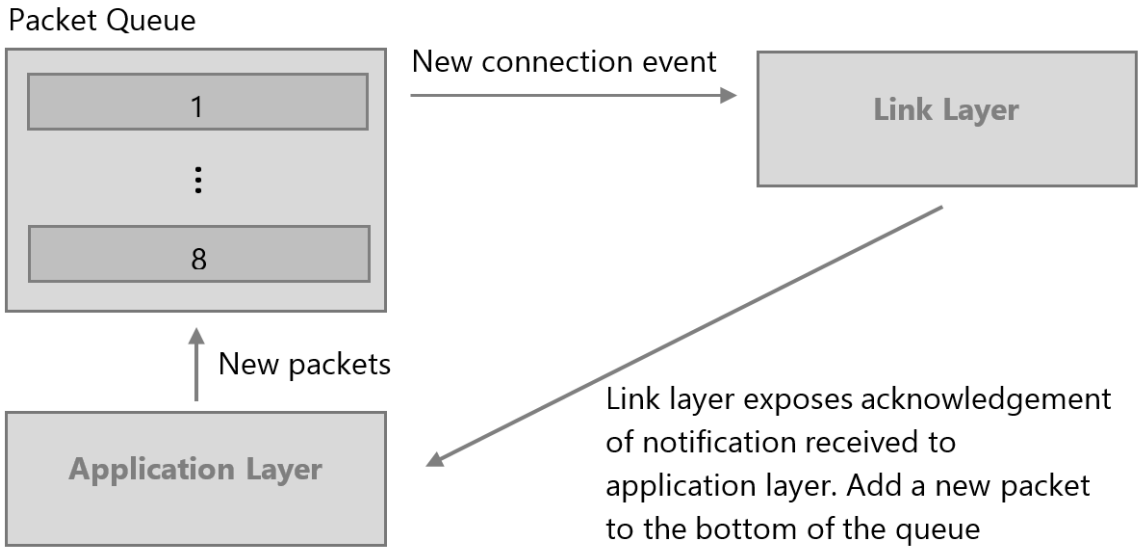


Figure 19: Upon packet acknowledgment

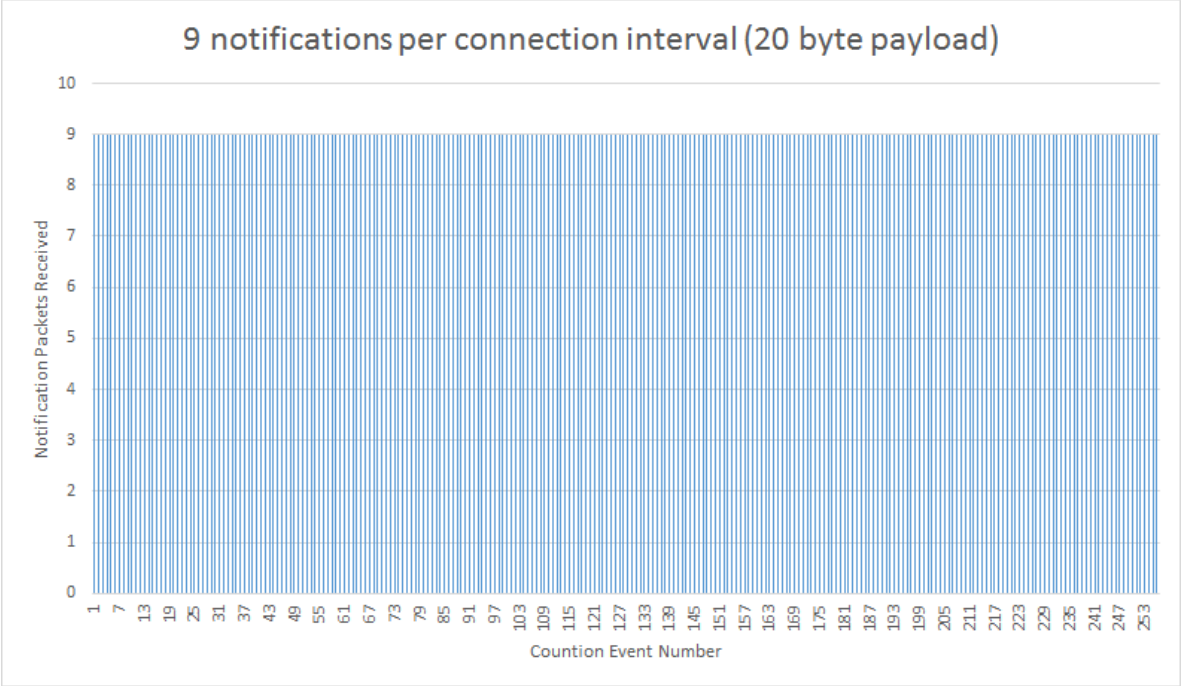


Figure 20: 9 packets being received per connection interval when through acknowledgment feedback mechanism

$$\frac{1200.30395137}{133.3333} = 9.0027\ packets/s$$

Therefore 9 packets per connection interval at 7.5ms. This corresponding to a throughput of 192kbps or over 80% of the theoretical maximum. A comprehensive explanation of the code used to achieved this is provided in the later section **??**.

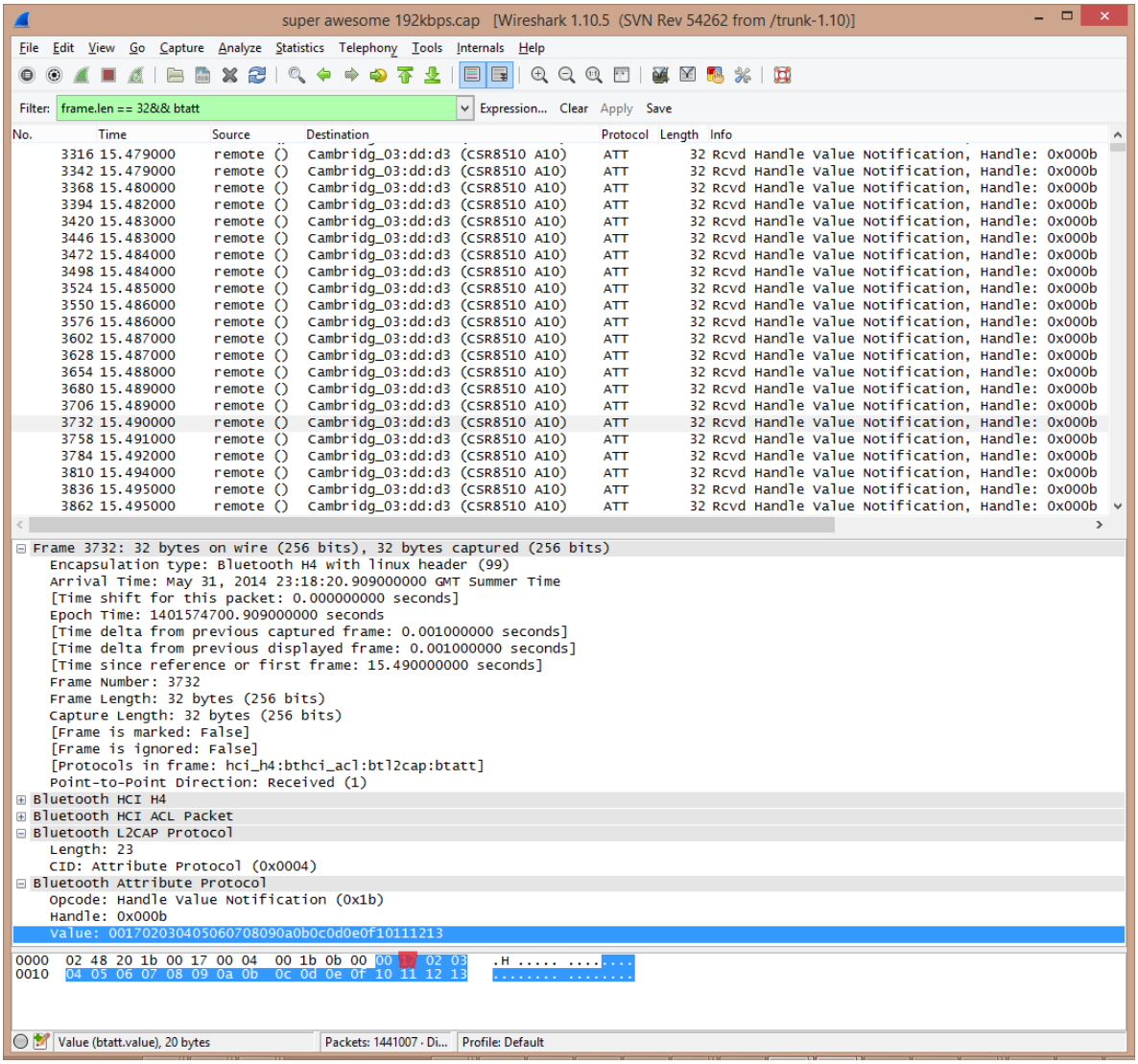


Figure 21: Packet capture. 20 byte data payload (blue) and connection event counter variable highlighted (red). Resolution of time stamps is to the nearest millisecond

4.1.4 CC2540 and CC2541

TI's CC2541 is part of the sensor tag package - a BLE one of the first, and arguably the most popular [need to CITE?] development kits. The CC2540 differs in that it's range and power usage are increased, though it's throughput is theoretically the same.

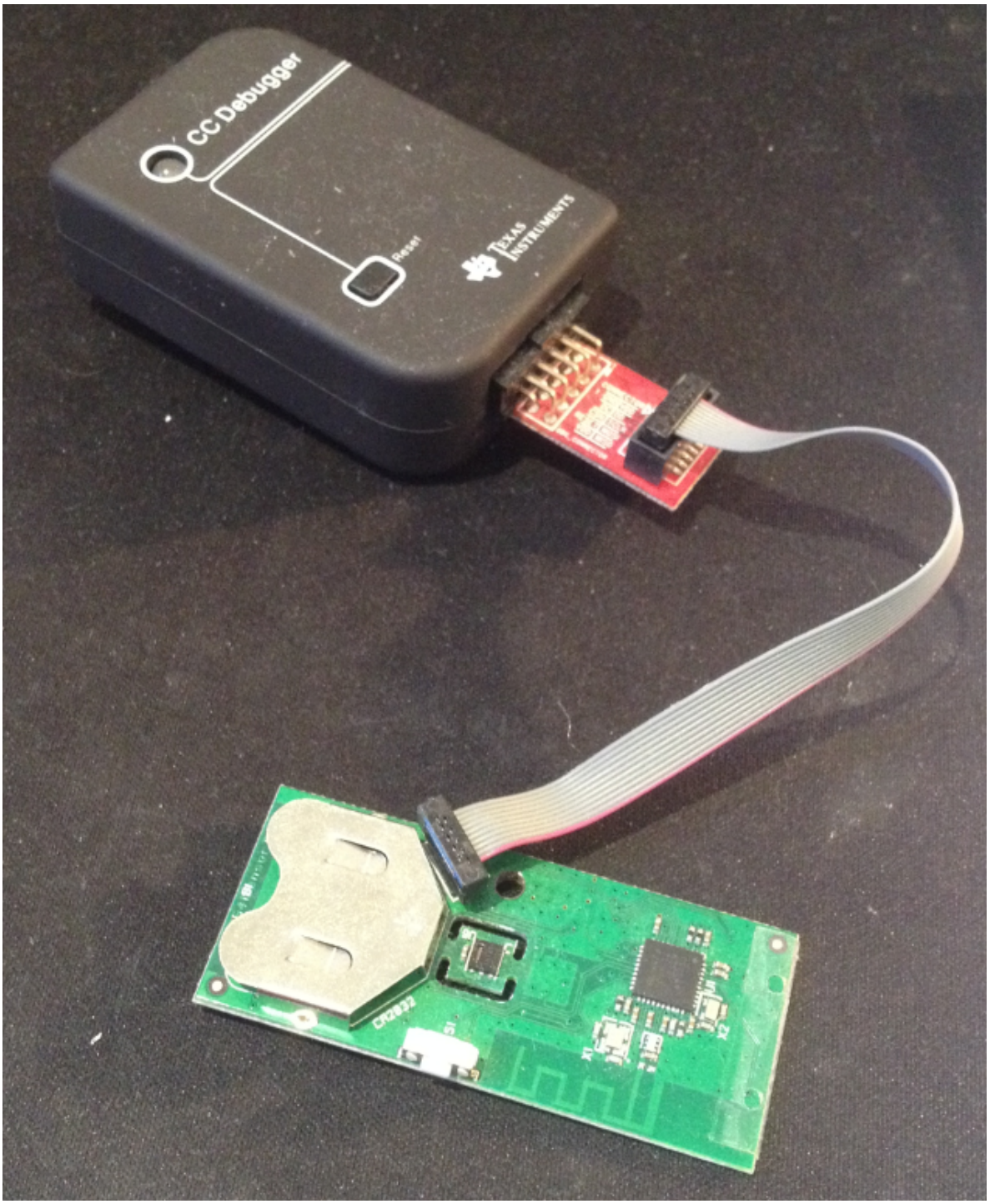


Figure 22: SensorTag hardware (contains CC2541) and programmer

Again, the associated white papers did not document the packet throughput per connection

event. Through contact with a TI employee [CITE], there is a hardware limitation of 4 buffers, limiting only 4 packets per connection event (each buffer contains 1 packet). Each buffer is 128 bytes in length, while BLE data packets can be a maximum of 41 bytes. Through using an exposed firmware (software) switch it was possible to increase the throughput slightly to achieve 5 packets per connection event by refilling the buffers within the connection event, once the packet has been link layer acknowledged. Again, this was discovered through a Texas instrument employee. This increased the average packets per connection event to approximately 4.6, bringing the total throughput between 10-12.3k byte/s.

The CC2541 (and also CC2540) is a full SoC - a programmable microcontroller (MCU) is packages togeter with the radio hardware. This has certain economical advantages, as common hardware and footprint is shared between the radio and MCU. However, the Nordic company Chipcon who originally developed the hardware (Texas Instruments has since acquired them), relied on the Swedish tool chain provider, IAR, for the development tool suite and chain. TI still relies up this tool software suite, haven't not invested in their own facilities to develop for the platform. A single software license is approximately 2000 USD. While IAR do offer a code-limited version, it is too small to contain all the necessary libraries. IAR also provide a 30 day trail, which is what is currently being used for evaluation, but proceeding with this chip for the prototype may be economically impossible.

For the measured packet througput, for 16 channels running simultaneously, at 8 bit resolution, the highest achievable detectable frequency is approximately 384 Hz.

The observant reader may notice that the BLE sniffer from TI is actually the CC2540 chip, which is reported as limited to between 4 and 5 packets per CE. However, when loaded with the firmware as a sniffer, it is capable of sniffing many more packets than this. After contacting a TI employee about this disparity, they reported that the sniffer uses a completely different stack:

> *The sniffer is capable of listening to any number of packets during a single connection event. Note that the CC254x sniffer SW is completely different than the CC254x stack.*

The engineer highlights that the software used on the sniffer is completely different than that used on the normal CC254x stack - this is presumably means the sniffer runs a reduced/highly specific operating system, to reduce overheads associated with a more generic system. This may mean the hardware is capable of sending large numbers of notifications per CE but either no supporting firmware or a acMCU powerful enough can drive the chip to these operational levels.

4.2 Batteries

There are many different battery technologies available. For the device a small, low profile low footprint, a lightweight battery is required. Given that BLE can consume upto 15mA at peak current draw, taking this as the upper bound means that for 12 hours of continuous use batteries with capacities of 180mAh should be considered. Although the radio will dominate, other parts of the circuit such as the Microprocessor will consume a small amount of power, so this figure is arbitrarily upped to 200mAh. Note that peak current is an unfair way to measure current consumption in BLE, however it is being used here as a worst case scenario.

Two batteries technologies which satisfy these requirements are Lithium-ion (the popular

CR2032 is typically advertised as an energy source with many BLE modules) or Zinc-air batteries with capacities of 600mAh cheaply available. Common voltages are 1.4V to 1.65V (radios require between 1.8 -3.6V). However the batteries are light, less than 2g each (one CR2032 weighs, on average, over 3g), therefore they can be combined in series to provide a suitable voltage and a large capacity (over 1Ah). Further, this high voltage will allow the effective radio range of the BLE device to increase. If a Zinc-air cell exhibits damage, unlike most technologies, no dangerous chemicals are present. The downside to zinc-air batteries is that they must be vented to an oxygen environment (cannot be hermetically sealed) which in turn causes them to lose their charge over time relatively quick once exposed to air. Although the AEEG use case requirements will require the device to have lifetime of hours to days, not years.

Two Zinc-air batteries weighing in at 1.9g each, can supply over 1000mAh with a combined series voltage of 2.8V. In the worst case of BLE drawing 15mA, the battery lifetime is approximately 3 days. Recall 15mA is the peak current and BLE will never continuously draw this current, hence the lifetime of the device should be greater, depending on the throughput. acBLE was designed that current is never continuously consumed, and between radio events, the battery has recovery periods, extending the useful life of the battery.

Nordic semiconductor provides a battery lifetime calculator. Using a typical Lithium-Ion battery of 220mAh at a 7.5ms connection interval, yields a lifetime of 163hours. This makes sense as the current average current consumption is slightly above 1.3mA and the capacity of the battery is 220mA. Accuracy of this model will require vertification in the power analysis of the AEEG, but the simulation looks sensible. Figure 23 shows a notification simulation of the EEG service being sent continuously every 7.5ms. This model is only for 1 notification per connection event. If

$$\frac{7.5ms}{0.676ms} = 11.094\ notificationpacketsperconnecitoninterval \frac{163hoursfor1notificationperconnectioninterval}{11.094packetsperconnectioninterval} \approx 14.7h$$

This should be an overestimation of the power requirements, as there is a fixed cost payed for waking the MCU, which would be amortised for higher throughputs. Further in the model above the idle time contributes to the power consumption, but at full throughput, there will be little idle time. Finally, running at full throughput is not necessairly the end used case. Running at lesser throughputs will have a dramatic effect on the power consumption.

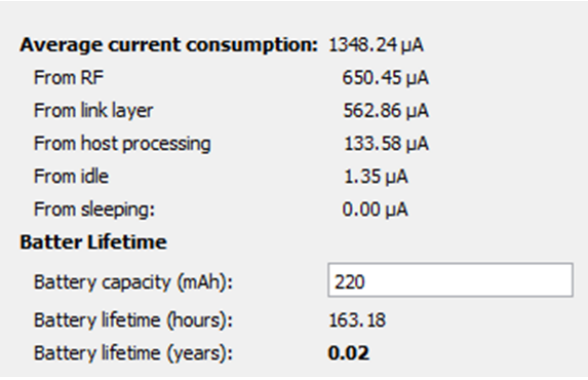


Figure 23: Power consumption and lifetime calculator

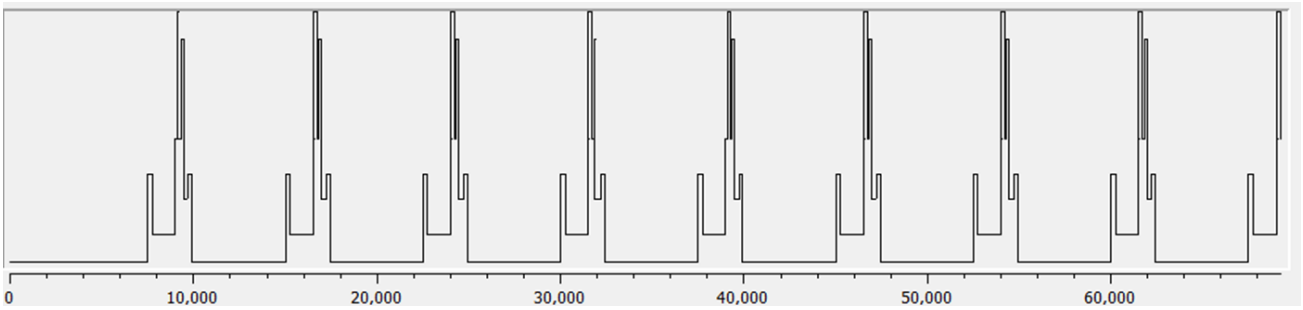


Figure 24: nRF8001 waveform (1 transmission per connection event

Figure 23 - Simulation plot of notifications being sent For 2 zinc-air batteries of 500mAh capacities, this model predicts the device lifetime under nominal operation begins to reach 10 days. An intelligent estimate to how this module is generated is linked to the throughputs calculated at the start of the Radios section. As each bit represents 1 micro second it is very easy to integrate for set parameters the time over time the current profiles to achieve an estimate of power usage. Max drain current is another important concept to consider. Batteries can only safely deliver a maximum amount of current. If the battery is stressed to deliver more current that it is rated, it will degrade the overall lifetime of the battery.

4.3 Microcontroller

There are a couple of options here. Often radios are provided as a system on chip (SoC), whereby the microchip contains not just the radio but also a fully functional microcontroller unit (MCU). Often these devices will consume more power than just the host and controller, but the system as a whole will typically consume more power overall if the application is run on an off chip MCU.

Figure 15 - SoC (a) and a typical 2 chip solution (b) (Heydon, 2012) It really comes down to the application, as having the application on an off-chip MCU may mean that power is saved overall as other features such as ADCs, SPI interfaces and other peripherals can be utilised and integrated into the application without having to run a separate application on both the external MCU and radio chip, which leads to increased overhead, complexity and power.

For development purposes the Arduino family of development boards, which sport many different processors seem like an excellent for development. These processors have many features such as GPIO, inbuilt ADCs, dedicated SPI pins, etc. Further, there is an extensive documentation, large knowledge base and a very active online forum. Of particular interest are the earlier Arduino models which utilise the very low power Atmel 8-bit family of processors. A further plus is that these processors are available in both DIP and SMT packages, meaning they are suitable breadboard prototyping. The ATMega328 seems like a particurly good choice, as a balance between support, power and simplicity.

Currently, Texas Instrument's SensorTag have been investigated along with Nordic semiconductors nRF8001. The former is a SoC solution while the latter only implements the host and controller layers. Implementing an application layer is no small feat, and fortunately it was possible to make

contact with someone who has developed libraries to drive the chip using an Arduino microcontroller. Please see the section titled Experimentation and Current for more information.

Currently the general feeling is towards using an Arduino micro controller simply due to the ease and rapid nature of development. Some components (an particular ADC and the nRF8001) have been tested and confirmed to work with the device.

4.4 Memory

A hard specification is the real time update of the EEG signals. Long term storage is not considered real time. NAND flash chips Further, high capacity memory modules come in small packages, some requiring factory assembly.

An option to achieve large storage capacity without difficult packages is to use popular SD cards. SD cards, and the variation, microSD cards are extremely easy to communication and an experiment was performed with an Arduino to discover power , However, during reading and writing the current consumption is between 25 to 100mA. From reviewing [san disk cite], a write can require 250ms. Assuming data is written in blocks, the average current consumption attributed by the NAND is 2.5mA.

4.5 Analogue to Digital Converter

Acquisition of EEG signals from the front end consists of using an analogue to digital converter (ADC). Many BLE SoC contain ADCs, though, for example the CSR1010 are only capable to gathering 700 samples per second. Operating the ADC requires a high speed controller. For high acqutisition rates (outside the standard operating range of EEG signals), the MCU will likely need to remain in an idle (active) state. For example, it takes the CSR1010 2.2ms to power up from a deep sleep state, meaning for acqisition rates over 450Hz, the chip would be required to remain in an idle (active) state as powering down to a lower power would miss samples.

The CSR1010 has an idle current of approximately than 1mA at 3.3V. The Arduino Uno micro controller is the ATMega328P. While the ATMega328's claims to consume only 0.2mA at 1.8V, this occurs when the device is clocked at 1MHz. This speed may not be sufficient to run the application layer with the responsiveness required, and the MCU is normally clocked at either 8MHz or 16 for most applications. Increasing the clock rate increases the current consumption. For the radio circuit and battery constrains, the expected voltage is expect to be 3V. While a voltage regulator can be used, there will be losses in conversion. Further power is required for running the hardware to enable communication between the radio and ADC. Running the ATMega328 at the same clock rate as the CSR1010, at 3.3Vs the report active (idle) current consumption is

The ATMega328 contains inbulit ADCs capable of suitably high sample rates, however it was found running these increases the current consumption by atleast 1.5mA.

All BLE chips tested without the application layer, was found to exhibit a low throughput. Given this, the only BLE radios worth running already contain a full SoC, whereby the MCU is

capable of communicating with an off-chip ADC. This and the power, complexity (communication and technology) and footprint size of using multiple active components, it was decided the best course of action would be too use an off-chip ADC connected to the BLE SoC.

The most obvious and popular protocols for interfacing with an external ADC are Serial Peripheral Interface (SPI) and I2C. The microchip MCP3008 has a resolution of 10 bits and uses the SPI protocol to interface. While confirmed to work with the ATMega328 MCU during preliminary prototyping (a dual in-line package (DIP) package was available), neither the CSR1010 nor the CC2541 support native SPI, containing no generic SPI driver libraries either. The protocol would have to be created in the application layer through modification of the programmable input/output (PIO) pins, commonly known as "bit banging". Achievable throughputs would not be known until implemented, leading to a development risk in the meantime.

The CSR1010 supports I2C natively along with provided a generic driver at the application layer. The most suitable device found was the Analogue Devices' AD7997 10bit 8 channel I2C ADC. The device can operate at different I2C speed, 100KHz and 400KHz. While there is the risk of interface issues, the level is less than that of the MCP3008, and believed tolerable. It is also highlighted that the correct working of the ADC is not crucial to the project's success. This, and the fact that no analogue front end is available.

4.6 Analogue Frontend

The biopoentials measured on the scalp required analogue filters and amplifiers to extract useful information. Ultimately the ADCs should interface with this frontend. At the time of writing, the circuits and systems group at Imperial College have recently taped out a full custom silicon analouge front-end design for manufacture, however these chips will not be available for use before the project deadline. Therefore no analogue front-end electronics will be present on the prototype board.

4.7 Component Selection

The most important decision for this project was the choice of radio device. While the CC254x showed promise as with enough hardware and firmware understanding, it may mean the device could approach the theoretical limit. Infact, recently (after initial testing) a technical article was created on TI's website [6] showing evidence of the device being capable of reaching high throughputs in league with other competiting devices. However considering the time, complexity and uncertainty that it can reach throughputs greater than competeting devices and the unavailability of an inexpensive tool chain renders this device undesirable. The CSR1010 is supplied with a free tool chain and IDE, as CSR engineers confirmed experiments with the device operating with high throughputs. The package of the radios were both similar, being of roughly the same size quad flat no-leads package (QFN) type, which while reported outside the department's PCB production capabilities, was still considered of sufficiently large for in-house prototyping. Therefore the CSR1010 was the chosen device for the prototype.

Considering the requirement for real time throughput, footprint, manufacturing and assembly

challenges, power consumption and added design complexity, there will be no use of NAND memory in the immediate prototypes, except that used internally inside a MCU or externally as a bootloader (i.e. CSR1010 device requires a small external storage unit which contains the program code, loaded at power on time).

For the EEG signal acquisition, use of an off-board ADC module, the AD7997 seems the most sensible choice due to power and overall time and complexity minisation. The prototype will contain two devices to acheive 16 channels and interface with the MCU (CSR1010) over the I2C bus.The choice of power source is still open between Lithium-ion and Zinc-air. To achieve longer lifetimes, it's most Zinc-air is the final choice as the device's energy source.

The final component selection for the EEG prototype is

- 2 xAD7997 - 8bit I2C ADC (two are required for 16 channels)
- CSR1010 single mode BLE device
- 2 x P675 Zinc Air batteries. Batteries of capacity 620mAh are readily available and inexpensive (two are required in series to achieve an operable voltage)
- AT24C512C-SSHD 512K Serial EEPROM flash for the CSR1010's non-volatile storage

The application will be developed on a tablet, due to the expected ease of development over other platforms. It is difficult to source either an Android and iPhone device capable of many notifications per CI, little information available and costly. The target platform is the Surface Pro, running Windows 8.1. While the Surface Pro contains an internal multi-mode wireless chip, BLE capable, it is not known how well this will fare for higher throughputs. CSR provided a dongle, which after preliminary testing, appears to be able to achieve high throughputs. This is another reason for using the tablet, as it contains a USB port. Further, developing for Windows 8.1 on a tablet is similar to developing for Windows Phone 8, hence limited effort is required to port the application to Microsoft Windows smartphone.

5 Specification

6 Implementation

6.1 Hardware

A challanging part of this project develping hardware. It was desired that the device should be small and light so it can be shown into a EEG hat and worn by the user with no discomfort. Both the schematic capture and layout was done within the **CAD!** (**CAD!**) software EAGLE. To achieve a small size components were placed within close proximity of each other, and at angled orientatinons (tesselated). The design used 2 layers, but component placing was only done on one side to allow for easier assembly. However, the bottom layer contained the batteries and solder bridges for ease of access. Ultimately the final achieved size was 26.5mm by 33.75mm, and with the battery connectors, the total height is less than 6.5mm.

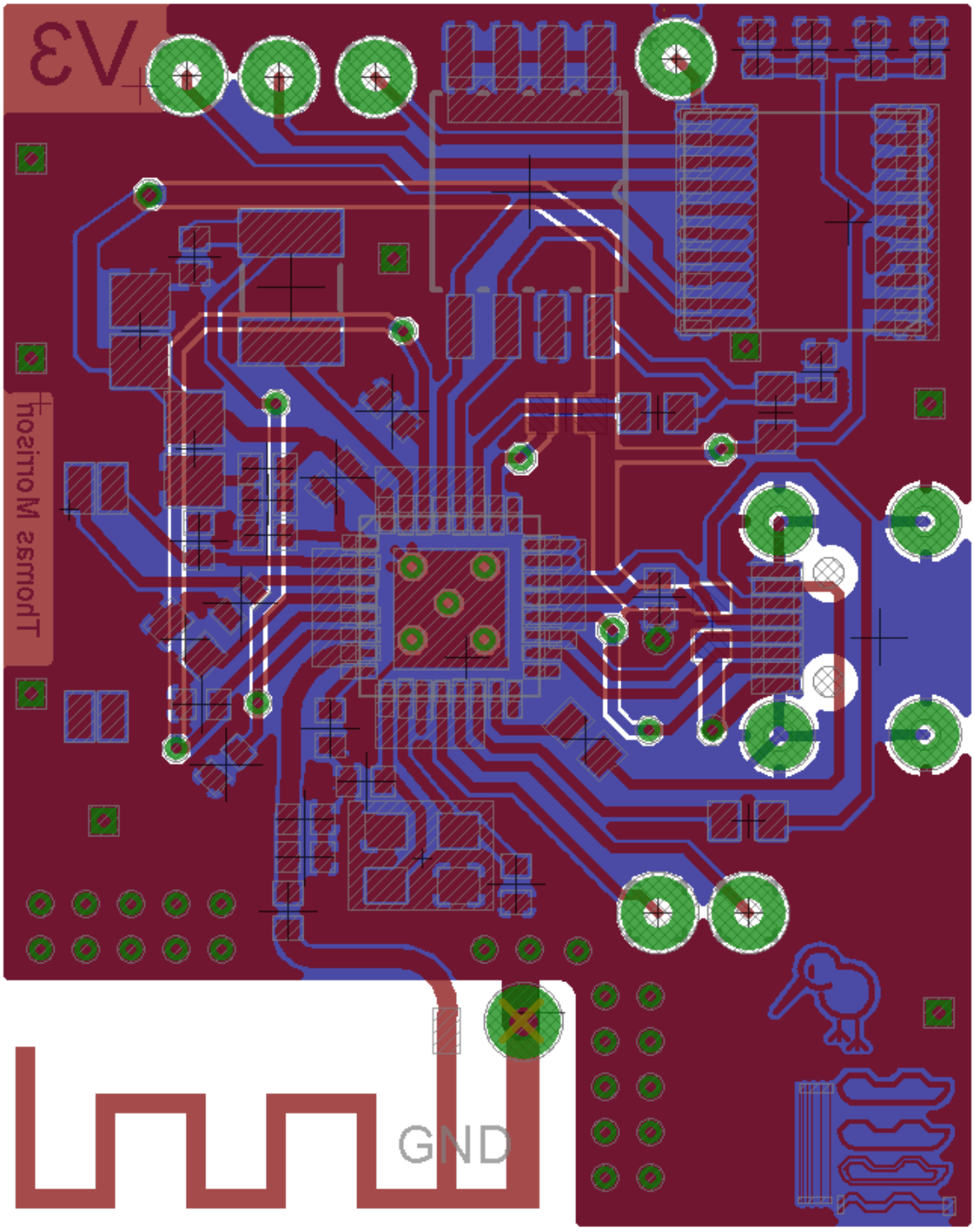


Figure 25: First PCB Design

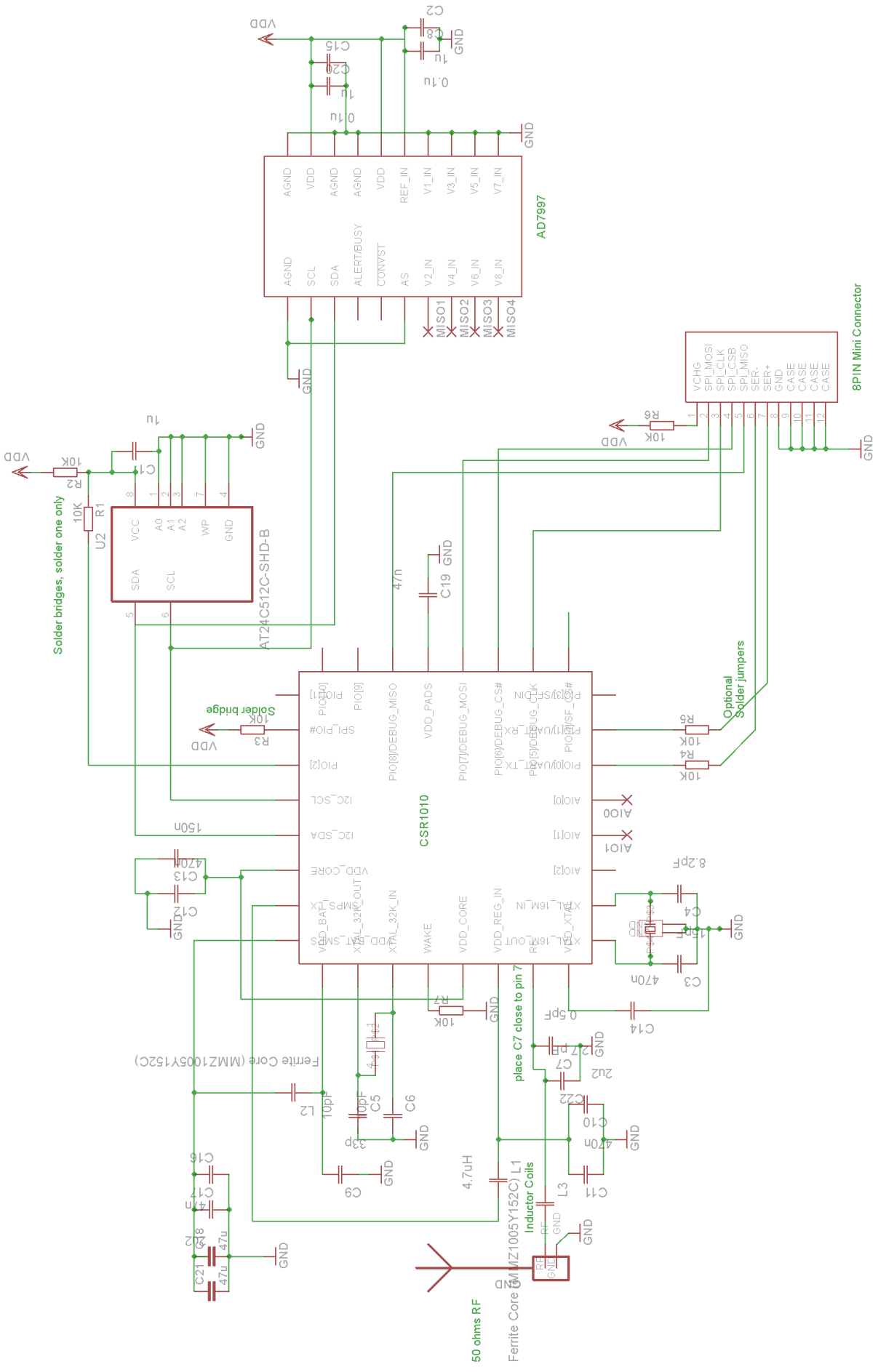


Figure 26: Schematic (showing only one ADC here)

The board was assembled by hand, and reflowed in an oven

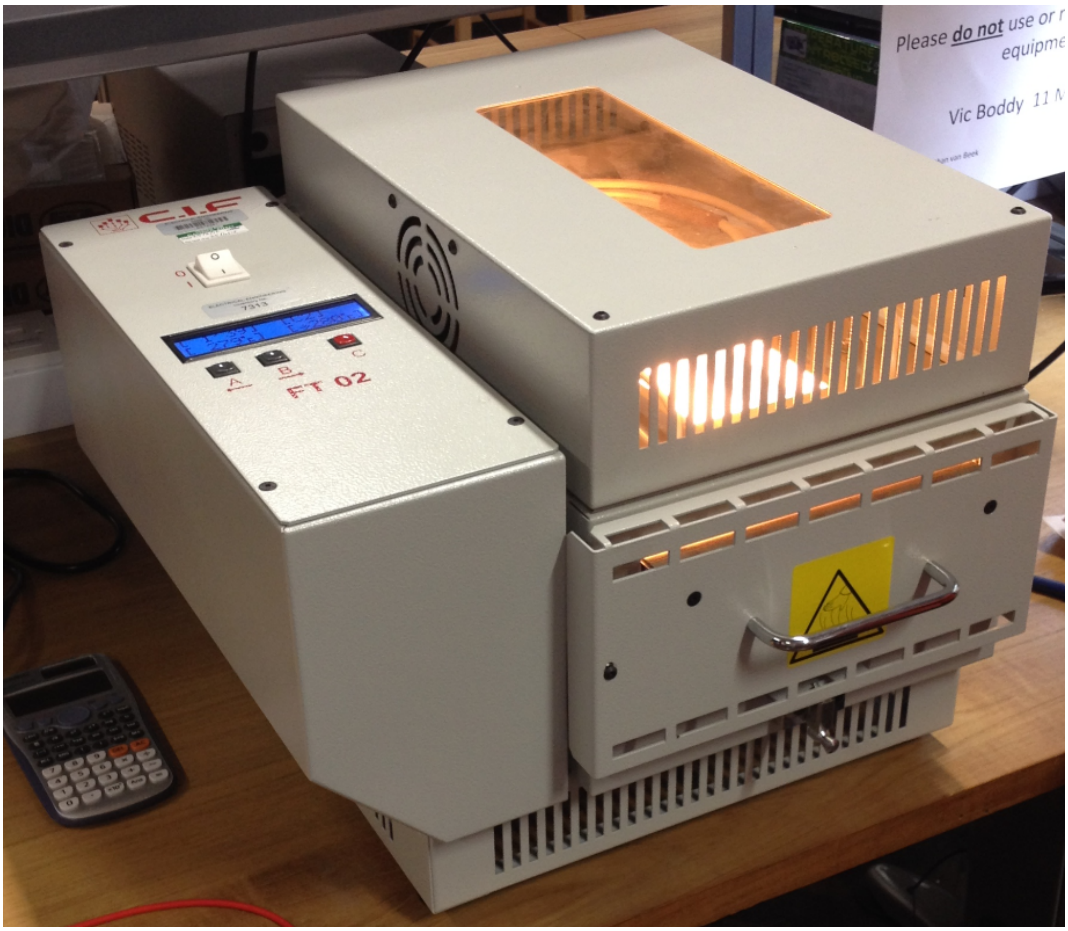


Figure 27: Reflow oven

Unfortunately, due to the size and complexity of some of the components, the oven did not give perfect results. Around the active component packets, the solder resist was remove to allow access for a soldering iron to "touch up" when the oven reflow failed to perform. To test both the process capability and variation, lines of decreasing thickness along with convex shapes were drawn in the bottom right corner of the board.

The first iteration of the board. Due to the process capabilities and materials used, the radio frequency traces were not impedance matched, and the device range was limited to approximately 7 meters within a building or 15 meters line of sight (LOS).

The connector used was difficult to source, being a variant of micro-USB, yet with more pins. The pin spacing was beyond the departments capabilities, but under magnification and with the right technique it was possible to make the connections. Unfortunately, with the **CAD!** package did not have capabilities to draw slots required for correctly seating the 8-pin connector. The connector had to be modified and filed down to correctly sit on the board. Even with super glue, small amouts of flexing from the cable, can give enough leverage to rip the connector off, along with the tracks, destroying direct connectivity. Two boards were damaged due to this and caused serve delays in developing the firmware for the AD7997 ADC. A workaround, although cumbersome, was to use a smaller, breakout board to connector to the mainboard to prevent any damage to the mainboard.

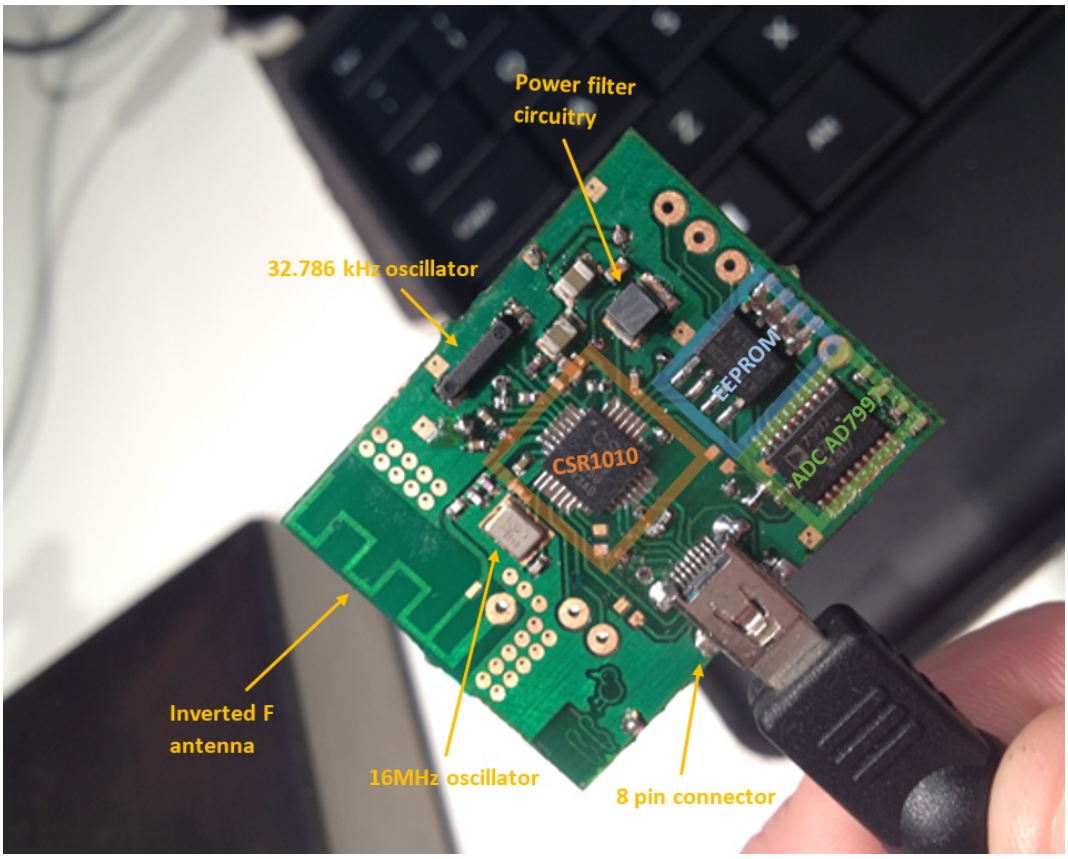


Figure 28: First (working) PCB iteration. Main components labeled

The second design added connections for Zinc-air battery holders, another ADC, channel and debug pads/test points, reduced weight and better RF properties. Further, the final design was produced in a professional off-site PCB house, making use of plated through hole (PTH) technology.

This board, while small, can be shrunk further in size

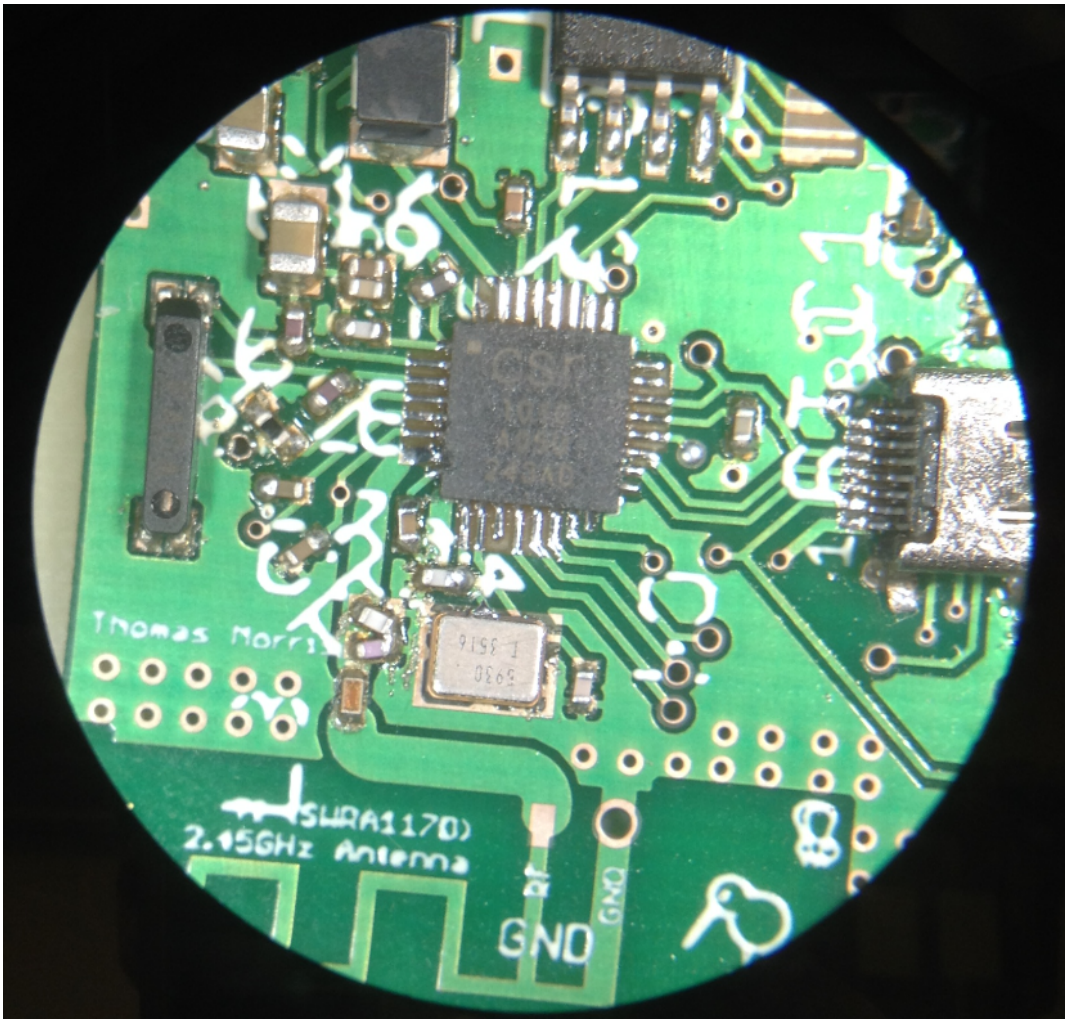


Figure 29: QFN touch up with a iron. Solder resist purposefully removed around small active packages, and straight tracks used to encourage solder capillary action

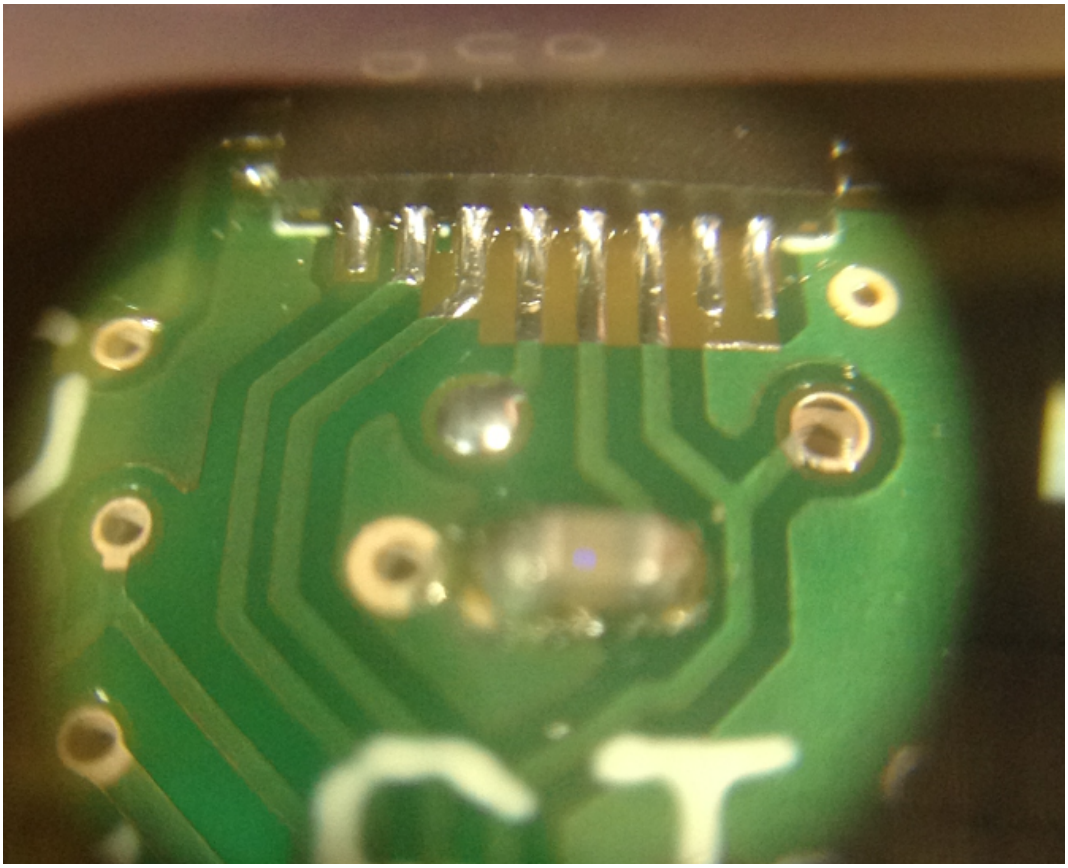


Figure 30: High magnification, shallow depth of field. Pads on side of QFN useful to ensure connection

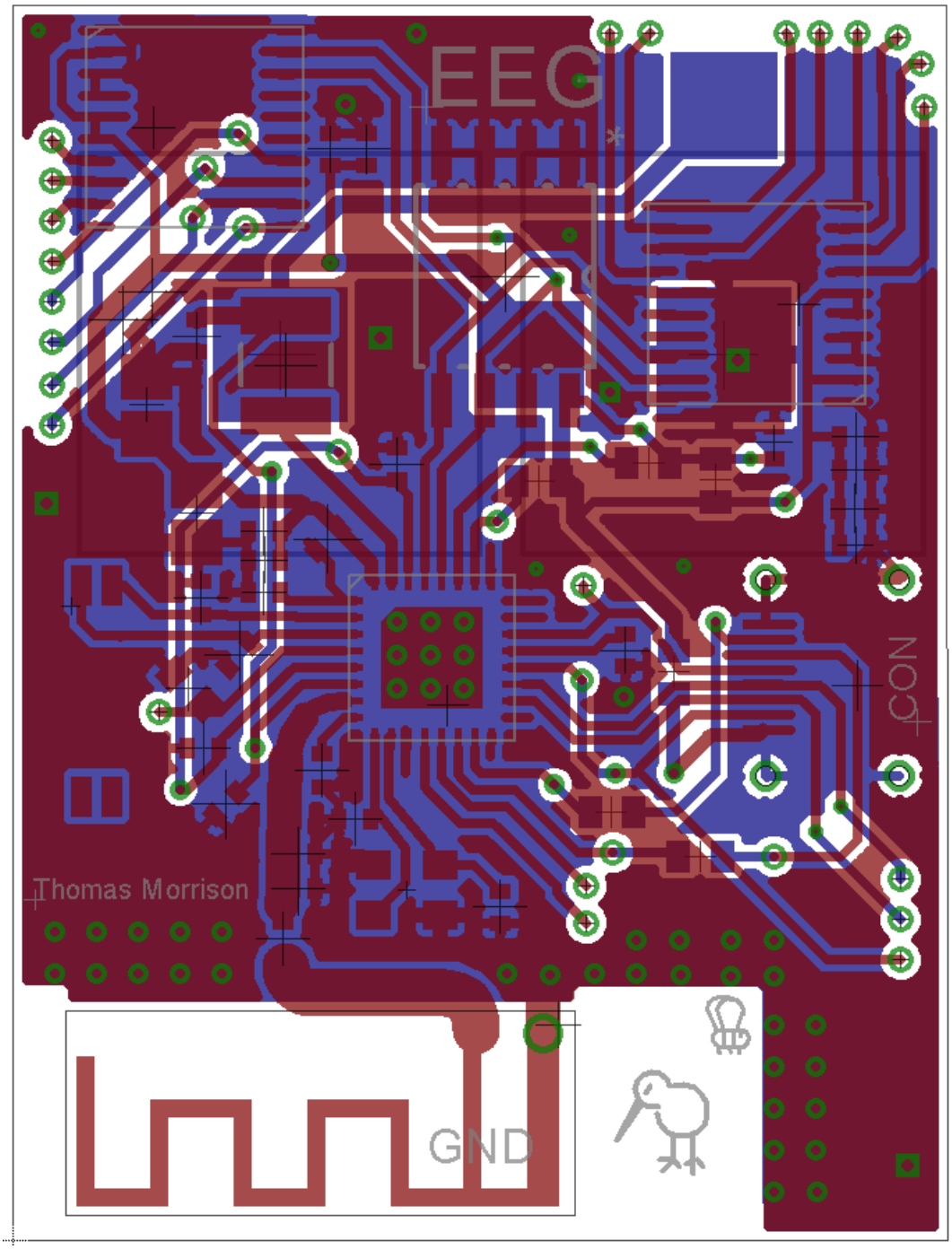


Figure 31: Final PCB Design

6.2 Firmware

The CSR1010 microcontroller contains a custom processor known as the XAP. acCSR provides the necessary tools to translate high level structured C-code into byte code instructions for the XAP processor

Fortunately, CSR provided example projects which contains the majority of the application layer functionality (see **??**) to handle basic device operations and radio. The MCU is a single threaded processor, yet the application design is event driven using a large central message pump(s) to service events and provide the control flow. Events can be of software origin (i.e. a flag set) or hardware invoked. Further, hardware interupts (such as PIO) can be supported outside the software message pump. Being event driven means the use of callbacks are used frequently througout the code base. CSR exposes APIs to communicate with the time critical and complex host and controller layers beneath.

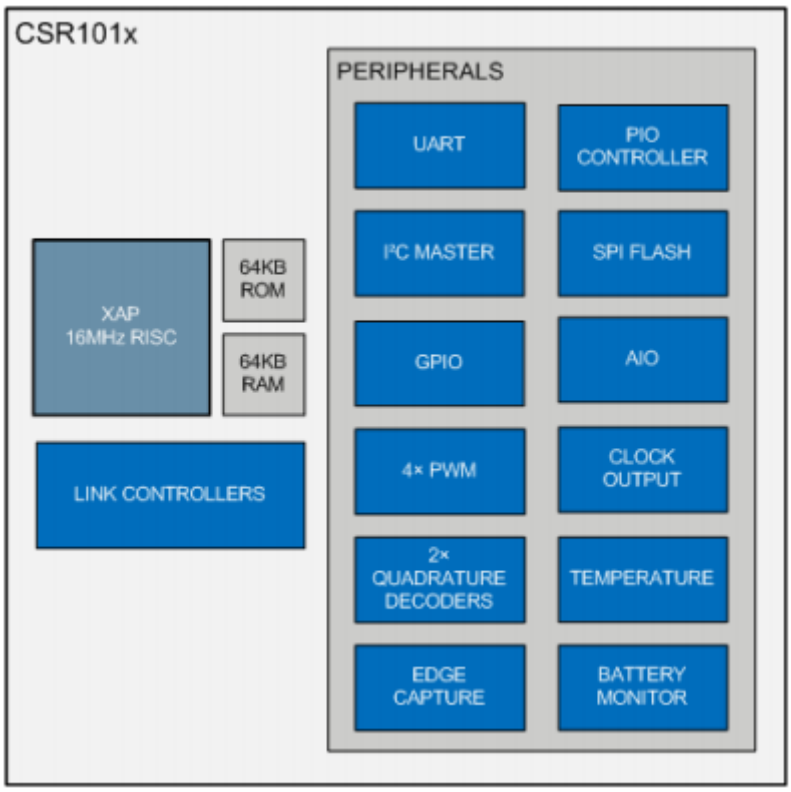


Figure 32: MCU overview [3]

When the device first powers up it enters the application layer through the AppInit procedure. This procedure initialises hardware and high level stack components such as the GATT server database. Once complete the control flow enters the system event message pump (AppProcessSystemEvent), which is called by the firmware to handle system level events such as PIO level changes such as configured interrupt from pins or wake events. This also handles low battery events. AppProcessLMEvent handles link manager related events from firmware. Such events include radio events (such as the acknowledgment or reciept of data) along with sercurity manager and GAP and G(ATT) messages. Making an insertion into this latter message pump was crucial for enabling the device to transmit more notifications once data was acknowledge (indicating the buffers were free). The link manager event caught is known as LS_RADIO_EVENT_IND, where the IND is short for indication -

despite the packet types used being notification, this event structure also corresponds to notifications. Finally, the timers run ontop of the hardware with microsecond accuracy and can be used as interupts. They are used to wake the device out of sleep ready for communicating or doing other work. A timer structure contains a pointer to a event handler which is fired upon timer exhaustion. Once the event handler returns, the chip will power down to a dormant state until the next CE or timer is available. It is highlighted that the application timers are not used for radio timings (i.e. synchronisation between devices happens in the base levels of the stack and/or use of dedicated hardware timers).

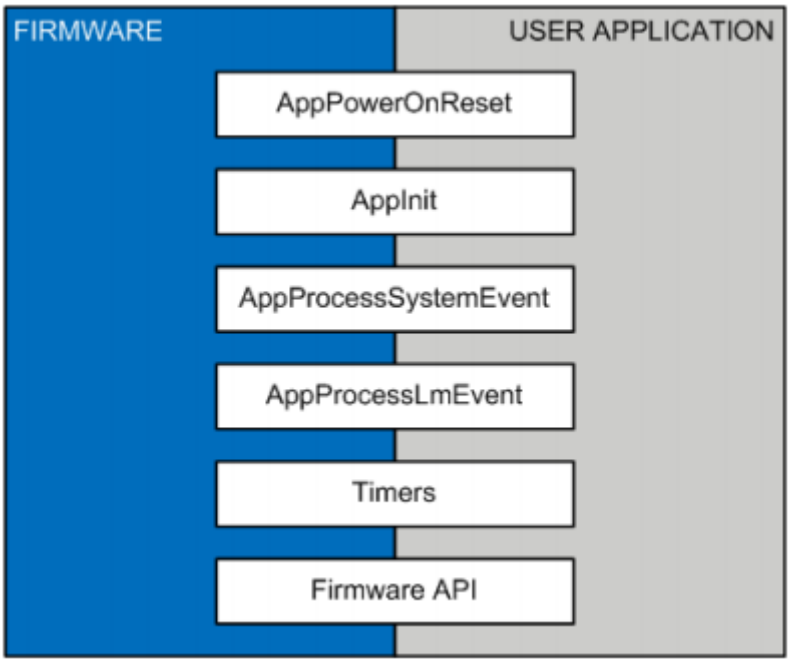


Figure 33: Firmware - Overview of the application layer control flow [3]

The GATT services and characteristics are stored in a flat-file database and defined as structures in JSON. The EEG characteristic is shown below in Listing lst:eeggatt. The EEG measurement characteristic provides a notification property which allows a device to subscribe to notification messages when generated from the device.

Listing 1: GATT database entry for EEG characteristic

```
#ifndef __EEG_SERVICE_DB__
#define __EEG_SERVICE_DB__

#include "eeg_service_uuids.h"

primary_service {
    uuid : UUID_EEG_SERVICE,
    name : "EEG_SERVICE",

    /* Characteristics support IRQ flag, thereby reads
     * and writes on characteristic configuration descriptor and notifications
     * on characteristic value are handled by application.
     */

    /* Notifications for the data*/
```

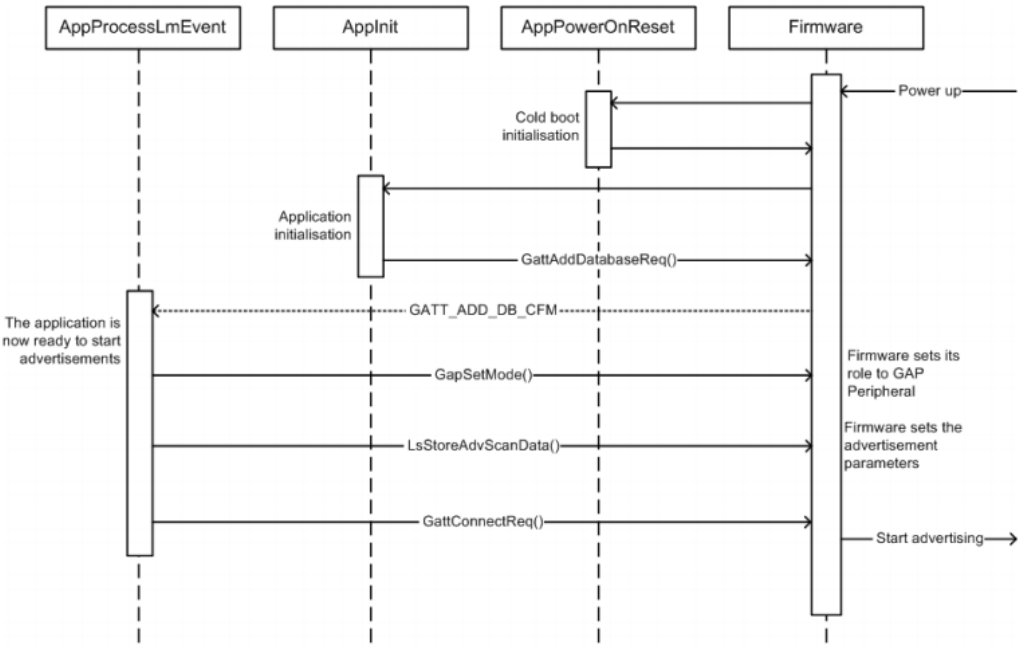


Figure 34: UML sequence diagram detailing power on to advertising packet events [3]

```
    characteristic {
        uuid : UUID_EEG_MEASUREMENT,
        name : "EEG_MEASUREMENT",
        properties : notify,
        flags : FLAG_IRQ,

        client_config {
            flags : FLAG_IRQ,
            name : "EEG_MEASUREMENT_C_CFG"
        }
    },

    /* Acquisition rate (i.e. frequency)*/

    characteristic {
        uuid : UUID_EEG_ACQUISITION_RATE,
        name : "EEG_ACQUISITION_RATE",
        properties : [read, write],
        flags : FLAG_IRQ,
        value : DEFAULT_ACQUISITION_RATE
    },

    /* Number of channels*/

    characteristic {
        uuid : UUID_EEG_CHANNELS,
        name : "EEG_CHANNELS",
        properties : [write, read],
        flags : FLAG_IRQ,
        value : 0x00
    }
},
```

```
48 #endif /* __EEG_SERVICE_DB__ */
```

Each service and characteristic requires a Universally unique identifier (UUID), which is a string of 128 bits and due to the enourmous space are very unlikely to appear elsewhere. Only 16 bits (known as the short UUID) are required to be set as the remaining bits are previously tied to the device address, which includes specific bit patterns allocated to chip manufacturers. Fruther some UUIDs are reserved for predefined services and characeteristics (see `http://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx`). The characteristic EEG service has the short UUID 0xEE0 (hexidecimal), while the characteristics share a similar short-UUID. The full service UUID can be seen in the User Guide section.

When the program is compiled the GATT data flat file is translated into a managed binary data structure and the tool chain creates events that can be referenced in the C-program. For example, EEG_ACQUISITION_RATE is associated with the event handle HANDLE_EEG_ACQUISITION_RATE, which is contained with the EEG service message pump AcqusitionHandleAccessWrite. Acqusition-HandleAccessWrite is in turn part of the more generic HandleAccessWrite that sits within the EEG service. Interestingly, this type of pattern is well suited to object oriented paradigm, however, the compiler (as with most embedded systems) was not C++ capabale.

Listing 2: UUIDs EEG service

```
1  #ifndef __EEG_SERVICE_UUID_H__
2  #define __EEG_SERVICE_UUID_H__
3
4  /*===========================================================================*
5   *  Public Definitions
6   *===========================================================================*/
7
8  /* UUIDs for EEG service and Characteristics*/
9
10 #define UUID_EEG_SERVICE   0x0EE0
11
12 #define UUID_EEG_MEASUREMENT  0x0EE1
13
14 #define UUID_EEG_ACQUISITION_RATE 0x0EE2
15
16 #define UUID_EEG_CHANNELS   0x0EE3
17
18 #define DEFAULT_ACQUISITION_RATE  0xFF
19
20
21
22 #endif /* __EEG_SERVICE_UUID_H__ */
```

To communicate with the off-chip ADC, the I2C bus was used. The datasheet for the ADC (AD7997) [4] shows the device has 3 modes of operation. Mode 1 uses edge triggering on the CONVST pin to begin a conversion. Mode 2, known as command mode allows a conversion to occur from a command sent over the I2C bus. The third and final mode is automatic cycle interval, which will cause the ADC to power up and take measurements periodically. The mode selected depends on the bits set in the control register. When enabling channels

Between all the code that provides the utility methods of dealing with data and firmware operations, the program is very simply - wake up periodically and sample data, perhaps store this data, then send it on.

6.3 Tablet Application

The highlevel tablet application was written for Microsoft's .NET framework platform in C#. The application relied upon the GenericAttributeNamespace (specifically $Windows.Devices.Bluetooth.GenericAttributeProfile$) class library to communicate with BLE devices. The application uses the new WinRT application design introduced into the Windows operating system since Windows 8. Writing for WinRT means the application can be easily The application archiecteure makes full use of object oriented programming (OOP), with an emphasis on the model-view-view model (MVVM) design pattern. The application was design to allow the user to be able to connect to a device, select the number of channels measure and the frequency.

The View is written in a markup language known as Extensible Application Markup Language (XAML). This is a verbose, but rich markup that allows a graphical user interface (GUI) to be highly flexible. It is what describes the view of the model (data) and binds to the view-model. The view-model, which is similar to the controller in the model-view-controller (MVC) pattern is the code behind (C#) which can be thought of as a "value converter", meaning it is tasked with exposing the model's data objects for easy cosumption and managability. This is subtly difference from the controller, which tells the view what to display and when. An example of the View-Model would be a collectional (e.g. a list of eeg measurements), that when mutated, automatically update the GUI. In the context of XAML and MVVM, this is known as data binding, and allows the View to easily consume the model.

In the context of a BLE service, the EEG service is described by the data model shown in Figure 36. The model contains EEG specific members such as the acqusition rate, channel map and the data for each channel. The model also contains some house cleaning members, such as the service and IsInitialised variables. As only one EEG service can exist, it made sense to implement a singleton pattern to allow easy lifetime and access from the model to the view-model. While singletons are often assoicated with poor OOP design, the use case here and simplicity makes sense. The SwapUInt16 moethd is required when writing data back to the CSR1010, as the device word size 16 bit, with big endianess (x64 is little-endian). The eegChannels data member contains . However, the method eegNotification can be used to provide direct access to the incoming notifications, acting as the callback from notification events

The GenericAttributeNamespace provided by the .NET framework only provides functionality on the work at the ATT level. The mechanism for interfacing with the BLE devices to to enumerate all connected devices then filter on the short UUID. Connected devices are devices connected outside of the application (i.e. through the Windows OS bluetooth settings from control pannel - Figures 38, 39. This will return objects for all devices with a service matching that of the short UUID. These objects can be further queries to find characteristics belong to this service along with the properties (read, write, indicate or notify). It then possible to read, write or assigned event callbacks from these characteristics. Attempt to write to a read only characteristic will result in an access denied, which will need to be handled to prevent program termination.

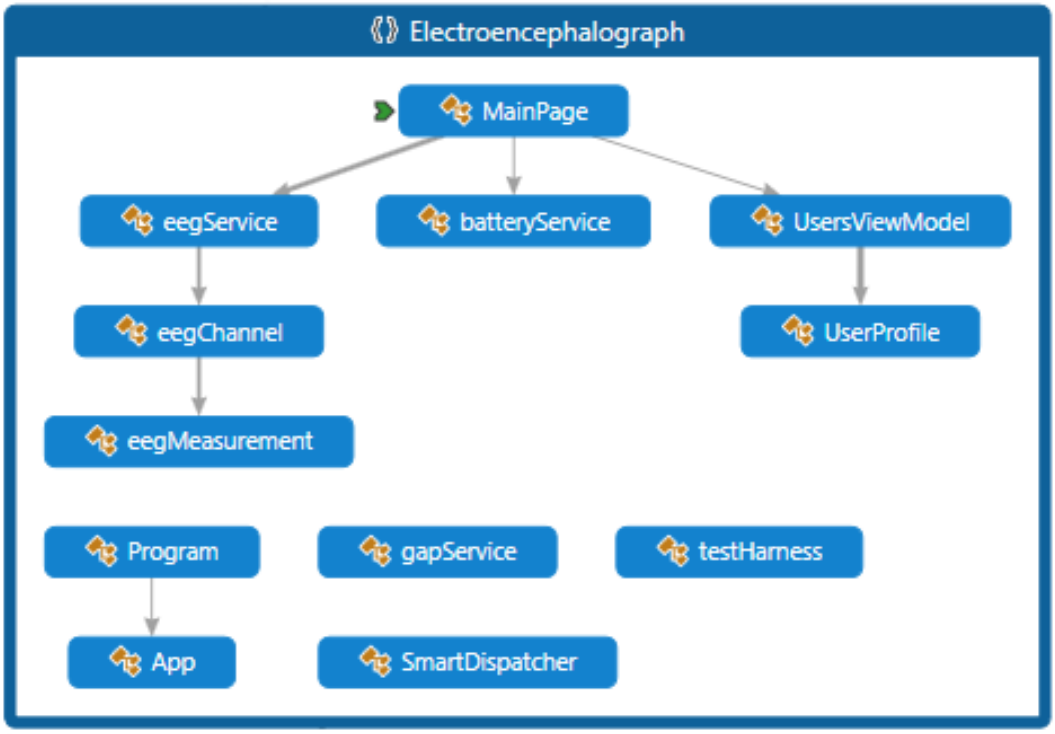


Figure 35: Dependencies

Listing 3: Connecting to a BLE device's EEG service

```
1
2           var devices = await Windows.Devices.Enumeration.
3     DeviceInformation.FindAllAsync(GattDeviceService.GetDeviceSelectorFromShortId(0xEEE0));
4
5           if (devices.Count < 1)
6           {
7               await new MessageDialog("Could not locate any EEG devices in the vinicity").ShowAsync();
8               return;
9           }
10
11          //By default connect to the first EEG service found
12          eegService.Instance.service = await GattDeviceService.FromIdAsync(devices[0].Id);
13    //Use long GUID with 16 bit short UUID (could of used just shortID)
14          var eegData = eegService.Instance.service.GetCharacteristics(new
                Guid("0000EEE1-0000-1000-8000-00805f9b34fb"))[0];
15    //Define call back function
16          eegData.ValueChanged += eegData_ValueChanged;
```

The applications makes extensive use of the task parallel library (TPL) for a fluid and responsive GUI. The TPL uses asynchronous procedures to add parallelism and conccurrency. This is particularly useful in this application, as the GUI thread needs to be constantly updated from incoming data without preventing usability. Despite use of the TPL further timing mechanisms were required to prevent interface lock up. When new data comes from the device, it will be transposed into the model. This will cause the data binding between the view and model to signal a change, meaning the view updates. Due to the high throughput, this mechanism can cause great delay, and hence, batching

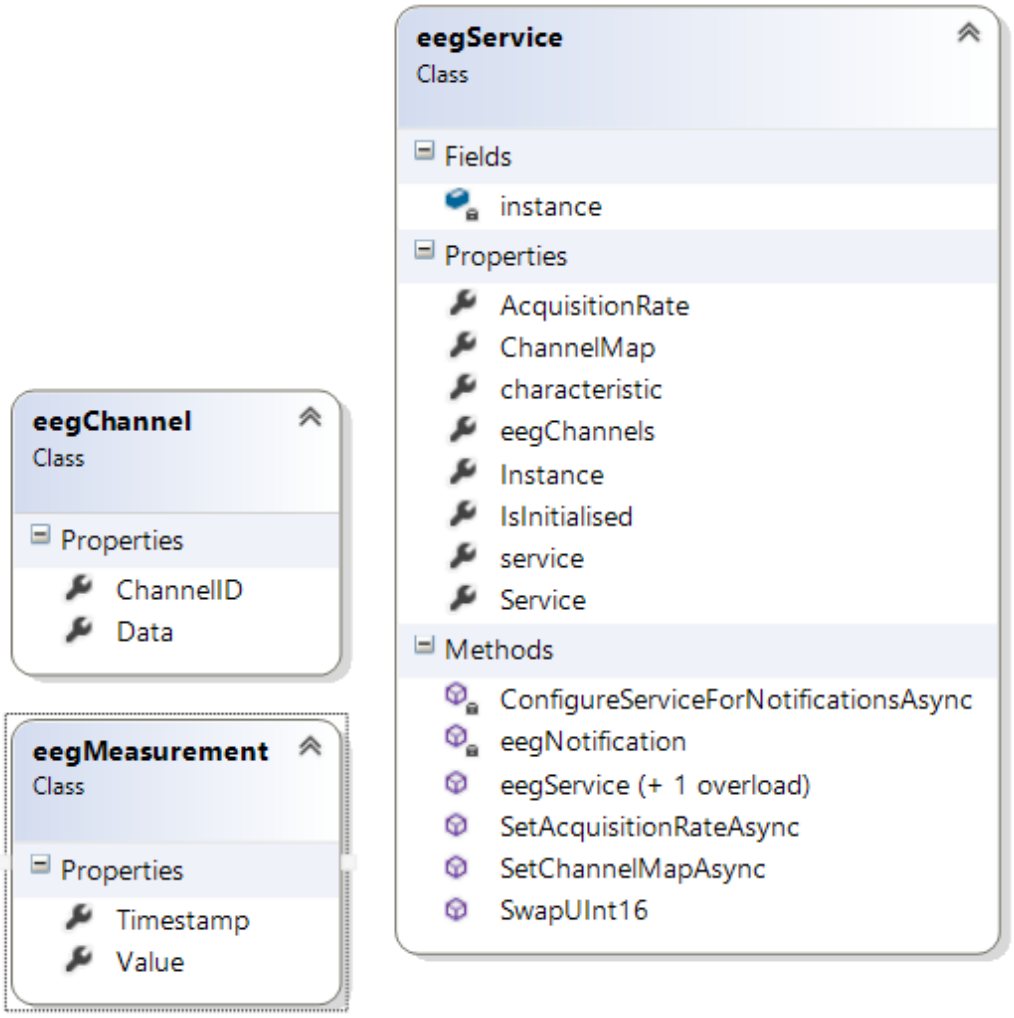


Figure 36: Data model (class diagram) for the EEG service

the updates is preferred.

Listing 4: Example batch update code

```
1
2  //For a single graph (can be called for each graph)
3        private void batchUpdate(ThreadPoolTimer source)
4        {
5
6                AddItem<FinancialStuff>(financialStuffList, lst);
7
8        }
9
10 //Pass an obversable collection (databound object between the model and the view), and a list of
       items to append to the collection
11       public async void AddItem<T>(ObservableCollection<T> oc, List<T> items)
12       {
13
14               //items to keep on the graph
15               const int maxSize = 500;
```

```
16
17              // Make sure it doesn't index out of bounds
18              int startIndex = Math.Max(0, items.Count - maxSize);
19              int length = items.Count - startIndex;
20
21              List<T> itemsToRender = items.GetRange(startIndex, length);
22
23              lst.Clear();
24
25               await Task.Factory.StartNew(() =>
26              {
27                  lock (oc)
28                  {
29                      oc.Clear();
30
31                      for (int i = 0; i < itemsToRender.Count; i++)
32                      {
33                          oc.Add(itemsToRender[i]);
34                      }
35                  }
36              });
37      }
38
39 ...
40
41  //Establish the timer object with a handler - 100 milli second update
42          periodicTimer = ThreadPoolTimer.CreatePeriodicTimer(new TimerElapsedHandler(batchUpdate);,
                new TimeSpan(0, 0, 0, 0, 100000));
```

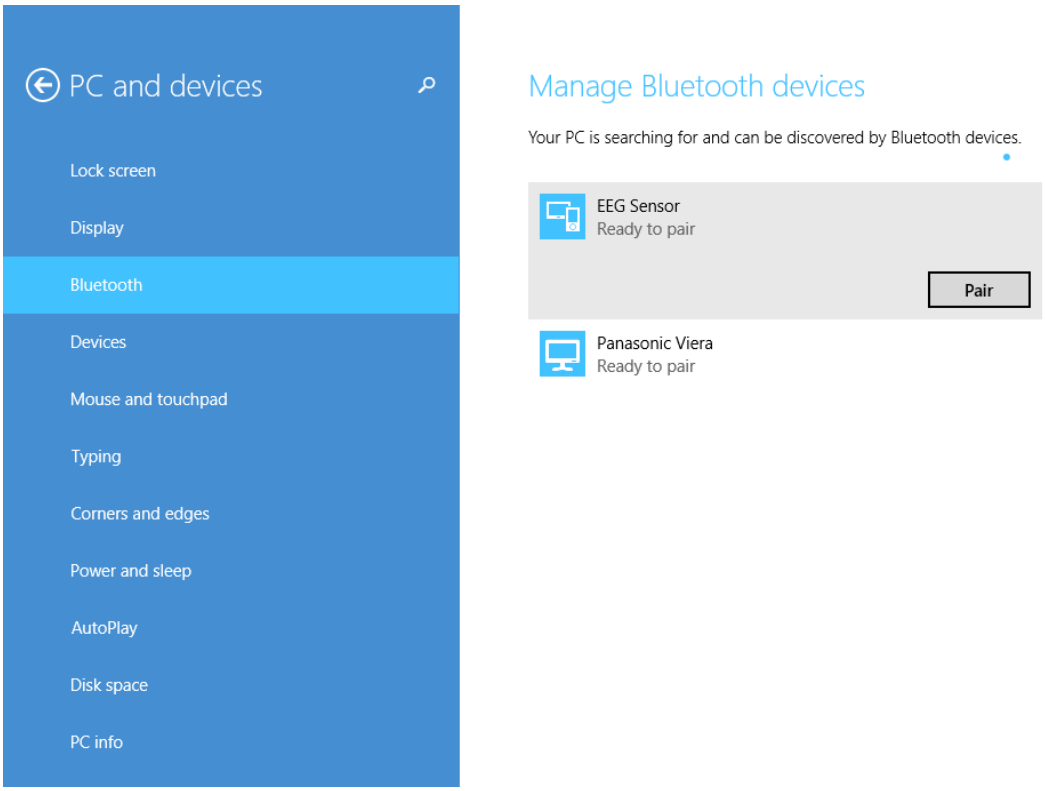


Figure 37: Searching for Bluetooth devices in windows

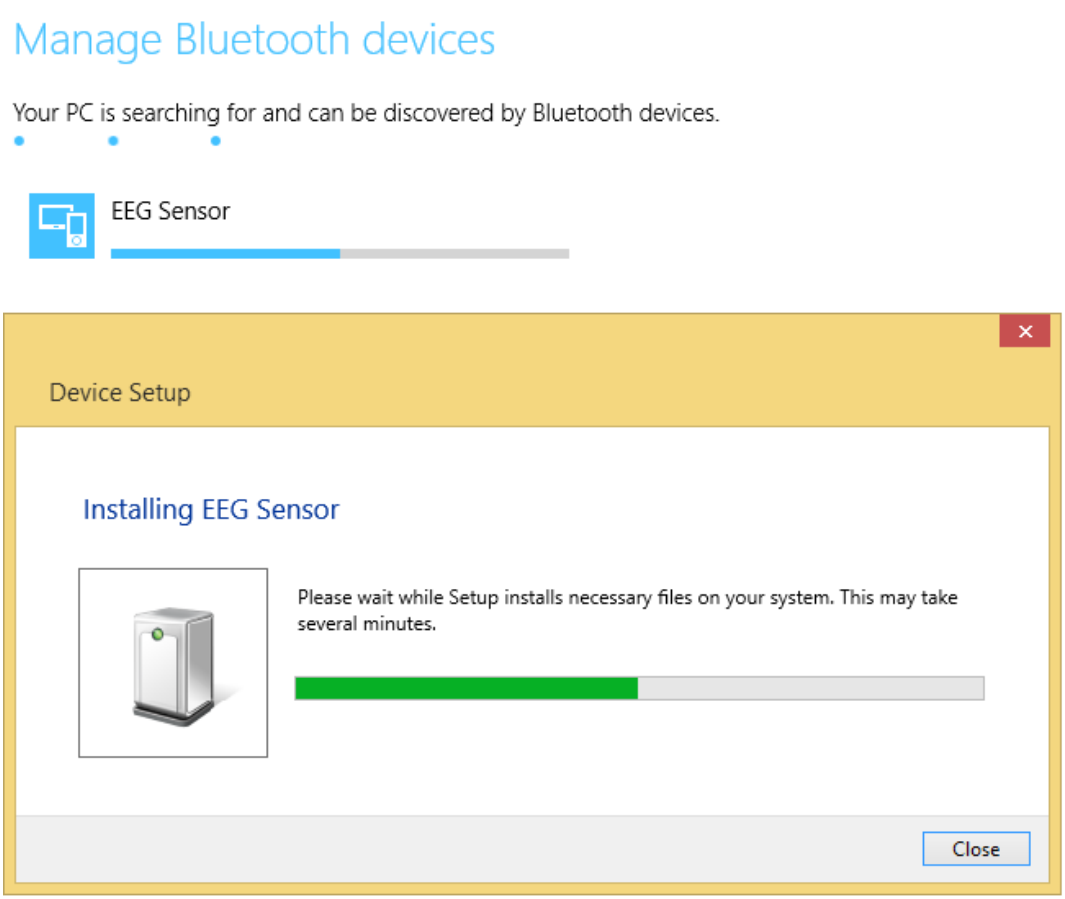


Figure 38: Connecting and installing to a BLE device

At the time of writing the application looks like

An issue was found whereby at higher throughputs, the order in which notifications are recieved become handled out of order. The reason for this occuring is that there is no guarentee when events are dealt with. Every time a packet is recieved, it will fire an event handler, however this will occur in batches. If the rate at which these batches is processed is greater than the rate at which new events arrive, then the events may no be handled in order. Unfortunately, there has been no fix for this, and it is assumed a current limitation of Microsoft's .NET GenericAttributeNamespace. When using the CSR stack and software development kit (SDK), the events are handled in the order they occur. Despite this causing issues in the highlevel application, for the remainder of this report it is ignored. Microsoft has since been contacted about the issue and are looking into it. It is understandable that they wouldn't of built the system for it (much like how some manufacturers only permit a limited amount of packets per CE), but even at smaller throughputs, notifications from the same service may be serviced out of order.

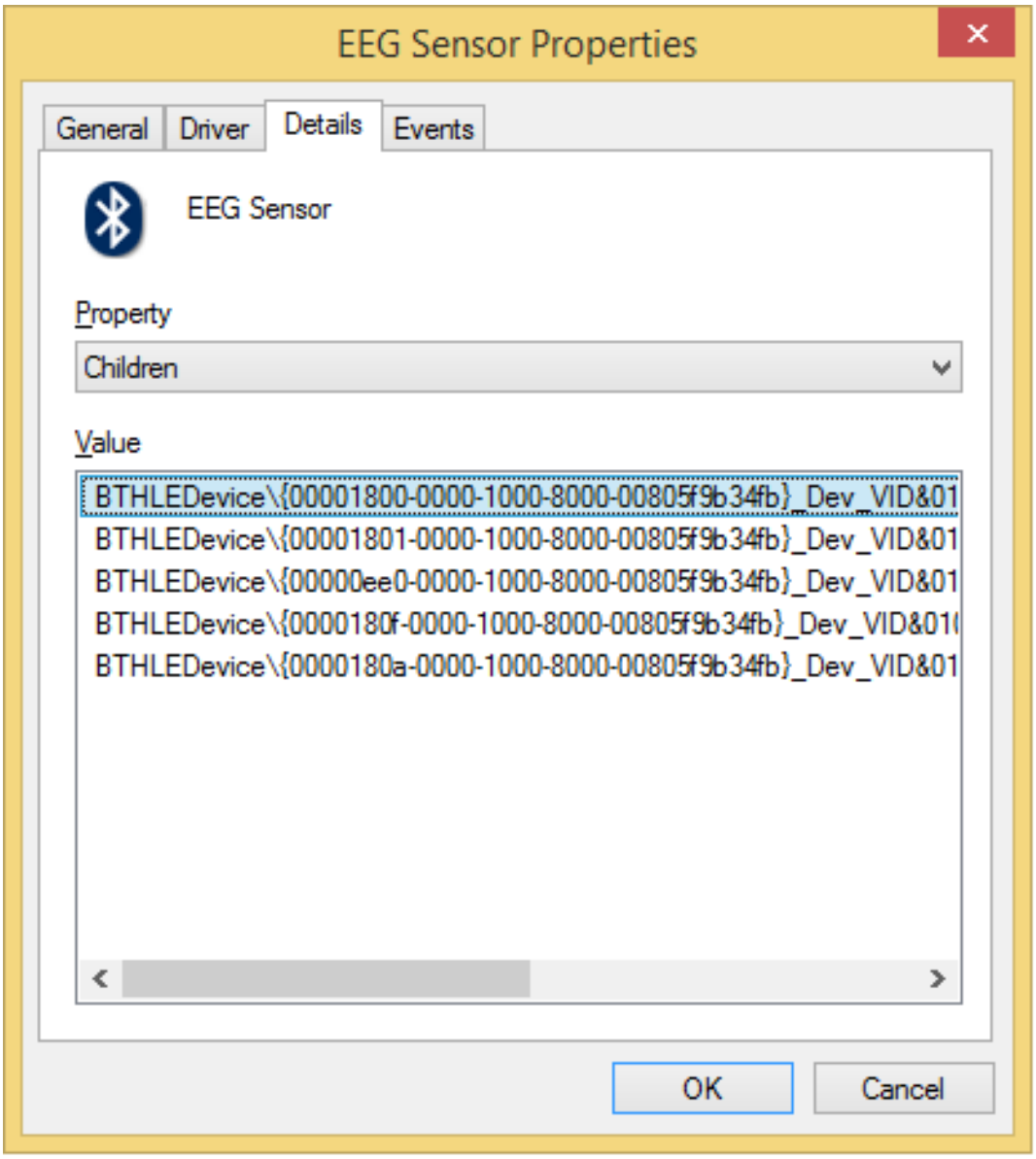


Figure 39: EEG sensor installed as a Bluetooth device. Long UUIDs for each service present on the device

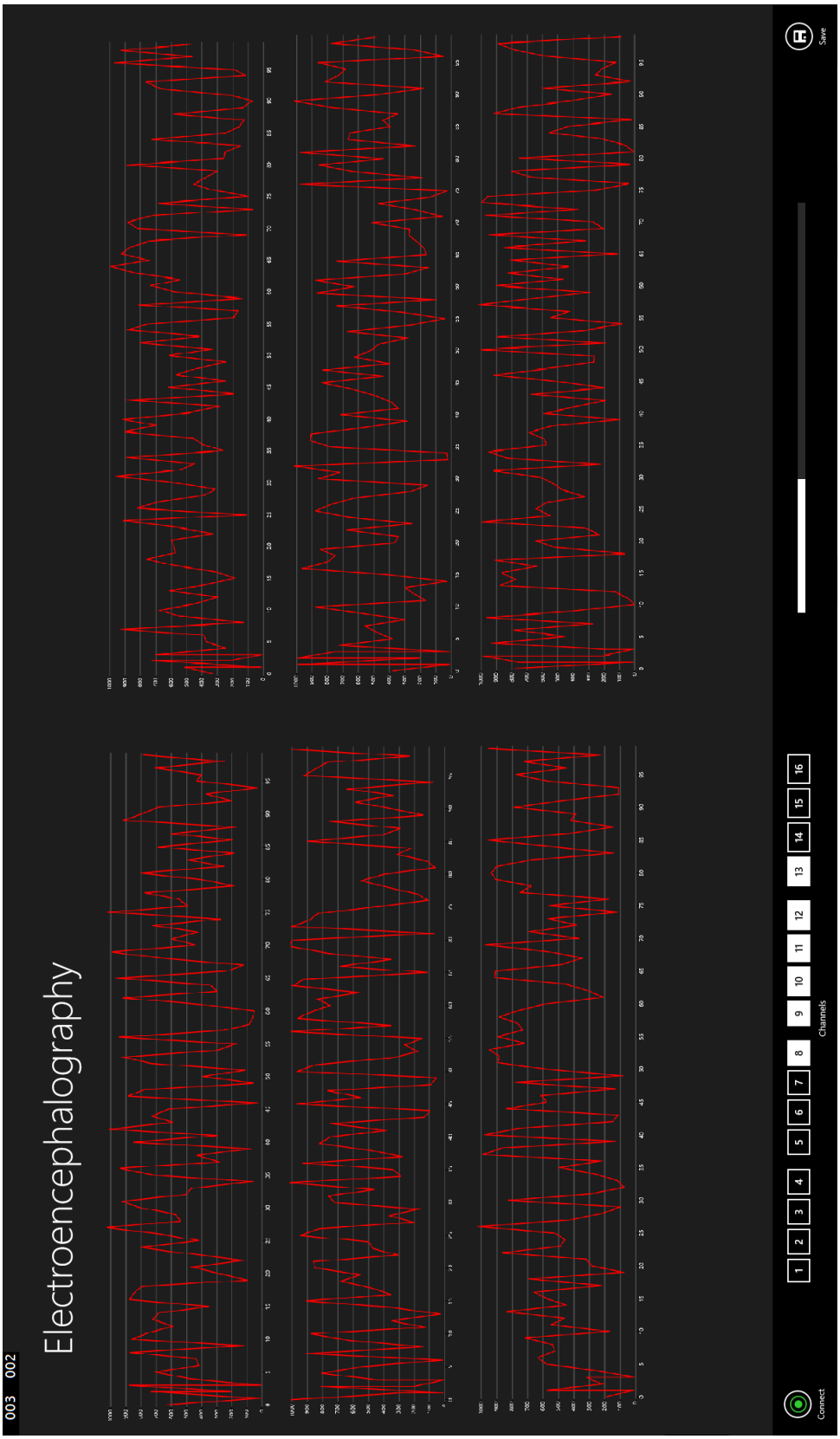


Figure 40: Application - data recieved with 6 channels selected

7 Results

May merge this serction in with evaluation. This section will be short

Weighing in at a total of

8 Evaluation

8.1 Throughput

In the sole pursuit of maximizing throughput, the idea to try and pack as many notifications into a CE is a superficial tactic. Taking the smallest CI of 7.5ms, upto $\frac{7500}{676} = 11.094675$ packets can fit into this interval. This 0.094675 remainder may seem small, and indeed equates to only 13 packets lost of the maximum throughput within a second, thus reducing the total throughput by less than 1%. This remainder changes with the CI and in the worst case, just less than a whole packet fits into the remaining space. The profile which desrcibes the "remainder" packets is formally written as

$$\frac{1250 \times x}{676} - \lfloor \frac{1250 \times x}{676} \rfloor$$

where 1250 is the CI unit, x the value, 676 the round-trip time for a maximum size data packet. A connection interval must be between 7.5ms and 4000ms long, in incremental steps of 1.25ms. Therefore x must exist in the range of 6 to 3200. This can be visually depicted as

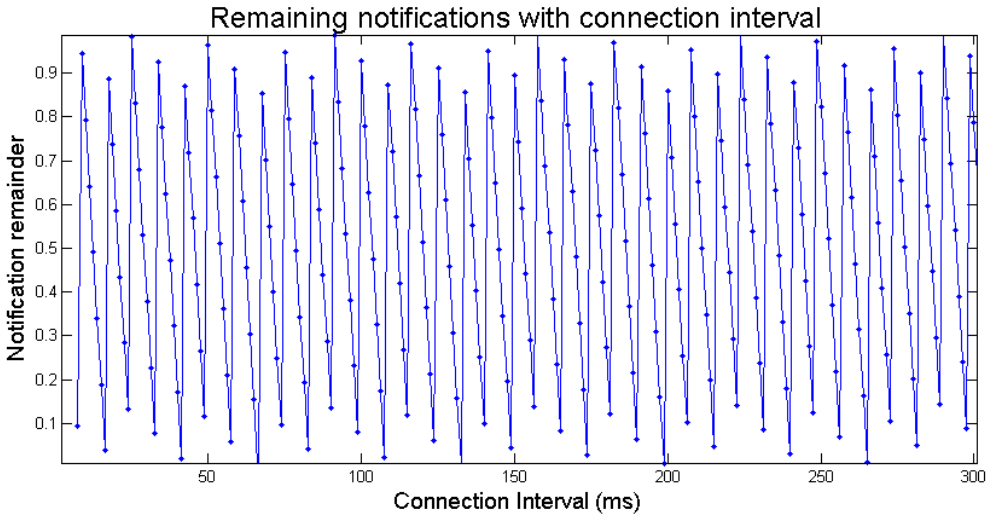


Figure 41: Remainder of packets unabel to fit into CI

However, this is simply the plot of the remainder, and for higher CI there will be more packets per CE and therefore while the remainder may be bigger, the throughput may be higher than a smaller CI with a smaller remainder. That is, the throughput is the number of CEs in a given time frame multiplied by the number of packets per CE. Factoring this in, the description changes to

$$\frac{1}{1250\mu s \times x} \times \lfloor \frac{1250\mu s \times x}{676\mu s} \rfloor \times 160 bits$$

This produces an expression for the maximum throughput against CI, shown graphically below

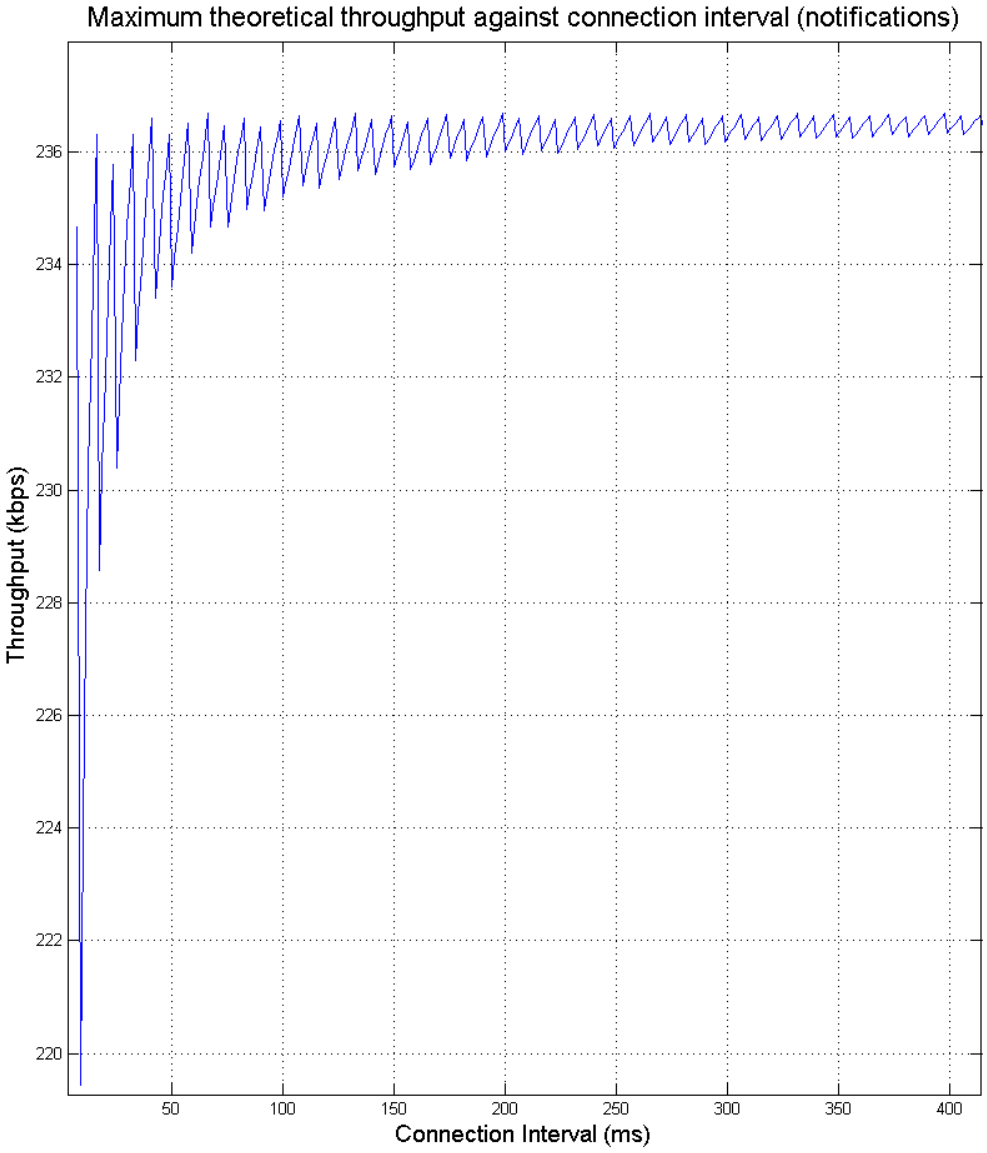


Figure 42: Throughput varying with connection interval

The throughput tends assymtopically to the maximum of 236.686 kbps, and min to max throughput is above 7%. To investigate how the theoretical maximum profile compares to the actual in practice, a test harness was created to investigate the relationship between throughput and CI. The test harness writes to the GAP characteristics the desired CI then takes a measurement of average throughput over a few minutes. It then moves onto another CI and does the same thing. This is called a test set. Sets are repeated a desired number of times (atleast 3) and averaged as it was found that there is a variation between sets. The results produced the profile shown in Figure 43.

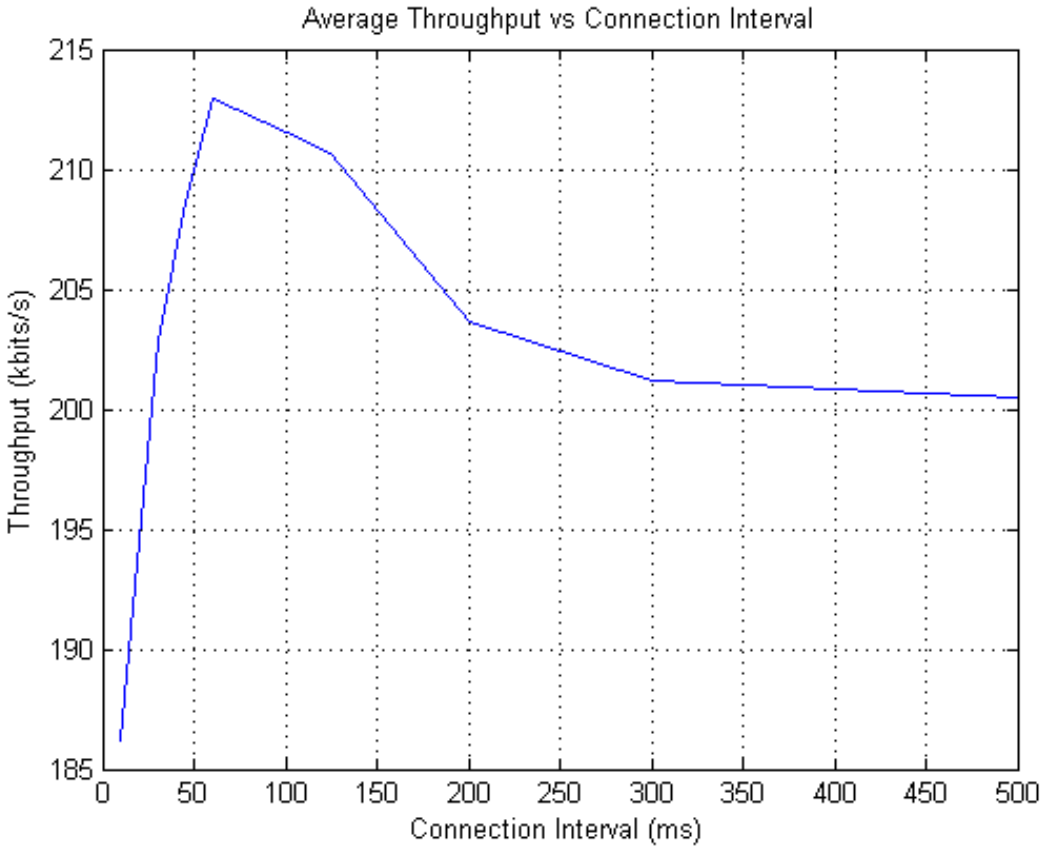


Figure 43: How throughput changes with varying connection interval

It was found that the optimum throughput did not exist at the longest connection interval. Rather, the profile appears have a single peaked platykurtic shape. The specification requires that during a CE, transmission errors (e.g. a CRC error), should cause the devices to "give up", power down and wait until the next interval to begin communicating again. This means that while longer CIs have marginally more throughput, there exists a trade off between the CI and risk of premature link termination. Figures 44, 45 shows packet waveforms for short and long CI. It can be seen that the longer CI intervals are prematurely terminated much more frequently than the shorter CIs. It is proposed that this exists because all communications takes place on the same channel during a single CE. In the subsequent CE the channel channel. For longer CIs, the chance of interference on that channel increases, while this is vice versa for shorter CIs.

Radio interference can described as a poisson process, whereby radio interference is an event occuring randomly in time. This shape will change depending on the conditions of the enviroment the raido is used in. Therefore, the task of achieving the highest throughput collaspes to being able to discover this profile. In practice this can be achieved by sampling throughput against CI using a feedback system between the master and slave. Investigating the range of variation between different radio enviroments may be an interesting piece of further work.

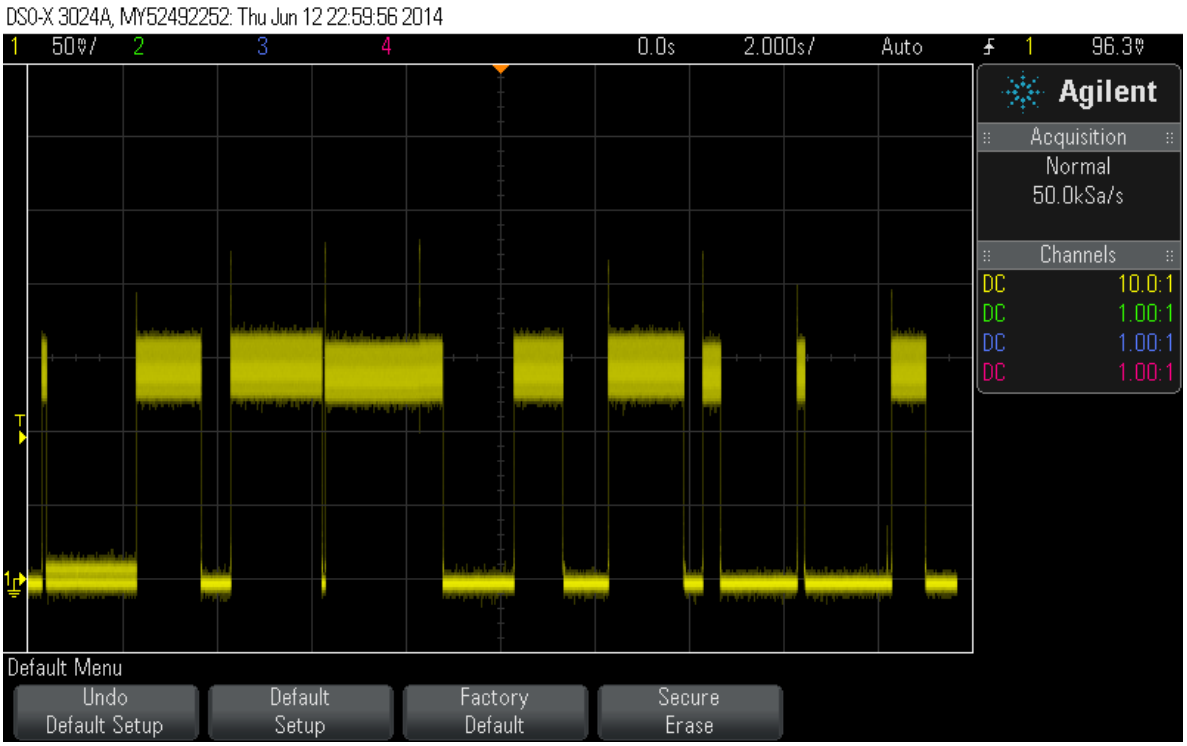


Figure 44: Waveform of notification packets for a CI of 2 seconds. Majority of CEs exhibit premature termination

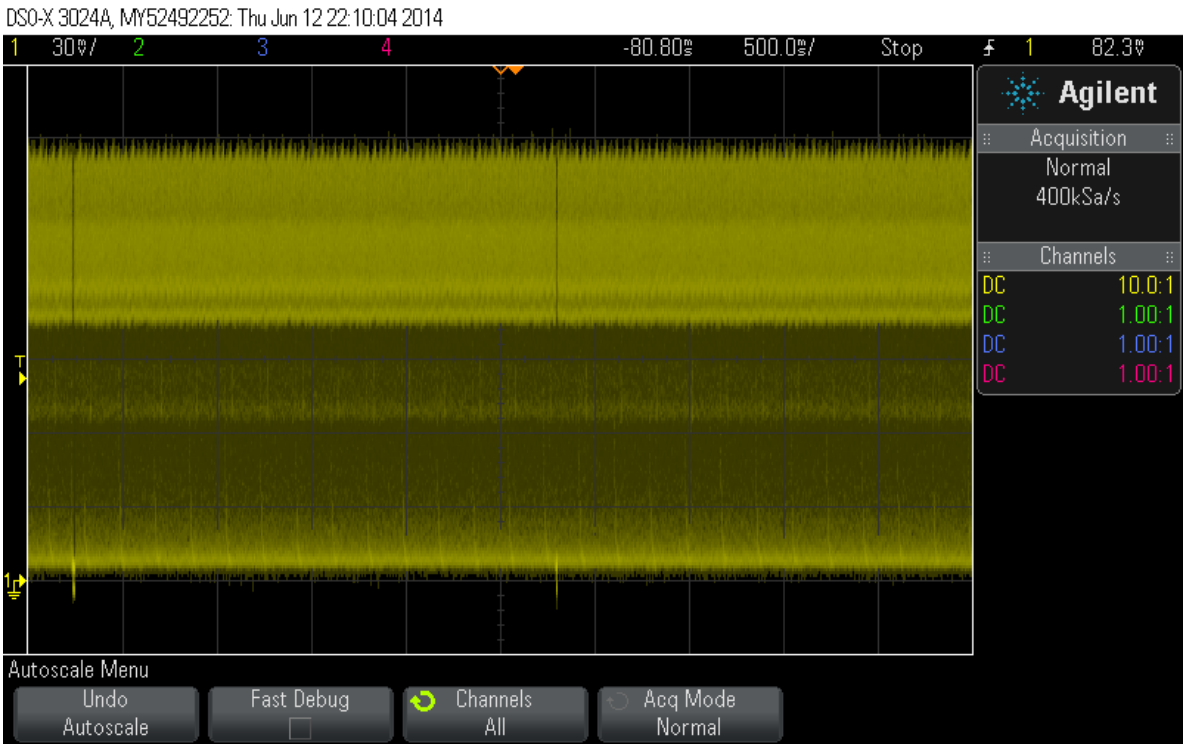


Figure 45

It is believed that the maximum throughput was not reached due to transmission errors

causing premature termination and device limitations. At a CI of 60ms, transmission errors still occur as shown in Figure 46. Device limitations includes the apparent 1.6ms of time between each connection. This 1.6ms represents 2 full notifications packets, and can be explained as to why at 7.5ms, only a throughput of 9 packets is achieved, not 11, like the theoretical maximum predicts. No requirement of this time has been found within the specification, and it is assumed that this is a limitation of the CSR1010 device. Figure 47 shows this.

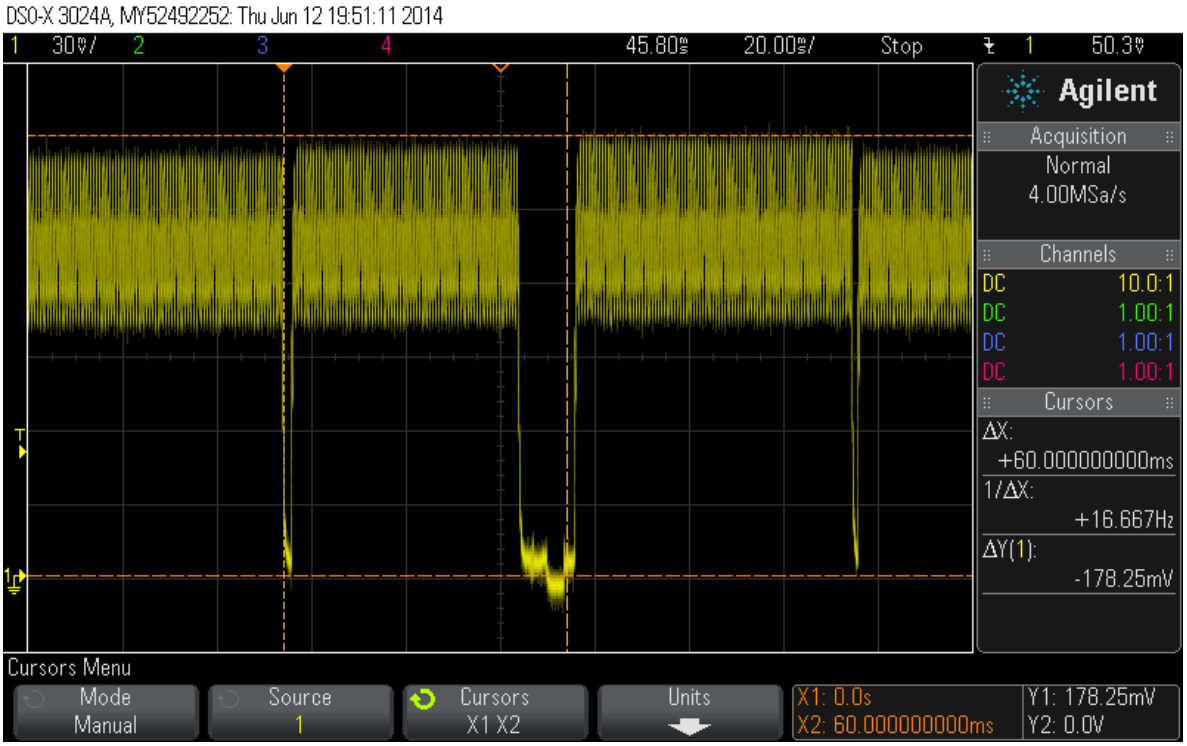


Figure 46: Transmission errors causing premature disconnection at 60ms

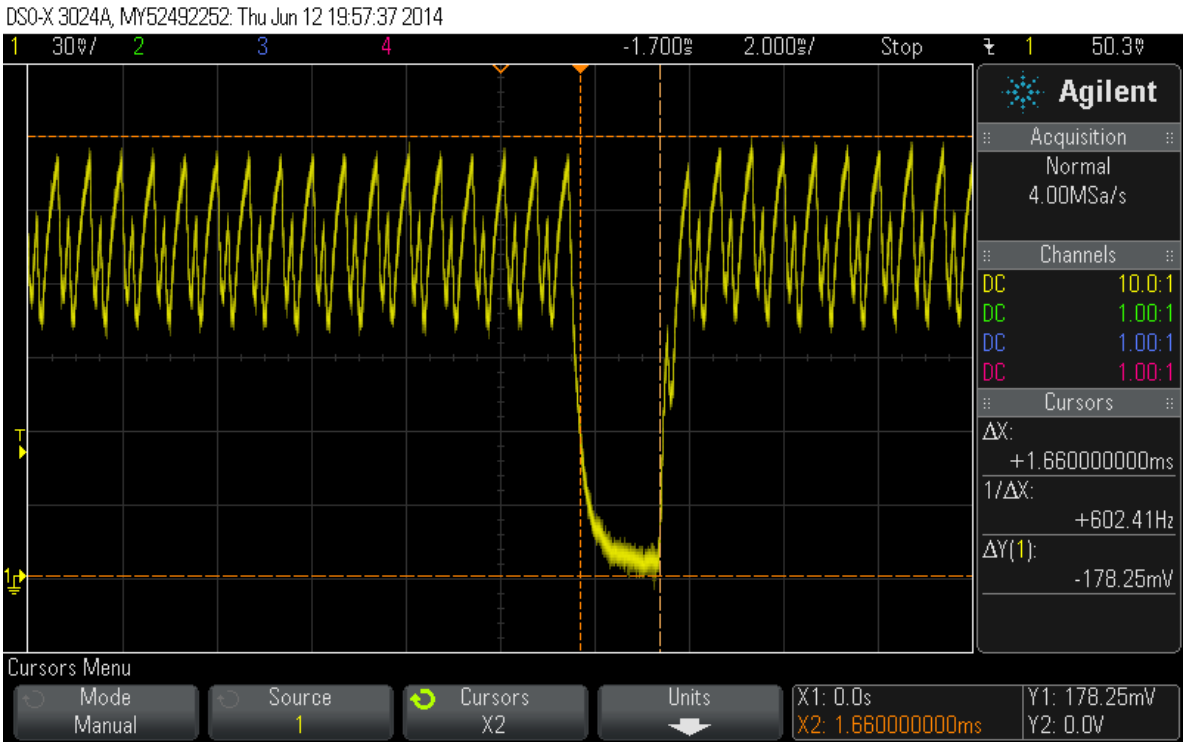


Figure 47: Radio power down present at every connection interval

Premature termination was observed to be frequent even at relatively low CIs. For example, at 60ms approximately 55% of 1 second samples achieved the full recorded throughputl (e.g. Figure 48). This differs greatly to a CI of 7.5, whereby this figure is close to 100

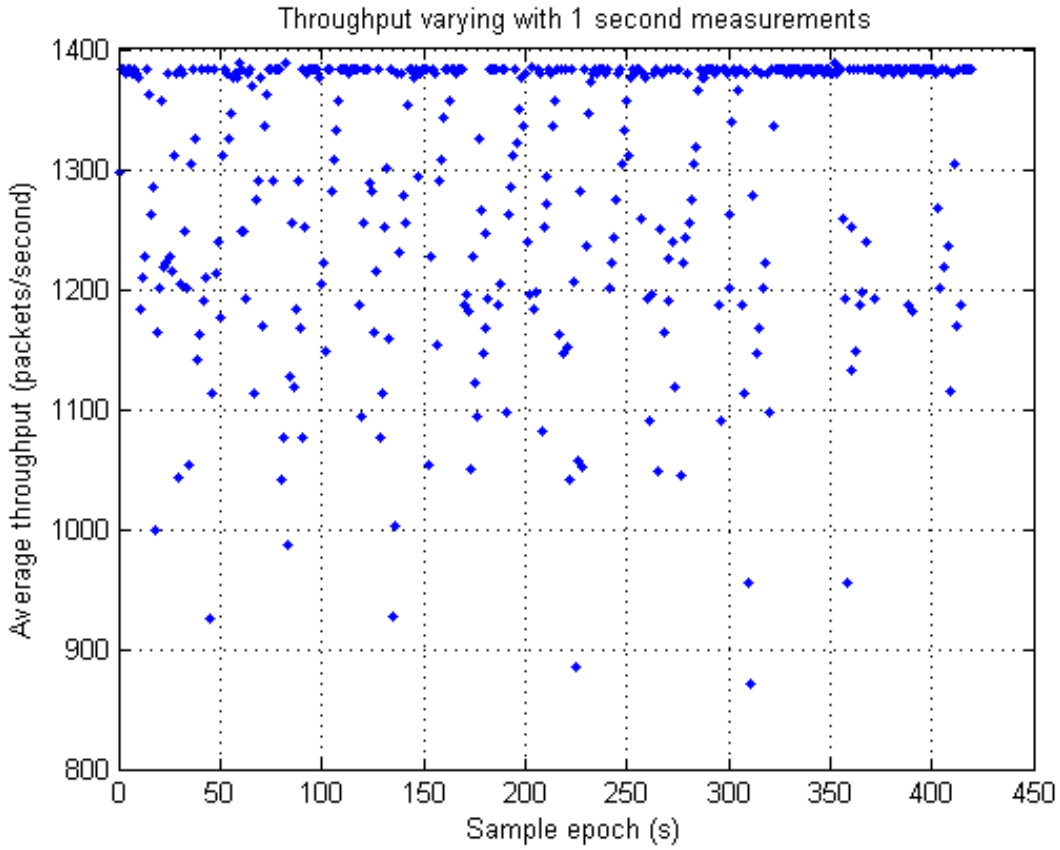


Figure 48: Radio power down present at every connection interval

The CI also determines the latency from when a measurement is taken to when it is displayed to the user. If data is buffered over many CEs and/or the frequency is low, the latency can become very large. In contrast, at high frequencies, the 64 k/bytes of ram (much used for the firmware application itself), may overflow and hence buffering is not a suitable. Further, with premature CE termination at higher CIs, data may have to remain buffered for long as early termination prevents communication until the subsequent CE. Providing the data can be serviced so that the buffer depth remains roughly the same over time (and does not variate to above full capacity), than selecting CIs that cause higher premature termination but higher average throughput is a rational choice in the pursuit of high datarates, but may suffer from power.

The difference between the minimum and maximum throughput for the theoretical maximum vs CI curve is approximately 7%, while that derived empirically was closer to %. At the peak measured performance, the device was capable of an average data rate of 90

From the background and literature review, most systems are only concerned with acquisition rates between

Table 1: Example achievable specifications from throughput

Channels	**Resolution (bits)**	**Frequency (Hz)**	**Required datarate (kbit-s/s)**
16	8	1500	192
16	8	1600	205
16	10	1300	208
2	10	10000	200
16	8	1000	128
16	10	200	32
16	8	1300	25.6
16	8	1300	25.6

Many devices restrict how many packets per CE can be received, constraining the system further, and in such cases, having a smaller CI is of benefit. The iPhone was observed to allow, on average only 6 packets per CE and limit the CI to a minimum of 30ms. From the Apple developer guidelines, it is possible to reduce this interval to 20ms. Many andriod devices have also have restrictions on the number of packets processed. At 6 packets per CE, and 7.5ms CI, the throughput achievable is 128 kbit/s, which is theoretically just enough bandwidth to support 16 channels with an 8 bit resolution at 1000 Hz.

Ultimately, to maximise throughput it is important to choose a CI for the trade off between CE notification packing wastage, risk of premature CE termination, latency and data integrity. The choice will further be constrained if BLE devices are limited by the number of packets per CE. It is believed this limitation arises from the firmware of the device as permits simplier firmware and manufacturers do not consider the use case of maxising throughput significant as there are other technologies out there. From the work above,the radio is capable

8.2 Power Analysis

To measure the power usage of the system a 10 ohm shunt resistor was inserted between the board's V_{ss} and ground. Scope leads where then attached to measure the voltage, which due to ohm's law, will be measured at 10 times the current.

Firsly, the power usage of the just the radio was performed (the ADCs were disconnected). The data sheet claims the maximum peak current used during transmission or receiving is approximately 16mA at 3V. While the measurements in Figure 51 and Figure 50 are at 3.3 volts, the overall power consumption of the system is the system regardless of the voltage level used. If voltage is decreased, the current increases.

It can be seen from Figure **??** that the measured voltage is 150mV. This corresponds to 15mA, approximately what the data sheet claimed. This respresents an instaneous power of 45mW. As mentioned previously, the advertising state of the system happens so infrequently that the long run

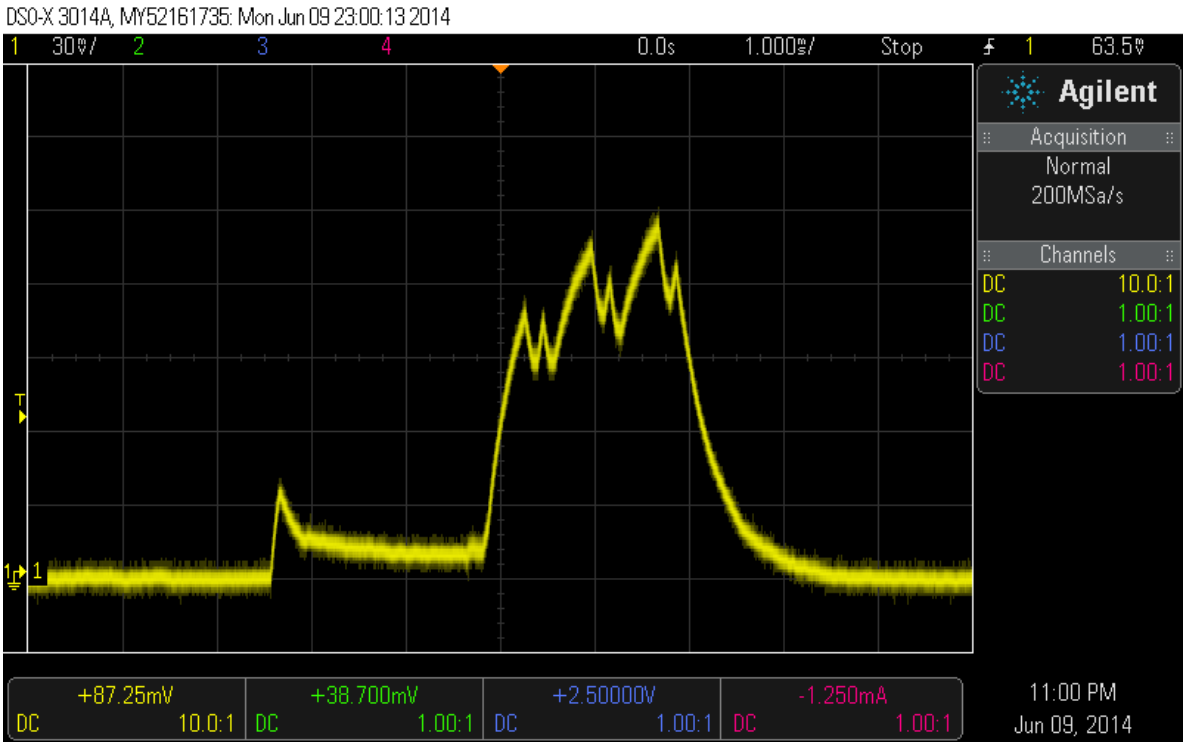


Figure 49: Advertising Event

cost is considered zero, hence the power analysis presented here does not include information from the advertising state.

CEs where no data other than correspondance (that the link was still alive), measured a peak current consumption less than 12mA. The CE can be divided into sections. The device wakes up approximately 2 milliseconds before transmission, remains idle, then listens/transmits (receive before transmission first for the slave device), before

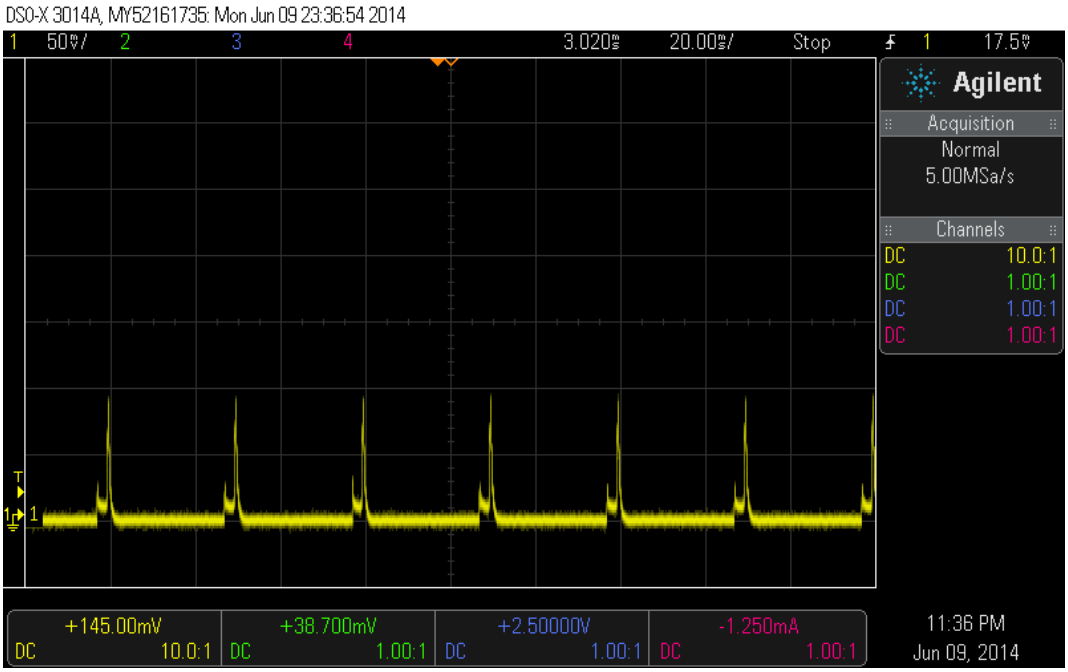


Figure 50: Connection events - connection interval set at 1 second with a slave latency of 0 (3.3 volt supply)

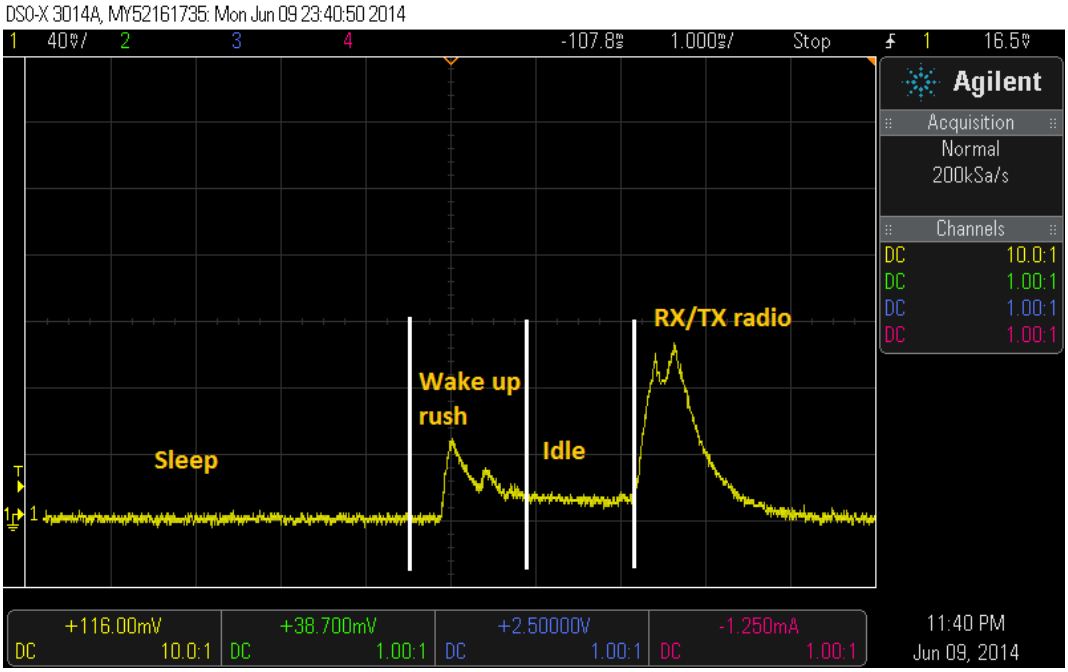


Figure 51: Connection event (3.3 volt supply)

To investigate the "base" power consumption the device requires, that is the power consumption required to maintain a connection indefinately. The data was logged and using an oscilloscope and saved to disk for later analysis. It was noticed that when the chip went into a deep sleep (inbetween connection events), the minimum value the scope recorded was negative. To adjust for this the absolute of the minimum value was added to the whole current vector. This will bring the minimum

consumption to 0, however this is still not correct as the device will consume a small amount of power in deep sleep. To correct to this, the data sheet value of 5μA was taken and applied to current vector.

To calculate the power consumption the trapezium rule is used to find the area under the profile. For 2.8V the area found under the curve equals. This area represents the current consumption in milli-amp-hours for a single CE with spacing either side to tesselate with the next CE. For each CE, a total of 2.69mA is consumed. To convert this to the current consumption in units milli-amp-hour not per connection event, the number is multiplied by the number of connection events (set by the connection interval) within an hour period, and divded through by the number of milliseconds in an hour to convert the time units (the x-axis in Figure 52 is in milliseconds).

Below shows the corrections and current consumption script for MATLAB.

```
current = Volt / 10; %adjust voltage to current (divide by 10 as 10 ohm shunt resistor)
currentAdjusted = current + abs(min(current)) + 0.000005 %voltage/current should not be negative. Shift
    up by absolute most negative value and add 5 micro amps taken from datasheet
```

However this often produces idle periods greater than 5 μA, so an alternative method can be used:

```
currentAdjusted = current - mean(current(1:300)) + 0.000005 %first 300 samples should be while device
    is sleeping. This should average to 5 micro amps
```

This is more accurate in producing profiles where at sleep time, the current is very close 5 μAs.

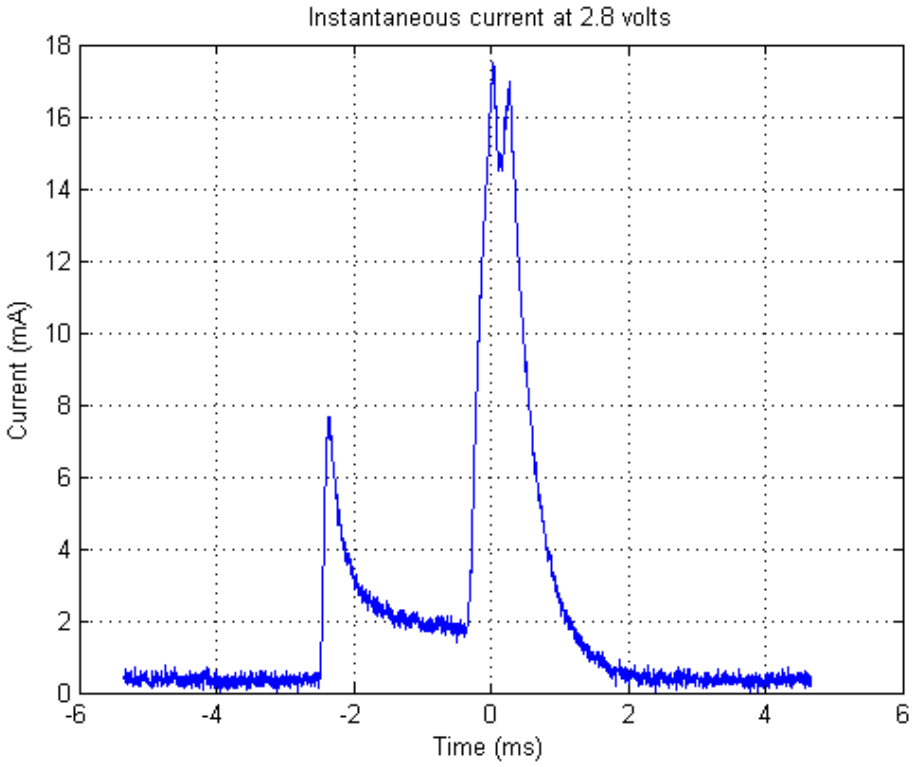


Figure 52: Connection event (3.3 volt supply)

This 2.69mA or (7.532mW power for all voltages) is the "base" amount, which must be paid every time data is sent. To achieve a higher economical value of energy for a given throughput, this fixed cost needs to be amortised as much as possible.

In the parameterised form

The approximated area can be further parameterised (e.g the rush current and duration), however it is presented here as a fixed cost

Increasing the slave latency upto the maximum defined by the specification, results in the a system lifetime of 337 years for two small 620mAh capacity batteries.

To conclude, vector of voltages and times. Approximated the area using the trapezium rule to calcualte the fixed cost.

Talk about min - max lifetime of BLE generic system, then shrink that too an EEG system.

Contrasting this to BTC

Comparision with TI

The hardware

Assuming that the number of

The idea of minimising energy per bit is not the true picture, as BTC and WiFi have higher energy per bit ratios. To achieve the lowest energy per bit, the transmission rate needs to be at the highest throughput achievable on the hardware due to the fixed costs or sunk costs of waking the MCU and preparing the device for radio transmission. However, consistently running at such throughput invalidates the use case for BLE, as there are other technologies capable of transmitting at the same or higher throughputs for smaller energy requirements.

8.3 Device

The device is a small size and weight, suitable of being sewn into a hat (EEG

The layout can be shurnk further through reducing distances between components. This was not done previously as it significantly increases the difficultly in hand assembly, which was the method used for the prototype. For the analouge front end

8.4 Bill of materials

9 Conclusions and Future Work

The majority of interesting EEG signals exist between . Running an EEG device at that

The proposed device is comparable to a hearing aid, even using the same battery technology.

Smart phone not single end device -> contain other radios for high throughput, maybe why hardware only supports 4 packets

From the preliminary work, it has been shown the CSR1010 is capable of running at 9 packets per CE stable. The theoretical throughput should be 11 packets per CE, and when examined under the scope using a shunt resistor (see section Power Analysis), it was found that if

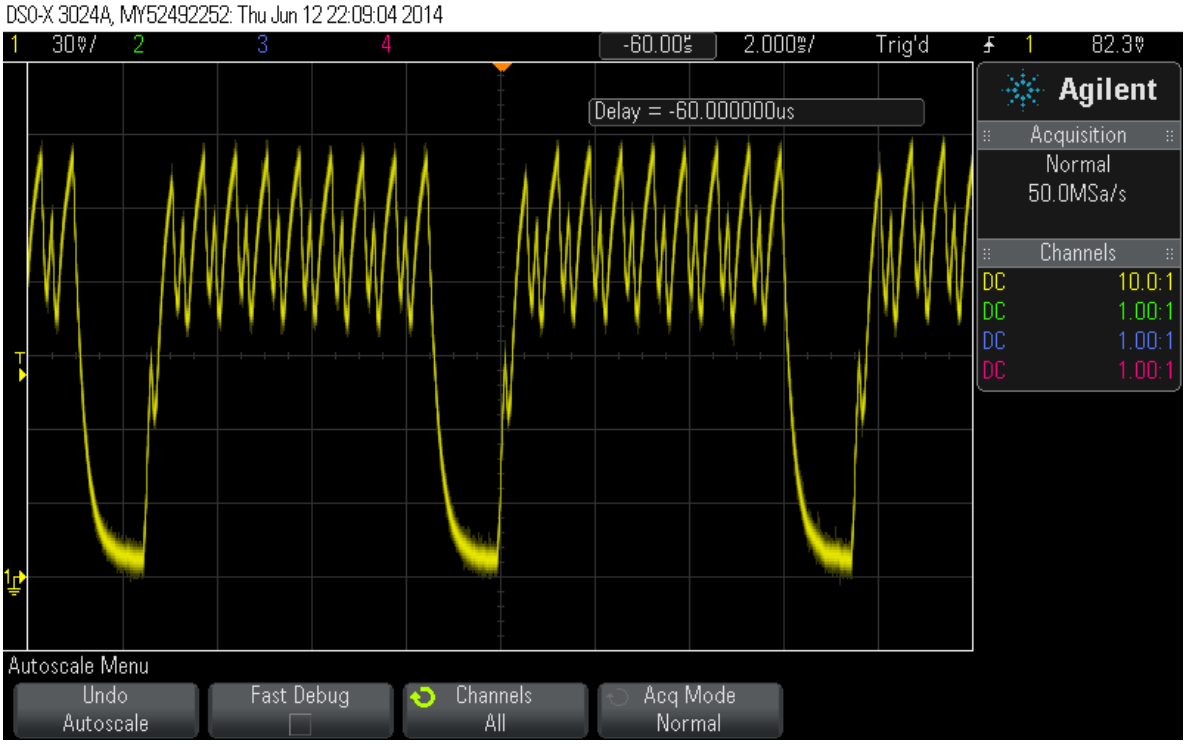


Figure 53: Waveform of notification packets for a CI of 7.5 milliseconds. Majority of CEs have no premature termination

Theoretically 11 packets should be able to fit into this interval

Unsure about when run a test set then run another, sometimes there can be big difference. Due to sharing radio channel?

//but unfortunately, there is an energy cost associated with hardware buffers from operation and leakage current

Hard to classify

On the whole, the project has

Both ends of the link

Not all tablets will be able to run at full speed

Ideally, it is desirable to write a conference style paper.

Never list packets per CI

Duplex

Combine with video

Ambulatory to wearable

Better understanding, big data

characeterise power with voltage

find point of indifference between bl and ble

10 Final Remarks

A vast array of technologies and tools were used throughout this project. Below highlights those that are non-trivial and were of significant importance Useful for CSR, monies worth out of me

- Git verisioning control - For versioning and segmenting the workflow
- xIDE - CSRs integrated development suite for compiling and debugging CSR-stack based chips. This is where the firmware for the CSR1010 MCU was written
- Wireshark - Invaluable in analysing the packets and control flow between BLE radios, unfortunately must be used offline
- SmartRF online packet sniffer - Useful as a low-speed packet sniffer. Extremely useful in the early days of the project, however the throughputs eventually obtained rendered the tool and hardware redundant as it was not capable of such speeds
- Visual Studio 2013 - Primairly used for developing the tablet application. Highly useful for "knocking up" quick throughput expirements
- Schematic capture and PCB layout using EagleCAD software

The large amount of both written and generated code is to vast to warrant being included in this report. Therefore is has been decided to make it publically available online at the git repositry address `https://github.com/proftom/AmbulatoryEEG`.

11 Bibliography

References

[1] SIG. *Fast, Simple Debugging for Bluetooth Low Energy Applications*. URL: https://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&ved=0CE0QFjAE&url=https%3A%2F%2Fdeveloper.bluetooth.org%2FDevelopmentResources%2FDocuments%2FWebinar_FastSimpleDebugging%2520Frontline-SIG.pdf&ei=lhaTU5OPJqGS7Ab-9IHwAQ&usg=AFQjCNHSkvdA4Vf2aGh-FJP4NpVUOmW0-A&sig2=wKQSG9ZYwzweJomkD6x2jg&bvm=bv.68445247,d.ZGU&cad=rja.

[2] Nick Hunn (WiFore) Bluetooth SIG Robin Heydon (CSR). *Bluetooth low energy*. URL: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=227336.

[3] Vincent Kuo (CSR) Bluetooth SIG. *Application development*. URL: https://www.bluetooth.org/en-us/Documents/Shenzhen%20CSR_uEnergy_SDK.PDF.

[4] Analog Devices. *AD7997 datasheet*. URL: http://www.roboard.com/Files/G211/AD7997_7998.pdf.

[5] John S. Duncan et al. *Adult epilepsy*. 2006. DOI: 10.1016/S0140-6736(06)68477-8.

[6] Texas Instruments. *Overlapped Processing*. 2013. URL: http://processors.wiki.ti.com/index.php/OverlappedProcessing.

[7] Guan Yang Jacob Rosenthal. *nRF8001 support for Arduino*. https://github.com/guanix/arduinonrf8001. 2013.

[8] Ole Morten. *Nordic Semiconductor employee*. 2013. URL: https://devzone.nordicsemi.com/question/3440/how-do-i-calculate-throughput-for-a-ble-link.

[9] Nordic Semiconductor. *nRF51822 Datasheet*. 2013, pp. 1–67. URL: http://www.100y.com.tw/pdf_file/39-Nordic-NRF51822.pdf.

[10] Anders Wimo, Bengt Winblad, and Linus J??nsson. "The worldwide societal costs of dementia: Estimates for 2009". In: *Alzheimer's and Dementia* 6 (2010), pp. 98–103. ISSN: 15525260. DOI: 10.1016/j.jalz.2010.01.010.

[11] World Health Organization. *Dementia: a public health priority*. Tech. rep. 2012, p. 112. DOI: 9789241564458. arXiv: 9789241564458. URL: http://whqlibdoc.who.int/publications/2012/9789241564458_eng.pdf.

[12] Maeike Zijlmans et al. "EEG-fMRI in the preoperative work-up for epilepsy surgery". In: *Brain* 130 (2007), pp. 2343–2353. ISSN: 00068950. DOI: 10.1093/brain/awm141.

12 Appendix

12.1 Acronyms

EEG Electroencephalography

AEEG Ambulatory Electroencephalogram

WHO World Health Organisation

BLE Bluetooth Low Energy

BTC Bluetooth Classic

CSR Cambridge Silicon Radio

TI Texas Instruments

NS Nordic Semiconductor

GATT Generic Attribute Profile

GAP Generic Access Profile

ATT Attribute Protocol

CI connection interval

CE connection event

SIG Special Interests Group

CODEC coder/decoder

CRC Cyclic Redundancy Check

PDU Packet Data Unit

SN Sequenc Number

NESN Next Expected Sequence Number

LOS line of sight

SoC system on chip

MCU microcontroller

API application programming interface

IDE integrated development environment

OSAL operating system abstraction layer

USB Universal Serial Bus

PCB printed circuit board

QFN quad flat no-leads package

ADC analogue to digital converter

DIP dual in-line package

SPI Serial Peripheral Interface

PIO programmable input/output

HCI host controller interface

MIC message integreity code

PTH plated through hole

UUID Universally unique identifier

MVC model-view-controller

MVVM model-view-view model

XAML Extensible Application Markup Language

OOP object oriented programming

TPL task parallel library

GUI graphical user interface

TPL task parallel library

SDK software development kit

13 User Guide