

**DUBLIN CITY UNIVERSITY**

**SCHOOL OF ELECTRONIC ENGINEERING**

**Real-time Limb Motion Capture for Human Performance Monitoring**

**Darren Brady**

March 2021

BACHELOR OF ENGINEERING

IN

ELECTRONIC AND COMPUTER ENGINEERING

MAJORING IN

THE INTERNET OF THINGS

Supervised by Dr Derek Molloy

# Acknowledgements

I would like to thank my supervisor, Dr Derek Molloy, for his guidance, enthusiasm and commitment to this project. Thanks are also due to Conor Murphy for supporting this work to date, etc.

# Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the DCU Academic Integrity and Plagiarism at [https://www.dcu.ie/system/files/2020-09/1\_-\_integrity\_and\_plagiarism\_policy\_ovpaa-v4.pdf](https://www.dcu.ie/system/files/2020-09/1_-_integrity_and_plagiarism_policy_ovpaa-v4.pdf%20) and IEEE referencing guidelines found at <https://loop.dcu.ie/mod/url/view.php?id=1401800> .

Name: \_\_\_\_\_\_\_\_\_Darren Brady \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_2/4/2021\_\_\_\_\_\_\_\_

# Abstract

This report details the development, implementation and testing of a wireless sensor network developed to perform real-time limb motion capture for use in human performance monitoring. The goal of this project was to create a wireless sensor network for tracking motion that could come close to the performance of some of the existing professional-grade solutions in this area. In this report some of the requisite background theory around this topic will be explored, the ideal design will be proposed broken down and explained.

The final solution that was developed at the end of this project meets many of the goals that were set out at the start and in the design phase, however, it also falls short in other areas,

In conclusion, this project succeeded in creating a wireless sensor network for motion tracking in its base form, but there's is also much scope for further development and refinement

Table of Contents

[Acknowledgements ii](#_Toc68584007)

[Declaration ii](#_Toc68584008)

[Abstract iii](#_Toc68584009)

[Chapter 1 – Introduction 1](#_Toc68584010)

[Chapter 2 - Technical Background 2](#_Toc68584011)

[2.1 Existing Solutions 2](#_Toc68584012)

[2.2 I2C Communication 5](#_Toc68584013)

[2.3 Software 7](#_Toc68584014)

[2.3.1 Code Composer Studio 7](#_Toc68584015)

[2.3.2 Sensor Controller Studio 7](#_Toc68584016)

[2.3.3 SmartRF Studio 7 9](#_Toc68584017)

[Chapter 3 - Design of Wireless Sensor Network System 11](#_Toc68584018)

[3.1 Hardware Design 11](#_Toc68584019)

[3.1.1 Challenges 11](#_Toc68584020)

[3.1.2 Microcontroller: LAUNCHXL-CC2650[14] 12](#_Toc68584021)

[3.1.3 Sensors: Adafruit MPU-6050[15] 12](#_Toc68584022)

[3.2 Software Design 13](#_Toc68584023)

[3.2.1 Challenges 13](#_Toc68584024)

[3.2.2 Interfacing with the sensors 14](#_Toc68584025)

[3.2.3 Wireless Communication 16](#_Toc68584026)

[3.2.4 Data Synchronisation 17](#_Toc68584027)

[3.2.5 Data Logging 17](#_Toc68584028)

[Chapter 4- Implementation of Wireless Sensor Network 18](#_Toc68584029)

[4.1 General Implementation 18](#_Toc68584030)

[4.1.1 Class Structure 18](#_Toc68584031)

[4.1.2 Radio Drivers and Smart RF Studio 7 21](#_Toc68584032)

[4.2 Sensor Node Implementation 23](#_Toc68584033)

[4.1.1 Sensor Node Operation 23](#_Toc68584034)

[4.1.2 Sensor Controller 25](#_Toc68584035)

[30](#_Toc68584036)

[4.3 Concentrator Node Implementation 31](#_Toc68584037)

[4.3.1 Concentrator Node Operation 31](#_Toc68584038)

[34](#_Toc68584039)

[Chapter 5 - Testing of Wireless Sensor Network & Results 35](#_Toc68584040)

[5.1 Scopy 35](#_Toc68584041)

[5.2 Sensor Controller Studio 37](#_Toc68584042)

[5.3 SmartRF Studio 7 38](#_Toc68584043)

[5.3.1 Received Signal Strength Indicator (RSSI) 38](#_Toc68584044)

[5.3.2 Packet Error Rate Testing 43](#_Toc68584045)

[5.3.3 Overall Results from Smart RF Studio Tests 45](#_Toc68584046)

[5.3.4 Eliminating Sources of Interference/Noise 46](#_Toc68584047)

[Chapter 6 – Ethics 49](#_Toc68584048)

[6.1 Sustainability 49](#_Toc68584049)

[6.1.1 Lead and Toxic Substances 49](#_Toc68584050)

[6.1.2 Packaging Waste and Recyclability 50](#_Toc68584051)

[6.2 Potential for Military Use 51](#_Toc68584052)

[Chapter 7 - Conclusions and Further Research 52](#_Toc68584053)

[References 53](#_Toc68584054)

[Appendix 56](#_Toc68584055)

Table of Figures

**Figure 2.1 -** Simple I2C Bus ………………………………………………………… Page 5

**Figure 2.2 -** I2C Communication Example ..………………………………………… Page 6

**Figure 2.3 -** Task Testing in Sensor Controller Studio ……………………………… Page 8

**Figure 2.4 -** Task Testing sensor output graphs ……...……………………………… Page 8

**Figure 2.5** – Device Control Panel ……...…………...……………….……………… Page 10

**Figure 3.1 -** Gyroscope Configuration Register…..………………………………… Page 14

**Figure 3.2 -** Gyroscope Range Select ………...……………………………………… Page 14

**Figure 3.3 –** LSB Sensitivity…………………..…..………………………………… Page 15

**Figure 3.4 -** Accelerometer Result Register ………....……………………………… Page 15

**Figure 4.1 –** Basic Node Operation ……….………....……………………………… Page 18

**Figure 4.2 -** Initialization Functions ……….………....……………………………… Page 19

**Figure 4.3 -** Task Initialization ……….………....……………………………….……Page 19

**Figure 4.4 -** DMPacket Struct ……….………....……………………………….……Page 20

**Figure 4.5 -** SmartRF Studio value for 4dBm …………….…………………….……Page 22

**Figure 4.6 -** Setting Tx Power in CCS …….……....………………….………….……Page 22

**Figure 4.7 -** Setting Frequency in CCS …….….…....……………….………….……Page 22

**Figure 4.8 -** Node Task Operation………….….….....……………….………….……Page 23

**Figure 4.9 -** Node RadioTask Operation………………..…………….………….……Page 24

**Figure 4.10 -** Sensor Controller Operation ………….……..…………………….……Page 25

**Figure 4.11 -** Constants and Data Structures ………….……..….……………….……Page 25

**Figure 4.12 -** Sensor Controller Initialisation……….……..….……………...….……Page 28

**Figure 4.13 -** Execution Code…………………………………..……..……...….……Page 29

**Figure 4.14 -** Sensor Node Output Sample……………………..…..……...…….……Page 30

**Figure 4.15 -** Sensor Controller Studio Output Sample…………...............…….……Page 30

**Figure 4.16 -** Concentrator Radio Task Operation ………………...............…….……Page 31

**Figure 4.17 -** Concentrator Task Operation ………………..……...............…….……Page 32

**Figure 4.18 -** Print Values Function…...…..………………..……...............…….……Page 33

**Figure 4.19 -** Concentrator Output for Sensor Readings …...………….......…….……Page 34

**Figure 4.20 -** Concentrator Output for Data Test...…....…………...............…….……Page 34

**Figure 5.1 -** Failed I2C Communication Example……...………................…….……Page 35

**Figure 5.2 -** Failed I2C Communication Example 2 ...…………................…….……Page 35

**Figure 5.3 -** Sensor Controller Burst Read ………………...……...............…….……Page 36

**Figure 5.4 -** Arduino Burst Read ………...………………………...............…….……Page 36

**Figure 5.5 -** Accelerometer y-axis spiking erratically …………...…………...….……Page 37

**Figure 5.6 -** Accelerometer y-axis normal behaviour …………...…….……...….……Page 37

**Figure 5.7 -** 2445MHz Noise Spikes………………. …………...…….……...….……Page 39

**Figure 5.8 -** 2435MHz low noise levels ………..…. …………...…….……...….……Page 39

**Figure 5.9 -** Packet Tx Function..…. …………………………...…….……...….……Page 43

**Figure 5.10 -** Packet Rx Function..…. ………………..………...…….……...….……Page 44

**Figure 5.11 -** Frequency Sweep Setup..……………..…..……...…….……...….……Page 46

**Figure 5.12 -** Antenna Bandwidth..…......……………..………...…….……...….……Page 47

**Figure 5.13 -** Erroneous Packet..…. …………………..………...…….……...….……Page 48

# Chapter 1 – Introduction

The main goal for this project was to develop a system of wearable sensors that was capable of capturing human limb movement in real-time for the purpose of human performance monitoring. Human performance monitoring is a practice that has been carried out for as long as humans have been competing with one another. Whether done consciously or unconsciously the desire to observe another human and gain an insight into their strength, speed, fitness and even health has been ever-present in our society since our hunter-gatherer days.

While the stakes may no longer be life and death, the desire to gain more insight into a person’s performance has only increased. The Sports industry is one of the main driving forces behind this desire and many of the advancements made in recent years can be attributed to the research and funding coming from this industry. Due to these technological advancements, there is no longer a reliance on the intuition and personal experiences of coaches and managers to make judgments on athletes’ performance. Technology can now provide us with accurate real-time data and exact performance metrics on athletes. This huge demand and drive for more accurate human performance monitoring equipment has lead to the market being saturated in high-end options for the professional sports industry.

In the next chapter, some of these existing solutions will be explored and the general market trends will be broken down.

# Chapter 2 - Technical Background

In this chapter, some of the existing solutions and design principles in this field are examined and compared against one another. Some of the software toolsets utilised in this project and the requisite background information required to understand them are explained.

## 2.1 Existing Solutions

The table below shows a basic review of the Hardware present in some of the existing designs and solutions in this field. While many implementations include additional sensors and monitoring methods to increase their versatility, the table below only displays the product specifications relevant to this project. The majority of the existing configurations are made up of at least one accelerometer and a gyroscope sometimes a Magnetometer is also included. Some of the lower prices listed in the table below are for single sensor units, multiples of which would be necessary for a full working system. If information was unavailable at the time of writing the field marked with a N/A.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Product | Size | Weight | Sensors | Battery | Networking |
| APDM Opal [1]  Price: €1977 | Dimensions:  43.7x39.7x13.7mm | 25g | Accelerometers:  ± 16g, ± 200g  Gyroscope:  ± 2000 deg/s  Magnetometer:  ± 8 Gauss | 12h-16h | Nordic Semiconductor  nRFL01+radio  Latency:  Up to 300ms  Data Rate:  Up to 2Mbps |
| Blue Trident [2]  Price: €1979 | Dimensions:  42 x 27 x 11mm | 9.5g | Accelerometers:  ± 16g, ± 200g  Gyroscope:  ± 2000 deg/s  Magnetometer:  ± 49 Gauss | Up to 12h | Bluetooth 5  Latency:  30-300ms  Data Rate:  Up to 2Mbps |
| Mbientlab MetaTracker  [3]  Price: €49 per unit | Dimensions:  52 x 35 x 15mm | 19.85g | Accelerometers:  ± 2, ± 4, ± 8, ± 16 Gyroscope:  ± 125, ± 250, ± 500, ±1000, ± 2000 deg/s  Magnetometer:  N/A | 2h-48h | Bluetooth Low Energy Smart  Latency:  100-300ms  Data Rate:  Up to 2Mbps |
| Noraxon MyoMotion  [4]  Price: Rented-€50/h Purchase-N/A | Dimensions:  37.6 x 52 x18.1mm | 34g | Accelerometers:  ± 1.7, ± 16  Gyroscope:  ± 500, ± 2000 deg/s  Magnetometer:  ±1.9 Gauss | Up to 8h | Unigen UGWG4USHN33  Latency:  140ms  Data Rate:  Up to 1Mbps |
| GaitUp  Physilog 5 [5]  Price: N/A | Dimensions:  47.5 x 25.5 x 10mm | 11g | Accelerometers:  ± 16g  Gyroscope:  ±2000 deg/s  Magnetometer:  N/A | Up to 20h | Nordic Semiconductors  Supporting Bluetooth Low Energy (BLE), Ant+, and Near field communication (NFC)  Latency:  100-300ms  Data Rate:  Up to 2Mbps |
| Shimmer IMU [6]  Price: €359 per unit | Dimensions:  51 x 34 x 14mm | 23.6 | Accelerometers:  ± 2, ± 4, ± 8, ± 16,  Gyroscope:  ± 250, ±500, ±1000, ± 2000 deg/s  Magnetometer:  ± 49 Gauss | N/A | Bluetooth 5  Latency:  100-300ms  Data Rate:  Up to 2Mbps |
| Xsens Dot  [7]  Price: €89 per unit | Dimensions:  36 x 30 x 11 mm | 10.8g | Accelerometers:  ± 16  Gyroscope:  ± 250, ±500, ±1000, ± 2000 deg/s  Magnetometer:  ± 8 Gauss | N/A | Bluetooth 5  Latency:  30ms  Data Rate:  Up to 2Mbps |
| Yost 3 Mini  [8]  Price: €107 per unit | Dimensions: 30 x 30 x 13mm | 9g | Accelerometers:  ± 2, ± 4, ± 8, ± 16  Gyroscope:  ± 125, ± 245, ±500, ±1000, ± 2000 deg/s  Magnetometer:  N/A | N/A | Bluetooth Low Energy Smart  Latency:  100-300ms  Data Rate:  Up to 2Mbps |

This review of some of the existing solutions highlights the high entry-level cost that makes many of these systems only suitable for use in high-level professional industries. However, while there are some discrepancies between existing solutions many of them follow similar design principles. A consistent feature of all the designs was the focus on maintaining a low profile and weight. The average dimensions and weight of a unit were found to be:

**Dimensions =** 42.5 x 34.2 x 13.2mm, **Weight** = 17.8g.

#### Size and Weight

The driving factor behind this desire for as low a profile and as low of a weight as possible is the effect that the size and weight of anything an athlete/user wears can on their performance and movement[9]. This criterion is clearly considered by some of the solutions that are aimed primarily toward the sporting markets such as the “Blue Trident”[2] and “Yost 3 Mini”[8] who implement extremely low profile and low weight designs. While the opposite is true for some of the solutions that may be designed for more lab and clinic-based use cases such as the “NORAXON myoMotion”, as they have significantly larger profiles and heavier weights.

#### Sports Injury Detection

In recent years there has been a large push for the implementation of some of these solutions in contact sports[10]. The increase in the physical strength and speed of professional athletes and the greater understanding that we now have of concussions and what can cause them has created an ideal scenario for the use of some of these designs. Concussions are usually caused as a result of forces applied to the head. These forces can be a result of direct impact or rotational force, but with the correct hardware design and sensor choice, these forces could easily be detected and flagged possibly even before a serious injury has even occurred[11].

## I2C Communication

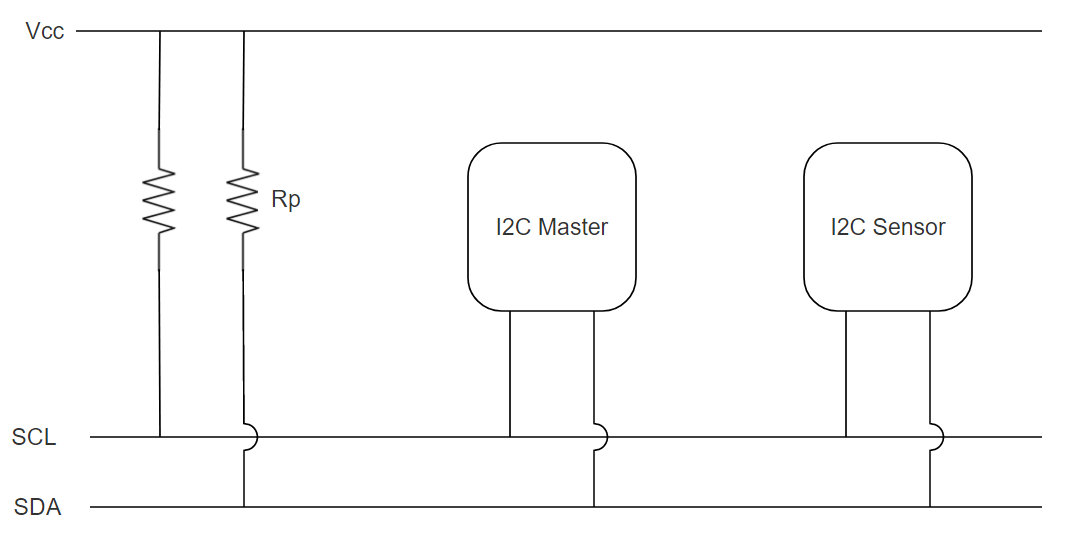
This project utilises the I2C communication bus to interface the main microcontroller with the sensor. The I2C bus is a communication method that allows a master device to communicate with a single or multiple slave devices. One of the largest benefits too I2C is that only two lines are required for this communication between devices the Serial Data (SDA) line and the Serial Clock (SCL) line. Figure 2.1 below shows a simple example of an I2C Bus with just a single master and slave device (I2C Sensor) connected. I2C uses an open-drain/open-collector with an input buffer on the same line, this allows for a single data line to be used for bi-directional data flow[12]. The open-drain nature of the I2C bus means that output can both pull the bus down to a voltage such as ground, or it can release the bus and allow it to be pulled up to Vcc by the pull-up resistor Rp. As no device can force high on a line, the bus will never encounter an issue where one device may transmit a high and another device may transmit a low simultaneously which could cause a short[12].

Figure 2.1 – Simple I2C Bus

#### I2C Transactions

Figure 2.2 below shows an example of an I2C write communication to access a sensor with an I2C address of 0x68 and to access the sensors internal register at 0x6B and to write a value of 0x80 to this register.

The I2C communication begins at the Red circle at the bottom of the figure, this highlights the start condition which occurs when there is a high to low transition on the SDA line while the SCL line remains high. The master device then “writes” the address of the slave with a zero bit appended to it, this zero bit informs the slave that a write transaction is occurring, if we wanted to perform a read transaction one bit would be appended. The master then waits until it receives an ACK (acknowledgement) from the slave device before writing the address of the internal register (0x6B) of the slave’s that it wishes to access. Finally when the master receives ACK from the slave for that data write it then writes the value (0x80) to the register 0x6B. The master then receives a final ACK from the slave informing it that its final write was successful. The transaction then ends with the green circle this denotes a stop condition which occurs when the SDA line transitions to high while the SCL line is high.

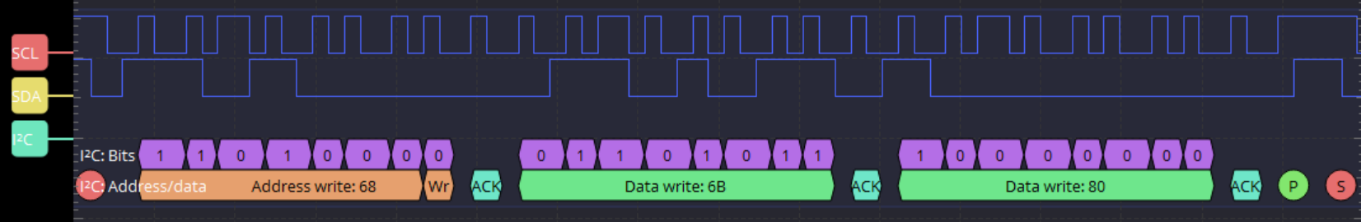


Figure 2.2 - I2C Communication Example

A read communication would follow an identical pattern to the write communication however one bit would be appended to the initial transmission of the Slaves I2C Address (Address Write 0x68). The internal register which the master desired to read would then be written to the slave device. The master would then read the contents of this register from the slave.

## 2.3 Software

The LAUNCHXL-CC2650 was chosen as the main microcontroller for this project[[1]](#footnote-1). This microcontroller has a few useful proprietary IDEs for programming, testing and debugging various elements of the chipset’s operation. This section gives a brief overview of some of these IDE and explains their uses and benefits.

### 2.3.1 Code Composer Studio

Code Composer Studio is an IDE that was developed to support Texas Instruments microcontroller and embedded Processor range it was used for the testing and development of the main program for this project. It can provide a user with much more in-depth low-level control over the program they write for their TI device than a regular IDE can.

### 2.3.2 Sensor Controller Studio

Sensor Controller Studio is an IDE for writing, testing and debugging code for the Sensor Controller on CC26xx/C13xx Texas instruments products. The Sensor Controller contains circuitry that can be enabled and programmed separately to that of the main microcontroller CPU. The sensor controller engine CPU can read and monitor sensors autonomously of the main CPU, which can reduce power consumption and offload work from the main CPU [13]. The sensor controller engine was utilised to handle the I2C communication between the LAUNCHXL-CC2650 and the sensor module in this project. Once a working program or task has been developed that communicates with and reads data from a sensor, a user can then test the output values produced by this task. Figure 2.3 shows the Task Testing functionality provided by Sensor Controller Studio. The Task Testing functionality allows a user to test all aspects of their program individual and can halt the program if certain events or interrupts occur. Figure 2.4 shows the live sensor data output from a sensor monitoring task being run with the Task Testing functionality

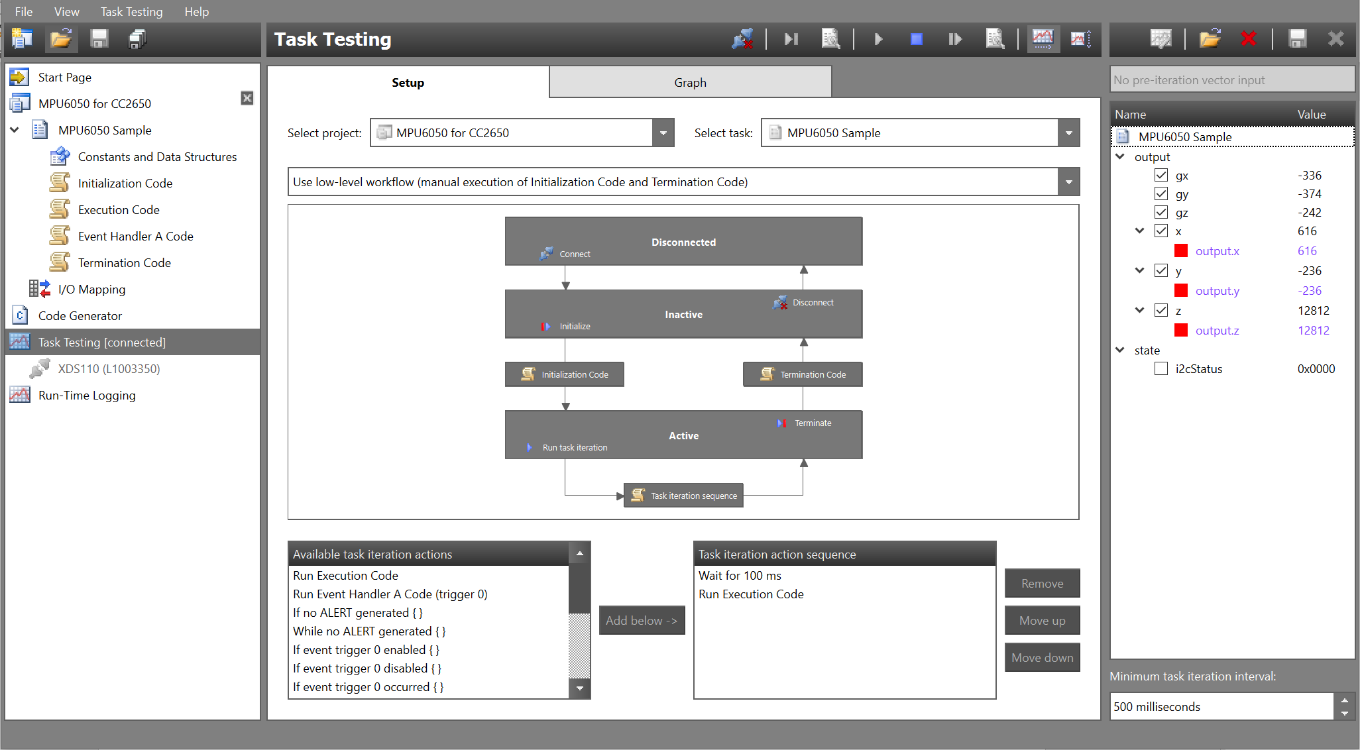


Figure 2.3 – Task Testing in Sensor Controller Studio



Figure 2.4 - Task Testing sensor output graphs

### 2.3.3 SmartRF Studio 7

SmartRF Studio is a program developed by Texas Instruments (TI) to allow users to customise and test different radio configurations on supported TI Devices. The functional and performance testing methods provided by SmartRF Studio allows users to accurately measure and tune the performance of their radio implementation. Users can then generate and export the customized radio register values for compilation in other IDEs such as Code Composer Studio[[2]](#footnote-2). The generated code will be stored in smartrf\_settings.c and smartrf\_settings.h files and can be easily interfaced using the Easylink API which is designed to allow the simple integration of RF drivers into a program.

When a compatible device is connected to the PC the user will be able to open the Device Control Panel, see Figure 2.5. This is the GUI that allows the user to customize and test different radio configurations. The RF Parameters section allows the user to customize the frequency of transmission and the output transmission power or Tx Power of the device. The software also provides various testing functions for measuring the performance of a radio network these are displayed at the bottom of the GUI[[3]](#footnote-3) .

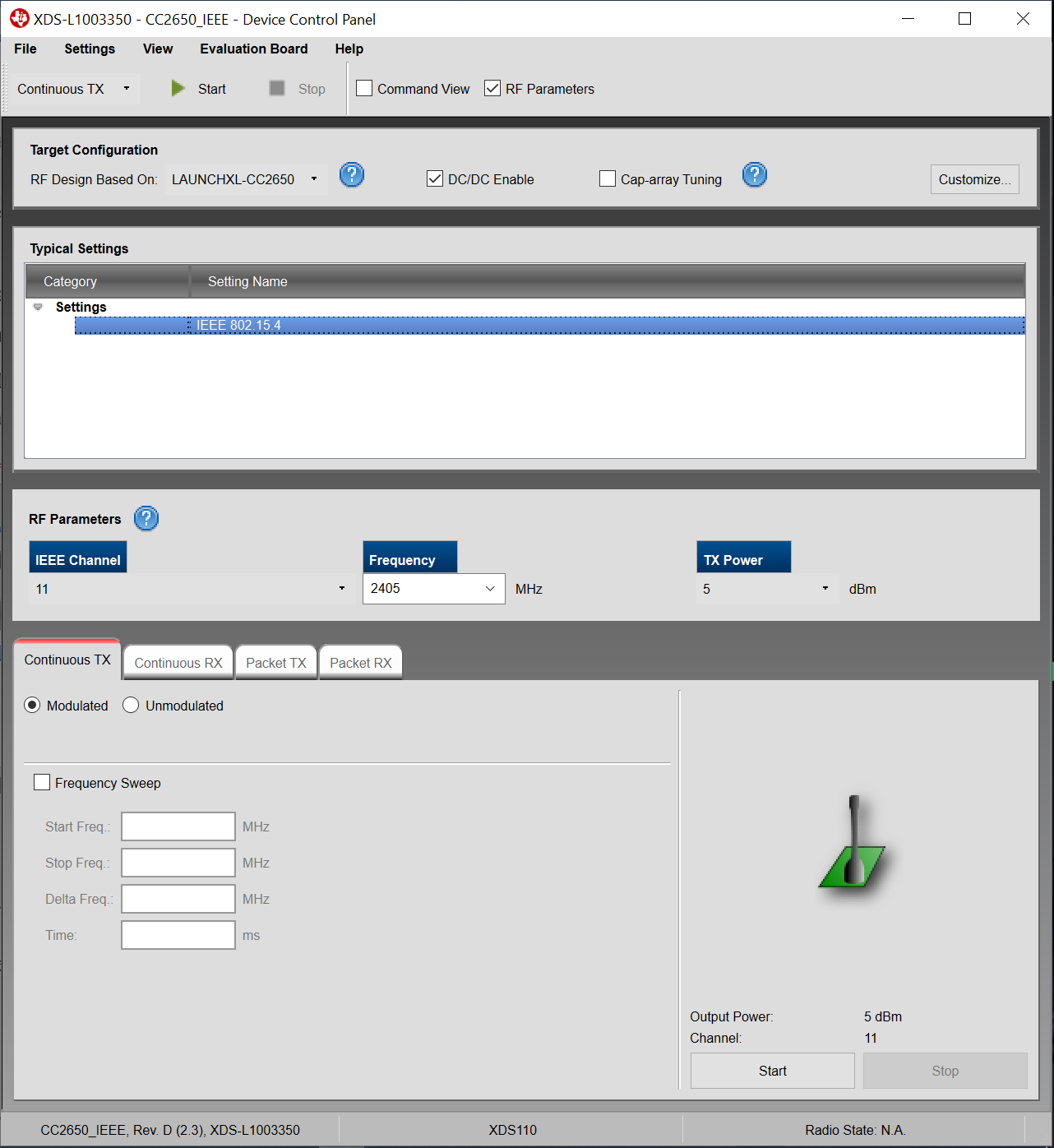


Figure 2.5 - Device Control Panel

# Chapter 3 - Design of Wireless Sensor Network System

The High-Level Design of this wireless sensor network was comprised of a system of Sensor Nodes that could be placed in different locations on a users body, and a single main Concentrator Node that would be placed at some central position on the user's torso. Each Sensor Node collects data from their onboard accelerometers and gyroscope, a single data reading is then comprised of data from both sensors. Once a data reading has been completed the Sensor Node then timestamps the reading and transmits it to the Concentrator Node. The Concentrator Node then ensures that the data is synchronized with the incoming sensor data from other sensors in the network and stores the data internally

## 3.1 Hardware Design

The challenges involved in the hardware design section was the selection of suitable hardware that was compatible with one another and could meet all the requirements of our wireless network design.

### 3.1.1 Challenges

#### Microcontroller choice

The microcontroller choice for this design must be able to meet all the requirements of this system while remaining cost-effective in price. It must be wirelessly capable with robust enough of a wireless connection that it can maintain communication with the other nodes in the system while they are not in direct line of sight (if they were to be obstructed by a users limbs or torso) or if they were actively moving away from one another. It must be capable of low power operation, this will be an essential feature to ensure the system has a long battery life while maintaining its performance. It must also be small and low profile enough that it could easily be deployed on a small PCB alongside the chosen sensor unit with little to no issue.

#### Sensor Choice

The sensor choice in implementation also strongly dictates its use case. Many of the existing solutions reviewed in Chapter 2 have adjustable ranges for their chosen sensors. This provides great flexibility to some solutions and can allow the system to be adapted to ensure the least amount of noise is present and that the sensors avoid over saturation. Some implementations utilise a dual accelerometer design, they use a low-g accelerometer to detect smaller forces and impacts and a high-g accelerometer for whenever a force exceeds the range of the low-sensor. This dual-sensor approach ensures the system is as adaptive as possible and further expands the use case of systems that utilise this approach.

### 3.1.2 Microcontroller: LAUNCHXL-CC2650[14]

The Texas Instruments LAUNCHXL-CC2650 supports a wide range of wireless protocols such as Bluetooth Low Energy, ZigBee® and 6LoWPAN, and ZigBee RF4CE. The device has very low active RF and MCU current and has a low-power current consumption mode that provides excellent battery life[14]. The CC2650 chip costs just €3.09 for up to 100 chips.

The microcontroller also features an ultralow-power sensor controller that can monitor and operate sensors independently of the main CPU. The microcontroller is also supported by some proprietary Texas Instrument software toolsets that allow for enhanced customization and control over both the wireless communication protocol of the MCU and the low power sensor controller engine.

### 3.1.3 Sensors: Adafruit MPU-6050[15]

The “Adafruit MPU-6050” was selected as the primary sensor unit for the system. The board consists of a 3-axis accelerometer with a selectable range of ±2g, ±4g, ±8g, and ±16g and a 3-axis gyroscope with a selectable range of ±250, ±500, ±1000, and ±2000 deg/sec. The combination of these two sensors allows for the accurate capture of the motion data of a user by measuring both the directional and rotational forces being applied to the board. The MPU-6050 also features a Digital Motion Processor which can be used to offload the computation of motion processing algorithms from the main CPU[15]. The DMP reads the data from the individual sensor result registers in the MPU and creates a signal output for each value which can then be read from the DMP’s registers. The Adafruit MPU-6050 costs just €6.70 for 100 chips.

## 3**.2 Software Design**

The main challenges involved in the software design section of this system were the interfacing and communication between the Sensor Nodes and their sensors, the transmission of data from the Sensor Nodes to the Concentrator Node and finally the synchronization of data and data logging performed by the Concentrator Node.

### 3.2.1 Challenges

#### Interfacing with the sensors

An interface must be put in place to allow the Sensor nodes to be easily and quickly able to communicate with their attached sensors. This interface must allow for the sensor node to read and output the raw data from the sensor result registers in real-time to ensure the most recent sensor data is being sent to the main Concentrator Node.

#### Wireless Communication

For the Sensor Node, a system must be put in place to manage the data collected by the sensor controller and package and transmit it to the Concentrator whenever it is ready to receive it.

For the Concentrator Node, a system must be developed that allows for it to communicate with and receive data from multiple sensor nodes, with as small a delay as possible.

#### Synchronizing data

The Concentrator Node must ensure that all the Sensor Nodes in the network remain synchronised with one another and that all the data it collects and logs is properly synced.

#### Data Logging

The Concentrator Node must be able to log all the data it receives in real-time while managing the new incoming data from multiple sensor nodes simultaneously. It must also format and convert the raw sensor readings from the Sensor Nodes into a more human-readable format before logging.

### 3.2.2 Interfacing with the sensors

This design utilises the onboard sensor controller engine of the LAUNCHXL-CC2650 to interface with the MPU-6050’s sensors. The sensor controller will interface with the MPU-6050 using I2C communication. This will allow the controller to write values to and read values from the MPU-6050 using just two lines and Serial Clock(SCL) line and a Serial Data(SDA) line. The sensor controller engine features an input buffer which allows it to actively receive input values from the main CPU. This input buffer allows for the sensor controller engine to be configured remotely. The sensor controller engine will initialise the MPU-6050 and configure the onboard sensor ranges and the filter bandwidth with the input values received from the main Sensor Node CPU. Figure 3.1 shows the configuration register for the MPU6050’s gyroscope, bit 3 and bit 4 control the FS\_SEL these bits allow us to control the range of the gyroscope using the values defined in Figure 3.2.

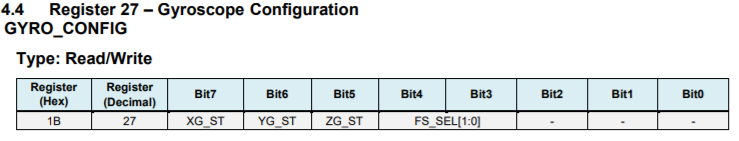
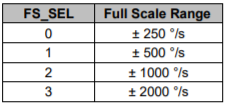


Figure 3.2- Gyroscope Range Select [15]

Figure 3.1 – Gyroscope Configuration Register [15]

These sensor ranges will be decided by and wirelessly transmitted from the main Concentrator Node whenever a new Sensor Node is connected to it. The Concentrator Node will also be able to configure all the connected Sensor Nodes in the network at once if a user wishes to update the sensor ranges or other parameters. As the sensor configuration registers and the values required for certain sensor ranges are known, the ability to configure the sensor ranges and settings could easily be tied to a button interrupt on the Concentrator Node. This would allow users to easily switch between different sensor ranges for different use cases. The sensor controller engine task could also be adapted actively to filter out different forces depending on the chosen sensor range. This feature would allow the network to easily adapt to different use cases such as limb motion tracking where an accelerometer range of ±8g would be sufficient to something like high-force collision detection where an accelerometer range ±16g would be more appropriate.

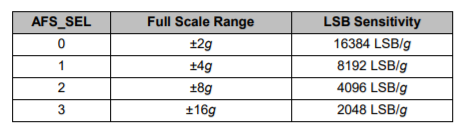
The chosen sensor range will also determine the Least Significant Bit (LSB) Sensitivity of the output values generated, see Figure 3.3. The raw data readings received by the Concentrator node will have to be scaled according to this value, but as the Concentrator node also sets this range it will be easy to determine which LSB Sensitivity modifier is appropriate.

Figure 3.3 – LSB Sensitivity

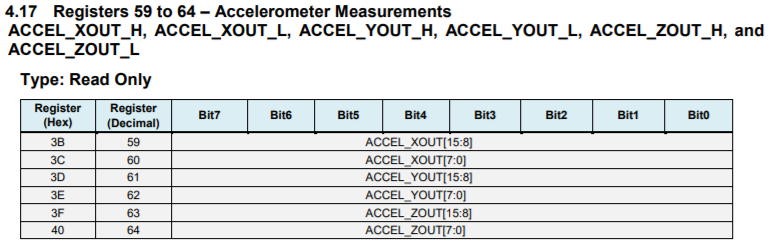


Figure 3.4- Accelerometer Result Register

The sensor controller also features an output buffer which allows the main Sensor Node CPU to easily access the data readings from the sensors. Figure 3.4 shows an example of the accelerometer result register, it shows that each axial reading is split into an upper and lower portion of the overall 16-bit 2s complement number. The sensor controller engine will read these values from the sensor result register then format them into singular outputs values for each axis. The same procedure will be carried out for both the accelerometer and gyroscope readings, this will allow the main CPU to directly transmit the data from the sensor controllers output buffers to the Concentrator Node without the need for extra formatting.

### 3.2.3 Wireless Communication

This design will implement a point to multi-point network topology. The Concentrator Node will be the central point that multiple Sensor Nodes will connect to. A set channel frequency will be selected for both the Concentrator and Sensor Nodes somewhere in the 2.4 GHz band as this is one of the most common frequencies used for this will allow for the system to have compatibility with a large number of other wireless communication devices. The frequencies around 2.4GHz are designated as the Industrial, Medical and Scientific bands, which this system would fall under as a fitness tracker[16]. Another benefit to this frequency is that they are free to use and no license is required to operate on them[16].

The Sensor Node will know the preset address of the Concentrator node, it will append this destination address and its personal address to the packet it creates alongside the packet’s payload of sensor data readings. The Sensor Node will then transmit this packet until it either receives an acknowledgement (ACK) packet from the Concentrator or until it has reached the max number of resend attempts that was set. If the transmission fails the node will flag that this packet’s transmission has failed and will move to a new packet. If the transmission succeeds the Sensor Node will flag the transmission as a success and move onto a new packet.

The Concentrator Node will actively listen for incoming packets on its set frequency. If a packet arrives the Node will check the destination address of the packet and if it matches its own set address (which will only be true for packets from the Sensor Nodes) it will accept the packet and respond to the Sensor Node with an ACK packet. The Concentrator Node will then store this Sensor Nodes now known address in a list of known Sensor Nodes. The Concentrator node will then be able to actively receive updates from this know sensor node while simultaneously listening for new nodes. Having a list of known sensor nodes will also allow the concentrator to actively store, track and manage the sensor data and synchronization information from each node.

Texas Instruments also provide a “TI Simple Link Starter” app that allows for a user to connect to supported TI devices via Bluetooth and WiFi. The software could be adapted to have two modes that could be toggled by a button interrupt. The first mode would be the default settings which was described above. The second would allow the Sensor Nodes to actively transmit data to the app, users could then review and monitor the raw sensor data values from their mobile phones.

### 3.2.4 Data Synchronisation

To maintain synchronisation in the network the Concentrator node will intermittently send a synchronisation packet to all the Sensor Nodes in the network. This method will reduce the strain of keeping synchronized on individual nodes in the system to a certain extent, but it does leave a timeframe between the Concentrator node’s synchronisation packets in which desync could occur. As a result, it will be necessary for the Concentrator node to analyse the timestamps on the Sensor Node data packets that it receives. The Concentrator will be able to compare the current sample from a particular Sensor Node to a past sample from the same node, then by comparing the values against its own internal clock it would be able to determine if desync has occurred at that particular node.

### 3.2.5 Data Logging

In order to log the data, it collects the Concentrator Node will have a memory card attached to it with a file for storing all the data collected. The data printing statements on the Concentrator will have to be adapted to format and print the data values recievd in a .csv format as then the values could easily be imported from the file into a program such as Microsoft Excel. Once the correct print statements are in place the console output of the Concentrator Node can then be redirected to print to that file on the memory card.

# Chapter 4- Implementation of Wireless Sensor Network

This chapter will discuss the Implementation of this wireless sensor network and will break it down into three main sections. General Implementation, will explain all features that were implemented in both the Concentrator Node and the Sensor Node. Sensor Node will explain the features unique to the Sensor Node and Concentrator Node will explain the features unique to the Concentrator Node. The full body of code for this project is available in the [Appendix.](#_Project_Code_Repository-)

## 4.1 General Implementation

### 4.1.1 Class Structure

Both the Sensor Node and the Concentrator Node have a very similar class structure with a few extra classes being present for the Sensor Node to manage the sensor controller engine. While they have a similar structure they operate slightly differently. Figure 4.1 shows the basic operation of each of the Nodes. Once either Node is powered it will initialize all of its tasks before starting the BIOS, However, they do differ slightly. The Sensor Node will initially be performing its sensor node task which waits on an event from its sensor controller, then collect the data readings when the event occurs and transmits them to the Concentrator, before returning to wait for a new sensor controller event. The Concentrator initially performs its radio task which involves it waiting for a Sensor Node packet to arrive before extracting the data from it and printing it to the console.

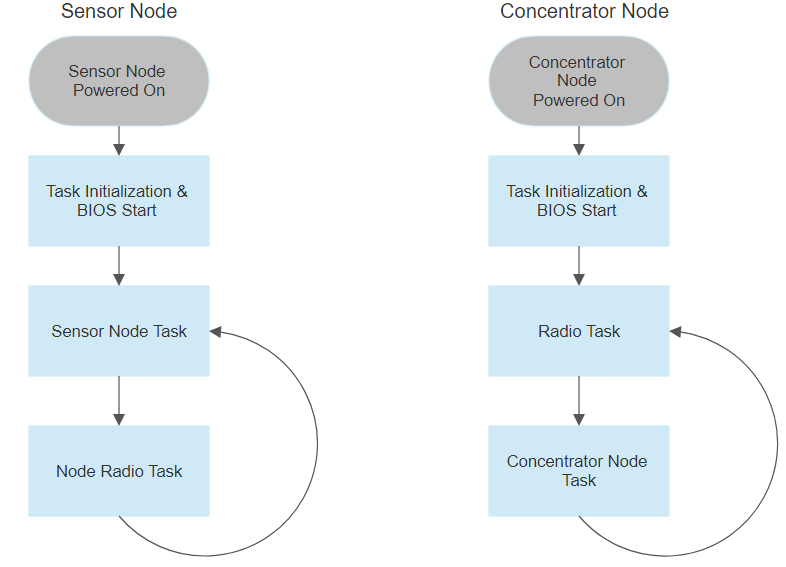


Figure 4.1- Basic Node Operation

While they differ slightly in operation the basic class structure is the same for both. A brief explanation will be given for each of the classes below, as they will be explained more in-depth for each node in their respective implementation sections.

#### rfWsn(Node/Concentrator).c

Figure 4.2- Initialization Functions

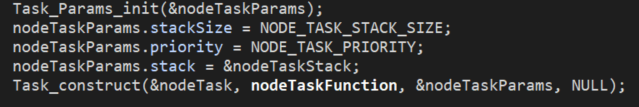
Figure 4.2 shows the int main() from the rfWsnNode.c class. This class performs the same function for both the Concentrator and Sensor Node. It begins by calling the general board initialization function, then it initializes both the radio communications task and the node task before starting the BIOS. The initialization tasks generally construct and initialize all the necessary Events and Semaphores that will be used by that task. The initialization task ends by constructing the task itself and assigning the task a priority and the space it can utilise in the stack. Figure 4.3 shows the initialization and construction of a task, the Task\_construct() function creates the task and prepares it to be run as soon as the system BIOS is started.

Figure 4.3 – Task Initialization

#### (Node/Concentrator)RadioTask

The radio task controls both of the nodes wireless communications protocol and setup. In the case of the Sensor Node, this task is used to transmit the sensor readings data it has collected to the Concentrator Node. In the case of the Concentrator Node, it actively listens for incoming packets, and once a packet arrives it passes it to its main task.

#### (Node/Concentrator)Task

This task performs the basic function of each of the nodes. The Concentrator node extracts the sensor reading data from the packet it has received and prints it to the console. The Sensor Node waits for a reading event from the sensor controller engine, it then stores the output data when an event occurs and passes it to its radio task to be sent to the Concentrator.

#### RadioProtocol.h

The Radio Protocol class is identical for both the nodes, it defines the address of the Concentrator, the RF driver settings that will be used which for this implementation was the “EasyLink\_Phy\_2\_4\_200kbps2gfsk” config, and the structure of all the packets that will be transmitted. First, a packet header struct was defined that consists of the Source Address of the packet and the packet type. The structs were then created that would store the sensor data produced from the sensor controller. Figure 4.4 shows an example of one of these structs where there are parameters to store the triaxial data from both the accelerometer and gyroscope.

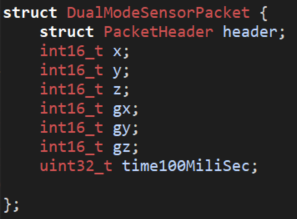


Figure 4.4 - DMPacket Struct

### 4.1.2 Radio Drivers and Smart RF Studio 7

This implementation uses the EasyLink API[17] and smartrf\_settings files generated from SmartRF Studio to set up and configures the RF Drivers on both nodes. SmartRF Studio allows users full customizability for their RF drivers and will generate custom smartrf\_settings.c and smartrf\_settings.h files based on the settings configured in it.

For some reason in this implementation, the custom smartrf\_settings files generated based on the testing done in [Chapter 5](#_5.4_SmartRF_Studio), would fail to compile in Code Composer Studio due to missing driver files and even when these driver files were re-downloaded and placed in the required directory the code would still fail to compile. As a result of this, the included “smartrf\_settings\_ predefined” files were used and the 2.4GHz 200kbps RF driver profile was chosen and adapted to meet the parameters decided upon from the testing in [Chapter 5](#_SmartRF_Studio_7). The parameters chosen from this testing were a frequency of 2435MHz and transmission power of 4dBm.

#### Code generated

Figure 4.4 shows the value set by SmartRF Studio when a transmission output power of $dBm is selected. Figure 4.5 shows the same parameter in the “smartrf\_settings\_ predefined.c” file in Code Composer Studio(CCS). Figure 4.6 shows the frequency being set to 2435MHz in the “smartrf\_settings\_ predefined.c”. These values were edited to match the radio parameters chosen in [Chapter 5](#_SmartRF_Studio_7) after this the driver files successfully compiled.

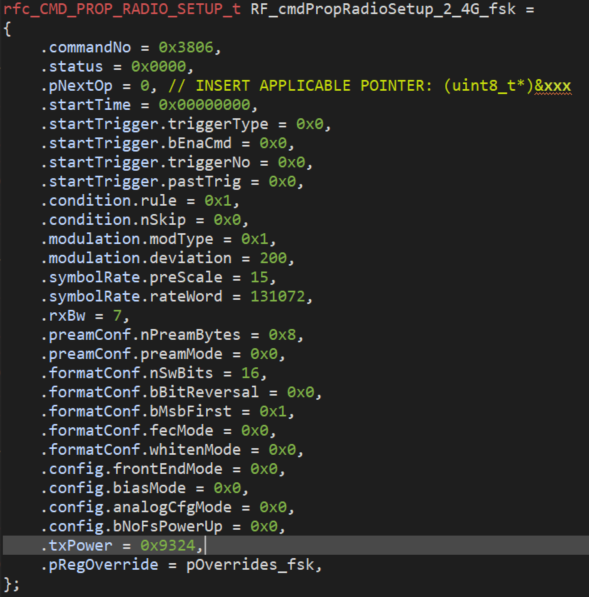
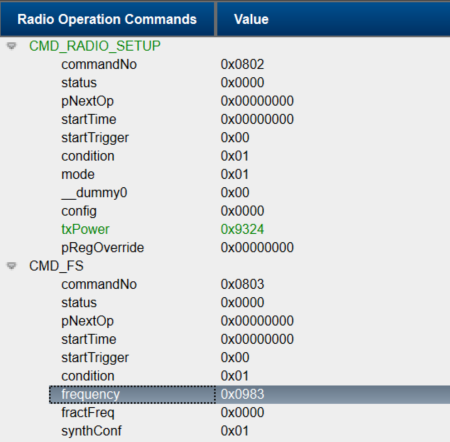
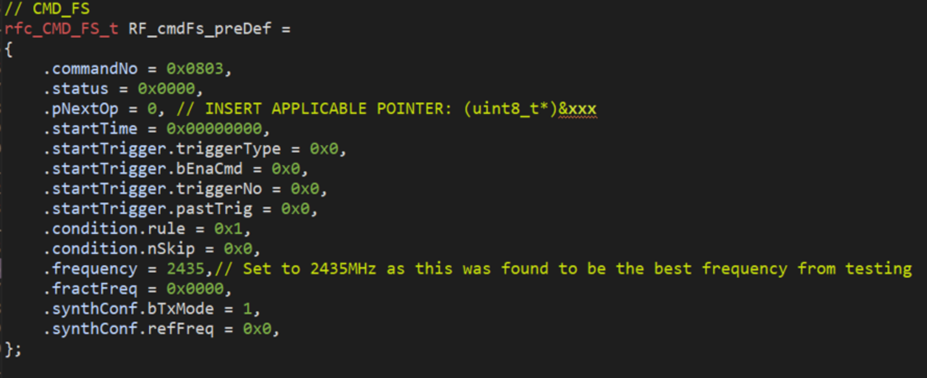


Figure 4.7 – Setting Frequency in CCS

Figure 4.5 - SmartRF Studio value for 4dBm

Figure 4.6 - Setting Tx Power in CCS

## 4.2 Sensor Node Implementation

This Sensor Node Implementation was derived from the “rfWsnNode Example” from SimpleLink Academy[18].

### Sensor Node Operation

This Section will explain the operation of the NodeTask and NodeRadioTask.

#### NodeTask

Figure 4.8 - Node Task Operation

Figure 4.8 shows the basic operation of the NodeTask. When the Node task begins it initializes and starts the Sensor Controller Engine and the LED Pins, as it uses an LED to indicate when sensor data is received or transmitted. The Node Task then enters its main loop, where it waits for a new NODE\_EVENT\_NEW\_MPU6050\_VALUE. This event is created by the function getMPU6050Data(), this function reads data from the sensor controllers output buffers using the call “scifTaskData.mpu6050Sample.output.x” to access the x value buffer for example. The ‘scifTaskData’ portion of the call refers to the task data being generated by the sensor controller, the “mpu6050Sample” is the name of the task, the “output” portion refers to the buffer you are accessing and the “x” portion is the specific data output in this buffer. Once the getMPU6050Data() function has collected all the output data it stores it in an array. The activity LED is then toggled and the data is sent to the radio task and a RADIO\_EVENT\_SEND\_MPU6050\_DATA event is posted.

#### NodeRadioTask

Figure 4.9 - Node Task Radio Operation

The Node Radio Task begins by initializing its RF Drivers using the EasyLink API, it then generates its own address that it will use for wireless communication using a random number generator. Once it has finished this initialization process it enters its main loop and waits for RADIO\_EVENT\_SEND\_MPU6050\_DATA to be posted. Once this event is posted it writes the data it was passed from the NodeTask into one of the packet structs that were defined in RadioProtocol.h. It then transmits this packet using the Easylink API’s EasyLink\_transmit() function. In order to send the packet struct containing the sensor readings with this function, the struct must be wrapped in an EasyLink\_TxPacket, this packet consists of a destination address which is set as the Concentrator address defined in RadioProtocol.h, a payload which will contain that data struct we created and length parameter which takes the length of the payload, this parameter can then be used later for error detection/correction. The task then attempts to send the Easylink packet to its destination address and waits for an ACK packet from the recipient. If the task receives an ACK packet in response it will post the status of transmission as a success and return to wait for a new RADIO\_EVENT\_SEND\_MPU6050\_DATA event to be posted. If the task does not receive an ACK packet in response it will attempt to resend the EasyLink packet for a set number of retries before it will post the status of transmission as a failure and return to wait for a new RADIO\_EVENT\_SEND\_MPU6050\_DATA event to be posted.

### 4.1.2 Sensor Controller

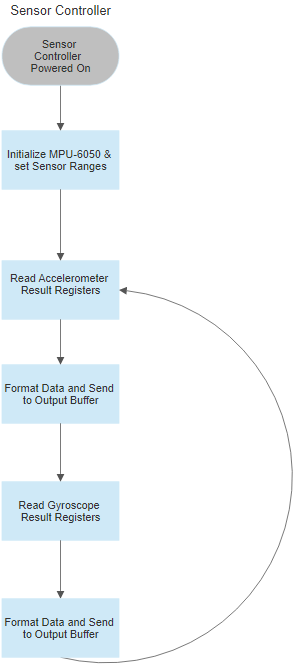
The CC2650’s built-in sensor controller engine was utilised for this project. This involved developing a program to run directly on the sensor controller using Sensor Controller Studio. The Sensor Controller Studio development was composed of three main parts, Constants, Data Structures and I/O Mapping, Initialization Code, Execution Code. The Sensor Controller also required the inclusion of two additional files in the Code Composer Studio project Sce.c and Sce.h these files initialize the drivers generated by Sensor Controller Studio and start the Sensor Controller. Figure 4.10 below shows then the basic operation of the Sensor Controller.

Figure 4.10 - Sensor Controller Operation

#### Constants, Data Structures and I/O Mapping

The Constants and Data Structures section is shown in the Figure below it required the user to define all constant values, input and output buffers that would be used in the program. The required MPU-6050 address’ and set some set values required to configure them were created as constants. The required output buffers for each of the sensor readings were created x,y and z for the accelerometer’s outputs and gx, gy and gz for the gyroscope’s outputs. The I2C pins of the CC2650 DIO4(SCL) and DIO5(SDA) were then defined in the I/O mapping section

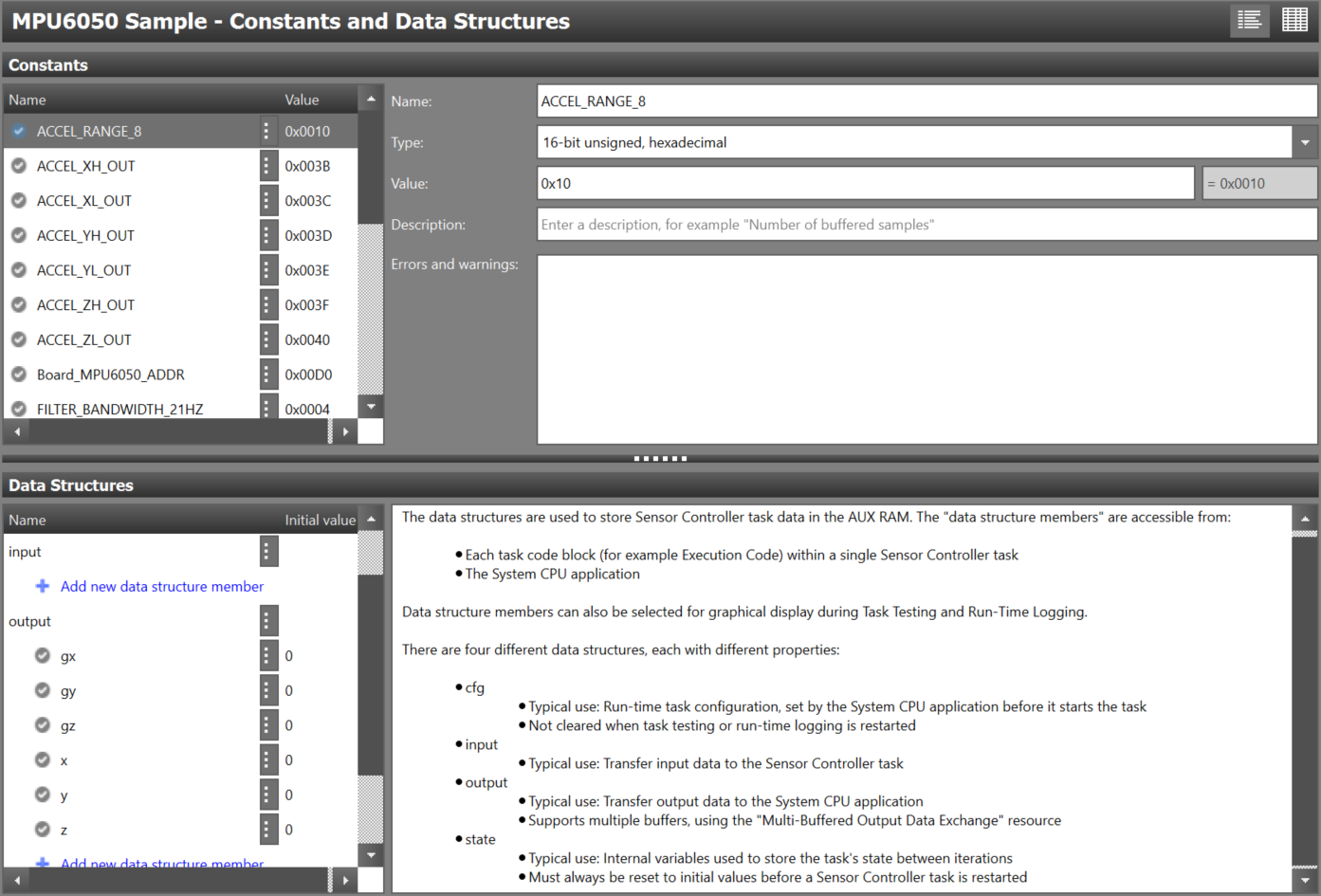


Figure 4.11 – Constants and Data Structures

The table on the next page lists some of the data constants sets and their purposes.

|  |  |
| --- | --- |
| Constant | Purpose |
| ACCEL\_RANGE\_8 | Constant to set Accelerometer range to 8g when in the Accelerometer Configuration Register |
| ACCEL\_XH\_OUT | This is the first register of the Accelerometers Result Registers |
| Board\_MPU6050\_ADDR | This is the main I2C address of the MPU-6050 |
| FILTER\_BANDWIDTH\_21HZ | Constant to set DLPF bandwidth to 21Hz when in the main MPU Configuration Register |
| GYRO\_RANGE\_500 | Constant to set Gyroscope range to 500 when in the Gyroscope Configuration Register |
| GYRO\_XH\_OUT | This is the first register of the Gyroscopes Result Registers |
| MPU6050\_ACCEL\_CONFIG | Address of the Accelerometer Configuration Cegister |
| MPU6050\_CONFIG | Address of the main MPU Configuration Register |
| MPU6050\_GYRO\_CONFIG | Address of the Gyroscope Configuration Register |
| MPU6050\_PWR\_MGMT\_1 | Address of the Power Management Register |
| PWR\_MGMT\_1\_RESET | Constant to reset the entire MPU when in the Power Management Register |
| PWR\_MGMT\_1\_WAKE | Constant to wake the MPU from sleep when in the Power Management Register |
| SIGNAL\_PATH\_RESET | Address of the Signal Path Register |
| SIGNAL\_PATH\_RESET\_ALL | Constant to reset the all signal paths when in the Signal Path Register |

#### Initialization Code

Figure 4.12 – Sensor Controller Initialisation

Developing functional initialization code was at first was one of the more challenging aspects of this project as it was unclear what values needed to be set at boot-up for the MPU-6050 to operate correctly. However, the Adafruit MPU-6050 has a custom Arduino library and example-set developed by Adafruit. By examining these drivers and observing the boot process for one of the examples using Scopy a digital logic analyzer[[4]](#footnote-4), the values required to initialise the MPU 6050 became clear. Figure 4.11 shows the final version of the initialization code for the Sensor Controller. The code first resets the entire MPU before resetting the signal paths to sensors. It then configures the sensor ranges desired ±8g for the accelerometer and ±500 degrees/sec for the gyroscope. Finally, it configures the bandwidth of the Discrete Low Pass Filter and now that the MPU is fully configured it wakes it from its low power sleep state.

#### Execution Code

Figure 4.13- Execution Code

Figure 4.12 shows the accelerometer section of the execution code for the Sensor Controller, the code for accessing the gyroscope follows the same principles but using the gyroscopes registers. The code handles the reading of the sensor data from the MPU-6050’s sensor result registers. It first writes the address of the first result register of the accelerometer to the MPU-6050 then using the “i2cRepeatedStart()” function it reads the values from each accelerometer result register consecutively using the “i2cRxAck()” and stores them in the variable passed in the Rx function [19]. For the final register the function the ” i2cRxNack()” is used as we no longer need to send an acknowledgement after our read as we are now ending the I2C communication. The 8-bit values from the upper and lower portions of each axis’s result registers are then formatted to form a 16 bit 2’s complement number that represents the actual output of that axis’s sensor. These formatted representations of the axis values are then written to their respective output buffers.

#### Sensor Node Console Output

Figure 4.14 below shows a sample of the output values the main sensor node CPU receives from the sensor controller. The values produced are in line with those that were observed when testing the Sensor Controller task in Sensor Controller Engine, see Figure 4.15.

## 

Figure 4.15 – Sensor Controller Studio Output Sample

Figure 4.14 - Sensor Node Output Sample

## 4.3 Concentrator Node Implementation

This Concentrator Node Implementation was derived from the “rfWsnConcentrator Example” from SimpleLink Academy[20].

### 4.3.1 Concentrator Node Operation

This Section will explain the operation of the NodeTask and NodeRadioTask.

#### ConcentratorRadioTask

Figure 4.16 - Concentrator Radio Task Operation

Figure 4.16 shows the basic operation of the Concentrator Radio Task. The task begins by initializing its RF drivers and configuration using the EasyLink API. It then sets up its own address using RADIO\_CONCENTRATOR\_ADDRESS that was set in Radio Protocol.h, and creates an ACK packet to send in response to any incoming packets it may receive. It then enters its main loop where it listens for all incoming packets and waits for a RADIO\_EVENT\_VALID\_PACKET\_RECEIVED event to be posted. This event is only posted if the packet received by the Concentrator, has the correct destination address that matches that of the concentrator and if its payload contains one of the known packet data structs declared in RadioProtocol.h. If the packet is valid the RADIO\_EVENT\_VALID\_PACKET\_RECEIVED event will be posted and the Concentrator will then send an ACK packet back to the source address of the packet before passing the packet and its payload to the ConcetratorTask. The ConcetratorRadio Task then Flags the packet reception as a success and returns to waiting for the next valid packet event.

#### Concentrator Task

Figure 4.17- Concentrator Task Operation

Figure 4.17 show the basic operation of the Concentrator Task. The Concentrator Task begins by waiting for a CONCENTRATOR\_EVENT\_NEW\_MPU6050\_SENSOR\_VALUE event to be posted. This event is posted by the packetRecievedCallback() function which is called by the Concentrator Radio Task when it knows a valid packet has been received. Once the event has been posted the task will check if the node that sent the newly received packet is already known i.e. it is stored in the array of known sensor nodes which is composed of the sensor node structs defined in RadioProtcol.h. If the Sensor node is known the task will update the values associated with that Sensor node in the array, if the Sensor node is not known it will be added to the array of know Sensor nodes. The Task then calls the PrintValues() function which prints all the values from all of the known Sensor Nodes in the network.

Figure 4.18 show the PrintValues() function. The function creates a pointer to the array of know sensor nodes, then loops through the array of known Sensor nodes while the current node pointer does not equal the last position in the array. It then prints every value associated with the current node in the array. The print statement also applies the necessary scaling factor to each of the raw sensor data values. Once all the values have been printed it increments to the next node in the array or ends the loop if it is at the end of the arry.

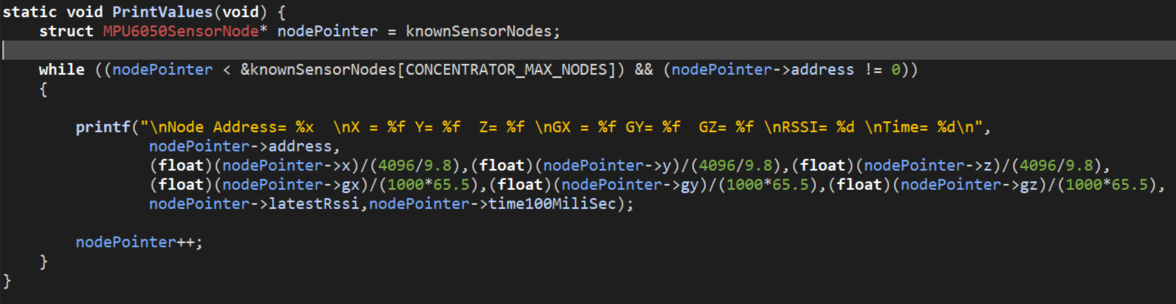


Figure 4.18- Print Values Function

#### Concentrator Node Console Output

Figure 4.19 shows the console output of the Concentrator Node Task when it is receiving real-time sensor readings from the Sensor Node, this is representative of the actual operation of this Wireless Sensor network. Figure 4.20 shows the console output of the Concentrator Node Task when it is receiving a string of consecutive numbers from the Sensor Node, this is a test demonstration to show that the Concentrator is correctly receiving consecutive information from the Sensor Node.

## 

Figure 4.20 - Concentrator Output for Data Test

Figure 4.19 - Concentrator Output for Sensor Readings

## 

# Chapter 5 - Testing of Wireless Sensor Network & Results

This chapter explores the functional performance testing approaches undertaken during the development of this wireless sensor network

## 5.1 Scopy

Scopy a software toolset used for signal analysis, it has a Logic Analyzer and i2c decoder tool that can allow a user to monitor I2C communications between two devices. During the early stages of the development of the I2C communication between the CC2650 and the MPU the ability to actively monitor and view the I2C communication as it took place was hugely beneficial. Scopy made it easy to identify failed Communication attempts , such as in Figures 5.1 and 5.2.

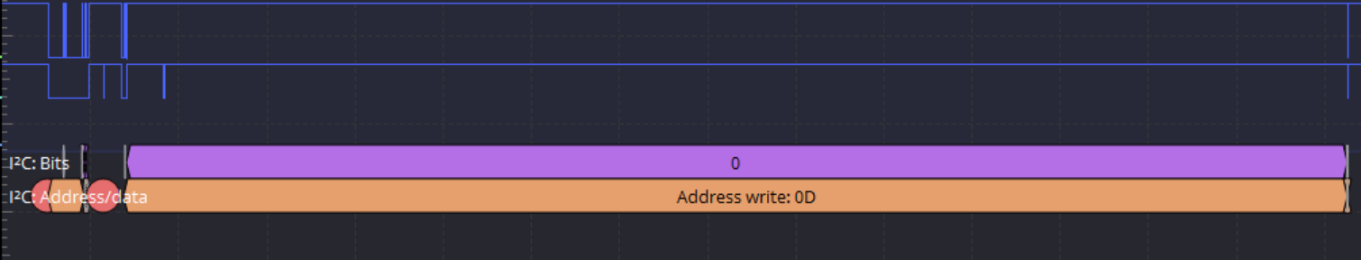
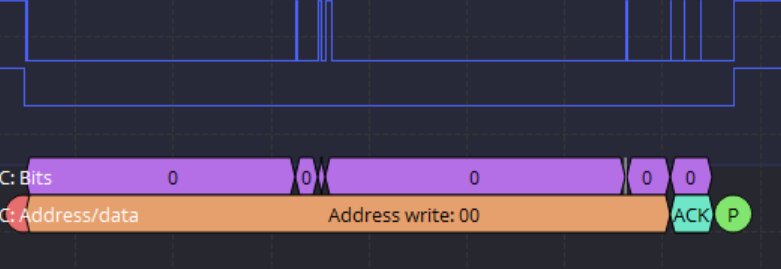


Figure 5.2- Failed I2C Communication Example 2

Figure 5.1 – Failed I2C Communication Example

Scopy also allowed for the comparison of this projects boot processes and data reads against the same processes but being executed by the official Arduino library that was created for the Adafruit MPU-6050. Figure 5.3 shows this project’s implementation of a burst data read of the result registers of the MPU-6050, the break between the accelerometer burst read and the gyroscope burst read can clearly be seen. The accelerometer read takes place after the write of 0x3B the address of the first accelerometer result register it burst reads all 6 of the accelerometer result registers before ending at the red stop circle in the center of the picture. The gyroscope register then performs the same operation terminating at the end of the image.

This approach is slightly different to that implemented by the ardunio communication in Figure 5.4. The Arduino implementation simply writes the address of the first accelerometer register then burst reads all the remaining sensor results registers (including two additional temperature registers that were not utilised in the project).

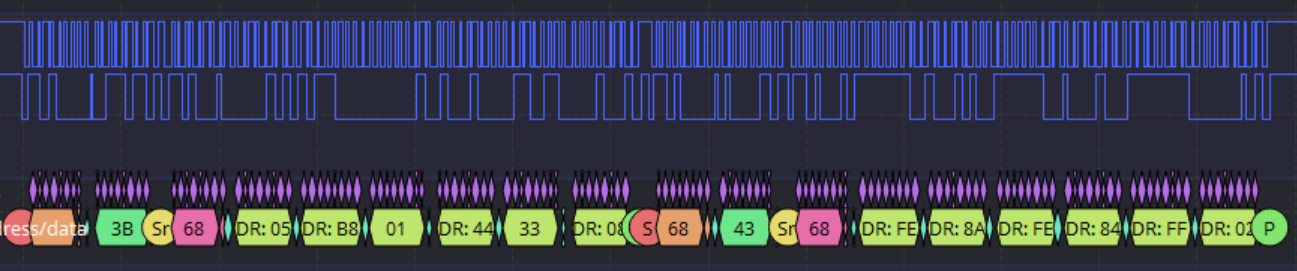


Figure 5.3 – Sensor Controller Burst Read

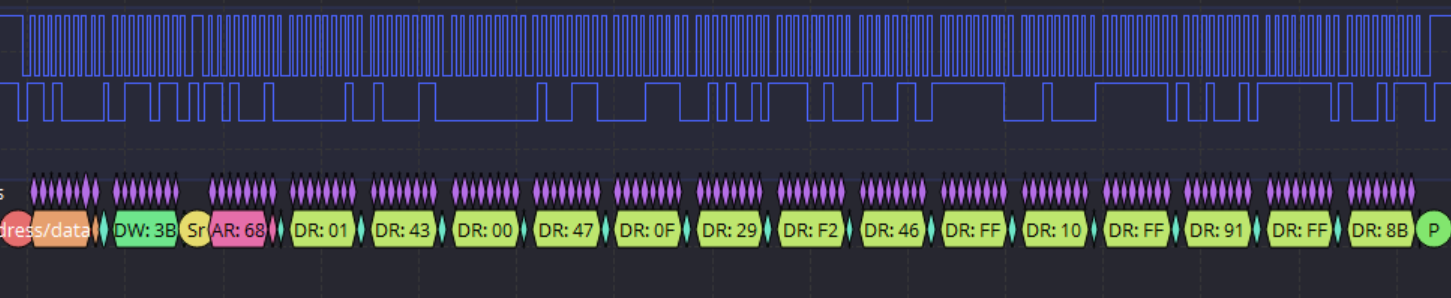


Figure 5.4 - Arduino Burst Read

## 5.2 Sensor Controller Studio

To ensure that all the sensors on the MPU-6050 were functioning correctly their output values were monitored using Sensor Controller Studio. Sensor Controller Studio displays the outputs of the sensors in real-time and can plot the values on a graph for easy analysis.

This made it easy to spot any irregularities such as a sensor not producing any input or a sensor going out of range. Figure 5.5 shows an example of an acting sensor erratically by wildly spiking in and out of range (up to a value of 65340). In this case, the erratic behaviour was due to the y output buffer being defined as an unsigned decimal instead of a signed one, the output value spike was due to whenever the y value fell below zero it would cause the sensor to seemingly go out of range. Figure 5.6 shows the y value behaving much more normally when the output buffer was redefined as a signed decimal.

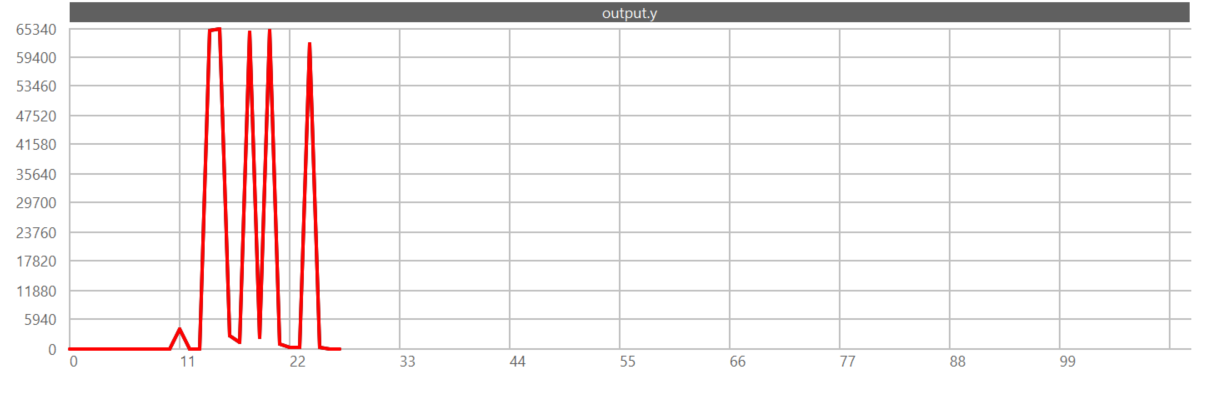


Figure 5.5 – Accelerometer y-axis spiking erratically

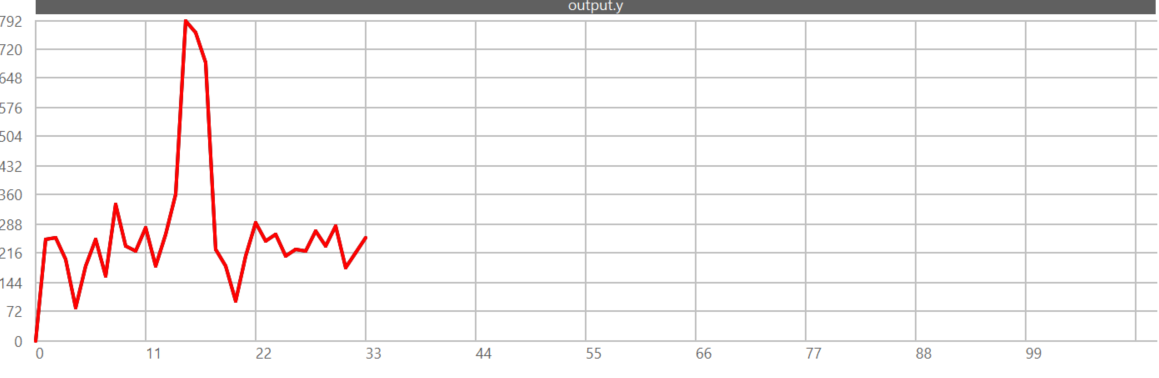


Figure 5.6 - Accelerometer y-axis normal behaviour

## 5.3 SmartRF Studio 7

SmartRF Studio 7’s inbuilt testing functions were used to test the performance metrics of the radio network implementation for this system. They allowed for testing of performance metrics such as RSSI and the testing of Packet Error Rates at different distances/Tx Powers. These testing functions also allowed an optimal frequency with the least noise in the testing environment to be found.

### 5.3.1 Received Signal Strength Indicator (RSSI)

The Received Signal Strength Indicator (RSSI) Test was performed using SmartRF Studio’s inbuilt “Continuous Rx” and “Continuous Tx” applications. The “Continuous Rx” function allows the user to continuously monitor the energy levels at a given frequency, on its own this function can allow a user to easily identify the frequencies with the least amount of noise and traffic which would be optimal for testing purposes. The “Continuous Tx” function allows the user to continuously transmit a modulated or unmodulated carrier signal, it also allows the user to set up frequency sweeps[21]. When used together the “Continuous Tx” and “Continuous Rx” functions can be useful for determining the sensitivity of a device to interference from nearby channels and for measuring antenna bandwidth.

#### Selecting a suitable Frequency

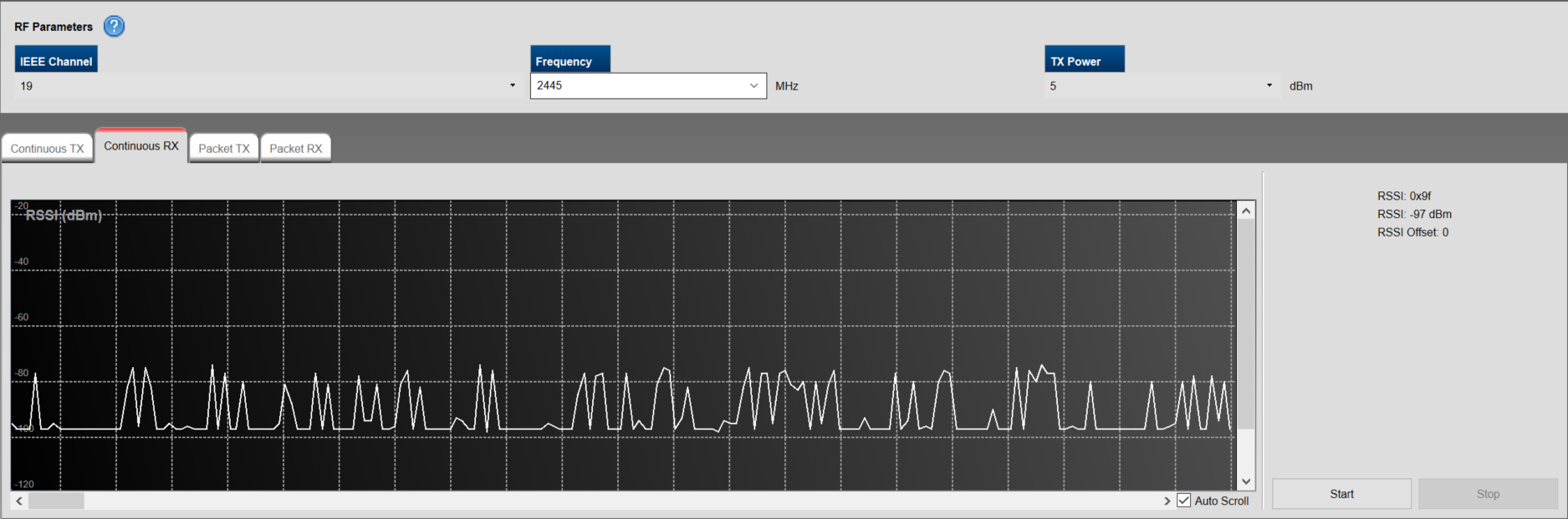
To ensure an optimal testing environment the multiple frequencies in the 2.4 GHz band were sampled using the “Continuous RX” function. Many frequencies were found to have seemingly random noise spikes and interference present at varying intervals, these frequencies were then ruled out as suitable for this use case, see Figure 5.7.

Figure 5.7- 2445MHz Noise Spikes

The first suitable frequency for testing was found to be 2435MHz, this frequency had the least interference and noise spikes out of all the tested frequencies, see Figure 5.8.



Figure 5.8 – 2435MHz low noise levels

#### Received Signal Strength Indicator(RSSI) Tests

The RSSI of the system was tested using a PC and a laptop to allow for the testing of different Tx Powers at varying distances. The purpose of this test was to determine the RSSI values that could be achieved with different Tx Powers for set distances. The goal was to find the lowest Tx Power that could provide a sufficient RSSI value at the maximum range that this system would require. Finding the lowest sufficient Tx Power should decrease the overall power draw of the system and extend its battery life. The range of Tx Powers selectable for the CC2650 in SmartRF Studio was {-21,-18,-15,-12,-9,-6,-3,0,1,2,3,4,5}dBm each of these powers was tested.

A value of -67 dBm was chosen as the minimum suitable RSSI value for the network as it is considered the “Minimum signal strength for applications that require very reliable, timely delivery of data packets”[22].

#### Testing RSSI at 1m

The first test performed measured the RSSI values of different Tx Powers at a distance of 1m. A distance of 1m was chosen as this is most likely slightly above the normal distance that a sensor node would be positioned from the concentrator when being used for limb tracking.

**Results**

The first test showed that almost all Tx Powers were above the minimum suitable RSSI of

-67dBm at a distance of 1m. The next step was to test the same values again to ensure they could meet the maximum performance requirements of the network.

#### Testing RSSI at 2m

The second test involved retesting the Tx Powers from the first test at a distance of 2m. While it is highly unlikely that a sensor node would ever be at a distance greater than 2m from the concentrator, this test ensured that the network would be able to operate at distances up to and beyond what would be realistically required of it.

**Results**

The second test showed that the Tx Powers {-21,-18,-15} dBm all failed to meet the minimum required signal strength for the network. Whereas the Tx Powers{-12,-9,-6,-3,0,1,2,3,4,5} dBm all made met the minimum required RSSI of -67dBm.

### 5.3.2 Packet Error Rate Testing

The Packet Error Rate Test was performed using SmartRF Studio’s inbuilt “Packet Tx” and “Packet Rx” functions. The purpose of this test was to gain further insight into the performance metrics that could be expected of different Tx Power levels and to discover the lowest Tx Power that could provide little to no packet loss to the network within the distances that the network would be required to operate.

The “Packet Tx” function allows the user to create a packet with a random or user-defined payload such as a string or and then include a sequence number in the payload. The function can then be configured to send a set number of packets and at a set interval of ms.

The “Packet Rx” function can be configured to receive a set number of packets and using the sequence number attached to each packet it can recognize whether a packet was received ok or not received ok. The function counts each occurrence of a packet received or not received and displays this to the user. The function also accounts for packets that were received but have a Cyclic Redundancy Check (CRC) Error i.e. the packet data or a bit in it was corrupted at some point, the function records and measures this as the Bit Error Rate. This information alongside the Packet OK and Packet Not OK counts can allow a user to determine the performance of the network and to find the overall Packet Error Rate(P.E.R) using the formula[23]:

#### Testing Packet Error Rate over Distances

The first Packet Error Rate test was performed using the chosen 2435MHz frequency by testing the maximum Tx Power of 5 dBm at various distances to discover if and when packet loss occurs. A test string was created in the form of a mock accelerometer reading: “X= 0.15 Y= -0.09 Z= 8.76 m/s^2” this string was used as the payload for each packet. For each test, ~1000 packets were sent across the network the results were recorded. Figure 5.9 below shows the “Packet Tx” setup for this test.

Figure 5.9 – Packet Tx Function

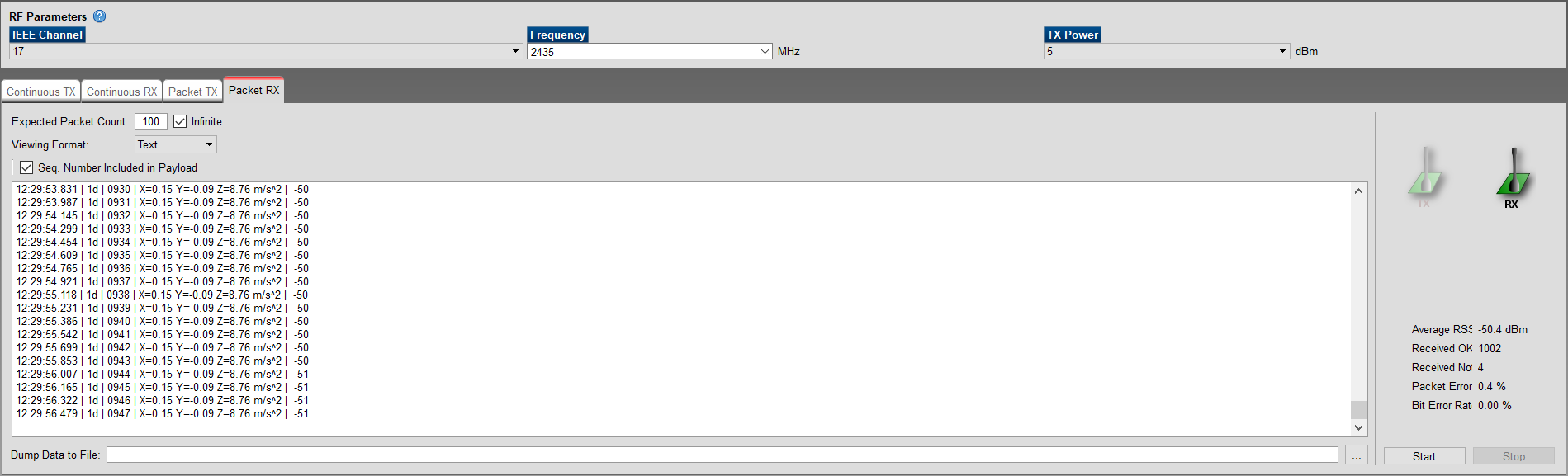
Figure 5.10 below shows the results shown in the “Packet Rx” function from testing the network at a distance of 5m. Each Packet that was received is displayed in the large text box in the centre of the figure. All packets that are received are printed with the time of reception, the length of the packet ‘1d’ ( the hex value for 29 in this case), the sequence number and finally the string payload of the packet. The Average RSSI, Packets Received OK, Packets Received Not OK, Packet Error Rate and Bit Error Rate can be seen on the Right-hand side.

Figure 5.10- Packet Rx Function

**Results**

The result for the test at 1, 2 and 5 metres are shown in the table below. These results show that there was no packet loss experienced in the network over distances of 1 and 2 metres, however, when the system was tested at 5m packet loss did occur.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tx Power** | **Distance** | **Packets OK** | **Packets NOK** | **Packet Error Rate** | **Bit Error Rate** | **Average RSSI** |
| 5 dBm | 1m | 1000 | 0 | 0% | 0% | -36.9 dBm |
| 5 dBm | 2m | 1000 | 0 | 0% | 0% | -43.7 dBm |
| 5 dBm | 5m | 996 | 4 | 0.4% | 0% | -50.4 dBm |

#### Testing Packet Loss with Different Tx Powers at 2 metres

The second test was performed using the same frequency of 2435MHz and payload string of “X= 0.15 Y= -0.09 Z= 8.76 m/s^2” however, for this test a range of different values were used for the Tx Power, additionally, the tests were all performed at a fixed distance of 2m. This distance was chosen as the maximum distance requirement for the network so it is necessary to understand the performance of different Tx Power levels at this range. The main goal of this test was to find the lowest Tx Power that could provide little to no packet loss across the network at this maximum distance.

**Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **TX Power** | **Distance** | **Packets**  **OK** | **Packets**  **NOK** | **Packet Error Rate** | **Bit Error Rate** | **Average RSSI** |
| 5 dBm | 2m | 500 | 0 | 0% | 0% | -37.2 dBm |
| 4 dBm | 2m | 500 | 0 | 0% | 0% | -39.1 dBm |
| 3 dBm | 2m | 499 | 1 | 0.002% | 0% | -45.6 dBm |
| 0 dBm | 2m | 498 | 2 | 0.2% | 0% | -49.2 dBm |
| -9 dBm | 2m | 495 | 5 | 0.4% | 0% | -61.7 dBm |
| -21 dBm | 500 | 495 | 5 | 1.0% | 0% | -79.7 dBm |

The results from this test show that while most Tx Power levels performed very well with Packet Error Rate’s less than 1%, only 4 dBm and 5 dBm can provide the network with a Packet Error Rate of 0%.

### 5.3.3 Overall Results from Smart RF Studio Tests

These tests were performed in a relatively ideal scenario for the network, where each node was stationary and there was a clear line of sight between the two points in the network. What these test didn’t account for is many of the likely scenarios the network will encounter while in use for actual motion tracking. In these scenarios, the points in the network will be moving towards and away from one another and they will more than likely not have a clear line of sight to other points in the network, as they are likely to be obstructed by a users limbs or torso. However, these tests allow us to rule out the Tx Powers that did not perform perfectly in even these ideal scenarios as their performance would likely worsen when used in non-ideal scenarios. As a result of this 4dBm would likely be the most suitable Tx Power for this implementation as it suffered no packet loss in any of the tests and maintained a strong RSSI at the maximum distance requirement of the system.

### 5.3.4 Eliminating Sources of Interference/Noise

To ensure that network was tested accurately it was imperative to remove as many sources of noise from the testing environment as possible to provide an ideal scenario for the network. During the testing phase, many sources of Noise were discovered. To eliminate this noise and interference all Bluetooth and other wirelessly communicating devices were removed from the testing environment. Some of the Issue related to noise and interference are explored further below.

#### Channel Interference

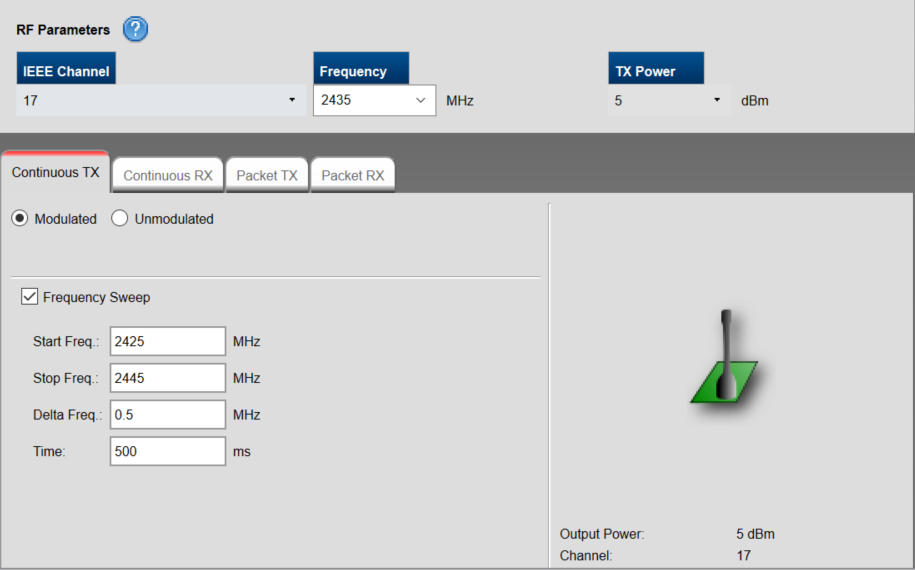
A frequency sweep was performed around the chosen 2435MHz using the “Continuous TX” function. The second node in the network performed a “Continuous RX” at the centre 2435MHz frequency. The sweep tested the sensitivity of the antenna to noise from nearby frequencies. The sweep started at 2425 MHZ and increased in frequency by 0.5 MHz every 500ms, see Figure 5.11.

Figure 5.11 – Frequency Sweep Setup

The frequency sweep showed that the network was susceptible to noise from nearby frequencies with the antenna receiving a “good ” signal strength of greater than -67 dBm from all frequencies within 2.5MHz of its set frequency of 2435MHz, see Figure 5.12.

Figure 5.12- Antenna Bandwidth

#### Sources of Noise

During the Packet Error Rate testing phase it was discovered that in certain circumstances erroneous packets were received and accepted by the CC2650. These packets would have a very high sequence number and a seemingly random character sequence as their payload, see Figure 5.13. The high sequence number of the packet would then cause the “Packet Rx” function to believe it has missed all the packets between its current actual sequence number and the erroneous high sequence number. This would cause the function’s estimation of packet loss in the network to be much higher than it was in reality.

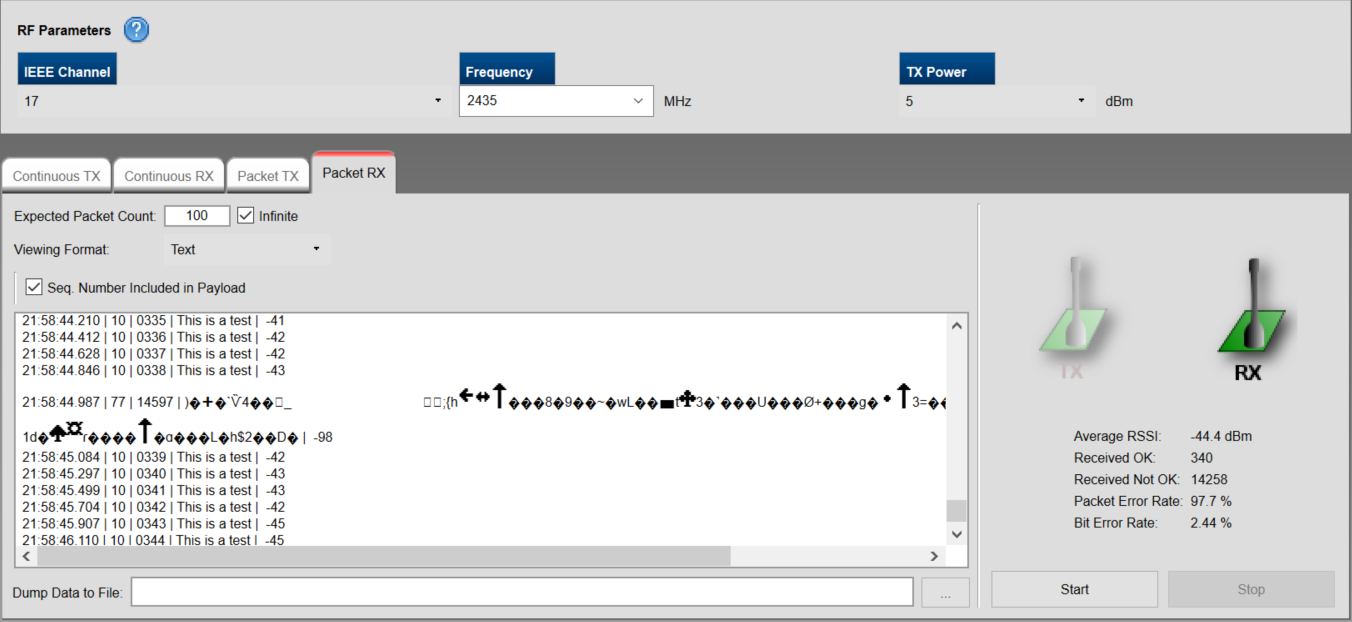


Figure 5.13- Erroneous Packet

Through further testing, it was discovered that the erroneous packet was likely caused by a Fitbit smartwatch. Whenever the watch was close to one of the nodes the erroneous packet would appear. When the watch was removed from the testing environment the packet was no longer detected.

# Chapter 6 – Ethics

## 6.1 Sustainability

The topic of sustainability and environmental impact raised some ethical concerns over the course of this project. The main ethical issues in relation to sustainability in this project can be broken down as

* Sustainability in regards to the efficient management of resources.[24]
* Sustainability as the preservation of wildness, with the rejection of the ethic of human use in favour of the ethic of human respect and non-interference [24]

There are two main areas in this project that raise concerns over these ethical issues. The first is over the use of toxic substances such as lead in the manufacturing of electronic components, this raises concerns over the effect the toxic substances could have on wildlife or the environment during the manufacturing process or when the product is disposed of.

The second area for concern is the packaging waste produced by the products used. It is important to use products from companies who ensure their packaging and shipping methods are as environmentally friendly as possible. This raises concerns over the sustainability of the products in regards to the efficient management of resources and the effect on wildlife and the environment.

### 6.1.1 Lead and Toxic Substances

Electronic waste is becoming a growing problem in many countries worldwide[25].

Many electronic components are comprised of both valuable materials such as gold and silver and toxic substances such as lead. In the past lead has been a vital component in many electronic devices, it was often used in batteries, solders and metal alloys.

In 2003 the EU introduced the Restriction of Hazardous Substances Directive (RoHS 1). This directive restricted the use of ten hazardous materials including lead in the manufacturing of electronics and electronic components. In July 2011 the Eu introduced the (RoHS 2) directive evolution of the original directive. This made it a legal requirement to demonstrate conformity to the directive through detailed files and documentation for products in its scope, failure to implement these measures during production was made a criminal offence[26].

As a result of these concerns, the LAUNCHXL-CC2650 from Texas Instruments(TI) was chosen for this project. Since the 1980s TI has been converting its products to lead-free alternatives[27]. Today the majority of TI’s lead-free products use annealed matte tin or Tin/Silver/Copper alloys as alternatives to lead. Texas Instruments also provide full materials breakdowns for all of their products[[5]](#footnote-5). This allows users to make informed choices on the products they are purchasing. The CC2650 which was chosen for this project is compliant with both RoHS 1 and RoHS 2.

While during the introduction of the RoHS directive and the initial phase of the move from lead-based to lead-free products there were greater losses and expenses incurred by companies as they adapted their old manufacturing methods to comply with the new standards. These days as more and more companies have developed lead-free solutions, the cost of moving to a lead-free platform has significantly decreased and in some cases can be a cheaper alternative due to the difficulty of obtaining an exemption from these directives.

### 6.1.2 Packaging Waste and Recyclability

Due to the sensitive nature and fragility, many electronic components are very well packaged in order to protect the devices. However, this can lead to a large amount of waste being produced when the device arrives at its destination.

Texas Instruments try to only use packaging suppliers who use environmentally-friendly materials in their products. This ensures the packaging of their products will have as small an environmental impact as possible. They also provide a comprehensive list of the composition of all the packing materials they use [28]. This full transparency for the materials used in the packaging can allow a customer to make an informed choice when purchasing one of their products. The LAUNCHXL-CC2650 was packaged in a recyclable anti-static bag and a full cardboard box which is also easily recyclable.

In the past, the impact of using easily recyclable materials for product packaging could raise costs of shipping and packaging of a product however, due to the worldwide desire for making more products easily recyclable in many cases switching to recyclable materials can cut costs for product shipping and packaging.

## 6.2 Potential for Military Use

The potential for the adaption of elements of this project for military or weapons use case is another area of concern. In 1997 Texas Instruments sold their defence operations contract with the United States military to Raytheon for $2.95 billion, before this they utilised their knowledge of integrated circuitry sensor, and embedded systems to develop top of the line missile guidance systems[29]. Some of these systems likely utilised early predecessors to the CC2650 chip used in this project. As a result, they likely had similar wireless communications and accelerometer and gyroscope based sensor readings to those used in this project, albeit at a much more advanced level and alongside numerous other sensors. However, this still leaves the possibility that this project could be adapted in some way to suit a military or weapons use case.

While the issue of military use is a concerning one by itself, the potential use case for this product in missile guidance systems is particularly worrying. These technologies challenge the principles of Discrimination and Proportionality in War. Discrimination refers to who is the actual target of an attack and proportionality refers to how much force is appropriate to use in a given scenario[30]. Due to their nature missiles systems do not allow for discrimination to an extent that one can guarantee no unnecessary loss of human life and no unintended casualties. There are also many scenarios where the proportionality of using missiles and the justifications of them as an appropriate level of force are extremely questionable.

One method Texas Instruments use for avoiding the potential for its products to be used for weapons or military purposes without prior agreement is their terms and condition which must be accepted before downloading any of their proprietary software or receiving samples of their products[31]. Users must verify that are using the product/software for only civil purposes only or they will be denied access to the product or software. This is one method that could be utilised to safeguard this project against the potential for military/weapons use. Preventing a user from accessing any information on the project until they agree to the terms and conditions forbidding military use could alleviate some of the concerns over this issue.

The potential impact of removing this project from the military and weapons markets is a loss in the opportunity for lucrative military contracts for non-weapons use cases such as sensor networks or transport and shipping applications. However, removing it from this market is justifiable in order to prevent the project from misused for weapons purposes.

# Chapter 7 - Conclusions and Further Research

While the current form of the implementation of this project meets many of the aims/deliverables that were set out at the beginning of this project, there are some areas in which it falls short of these goals. In its current form, the system is capable of the real-time motion capture of data, and while it currently does not store the data on board with a small amount more development time it could easily be adapted to stream the data to an external source which could be used for data handling or to store it in an onboard memory card. This also holds true for some of the other features laid out in the Design in [Chapter 3](#_Chapter_3_-_1), such as developing the software to be actively adaptable using the input buffers of the Sensor Controller Engine, a feature like this would make the system extremely versatile and useful in sporting markets where there can be a need to track both low force movement and high force collisions.

Future areas for potential development in this project are in the analysis of the motion data collected, to help the system identify patterns and learn the movements that these patterns describe. Another potential area for development is the system's compatibility with other devices, as this device operates in the public 2.4 GHz frequency band it has the potential to interface with the numerous other smart devices that operate in this band. It could be possible to develop an interface to allow the sensor data collected to analysed and understood by some of the existing smart tracking apps. This project could also further developed to be easily interfaceable with a variety of different sensors.

# References

[1] ADPM Wearable Technologies, “The Opal”. (5 March 2021)

<https://www.apdm.com/wearable-sensors/>

[2] VICON, “Blue Trident” . (5 March 2021)

<https://www.vicon.com/cms/wp-content/uploads/2019/09/PS3439_Blue-Trident-factsheet_Final.pdf>

[3] MbientLab, “MetaTracker (MTR)”. (5 March 2021)

<https://mbientlab.com/metatracker/>

[4] NORAXON, “myoMOTION Portable Motion Capture”. (5 March 2021)

<https://www.noraxon.com/noraxon-download/portable-motion-capture-brochure/>

NORAXON, “myoMOTION Sensor and Receiver User Manual” (5 March 2021)

<https://www.noraxon.com/noraxon-download/myomotion-system-user-manual/>

[5] GaitUp “Physilog 5” 20.02.2019 - Physilog 5 datasheet.pdf - v1.03 - Copyright© 2019 Gait Up SA. CH-1015 SWITZERLAND. (5 March 2021)

<https://research.gaitup.com/wp-content/uploads/2020/03/Physilog5_Datasheet_V1.03.pdf>

[6] Shimmer “Shimmer3 Wireless Sensor Platform” Copyright© 2019 Shimmer S-S/S3-v1.8 (5 March 2021)

<http://www.shimmersensing.com/images/uploads/docs/Shimmer3_Spec_Sheet_V1.8.pdf>

[7] Xsens, “Xsen Dot”. (5 March 2021)

<https://www.xsens.com/xsens-dot>

[8] Yostlabs, “Yost 3 Space Mini Bluetooth”. (5 March 2021)

<https://yostlabs.com/product/bluetooth-mini/>

[9] Haugen, T., & Buchheit, M. (2016). Sprint Running Performance Monitoring: Methodological and Practical Considerations. Sports Medicine, 46(5), 641–656.

<https://doi.org/10.1007/s40279-015-0446-0>

[10] Gastin, P. B., Mclean, O. C., Breed, R. V. P., & Spittle, M. (2014). Tackle and impact detection in elite Australian football using wearable microsensor technology. Journal of Sports Sciences, 32(10), 947–953. <https://doi.org/10.1080/02640414.2013.868920>

[11] Scorza, K. A., Raleigh, M. F., & O’Connor, F. G. (2012). Current concepts in concussion: evaluation and management. American Family Physician, 85(2), 123–132.

<https://www.aafp.org/afp/2012/0115/p123.html>

[12]Texas Instruments. (2015).” Understanding the I2C Bus”. (5 March 2021)

<https://www.ti.com/lit/an/slva704/slva704.pdf>

[13] Texas Instruments,( June 2020).” CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual (Rev. I)”. (8 March 2021) p25

<https://www.ti.com/lit/ug/swcu117i/swcu117i.pdf?ts=1617569938280>

[14] Texas Instruments,(June 2020). ”CC2650 SimpleLink™ Multistandard Wireless MCU datasheet (Rev. B)”. (8 March 2021)

<https://www.ti.com/lit/ds/symlink/cc2650.pdf?ts=1617292981934&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCC2650>

[15] InvenSense, (August 2013). “MPU-6000-Datasheet1” (8 March 2021)

<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

[16] WIRED, (July 2010). “Why Everything Wireless Is 2.4 GHz” (9 March 2021)

<https://www.wired.com/2010/09/wireless-explainer/>

[17] Texas Instruments,”EasyLink API reference” (8 March 2021)

<https://software-dl.ti.com/simplelink/esd/simplelink_cc13x0_sdk/3.20.00.23/exports/docs/proprietary-rf/proprietary-rf-users-guide/easylink/easylink-api-reference.html>

[18] Texas Instruments,” rfWsnNode Example” (8 March 2021)

<http://software-dl.ti.com/simplelink/esd/simplelink_cc13x0_sdk/2.20.00.38/exports/docs/proprietary-rf/proprietary-rf-users-guide/proprietary-rf-guide/rfWsnNode_README.html>

[19] Texas Instruments,” I2C Master” (8 March 2021)

<http://software-dl.ti.com/lprf/sensor_controller_studio/docs/cc13x0_cc26x0_help/html/i2c_master__0.html>

[20] Texas Instruments,” rfWsnConcentrator Example” (8 March 2021)

<http://software-dl.ti.com/simplelink/esd/simplelink_cc13x0_sdk/2.20.00.38/exports/docs/proprietary-rf/proprietary-rf-users-guide/proprietary-rf-guide/rfWsnConcentrator_README.html>

[21] Texas Instruments, “SmartRF™ Studio 7 Overview Low Power RF Development Tools”, 2010. p7 (15 March 2021)

<https://www.ti.com/lit/ug/swru195b/swru195b.pdf?ts=1617283480523&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FSMARTRFTM-STUDIO>

[22] metageek, “Understanding RSSI, Acceptable Signal Strengths ” (15 March 2021)

<https://www.metageek.com/training/resources/understanding-rssi.html>

[23] Texas Instruments, “SmartRF™ Studio 7 Hands-on User Guide and Tutorial”, 2011. p25 (16 March 2021)

<https://www.ti.com/lit/ug/swru194b/swru194b.pdf?ts=1617284612698&ref_url=https%253A%252F%252Fduckduckgo.com%252F>

[24]Centre For Humans & Nature, Bruce Jennings, “ETHICAL ASPECTS OF SUSTAINABILITY” (2 April 2021)

<https://www.humansandnature.org/ethical-aspects-of-sustainability>

[25] Needhidasan, S., Samuel, M., & Chidambaram, R. (2014). Electronic waste - an emerging threat to the environment of urban India. *Journal of Environmental Health Science & Engineering*, *12*(1), 36.

<https://doi.org/10.1186/2052-336X-12-36>

[26]Wikipedia, “Restriction of Hazardous Substances Directive” (15 March 2021)

<https://en.wikipedia.org/wiki/Restriction_of_Hazardous_Substances_Directive#cite_note-4>

[27] Texas Instruments, “Lead-free (Pb-free) conversion” (2 April 2021)

<https://www.ti.com/support-quality/environmental-info/lead-free.html>

[28] Texas Instruments, “Packing material information” (2 April 2021)

<https://www.ti.com/support-quality/environmental-info/packing-material-information.html>

[29]wearethemighty, **“**5 best weapons from this famous calculator manufacture**”** (2 April 2021)

<https://www.wearethemighty.com/mighty-tactical/texas-instruments-weapons-manufacturer/>

[30] Medium, Trae Stephens,” The Ethics of Defense Technology Development: An Investor’s Perspective” (2 April 2021)

<https://medium.com/@traestephens/the-ethics-of-defense-technology-development-an-investors-perspective-45c71bf6e6af>

[31] Texas Instruments, “TEXAS INSTRUMENTS INCORPORATED ("TI")

TERMS AND CONDITIONS FOR THE PROVISION AND RECEIPT OF

SAMPLES OF TI PRODUCTS” (2 April 2021)

<https://www.ti.com/sc/docs/sampterms.htm>

# Appendix

#### Project Code Repository- <https://github.com/DarrenB-32/Wireless-Sensor-Network>

#### LAUNCHXL-CC2650 Materials Breakdown -

<https://drive.google.com/file/d/1TxX-zuNVGgqXaxe5sjeMmkHzFcTQe79e/view?usp=sharing>

#### Project Logs -

<https://drive.google.com/file/d/1tkdau6cIOnLg_0jiEKMewBDTk41JwYCi/view?usp=sharing>

1. The reasoning behind this choice will be explained further in the [Chapter 3](#_Chapter_3_-) [↑](#footnote-ref-1)
2. The code generated by SmartRF Studio 7 is explained further in [Chapter 4](#_3.1.3_Smart_RF) [↑](#footnote-ref-2)
3. The testing functions provided by SmartRF Studio 7 are explored further in [Chapter 5](#_SmartRF_Studio_7) [↑](#footnote-ref-3)
4. The testing performed with Scopy is explored further in [Chapter 5](#_Chapter_5_-) [↑](#footnote-ref-4)
5. The full materials breakdown for the CC2650 can be found in the [Appendix](#_LAUNCHXL-CC2650_Materials_Breakdown) [↑](#footnote-ref-5)