



**Murdoch**  
**UNIVERSITY**

# Reliable Data Transfer via Frequency Transmission

---

Bachelor of Engineering Honours / Bachelor of Commerce (*H1265*)

By  
Brandon James John Butler

Submitted to the Murdoch University School of Engineering Information and Technology to fulfil  
the requirements of the Bachelor of Engineering Honours degree

2017

## 2.1 Transmission Types

I, Brandon Butler, declare this thesis is an account of my own research. Its main content has not previously been submitted for a degree at any tertiary education institution.

Other sources will be appropriately acknowledged by full referencing.

Word Count: 11946

## 2.1 Transmission Types

### Abstract

Reliable single directional frequency data transfer is a method of electronic communication that is a potential alternative to a bi-directional or wired method. It is intended to determine whether or not single directional data transfer can be designed to perform at the same reliability level as other methods of data transmission. The reason for researching this is to see whether, two-way communication is necessary. Upon finding results to this question it will be determined if single directional frequency data transfer can be as power efficient as bi directional data transfer. It will also look into the overall performance of the method and how it can deal with inhibiting factors that will be introduced to simulate real world external variations in the signal.

How reliable single directional frequency transfer is, will be determined through experiments that are tasked at finding the maximum transmission rate of the devices made and the distance that data transmission can be conducted over.

The investigation will require the design of an experimental apparatus that will allow results to be found in the maximum possible transmission rate and distance transmission can cover.

The experimental apparatus will consist of two processors, one for encoding a message, the other for decoding a message. The apparatus will also require an integrated radio frequency transmitter circuit as well as a receiver radio frequency integrated circuit. The processors will need to be coded with a new protocol that will allow the incorporation of three forward error correction techniques. This is so that a basic, intermediate and advanced method of forward error correction can be compared when gathering results.

Throughout this research consideration into all aspects that can possibly improve the energy efficiency of the electrical apparatus will be addressed and implemented. By creating an experimental apparatus that is competitive against other data transmission methods in terms of energy efficiency. With an energy efficient experimental apparatus, then results found could be a close representation of the likely outcomes if single directional data transmission was to be implemented on a larger scale.

## 2.1 Transmission Types

### Acknowledgments

Throughout my university degree there have been numerous people who contributed towards my development in becoming an Engineer.

I would like to thank my supervisor, Gareth Lee, for his continuous help and approachability throughout my dissertation. The results were largely achieved due to Gareth's support.

Thank you to all academic and technical staff at Murdoch. My time at the university has been invaluable and will be a great foundation for the transition into workforce.

To my Friends and Family, thank you for motivating me continually and providing a healthy distraction from studies at times.

Without the support of my parents Allan and Lynda Butler, I would not be where I am today, I was constantly encouraged to achieve the best of my ability with their help this was made possible. As well as contributing to the experimental build my biggest supporter throughout my time studying has been Allan. I am so grateful for everything he has helped me to achieve. I would like to acknowledge Michael Crooke, for taking time out of his week to help record results for this report. Additionally, I would like to thank Glen Bleach for sharing his knowledge and discussing different focus points presented in this report to help bring them clarity.

## 2.1 Transmission Types

### Table of Contents

Abstract.....	iii
Acknowledgments.....	iv
Table of Contents.....	v
List of Figures .....	vi
List of Tables .....	vii
List of Abbreviations .....	vii
1.0 Introduction .....	1
2.0 Literature Review .....	3
2.1 Transmission Types.....	3
2.1.1 AM Transmission .....	4
2.1.2 FM Transmission.....	4
2.2 DM Transmission .....	5
2.2.1 Frequency Shift Keying (FSK).....	5
2.2.2 Amplitude Shift Keying (ASK) .....	5
2.2.3 Phase Shift Keying (PSK).....	6
2.3 Frequency Transmission History.....	6
2.4 Line Coding Method .....	8
2.4.1 Line Coding History .....	8
2.5 Error Correction Techniques .....	9
2.5.1 Error Correction History .....	9
2.5.2 Forward Error Correction .....	10
2.5.2.1 2D Matrix Error Correction .....	10
2.5.2.2 Reed Solomon Error Correction .....	11
3.0 Methodology.....	16
3.1 Experimental Apparatus Hardware Design .....	16
3.1.1 Radio Frequency Spectrum .....	16
3.1.2 Micro Controllers .....	17
3.1.3 TFT Arduino Screen Shield .....	17
3.1.4 User Interface.....	18
3.2 Software Design.....	18
3.2.1 Transmission Line Coding Method.....	18
3.2.2 Encoding Process with 2D Matrix Error Checking.....	19
3.2.3 Decoding Process with 2D Matrix Error Checking.....	20
3.2.4 Encoding Process with Reed Solomon Error Checking Method.....	21
3.3 Hardware Design Issues.....	21
3.3.1 Switch Mode Voltage Regulator.....	21
3.3.2 Adding a Comparator to Filter out Noise .....	22
3.3.3 Plug and Play Style Apparatus .....	23
3.4 Software Design Issues .....	23
3.4.1 Processor Timing .....	23
3.4.2 Reed Solomon Error Checking Code Footprint .....	27
3.5 Experimental Design .....	27
4.0 Results.....	29
5.0 Interpretation .....	31
5.1 Experiment One Distance Vs Errors.....	31
5.2 Experiment Two Transmission Rate Vs Errors.....	34

## 2.1 Transmission Types

5.3 Experiment Three Signal Strength Vs Errors .....	36
6.0 Economic Comparison .....	39
7.0 Future Works .....	41
8.0 Conclusions .....	42
9.0 References .....	43
10.0 Appendix (A) – RF Module Data Sheets .....	44
11.0 Appendix (B) – Frequency Spectrum .....	48
12.0 Appendix (C) – Line Coding Style Comparison .....	49
13.0 Appendix (D) – Code Alterations.....	53
14.0 Appendix (E) – Apparatus Diagrams and Wiring Schematic .....	67

## List of Figures

Figure 1 - AM Modulation Diagram .....	4
Figure 2 - FM Modulation Diagram.....	4
Figure 3 - FSK Modulation Diagram .....	5
Figure 4 - ASK Modulation Diagram.....	6
Figure 5 - Phase Shift Keying Diagram .....	6
Figure 6 - Matrix forward error correction encoding .....	11
Figure 7 - Matrix forward error correction decoding .....	11
Figure 8 - Reed Solomon forward error correction sent data .....	12
Figure 9 - Reed Solomon forward error correction decode equation .....	13
Figure 10 - Reed Solomon forward error correction decoding the code word .....	13
Figure 11-Reed Solomon workings 1 .....	14
Figure 12-Reed Solomon workings 2 .....	14
Figure 13 - Return to Zero Line Coding Method .....	19
Figure 14 - Encoding for 2D Matrix Error Checking .....	19
Figure 15 - 2D Matrix Representation .....	20
Figure 16 - Comparator Filter.....	23
Figure 17- Out of Sync Timing.....	24
Figure 18 Hardware Timer Loop .....	25
Figure 19 - Timer Improved Accuracy Data Readings.....	26
Figure 20 - Diagram Showing Rising Edge Resynchronisation .....	26
Figure 21 - Final Transmitter Experimental Apparatus.....	28
Figure 22 - Final Receiver Experimental Apparatus.....	28
Figure 23 - Experimental Apparatus Enclosure.....	29
Figure 24 - Experiment One Check Sum Vs Error Rate .....	31
Figure 25 - Experiment One 2D Matrix Distance Vs Error Rate .....	32
Figure 26 - Experiment One Reed Solomon Distance Vs Error Rate.....	33
Figure 27 - Experiment Two Check Sum Transmission Rate Vs Errors.....	34
Figure 28 - Experiment Two 2D Matrix Error Correction Transmission Speed Vs Errors .....	35
Figure 29 - Experiment Two Reed Solomon Error Correction Transmission Speed Vs Errors .....	35
Figure 30 - Experiment Three Check Sum Signal Strength Vs Error Rate .....	36
Figure 31 - Experiment Three 2D Matrix Signal Strength Vs Error Rate .....	37
Figure 32 - Experiment Three Reed Solomon Signal Strength Vs Error Rate.....	38
Figure 33 - Economic comparison cost breakdown of single direction wireless transmission. ....	40
Figure 34 - Economic comparison cost breakdown of bi directional wireless transmission. ....	40

## 2.1 Transmission Types

Figure 35 - Economic comparison cost breakdown of bi directional wired transmission..... 40

## List of Tables

Table 1 - Table showing the breakdown of the Impeding variables and their percentage for experiment three .....37

## List of Abbreviations

FEC	Forward Error Correction
ARQ	Automatic Response Request
RF	Radio Frequency
BPS	Bits Per Second
EEPROM	Electronically Erasable Programmable Read Only Memory
SRAM	Static Random Access Memory

## 1.0 Introduction

This research report will focus on frequency data transmission. The main research question is;

*'Is it beneficial to use forward error correction as an alternative to bi-directional communication?'*

There are three objectives that will lend supporting evidence to the underlying question of the research project should they be validated or proven through either experimentation or research. The hypothesis being that a system with forward error correction benefits over an alternative system due to the hypothesised decreases in power consumption and cost in apparatus to setup this form of frequency communication. This will be validated by conducting experiments, which look into the economic overheads created by forward error correction. As well as providing quantitative data to distinguish whether one-way communication with forward error correction performs better than a communication system that communicates bi-directionally.

These quantitative figures will be used as comparisons between the two methods of communication comparing transmission rate with minimal errors, the max distance between the receiver and transmitter in order for the system to work and the required signal strength between the two methods. The purpose of this research is to provide data which can be used as a reference for determining if a unidirectional transmission system will be capable of performing to the same reliable standard that a bi-directional communication system would.

Prior to conducting research into this field it is expected that a frequency communication system which communicates only one way will have the capability to add further software coding which can vary in complexity which will allow the system to perform to either an equal or higher standard than bi-directional communication. To simulate this a smaller scale version of a frequency communication system will be built using two Arduino's to process the signal and a radio frequency integrated circuit board transmitter and receiver. One Arduino will be built to simulate a message encoder the other will be used to simulate a



message decoder. The encoding and decoding software will be written in Arduino's IDE software and will be designed based on further research of line coding methods. This apparatus can then be used to carry out the experiments mentioned herewith.

## 2.0 Literature Review

Key features to help define the objectives of the experiments to be conducted to confirm reliable data transmission are, transmission types, protocol structure, methods of error checking and simplex versus duplex communication setups.

Each of these features was researched in order to further develop an understanding and to assist in the procedure taken to determine if the all-encompassing hypothesis of the report is valid.

### 2.1 Transmission Types

Transmission types, was an important factor to look into due to the focus being frequency transmitted data. There are many ways of transmitting data using a radio frequency. A few that you would come across in day to day activities could be frequency modulated, otherwise known as FM transmission, which you would come across when listening to the radio similarly Amplitude modulation known as AM transmission also can be heard on the radio [1]. There are also a few types of wireless technologies that use a frequency transmission type known as digital modulation (DM). Digital modulation is a common technique used in wireless computer networks, 3G, 4G and LTE mobile phone networks [2]. There is also digital modulation used in communication methods such as Bluetooth.

### 2.1.1 AM Transmission

'AM' transmission is an analogue form of transmission where to represent data a continuous analogue signal is imposed on carrier wave [3]. The carrier wave will scale its amplitude to represent the carrier wave. In Figure 1 the amplitude is high when the signal wave is positive and the amplitude is low when the signal wave is negative.

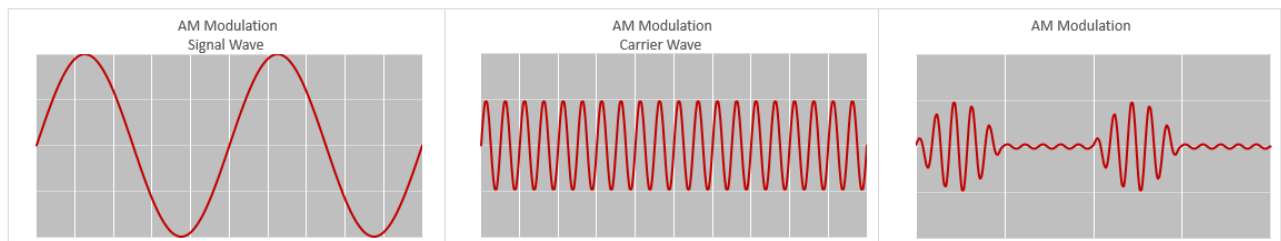


Figure 1 - AM Modulation Diagram

### 2.1.2 FM Transmission

'FM' transmission is another analogue form of frequency data transmission. In this method the same as in the 'AM' transmission form the message is scales so that the message is represented by the carrier waves frequency. This works by having a high frequency to symbolise the signal is high. If the signal is low so will be the frequency of the carrier wave [3]. This can be seen in Figure 2 where it is evident that the Frequency modulated wave looks more bundled when the signal wave is positive.

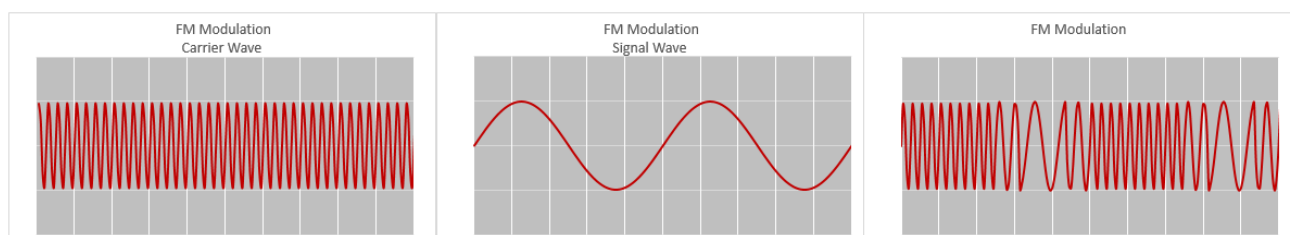


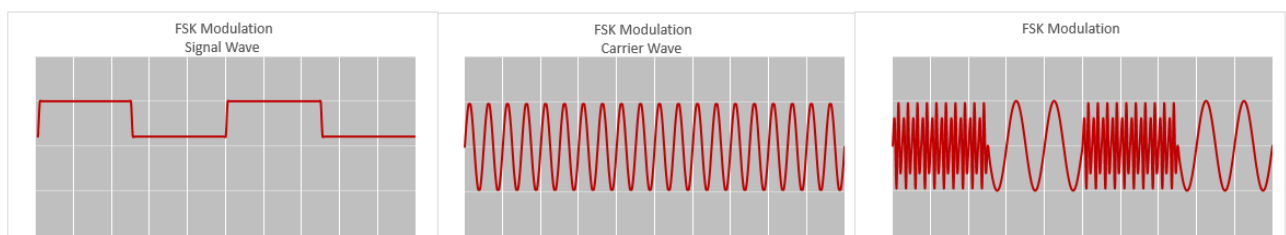
Figure 2 - FM Modulation Diagram

## 2.2 DM Transmission

Digital Transmission is another common transmission method. This form of frequency transmission is most commonly used in televisions, mobile phones, game console controllers, portable Bluetooth speakers and some computer components. Digital modulation can then be broken down further into subcategories known as, Frequency Shift Keying (FSK), Phase Shift Keying (PSK) and Amplitude Shift Keying (ASK) [2]. Digital Modulation is a technique that uses discrete signals to modulate a carrier wave in a particular bandwidth [2] [4].

### 2.2.1 Frequency Shift Keying (FSK)

This method is closely related to Frequency modulation. The only difference being the signal shown using the carrier wave is of a discrete type in nature. This method works by breaking down the signal into its binary form so that it can be represented as ones and zeros. FM frequency works by changing the period of the carrier wave to represent the ones and zeros within the data message being sent. Usually a smaller period represents a one and a longer period represents a zero [4].



*Figure 3 - FSK Modulation Diagram*

### 2.2.2 Amplitude Shift Keying (ASK)

Similar to AM modulation Shift keying refers to the data type of the signal. In shift keying, the signal is of a discrete type in nature. For ASK modulation to work data is broken down into a binary form that can be represented as ones and zeros. This message is then sent with a wireless signal at a chosen frequency,

where a change the amplitude modulation can represent either a one or a zero. Typically a high amplitude represents a one and a lower amplitude represents a zero [4].

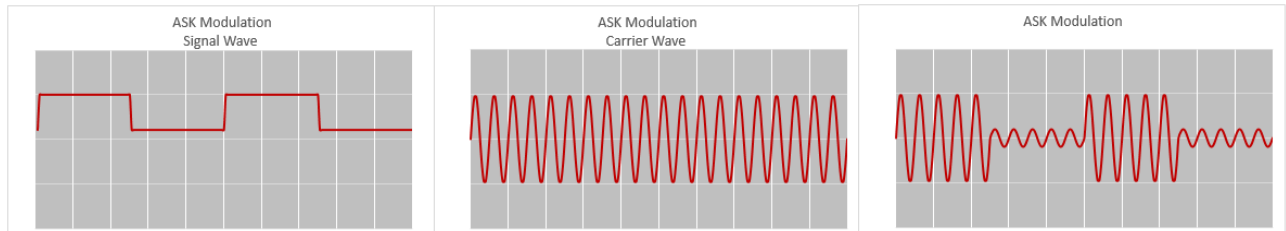


Figure 4 - ASK Modulation Diagram

### 2.2.3 Phase Shift Keying (PSK)

The third method, Phase Shift Keying, is how the name describes, a method where the phase angle at the start of each period changes depending on if the data being sent is a one or zero [4]. This method can also be done as an analogue transmission method although it requires more processing to decode this type of transmission.

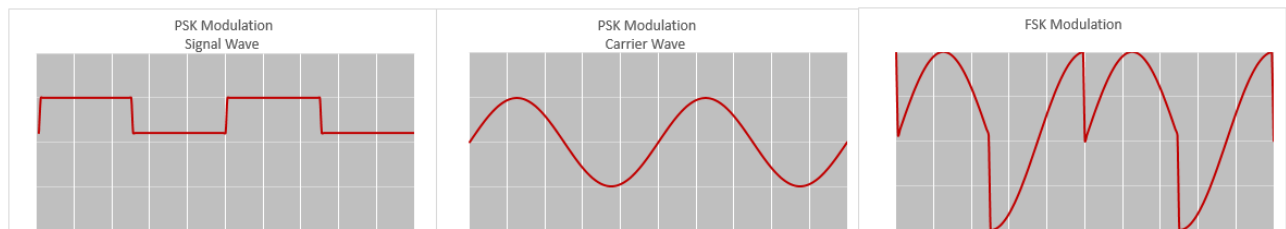


Figure 5 - Phase Shift Keying Diagram

## 2.3 Frequency Transmission History

In looking at the history of frequency data transmission, which for AM modulation started back in the late 1800's where the idea to vary the signal strength (Amplitude) to encode data for wireless communication was proposed. Several inventors and engineers worked on this concept. It was not until 1900 when a Canadian engineer and researcher, Reginald Fessenden made the first known Amplitude modulated

transmission [5]. Between the invention of AM transmission and FM transmission in 1928 many investigations into the capabilities of FM transmission were looked into due to the effect static noise would have on AM transmission. AM transmission works by varying the amplitude, the receiver for an AM radio would receive the signal as well as any static noise making the transmission difficult to decode. An American engineer, Edward Howard Armstrong conducted experiments that first uncovered the capabilities of FM modulation, which then led to the development of wide band FM transmission. This was less susceptible to noise in comparison to Amplitude Modulation [6].

These experiments highlighted that FM transmission will be less susceptible to static noise than AM making FM transmission the favourable method of transmission. Therefore this will be incorporated into the design of the experiment that will be later used as validation of the Report Question, being, is single directional frequency data transfer a reliable method for transmitting data.

Also upon investigating the methods of frequency transmission the methods that are of the type, which involves frequency modulation, are FM and FSK. These two methods as earlier explained both use a varying frequency the different in the two is the type of data being represented by the carrier wave [4]. Out of the two methods FM transmission and frequency shift keying Transmission the option that would best suit the research is the frequent shift keying method. Due to the ability to determine how effective a transmission was becomes a lot more obvious with a discrete signal in comparison to an analogue signal. With the discrete signal there are two possibilities that the transmission could be sending this is either a one or a zero. Where as in an analogue method there is a range of possibilities that could be sent within the accuracy level of the transmitter.

## 2.4 Line Coding Method

Once there is a transmission method for data then a method for how the decoder and encoder of the signal waves are required. This is known as line coding. Line coding can be done in numerous ways. As part of this research report line coding methods were compared against each another to distinguish which method would be the best method to incorporate into our experiment.

### 2.4.1 Line Coding History

Line coding has been researched in a number of different ways in order to create a method to suit a particular purpose. For example, a line coding method that is good for regulating clock speed so the decoder can stay in sync with the encoder's message [7]. A line coding method that achieves this is Manchester line coding. This line coding method was developed at the University of Manchester. It is a method that was created for the Manchester Mark 1 computer, which was one of the earliest stored program computers [8]. This is a method where the transmitter will send with a high or low depending on which version of Manchester line coding a one will be represent by a high pulse and at the middle of each period the signal will change to the opposite direction [8]. For example, if a one was being represented as a high half way through the clock cycle this would be represented as a low and vice versa if a zero was being represented on the line. The change in the slope half way through the period of the binary value being sent can be used to recalibrate the processors clock cycle so it stays in sync with the encoder [8]. There are other methods of line coding such as return to zero when simply a one represents a high and a zero represents a low. Return to Zero however, unlike the Manchester method doesn't offer the ability to the decoder to resynchronise its clock cycle. For instance if there is multiple ones the signal will stay high for the period of the ones being sent with no slope in between for the decoder to use as a time sync. It was favourable to use a method such as Manchester where there is the advantage that allows the communication equipment to stay in sync making it more accurate when reading a signal [7] [9].

Some of the methods researched in this report are Manchester line coding and various forms of the Return to Zero line coding method. Independent research was completed as part of the validation to determine

the best line coding method to incorporate with the experiment conducted as part of this report. This research will be discussed in further detail in the later stages of this report.

## 2.5 Error Correction Techniques

Error correction techniques are decoding techniques that allow a signal to be deciphered correctly even in the event where part of a message is affected by noise and is received incorrectly. Error correction can be as simple as having a checksum as part of the message string or as complex as having binary code words added to the ends of message strings as parity bits.

Some of the methods looked into at the early stages were adding simple checksums to each byte. Other methods were 2D matrix error correction, a simple version of Reed Solomon error correction and a more sophisticated version of Reed Solomon. Error correction techniques will be used in later experiments that aim to investigate the question that the research in this report is based around. The investigation requires tests to be done to validate the claim that frequency data transfer being used in a single directional communication setup is a reliable form of data transfer.

### 2.5.1 Error Correction History

In 1947 Richard W Hamming first documented and developed error correction in computer science [9]. Where a piece of error correcting code, which he had developed, appeared in a mathematical theory textbook [9].

There are two forms of error correction these are, automatic repeat request (ARQ) [9], where simply the message is re sent and the receiving end will decipher the message based on multiple messages received. There is also Forward Error Correction (FEC) [9] where information is sent with the message that can help determine whether or not the message received was actually the intended message. Since this report focuses on a single direction communication style automatic repeat request does not fit the criteria. The decoder will not have transmitting capabilities so the ARQ will not be an appropriate error correction type.



Forward error correction as an alternative allows the receiver to interpret the message being sent without the need to ask for the message to be resent.

## 2.5.2 Forward Error Correction

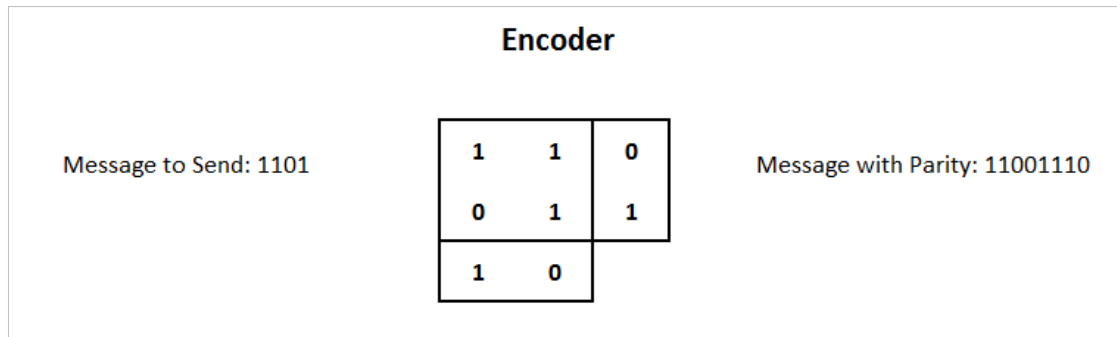
Forward error correction works by sending as well as the message a set number of parity bits that are calculated based on the message being sent [9]. This makes the parity bits being sent unique to the message that in theory helps to allow the receiver to decipher the message.

### 2.5.2.1 2D Matrix Error Correction

Matrix error correction works by representing the data being sent in a matrix from [9]. For each row and each column a parity bit is added. This allows a decoder to locate any errors in a message by identifying where the vertical parity and horizontal parity overlap showing the exact bit or byte incorrectly sent during transmission [9]. Experiments have been conducted and confirmed this method as having the ability to identify up to three errors for every byte sent and is able to correct up to two of those errors. Improvements to this method were made at a later stage for all three of the identified errors to be corrected. This was achieved by adding diagonal parity bits. Although once this improvement is incorporated the method becomes uneconomic in the way that more parity bits begin to be sent in comparison to message bits [9].

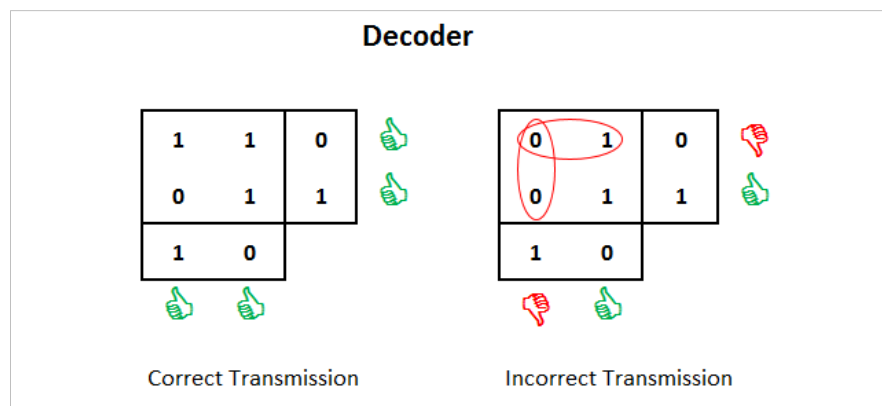
An example of how this works on a simplified level, a matrix is created with the data you wish to send. The encoder first does an odd or even parity bit for each row. It then does the same for each column. This message is then sent to the decoder where it will do its own check to see if the parity bits line up with what was received. If the parity lines up with what was sent then generally no error was received. When a parity bit does not line up with what was received then there is usually a parity that doesn't line up in both the rows and columns. In this case the incorrect bit to be fixed will be the section where the two circles as shown in Figure 6 overlap [9]. Both cases are shown in the figure below, the first case where the message

was received correctly and the second case where the message was received with errors. This example has been shown as a two by two matrix so that the methodology would become more evident.



*Figure 6 - Matrix forward error correction encoding*

The diagram above shows how the message is structured before being sent to the decoder, which is also shown in Figure 7.



*Figure 7 - Matrix forward error correction decoding*

### 2.5.2.2 Reed Solomon Error Correction

Reed Solomon Coding is the name given to algebraic formulas designed to detect and correct errors in transmitted signals [10]. Reed Solomon is an amalgamation of both the inventors' names. The two mathematicians who were also electrical engineers who brought about this form of error correction were Irving S. Reed and Gustave Solomon. The Reed Solomon Error checking technique is widely used in modern technology. Areas where this error checking has been implemented are in data storage, 2 dimensional bar

codes data transmission and space transmission [10]. The fundamentals of this form of error checking are quite complex to code from scratch. A sample of Reed Solomon Code was used then adapted to fit the purpose of the investigation. How this works on a simplified level is by making the message to be sent into matrix form. This matrix is then multiplied by the identity matrix with added parity rows. The amount of parity rows depends on the amount of errors that you want to be able to detect and correct [10] [11].

$$\text{Number of Parity} = \text{Number of errors to be detected} \times 2$$

For this example we wish to correct 1 transmission error, therefore two parity rows are added to the identity matrix. Figure 8 below shows the data message in a matrix format and the identity matrix with parity that the data message will eventually be multiplied by.

Matrix A	Matrix I	Matrix B
1 2 3	1 0 0	1 2 3
4 5 6	0 1 0	4 5 6
7 8 9	0 0 1	7 8 9
	9 11 13	144 177 210
	12 14 16	180 222 264
Message	Identity Matrix Plus Parity	Matrix A + Matrix I Parity

*Figure 8 - Reed Solomon forward error correction sent data*

Matrix B is a result of multiplying matrix A and the identity matrix together. Matrix B in our example will be the matrix of data that is to be sent to the decoder. Once matrix B is transmitted to the decoder the data is broken up so that it can be used in the following format illustrated by Figure 9.

<b>Matrix A</b>		<b>Matrix I</b>		<b>Matrix B</b>																																							
<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table>	1	2	3	4	5	6	7	8	9	X	<table border="1"> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr style="background-color: #f08080;"><td>9</td><td>11</td><td>13</td></tr> <tr style="background-color: #f08080;"><td>12</td><td>14</td><td>16</td></tr> </table>	1	0	0	0	1	0	0	0	1	9	11	13	12	14	16	=	<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> <tr><td>144</td><td>177</td><td>210</td></tr> <tr><td>180</td><td>222</td><td>264</td></tr> </table>	1	2	3	4	5	6	7	8	9	144	177	210	180	222	264
1	2	3																																									
4	5	6																																									
7	8	9																																									
1	0	0																																									
0	1	0																																									
0	0	1																																									
9	11	13																																									
12	14	16																																									
1	2	3																																									
4	5	6																																									
7	8	9																																									
144	177	210																																									
180	222	264																																									
Message		Identity Matrix Plus Parity		Code Word Matrix																																							

Figure 9 - Reed Solomon forward error correction decode equation

The decoder from this point is where you take any three rows from the Identity Matrix and multiply it with the same corresponding rows in matrix B. The result of this multiplication is the original message if no errors were sent. This example shows the method of this process of the Reed Solomon method in a case where no error was transmitted [10].

	<b>Matrix I</b>		<b>Matrix B</b>																														
	<table border="1"> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr style="background-color: #f08080;"><td>9</td><td>11</td><td>13</td></tr> <tr style="background-color: #f08080;"><td>12</td><td>14</td><td>16</td></tr> </table>	1	0	0	0	1	0	0	0	1	9	11	13	12	14	16		<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> <tr><td>144</td><td>177</td><td>210</td></tr> <tr><td>180</td><td>222</td><td>264</td></tr> </table>	1	2	3	4	5	6	7	8	9	144	177	210	180	222	264
1	0	0																															
0	1	0																															
0	0	1																															
9	11	13																															
12	14	16																															
1	2	3																															
4	5	6																															
7	8	9																															
144	177	210																															
180	222	264																															
R3 →		R3 →																															
R4 →		R4 →																															
R5 →		R5 →																															
	Identity Matrix Plus Parity		Code Word Matrix																														
<b>Matrix B</b>		<b>Matrix I<sup>-1</sup></b>	<b>Matrix B</b>																														
<table border="1"> <tr><td>7</td><td>8</td><td>9</td></tr> <tr><td>144</td><td>177</td><td>210</td></tr> <tr><td>180</td><td>222</td><td>264</td></tr> </table>	7	8	9	144	177	210	180	222	264	X	<table border="1"> <tr><td>1</td><td>-2.3</td><td>1.83</td></tr> <tr><td>-2</td><td>-2</td><td>-1.5</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> </table>	1	-2.3	1.83	-2	-2	-1.5	1	0	0	=	<table border="1"> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td></tr> </table>	1	2	3	4	5	6	7	8	9		
7	8	9																															
144	177	210																															
180	222	264																															
1	-2.3	1.83																															
-2	-2	-1.5																															
1	0	0																															
1	2	3																															
4	5	6																															
7	8	9																															
R3		R3																															
R4		R4																															
R5		R5																															
Message		Inverse Matrix B		Message																													

Figure 10 - Reed Solomon forward error correction decoding the code word

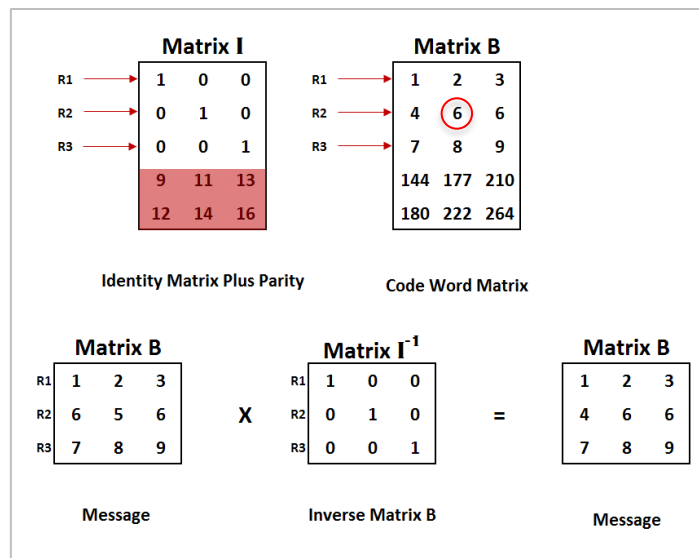
In a case where an error was transmitted to the decoder further multiplications of three different rows in matrix B and the Inverse of matrix I would need to be carried out. An example is if the decoder received an error in row two, column two, of matrix B. The decoder does not yet know this that there was an error, although after iterations of the Reed Solomon forward error correction method it is able to be determined whether or not the data was correct. The process is as follows, the decoder would perform the calculation

with 3 rows not necessarily order specific. The equation would be performed again using a different set of three rows. This process is repeated until all combinations of three rows for a five-row Matrix B. The total number of combinations that can be made is found the using formula below.

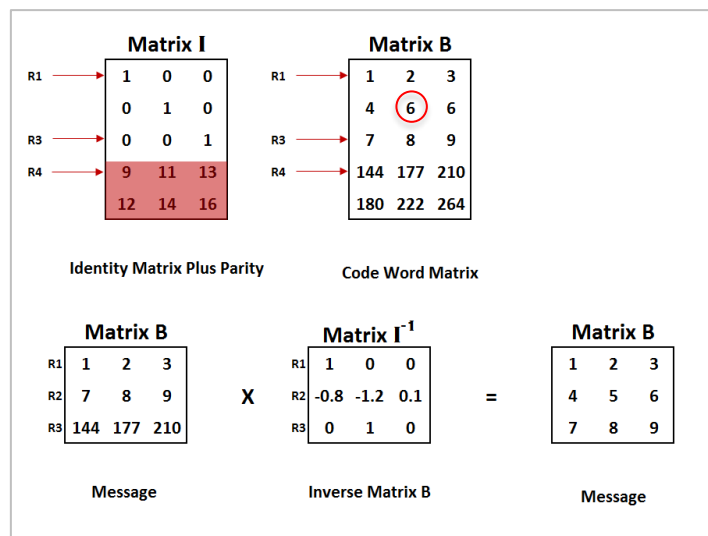
$$\frac{n!}{R!(n-R)!} = \text{maximum combinations}$$

This formula uses the number of rows in matrix B as the 'R' value and the number of rows in the resultant matrix as the 'n' value. For the example being explained 'n' would be five and 'R' would be three.

For simplicity Figure 11 & Figure 12 below show only two different combinations of rows and the resultant matrix [10].



*Figure 11-Reed Solomon workings 1*



*Figure 12-Reed Solomon workings 2*

The results from Figure 10, Figure 11 & Figure 12 show it can be seen in the data sent for two of the figures that the decoded message is correct, and in one of the figures the decoded message is wrong. Once all combinations of rows have been calculated the processor will then determine the message based on what the majority of the equations ended up decoding the message to be [10] [12].

## 3.0 Methodology

Throughout the beginning of this report it was stated that in order to gather quantitative data to determine reliable frequency data transfer an experiment would be conducted using a miniaturised version of the communication equipment you could potentially find in real world applications.

As well as building an experimental apparatus a literature review was conducted to determine the best methods / ideologies to incorporate into the experimental apparatus. This was necessary because, in order to gain the most accurate results the experimental setup needs to include the best techniques in order to achieve the greatest efficiency. The method to designing the experimental setup was first to design a physical apparatus capable of performing the required task. Followed by the software techniques being coded onto the encoder and decoder. Once the hardware and software was addressed, the experiments that allowed results for the reliable performance were conducted. These experiments are going to give an indication into how reliable our communications setup would be, were based on discussions in which it was determined that transmission speed, effectiveness over gradual increase in distance and signal strength to error ratio.

### 3.1 Experimental Apparatus Hardware Design

#### 3.1.1 Radio Frequency Spectrum

The design had to have a legal radio frequency transmitter and receiver. In order for a radio frequency module to be legal it must operate in a given bandwidth set out by the governing bodies of the area that you wish to operate in [12]. These restrictions are part of the frequency spectrum and there are international and national declarations that ensure certain frequency bands are not used without a license [12]. An example of an international frequency band, that is limited to who can operate within this band is, the Industrial Scientific and Medical Band (ISM) [12]. The limitations into which band within the frequency spectrum that you can operate in made it important that the RF components that would be incorporated in the design of the apparatus operated within a non-licensed frequency band. Generally it is assumed that if you are to buy a RF module from a third party retail store, that third party retail store would be required to ensure that the RF modules they are selling be within a non-licensed range. The RF modules acquired for

the experimental apparatus were the Z6905 (Receiver) and the Z6900 (Transmitter), the datasheets for these two components can be found in Appendix (A). The datasheets for these two RF modules state that the operating frequency of these two modules is 433.92MHz which is within a non-licensed amateur frequency band. This was confirmed using the Australian Radio Frequency Spectrum Allocations Chart, which can also be found in Appendix (B).

After confirming that the two RF modules would meet the requirements of the hardware the micro controller was then selected to control the RF Module.

### 3.1.2 Micro Controllers

Micro controllers were a necessary part towards building the experimental apparatus because the data being sent across the frequency channel between the two RF modules needed to be processed so that it was represented in binary form and included relevant parity bytes which would be required at later stages of the experimental apparatus design. Two micro controllers were needed to control the RF modules and also process the information to send and receive from the RF modules. Due to a limited time frame and budget the option that provided the needed functionality at a low price was the Arduino Uno. This micro controller was the best option due to its script coding style as opposed to other methods such as ladder logic or function block diagrams. Another advantage that suited the task was it had a high enough processing speed to make it possible for a decoder to take readings quick enough so the RF modules could be used at their highest possible bit rate (BPS). The Arduino has a possible bit rate of 4Mhz and the RF modules have a possible bit rate of 4.8kHz. The Arduino's cycle rate being much faster made it possible to implement a line coding and line decoding method at the highest rate possible for the RF modules. An added benefit of an Arduino is that it is a low powered device, which demonstrates single directional frequency transmission being achieved on a low energy cost device.

### 3.1.3 TFT Arduino Screen Shield

The experiments to be conducted would require repeat testing in order to achieve the best results. There was a need for repeat testing so a TFT Arduino Screen shield was used so the data can be displayed on both



the decoder and encoder. This makes it more efficient to distinguish if the data was transferred between the encoder and decoder correctly.

#### 3.1.4 User Interface

The experimental Apparatus needed to have controllability so the test could be conducted quickly and by one person in some cases. This required the use of pushbuttons for inputs that could be used on the Arduino. These pushbuttons were used in the initialisation of the two Arduino processors. They allowed features such as selecting the amount of parity bits to include and the overall transmission speed to operate at, as well as when to start transmitting. This made it possible to initialise the decoder so that it started waiting for the transmission from the encoder while an operator walked over to the encoder to start the transmission.

### 3.2 Software Design

#### 3.2.1 Transmission Line Coding Method

To distinguish the best line coding method it would need to be able to meet the criteria of being reliable. For this experiment to be as reliable as possible it needs to have a long life span where the apparatus can operate over a long period of time continuously. The apparatus having low power consumption would achieve this. A lower average power consumption will decrease the frequency that the power source needs to be replaced or replenished. To find the lowest power consumption research was conducted into the most energy efficient method of line coding. Multiple different types of line coding were looked at, of these different methods the average power consumption varied. In order to find out on a theoretical level which method would be the best option, a table of data showing the results on the average power consumption of each line coding method was created. This table showed calculations of the longest on time for each method using each different combination of a byte. For an eight-bit byte there are 256 different combinations of that byte that exist. For each of the 256 combinations, it was calculated how long a voltage signal would have to stay high for on average using the different line coding types. Of the different line coding types the method that showed the best results was the Return to Zero method. Return to Zero had an average high voltage on time of 25%. This 25% shows that for each data transmission, the signal would

be high for 25% of the time during transmission. The results of the research conducted in this area can be found in Appendix (C). Return to Zero line coding was the most efficient after testing and works by sending a high voltage for half of the bit rate time period when sending a 1 and it stays low for the entire bit rate period when sending a zero (Figure 13).

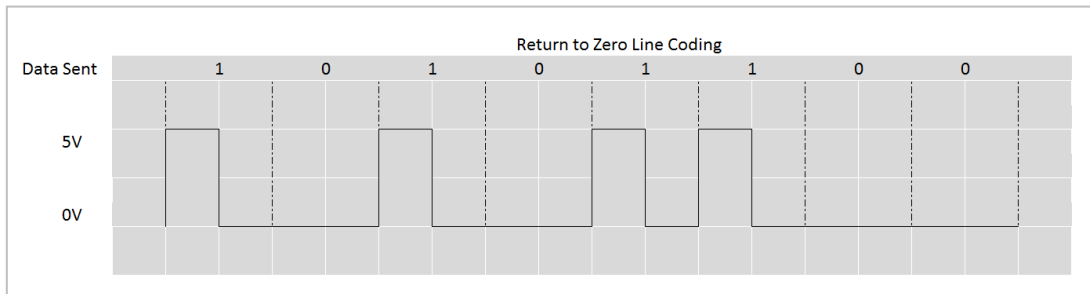


Figure 13 - Return to Zero Line Coding Method

### 3.2.2 Encoding Process with 2D Matrix Error Checking

The encoding process outlines how Returns to Zero line coding was incorporated into the Arduino so that data could be transmitted. By first generating a random bit pattern using an in built random function. The reason for this is so that the transmission success rate could be truly tested due to a different message being sent each time. Once a random bit pattern was generated this pattern was stored in an array. The array was eight positions long to represent a byte that was being sent. A new array was created with length equal to the byte plus the amount of parity bits. The new array was then rebuilt so the bits being sent to the receiver were in the correct order for 2D matrix error checking. The order of each individual bit being sent is shown in Figure 14.

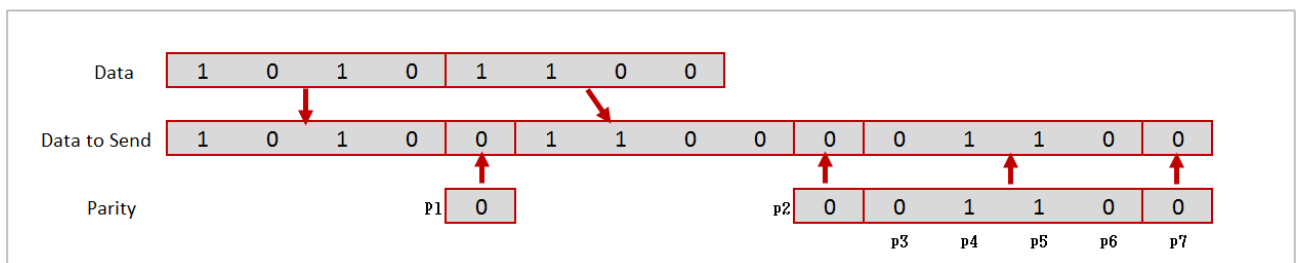


Figure 14 - Encoding for 2D Matrix Error Checking

The breakdown of the array in a single line is how the code is literally processed in theoretical form the 2D matrix for this method is as shown in Figure 15.

1	0	1	0	0	p1
1	1	0	0	0	p2
0	1	1	0	0	
p3	p4	p5	p6	p7	

*Figure 15 - 2D Matrix Representation*

Once the data was reconstructed into this correct format it was put into a loop where each iteration of the loop increased a counter variable. The counter variable was then used as a reference for a nested “if statement” to check whether the element in the array at position ‘x’ which was specified by the counter was a one or zero. If the ‘if statement’ read a one, a digital output connected to the transmitters data input pin would be sent high. If the statement didn’t read a one then the digital pin would not send any signal to the transmitter. This loop continued until all values in the data send array were transmitted. Before the data sent array is sent a simple handshake where the transmitter outputs a High, Low, High, Low is sent so the decoder knows when to start taking readings from the receiver RF module.

### 3.2.3 Decoding Process with 2D Matrix Error Checking

The decoder coding required more processing to incorporate the error checking method. The code begins with a while loop that will not be entered until the handshake is detected. Once a High, Low, High, Low handshake has been detected, then the while loop will be entered. This is where the decoder will take a reading from the data pin of the receiver RF module every bit period that is specified upon downloading the code to the processor. The Arduino waits for the first rise in the signal then waits for a third of the bit period before taking a measurement to ensure that the reading taken was in actual fact the signal being transmitted and not just an impulse caused by noise. Each reading is stored in an array until the receiver has made 19 readings that account for the start bits, the message and the parity bits. Once all readings are taken the decoder then has the necessary array of values needed to perform the forward error correction method. The processor first looks at the first 4 bits in the data sent array and determines whether an odd

or even bit number was sent. This process is continued for the 2D matrix rows and columns. Once the process has calculated whether the rows and columns are even or odd it compares its computed value to the parity bit value sent with the transmission. If the comparison of the two numbers do not match, then the decoder knows there was an error in the corresponding row or column. The rest of the forward error process is carried out as explain in section 2.5.2.1 of this report.

### 3.2.4 Encoding Process with Reed Solomon Error Checking Method

The encoding process for the Reed Solomon error checking method was the advanced form of error checking to be implemented into the experiments. This followed the same basic structure in how the ones and zeros would be sent to the decoder. The majority of the difference in the coding involved in this section was to do with creating the initial array that would be later sent. Due to the complicated mathematical theory involved in this error checking method a piece of sample code written by Henry Minsky was used as a basis. The sample code utilised the Reed Solomon error checking form and its more advanced implementation of using Galois Theory. The sample code by Henry Minsky would take a string of letters, convert it to binary form then add the appropriate code words before sending it to be decoded. This code was split in two so the encoding could be done completely separate from the decoding. The decoding part of the code was implemented on the decoding Arduino. The exact alterations to Henry Minsky's code are shown in Appendix (D).

## 3.3 Hardware Design Issues

### 3.3.1 Switch Mode Voltage Regulator

One of the hardware issues to come out of the test of the initial design was to replace the laptop with batteries to power the apparatus. This meant the apparatus would be more portable. The issue was AA batteries have a nominal voltage level of 1.5 volts and the apparatus required a 5 volt signal for the RF module. Available from an electronic retailer are battery holders that allow batteries to be placed in series so the combined voltage output could then be increased. Required was a battery holder big enough for four batteries that gave a total nominal voltage of 6 volts. if a battery holder with 3 slots was used the nominal voltage would only be 4.5 volts which would be under the level required for the apparatus. With the four slot battery holder being above the nominal voltage level required a voltage regulator was used to regulate

the voltage to stick within 0.2 volts of the nominal 5 volts. It was not noticed until after implementing this method the voltage regulator was of a switch mode type. This works by switching the power transistor on and off so that it is cutting the energy flow between the input and output. When the energy flow is cut the output is powered using energy storage devices such as inductors to buffer the energy [13]. This constant switch creates an electromagnetic change to the field around the regulator that impeded the RF signal during experimentation with the apparatus [13]. The power supply was causing the signal to become too noisy and non-controllable. The attempt to make the apparatus more portable by including an on-board power supply was unsuccessful due to the fact it would impede the results from testing. An alternative method in the end was to redesign the apparatus so a battery bank could power it. The battery bank had a nominal voltage output of 5 volts thus alleviating the need for the switch mode voltage regulator.

### 3.3.2 Adding a Comparator to Filter out Noise

During testing of the data transmission it became clear the RF signals were affected by an external environmental factors. This meant the receiver would be receiving random noise impulses that would jump up to a voltage of 2.5 volts. Due to the digital input from the Arduino being between the voltages of zero and five the threshold for a digital input to detect a high signal was at 2.5 volts. This was causing incorrect readings of the voltage signal to be made because the Arduino processor would see these pulses from, the noise and interpret it as a high voltage. This issue was corrected by incorporating a filter into the design. The filter was an operational amplifier that was wired up to act as a comparator for two voltages. A comparator works by comparing a signal voltage to a reference voltage, which in the case at hand 3 volts was used as the reference voltage [13]. If the signal voltage goes above the reference voltage the comparator outputs its positive high voltage. Alternatively if the signal voltage is below the reference voltage the comparator will output the negative low voltage [13]. By implementing this device into the experimental design the random noise pulses that were causing false readings are filtered out due to the reference voltage on the comparator being higher than the magnitude of the noise. Figure 16 below shows the wiring setup of the comparator.

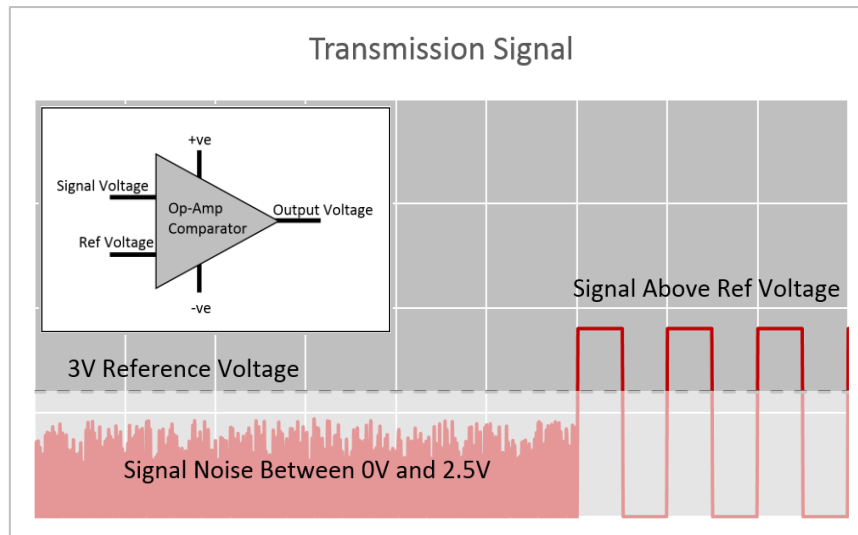


Figure 16 - Comparator Filter

### 3.3.3 Plug and Play Style Apparatus

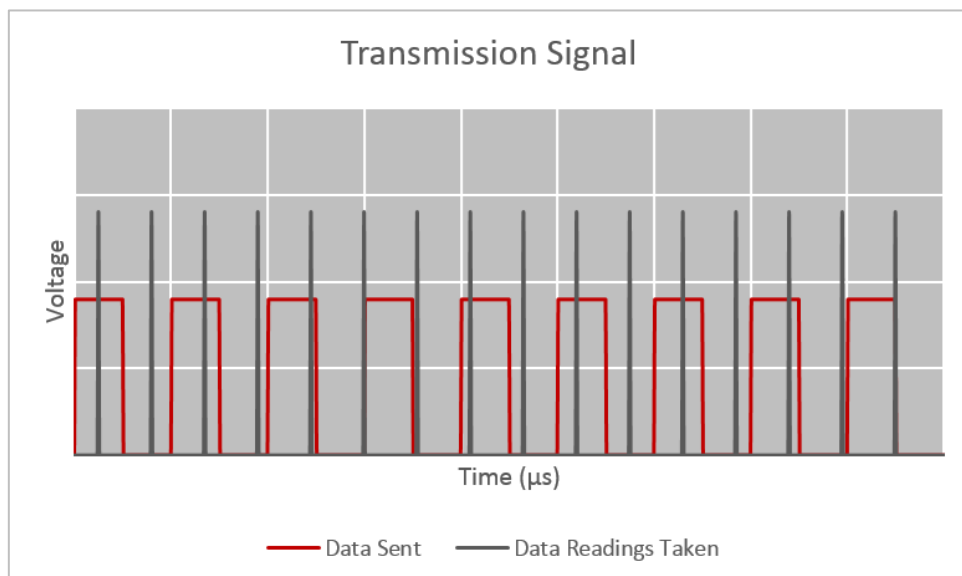
A lot of the electrical components being used in the experimental apparatus are not manufactured to be able to tolerate vigorous testing. It was found that the RF modules became temperamental over time with multiple tests and long running times. This does not affect the result because the RF modules used were to provide results into how reliable frequency transfer is, not how reliable low cost components are. It was found that after multiple tests where the RF modules were running for extensive periods of time and also not enclosed in a box to protect them from physical damage, the RF modules would eventually become unusable in the experiments. Due to the nature of this project requiring multiple experiments and testing to collect data the experiment apparatus was designed so that RF modules could be plugged into pre soldered female pin connectors making the system able to cope with plug and play functionality. This is a term used to describe a system that doesn't need to be stopped or turned off in order to make changes, also called hot swappable.

## 3.4 Software Design Issues

### 3.4.1 Processor Timing

The experimental apparatus required precise timing in order to achieve communication. Initially the processor coding was done so the pre-existing delay function in the Arduino language could be used. The delay function can be in milliseconds or microseconds, which was a high enough resolution to meet the

requirements for the project [14]. The Transmitter and Receiver were going to be run at 4500 Hz which sets the time period per bit being sent at 222 microseconds. The intention of using this function was to time the length an output pin was high and low for. On the other processor this timing would be used to determine when the processor should take readings. When this was first implemented the synchronisation between the encoder and decoder would become out of sync by around 1 millisecond per time period of data being sent. This effect can be seen in Figure 17 which shows how the decoder in the beginning was taking readings which were in sync only to become out of sync by the end of the message.



*Figure 17- Out of Sync Timing*

It was expected to see a reading from the decoder to be made in the middle of each rise in the data sent signal. What ended up happening was the readings started off being made in the middle of each rise in the signal then the readings began to shift to the right of the middle due to the timing obscurity which eventually ended up in a reading being made after a rise in the signal. The reason for this process becoming out of sync was due to the delay function being a software timer it was not able to take into account when delaying the processor the time it took for the code to process. This means that once the function is entered by the program the program will only process the wait function for the exact time it was intended. In some cases this was not a favoured approach because the delay function was used after the processing of the 'if loops' which determined if a one or zero was to be sent. These 'if loops' took a very small amount of time to process in the scale of microseconds which was adding on to the time that it took for the delay

function to execute. The result is shown in Figure 17 a build-up of processing time adding onto the delay time making the delay time greater than expected. To correct this timing issue the Arduino's timer interrupt was used as a way to incorporate the process timing in the delay [14]. The timer interrupt is a hardware timer on the Arduino board that starts timing from the time the board starts running the code downloaded to it. For the hardware microsecond timer this returns an unsigned long in relation to the amount of time it has been running. This unsigned long will overflow (go back to zero) after approximately 70 minutes of running. The way this hardware timer is used is to store its value in a variable then add on the time delay it is required for. Then use that variable as a condition for a while loop. This while loop will then run for the amount of time you added onto the variable taking into account any processing time from when the hardware timer was first stored in a variable (See Figure 18).

```

unsigned long current_time;           //
unsigned long delay_time = 222;      // delay time required for 4500 Hz

void setup ( ) {
}
void loop ( ) {

    current_time = micros();          // save the timer value into a variable
    current_time = Current_time + delay_time; // add desired delay time to variable
                                        //
                                        // processing of code, during which
                                        // the timer still counts
                                        //
                                        //
                                        // run loop for remainder of delay
    While(current_time > micros() ){
    }

}

```

Figure 18 Hardware Timer Loop

This new hardware timer method once incorporated the synchronisation of the two processors did improve to a point where the code would now be correctly interpreted due to the data readings being taken in the correct position. The fall back was the encoder and decoder relied on both processors staying at a pre-determined rate. This improved the accuracy of the processors but not the individual RF components. The RF component for the receiver did not always rise exactly when the transmitter would rise. The reason being the time it takes for the electromagnetic signal to travel between the transmitter and receiver. This caused the signal itself to be in intervals that were not exactly the specified time period. The effect caused



the two systems to become out of sync, not to the same magnitude as before the timers were corrected but enough to affect results gathered later in the investigation (see Figure 19).

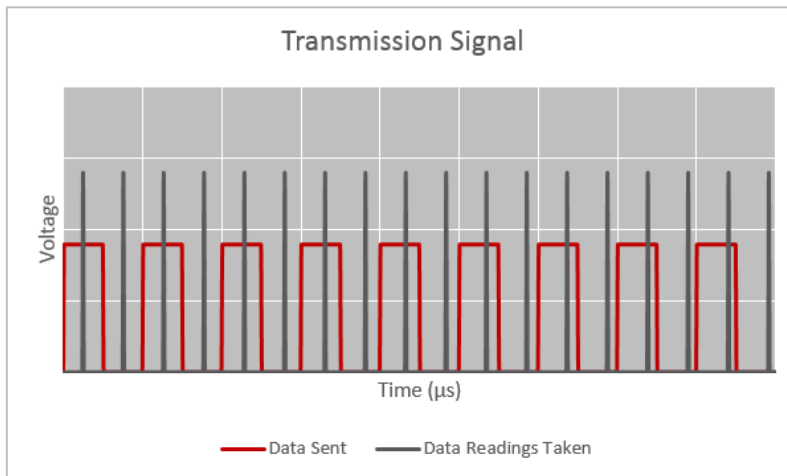


Figure 19 - Timer Improved Accuracy Data Readings

To change the code so it could account for this required a resynchronisation as often as possible. Initially the code would search for the handshake, as soon as the handshake was determined the decoder would take a number of readings at set intervals. This was changed so that that decoder would resynchronisation upon every rise in the signal. This was done by creating code which looked for a rising edge which represented the start of a bit period to reset the timer when voltage went from Low to High. This scenario is depicted in Figure 20.

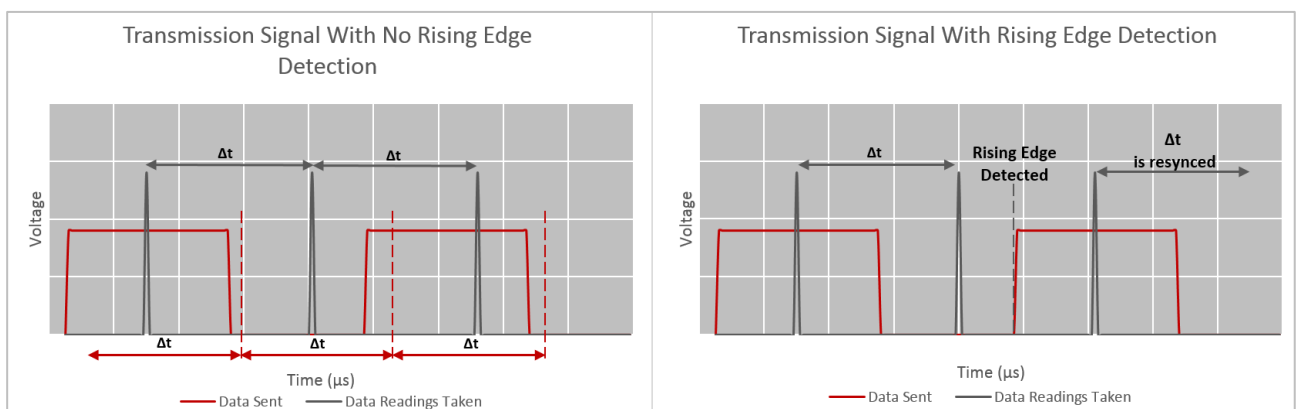


Figure 20 - Diagram Showing Rising Edge Resynchronisation

### 3.4.2 Reed Solomon Error Checking Code Footprint

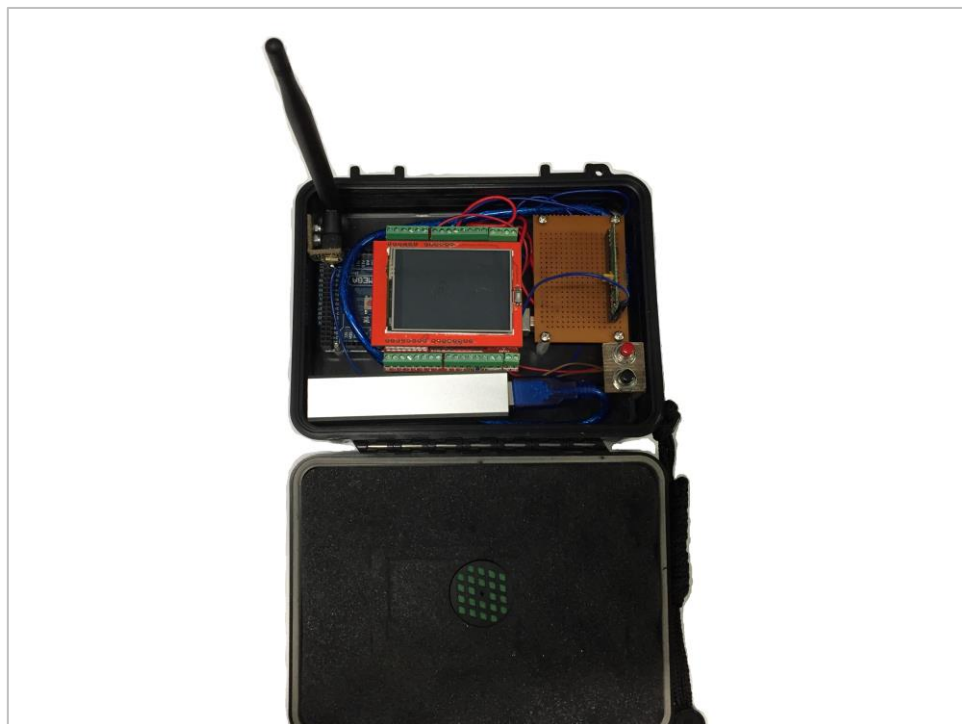
The Reed Solomon coding example that was used and adapted to fit the criteria for the investigations into reliable single direction frequency transfer ran into issues with the size of the example code file. The sample code was initially written in C++ and was then converted to the Arduino IDE language to make it compatible with the Arduino processors in the experimental apparatus. When trying to download this adapted code to the processors it became evident that the file size was too big for the on-board memory of the Arduino. The reason being the sample code used lookup tables in order to speed up processing of the Galois field calculations. The Arduino Uno (ATmega328P) was the initial specked processor with an on-board memory of 1kB Electronically Erasable Programmable Read Only Memory (EEPROM), 2kB Static Random Access Memory (SRAM) and 32KB of Flash Memory. In order for the Reed Solomon code lookup tables to be used as lookup tables, the SRAM needs enough storage space to save these lookup tables. The storage space required for these lookup tables was around 3.7kB which was 2 times bigger than the available SRAM space. An attempt was made to optimise this code and make the overall footprint of the code fit within the Arduino Uno without much success. The code was only able to be optimized to 2.2kB which was still too large. To overcome this issue an upgrade in the processor was made and the Arduino Uno was replaced with an Arduino Mega which had 8kB in SRAM with the ability to cater for the advanced error checking code.

### 3.5 Experimental Design

The final layout for the experimental apparatus was finalised after incorporating the necessary changes to account for the issues became evident along the way. The final experimental apparatus was mounted inside a pelican box so the circuitry could be protected. There were 2 versions of the experimental apparatus before the end result the first being the implementation with the Arduino Uno's and the second being the implementation with the two Arduino Mega's. The other noticeable difference was the HMI for the different versions of the electrical apparatus as the investigation went further and the experiments required more functionality from the apparatus the user interface was developed. Diagrams of the apparatus and the wiring schematic are shown in Appendix (E).



*Figure 21 - Final Transmitter Experimental Apparatus*



*Figure 22 - Final Receiver Experimental Apparatus*



*Figure 23 - Experimental Apparatus Enclosure*

## 4.0 Results

The results of the investigation are aligned with what was earlier determined as reliable data transmission. The key performance indicators that were determined earlier in this report are, error rate due to transmission speed, error rate due to distance and error rate due to an inhibited signal.

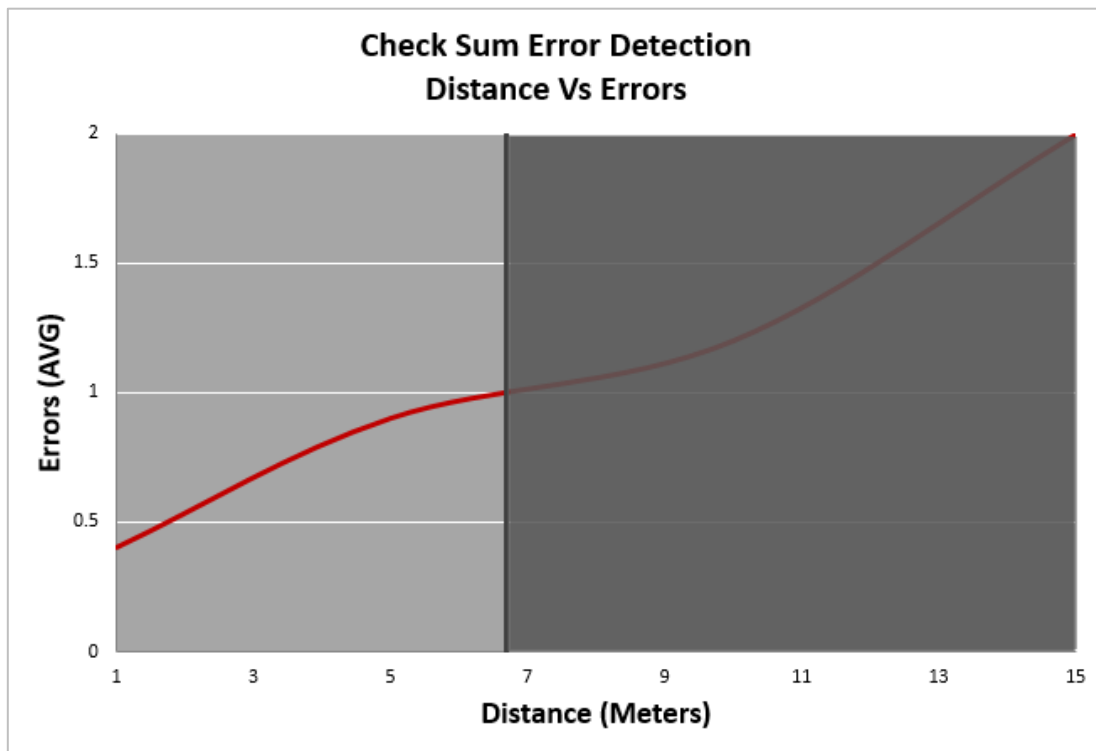
Of the performance indicators the three different types of forward error correction were tested. The three types being check sum, 2D matrix error Correction and Reed Solomon error correction. The intent of these experiments was to show how effective a frequency transmission could be, if supported by a method of forward error correction. As well as individually testing the performance KPI's, experiments were conducted where multiple KPI's were varied. An example of this could be, gradually increasing speed and decreasing signal strength. Another example was an experiment where the signal strength was decreased and the distance was increased. The multiple experiments were performed to try and record as many of the factors affecting frequency transmission and the significance of these factors both individually and when paired with other factors. The experiments carried out have been depicted in graph format to aid in the analysis of results. Where it was not clear to show a graphical representation of the results these were recorded in tables. Whilst conducting the following experiments a pass mark for what would be accepted

and what wouldn't be accepted was noted at one error on average per transmission. This meant that if in any of the experiments an average of over one error was transmitted this would represent the point in which the transmission was intolerable and would need to be improved in order for reliable frequency transmission. The aim of this being, to find the saturation point where the transmission could no longer handle different inhibiting variables.

## 5.0 Interpretation

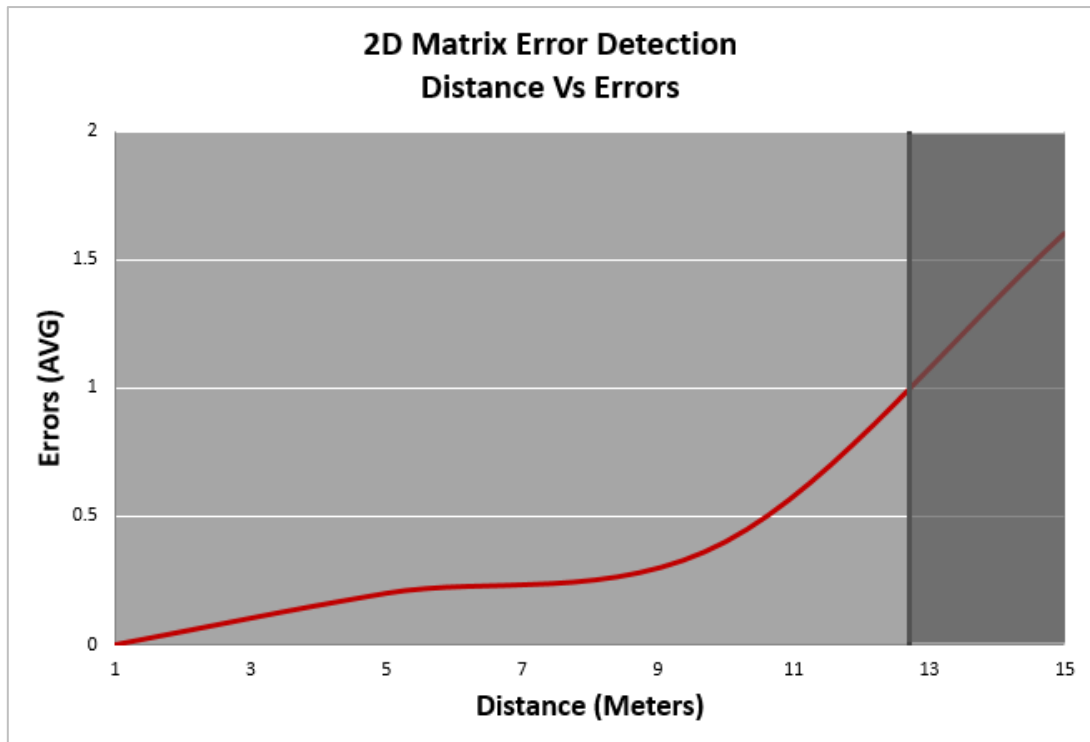
### 5.1 Experiment One Distance Vs Errors

The results gathered and shown in the figures below represent the maximum distance that the experimental apparatus was able to perform before failing the acceptance test of having less than one error average per transmission. The three graphs depict how each forward error correction method performed under test standards.



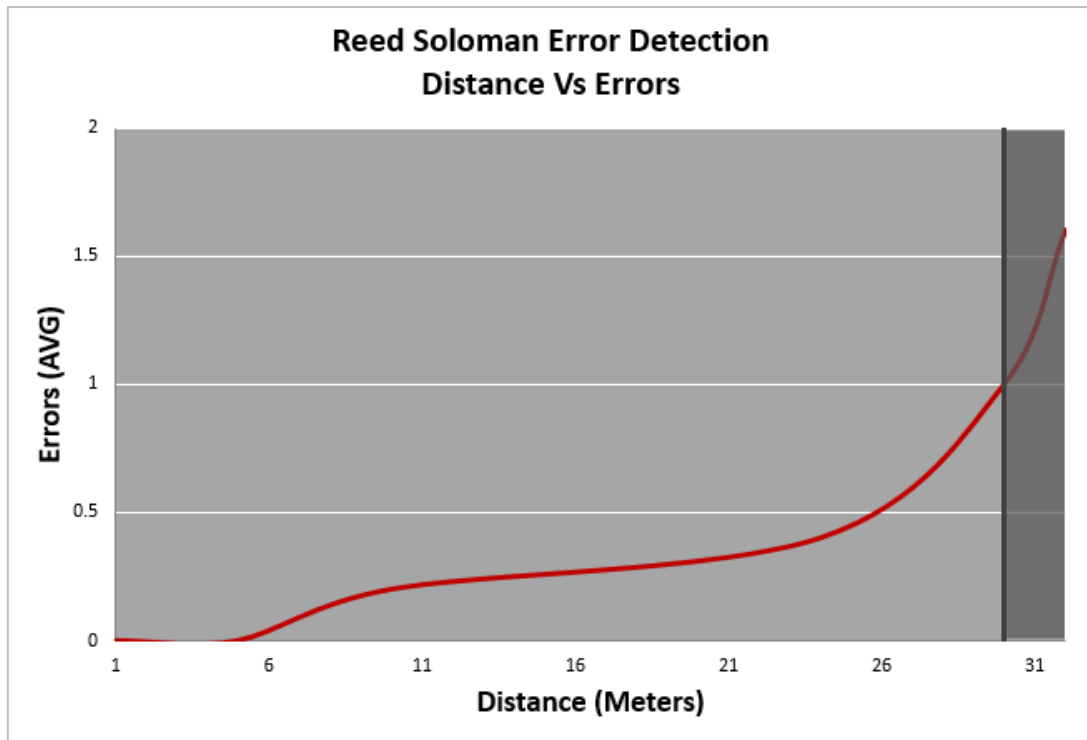
*Figure 24 - Experiment One Check Sum Vs Error Rate*

The first experiment was set in place so a baseline performance could be determined. The checksum method was not able to correct any errors but was able to detect if there was errors transmitted. The result shown in Figure 24 demonstrates that with no forward error correction the apparatus was able to perform to an acceptable standard up to approximately 7 meters.



*Figure 25 - Experiment One 2D Matrix Distance Vs Error Rate*

The second experiment incorporated the 2D matrix error correction method. This method is able to detect up to three errors but can only correct two. The results of this second experiment show the 2D matrix error correction was able to improve the performance of the experimental apparatus, which allowed it to perform to an acceptable standard for a further 6 meters in comparison to the first experiment. With a total distance of approximately 13 meters achieved.

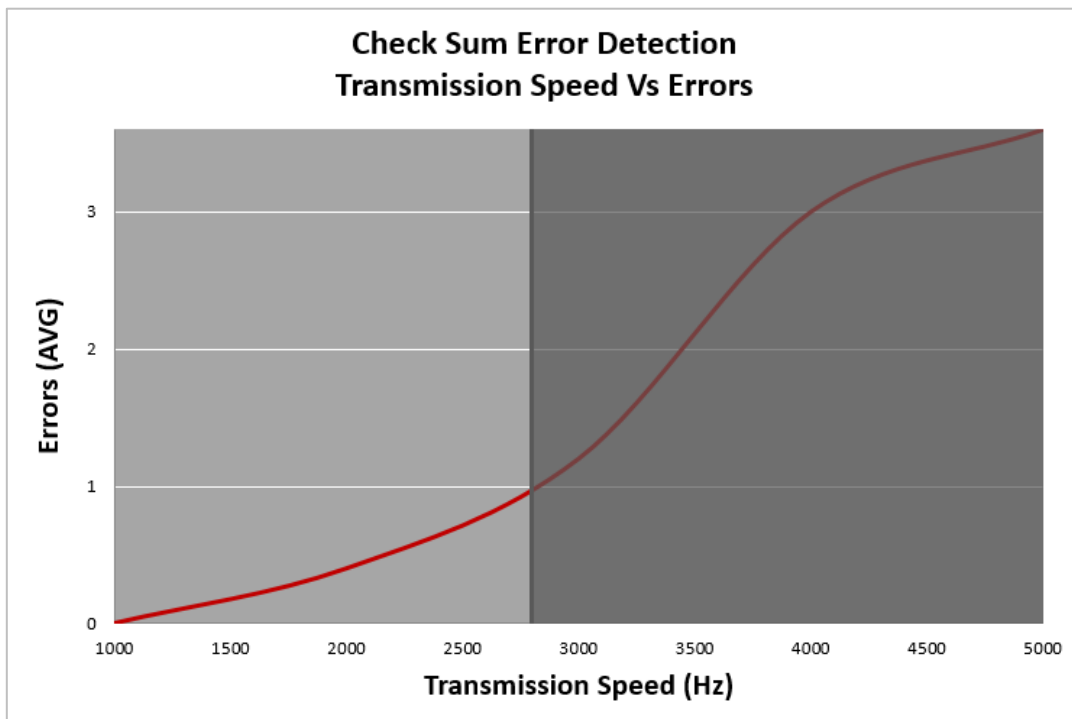


*Figure 26 - Experiment One Reed Solomon Distance Vs Error Rate*

The third experiment where distance was the variable being measured incorporated the advanced forward error checking technique. This method saw a significant improvement in the performance of the apparatus that clearly outlined the added benefit of having an advanced forward error correction method. The acceptable distance achieved was more than doubled that of the 2D matrix error checking method. The distance achieved as seen in Figure 26 was approximately 30 meters. The limitation to this experiment was that the distance had increased substantially and it was difficult to find a clear 30 meters unimpeded by external factors that would replicate the same conditions as the first two tests that were conducted indoors.

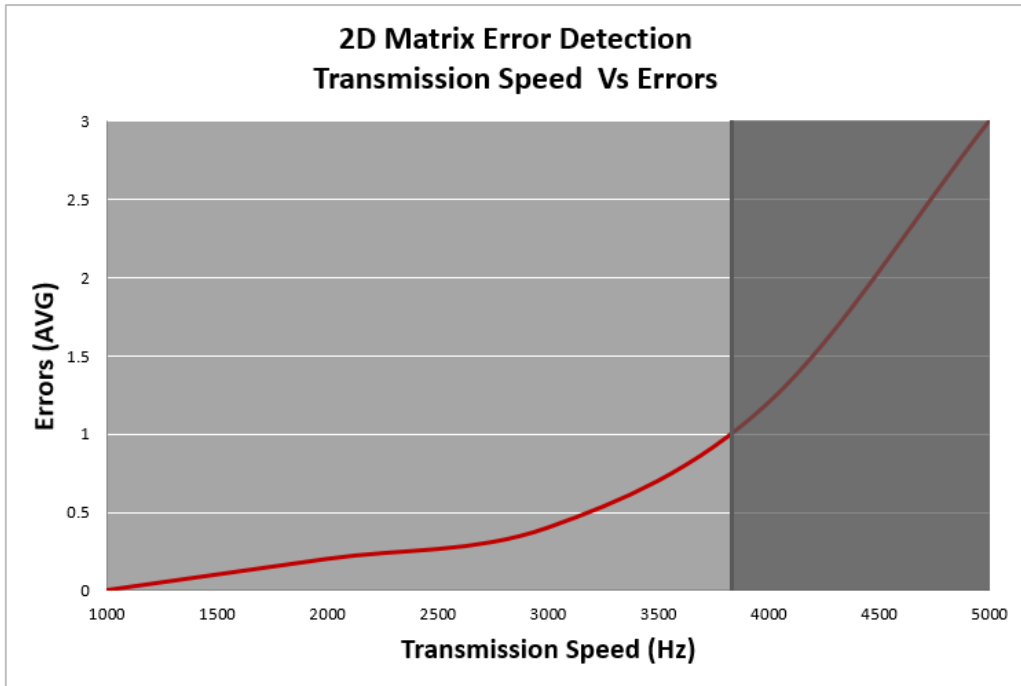


## 5.2 Experiment Two Transmission Rate Vs Errors



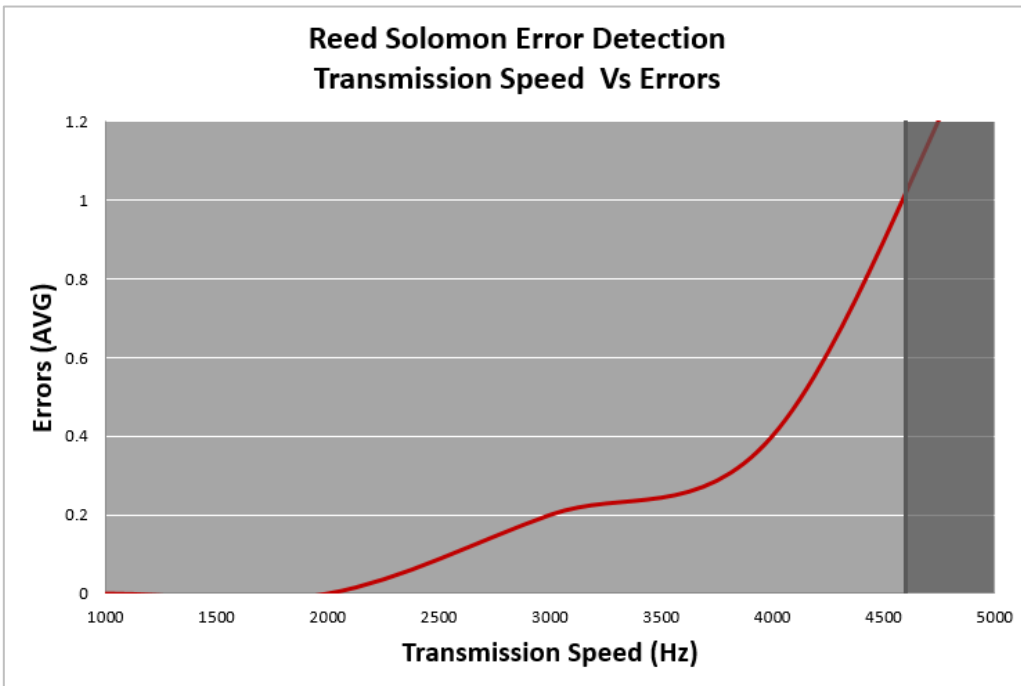
*Figure 27 - Experiment Two Check Sum Transmission Rate Vs Errors*

The second experiment conducted involved the transmission speed. The first test establishes a base line standard for how well the apparatus performs before any forward error correction is implemented. The baseline for the transmission speed was the maximum transmission speed that was attainable before failing the acceptance test of less than one error average was 2800Hz. This figure was founded by gradually decreasing the time delay in the code until the transmission speed resulted in more errors than what can be catered for in single direction communication.



*Figure 28 - Experiment Two 2D Matrix Error Correction Transmission Speed Vs Errors*

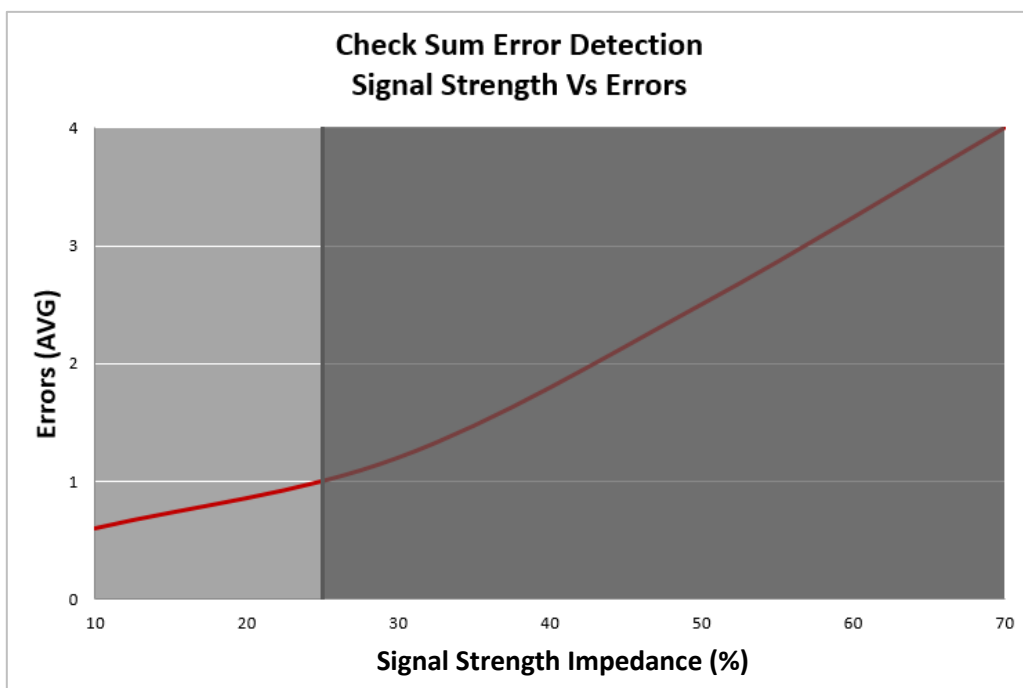
The second transmission experiment with the 2D matrix error correction technique managed to operate at 3800Hz before failing the acceptance test.



*Figure 29 - Experiment Two Reed Solomon Error Correction Transmission Speed Vs Errors*

The third test with Reed Solomon error correction achieved a better result than the 2D matrix error correction. What was noticeable though was since the first experiment where distance was tested the Reed Solomon method doubled the achieved distance that 2D matrix error checking achieved. In the second experiment Reed Solomon still performed better but the magnitude between the different types of forward error checking techniques was much more clustered when compared to experiment one. This suggests that the error correction technique can improve the achievable transmission rate of the signal but not by an amount of any significance. The assumed reasoning for this would be the RF devices were nearing on their maximum transmitting capacity were beyond the RF modules maximum transmitting speed the data saturates leaving the transmission leaving it no longer correctable. The final transmission speed achievable by Reed Solomon error correction was approximately 4500Hz.

### 5.3 Experiment Three Signal Strength Vs Errors



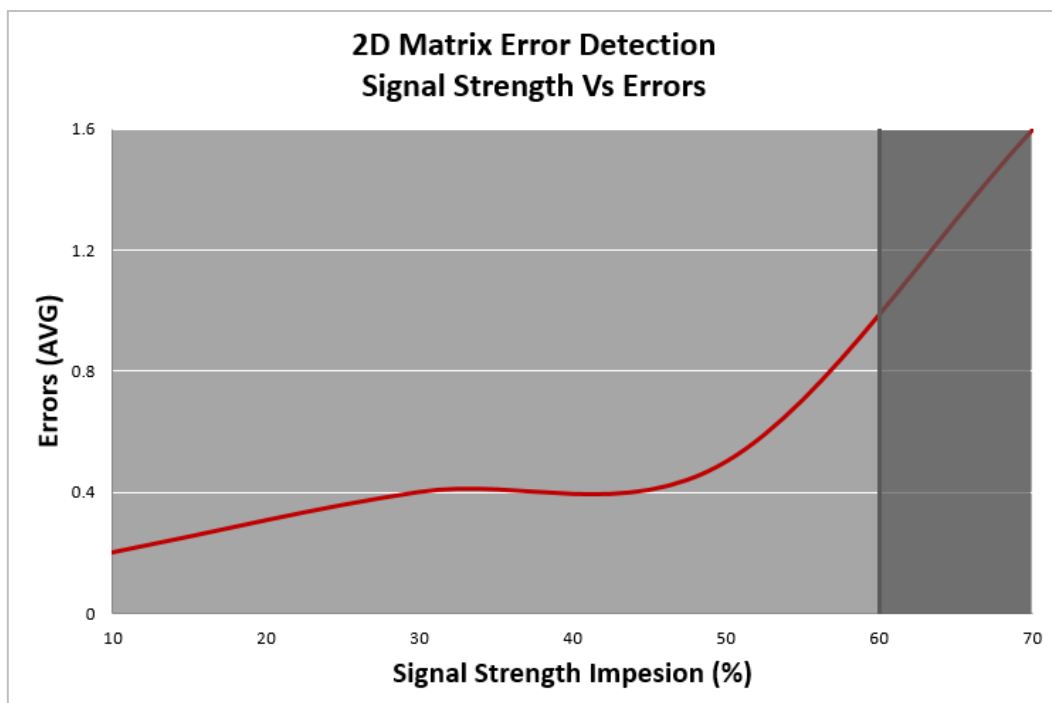
*Figure 30 - Experiment Three Check Sum Signal Strength Vs Error Rate*

The third experiment was a combination of inhibiting factors that were used to dampen the signal so that the effect of multiple external variables on the apparatus could be investigated. The first step to investigating this was to find the maximum amount of dampening possible that would allow transmission

to still occur. This would mean finding a percentage lower than 100 % inhibited. It was established that the most that the signal could be inhibited before becoming no longer transmittable was, with the voltage lowered to 3.3volts, the antenna being cut to a total length of 5cm, a transmission speed of 4288Hz and with 5 meters between the transmitter and receiver. Using this information a table was formed indicating different percentages of impression to the signal. Where each percentage increase shows an increase in the external variable relative to the maximum inhibited values previously found.

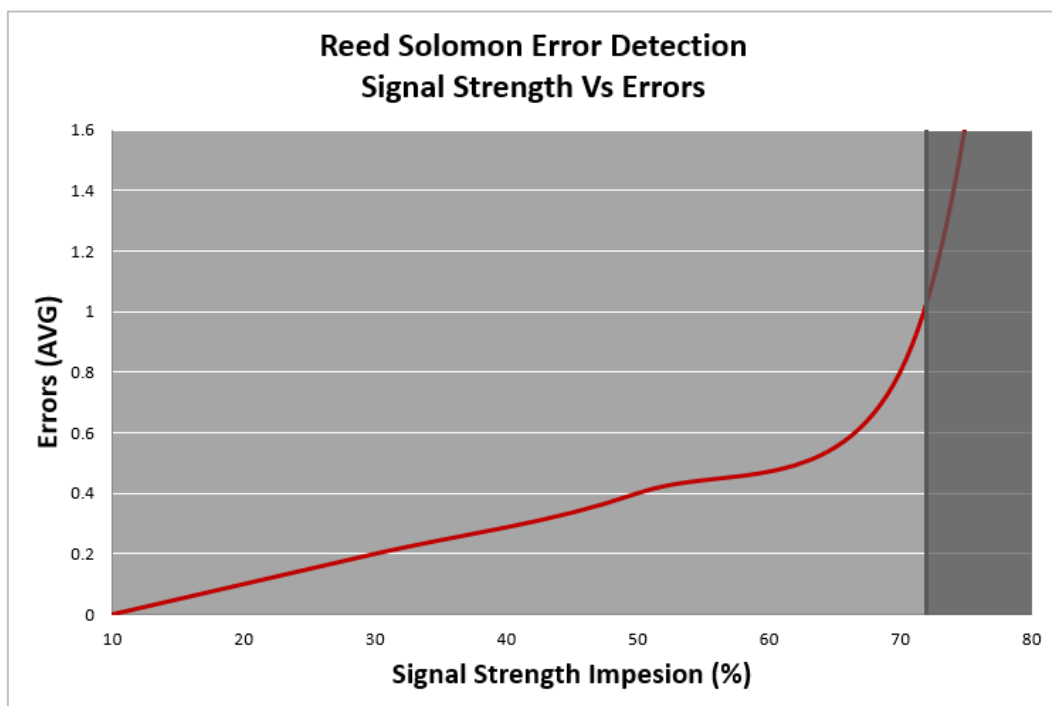
*Table 1 - Table showing the breakdown of the Impeding variables and their percentage for experiment three*

Transmission Inhibit Percentages Used For Experiment Three							
Signal Strength Impesion	10%	20%	30%	40%	50%	60%	70%
Power	5 Volts	5 Volts	5 Volts	5 Volts	3.3 Volts	3.3 Volts	3.3 Volts
Distance	5m	5m	5m	5m	5m	5m	5m
Transmission Rate	4288Hz	4288Hz	4288Hz	4288Hz	4288Hz	4288Hz	4288Hz
Antenna Length	17cm	15cm	13cm	11cm	9cm	7cm	5cm



*Figure 31 - Experiment Three 2D Matrix Signal Strength Vs Error Rate*

Figure 31 shows the performance of 2D matrix error correction under varying percentages of signal impedance. This form of frequency communication was able to tolerate 60% signal strength Impeded before not meeting the 1 error minimum per transmission requirement. The data found represents an exponential relationship where once 60% on the x-axis has been reached the error rate then saturates to an uncontrollable level. From the data shown it is an improvement from the checksum test with an overall acceptance at 60% which more than doubles the initial testing. This shows promising results towards determining if single direction frequency data transmission is reliable.



*Figure 32 - Experiment Three Reed Solomon Signal Strength Vs Error Rate*

The final experiment conducted with results shown in Figure 32. Using the advanced error correction method showed again an improvement how well it performed under different dampening effects to the transmission signal. The trend of this data also follows an exponential curve showing that the Reed Solomon method was able to maintain an acceptable level of communication even with 70% of the signal being impeded. This final experiment followed the trends of the experimental data that preceded by showing that more advance methods of forward error correction improved the performance of the Experimental apparatus. From this data is that a single directional communication method can be effective

at data transfer even when under poor signal condition. The experiments were able to provide a clear demonstration of an effective frequency data transfer apparatus, which addressed the majority of the research criteria.

Some limitations to the experiments were that there wasn't always a clear cut quantitative way to represent data. An attempt was made to control each of these variables in order to achieve the most accurate results possible. Variables that could not be controlled throughout the experiments were temperature and the operability of the RF modules.

## 6.0 Economic Comparison

As well as carrying out investigations into the performance of the experimental apparatus consideration towards the final cost of this implementation was compared. For a single direction frequency data transfer to be reliable it not only has to perform reliably it also needs to be an economically viable option. If the economic cost of this option outweighed a proven data transmission method by a great deal then the benefit of this investigated data transmission method would be superseded by a cheaper already established method for data transmission. The major difference in cost between bi-directional communication and single directional communication is that in single directional communication the cost is cut in half due two only one side of the transmission requiring a transmitter and a receiver on the opposite side, as opposed to bi-directional which requires both a transmitter and receiver. The other benefit is forward error correction allows the transmitter to send the message plus a given amount of parity rather than if the receiver requested a retransmit upon unsuccessful transmission. The outcome means that single directional transmission also has a lower overhead cost in energy consumption. Similarly if a comparison was to be between wired transmission and wireless transmission the wired transmission would not need the RF modules to send and transmit data. This is because the Arduino processors output pins can be directly used without the need for an external module. The added cost for a wired setup would be the price of the cable between the transmitter and receiver and potentially signal repeaters if the transmission is

occurring over a long distance. Below is a basic cost breakdown of the equipment required for single directional frequency transmission, bi-directional frequency transmission and wired transmission.

Single Direction Wireless Transmission	
1. X1 Transmitter	\$9.95
2. X1 Receiver	\$10.95
3. X2 Arduino Mega	\$32.95
4. X2 Antenna	\$2
<b>Total</b>	<b>\$90.80</b>

*Figure 33 - Economic comparison cost breakdown of single direction wireless transmission.*

Bi Directional Transmission Wireless	
1. X2 Transmitter	\$9.95
2. X2 Receiver	\$10.95
3. X2 Arduino Mega	\$32.95
4. X2 Antenna	\$2
<b>Total</b>	<b>\$111.70</b>

*Figure 34 - Economic comparison cost breakdown of bi directional wireless transmission.*

Bi Directional Wired Transmission	
1. 50m of 0.5mm <sup>2</sup> cable	\$22
2. X2 Arduino Mega	\$32.95
<b>Total</b>	<b>\$87.90</b>

*Figure 35 - Economic comparison cost breakdown of bi directional wired transmission.*

As seen in **Figure 33**, **Figure 34** & **Figure 35** the cost comparison between the different options shows single directional transmission as the cheaper option between frequency transmission methods.

## 7.0 Future Works

To further the research conducted in this report experiments could be conducted into the specific bandwidth that the RF modules are operating in. specifically looking into if the frequency of the carrier wave has any affect on the transmission of data. The reason it was not looked at up until this point was due to the required license to operate in alternative frequency bands. It would be beneficial to determine whether this would have any significant impact on the experimental apparatus.

Another area that could be improved upon is the Reed Solomon sample code footprint. The sample code had a large footprint and required a processor with more memory. If this code can be condensed then the processor would no longer need to be upgraded. This creates a positive economic argument that single direction frequency data transmission does not require a processor of advanced capabilities. Instead it improves the economic cost making the communication style more viable to implement.

With the experimental apparatus the components incorporated were limited to basic models due to a small budget. To have the ability to find results, which can stand up to more industrial methods, would require more advanced RF modules. It is for this reason that a possible future implementation for this project could be to trial more advanced hardware components to identify the difference in performance that advanced components will have on basic components.

Finally an experimental apparatus which incorporates bi directional communication and a wired experiment apparatus could be built so that long term power consumption studies can be conducted to provide definitive results on which method is the most power efficient.




## 8.0 Conclusions

After the studies conducted throughout, the results recorded showed a favourable lean towards single directional frequency data transmission, as being a reliable method of electrical communication. In the experiments carried out the results show an improvement in the systems capability in the three areas, which were set as KPI's at the beginning of the report. The transmission method was able to perform when deliberate inhibiting factors were introduced simulating external environmental factors. Based on the results currently gathered single directional frequency transmission is a reliable data transmission method. There are future works stated in this report which may provide constrictions to this statement such as single directional frequency transmission is only reliable in a given bandwidth or single directional frequency data transfer is reliable if accompanied by a more advanced processor. The findings also show that the more advanced the forward error correction method the better the performance of the overall system. After an economic comparison single directional frequency data transfer is also an option that can be justified in terms of cost due to its nature in not requiring a doubling up of transmitting and receiving components.

## 9.0 References

- [1] Charles S. Fitch. (2016, 27) Radio World. [Online]. <http://www.radioworld.com/headlines/0045/how-fm-stereo-came-to-life/337146>
- [2] Murat Torlak. (2015) Introduction To Digital Modulation.
- [3] Curtis L. Hembly, Richard W. Henry, and Martin Caulton, *Physical Electronics*, 2nd ed. New York: John Wiley and Sons, Inc, 1963.
- [4] G Kennedy and B Davis, *Electronic Communication Systems*, Fourth ed. ed. USA: McGraw Hill International, 1992, ISBN 0-07-112672-4.
- [5] IEEE. (2016) ETHW. [Online]. [http://ethw.org/Milestones:First\\_Wireless\\_Radio\\_Broadcast\\_by\\_Reginald\\_A.\\_Fessenden,\\_1906](http://ethw.org/Milestones:First_Wireless_Radio_Broadcast_by_Reginald_A._Fessenden,_1906)
- [6] Gary M Frost, *Early Fm Radio: Incremental Technology in twentieth Century America*, 9780801894404th ed., John Hopkins, Ed.: University Press, 2010.
- [7] Higher National Computing. (2016) Sqa.org.uk. [Online]. [http://www.sqa.org.uk/e-learning/NetTechDC01ECD/page\\_02.htm](http://www.sqa.org.uk/e-learning/NetTechDC01ECD/page_02.htm)
- [8] R Forster, *Manchester Encoding: Opposing Definitions Resolved*, 9th ed., IET Institution Of Engineering and Technology, Ed.: Engineering Science & Educational Journal, 2000.
- [9] Todd K. Moon, *Error Correcction Coding*, 9780471648000th ed. New Jersey: John Wiley and Sons, 2005.
- [10] Irving S. Reed and Gustave Solomon, *Polynomial Codes over Certain Finite Fields*, 1011370108018th ed., SAIM, Ed.: Journal of the Society for Industrial and Applied Mathematics, 1960.
- [11] David M. Lane. Onlinestatebook.com. [Online]. <http://onlinestatbook.com/2/probability/permutations.html>
- [12] Australian Communications and Media Authority. (2017) AMCA. [Online]. <http://www.acma.gov.au/theacma/australian-radiofrequency-spectrum-plan-spectrum-planning-acma>
- [13] Allan R. Hambley, *Electrical Engineering*, 6th ed., Andrew Gilfillan, Ed.: Pearsons, 2014.
- [14] John Boxall, *Arduino Workshop*, Serena Yang, Ed. San Francisco, CA, USA: No Starch Press, 2013.

## 10.0 Appendix (A) – RF Module Data Sheets

<p>*Frequency Range: 433.92 MHZ                  *Modulate Mode: ASK                  *Circuit Shape: LC                  *Data Rate: 4800 bps                  *Selectivity: -106 dB                  *Channel Spacing: 1MHZ                  *Supply Voltage: 5V                  * High Sensitivity Passive Design.                  *Simple To Apply with Low External Count.</p>	
---	--

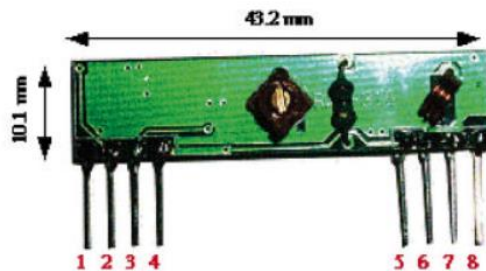
### DC Characteristics :

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
Vcc	Operating Supply Voltage		4.9	5	5.1	
I Tot	Operating Supply Voltage			4.5		
V Data	Data Out	1 Data = +200 uA ( High )	Vcc -0.5	Vcc		V
		1 Data = -10 uA ( Low )			0.3	V

### Electrical Characteristics :

Characteristics	Sym.	Min.	Typ.	Max.	Unit
Operating Radio Frequency	FC	433.72	433.92	434.12	MHZ
Sensitivity	Pref.			-106	dBm
Channel Width		-500		+ 500	KHZ
Noise Equivalent BW	NEB		5	4	KHZ
Baseboard Data Rate				3	KB/S
Receiver Turn On Time				3	ms

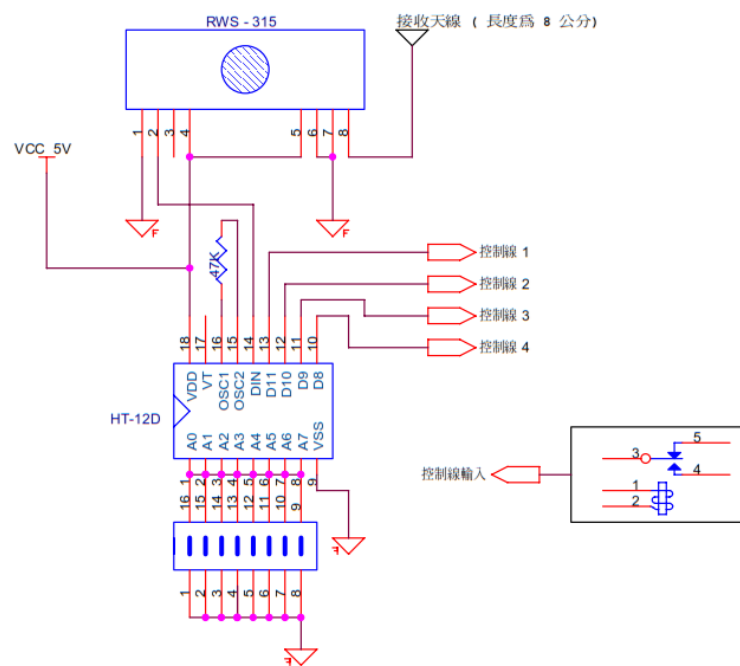
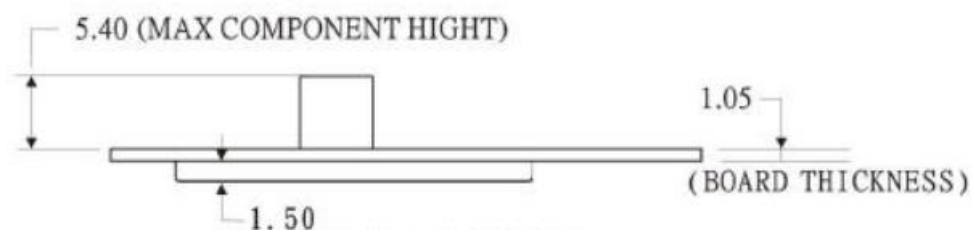
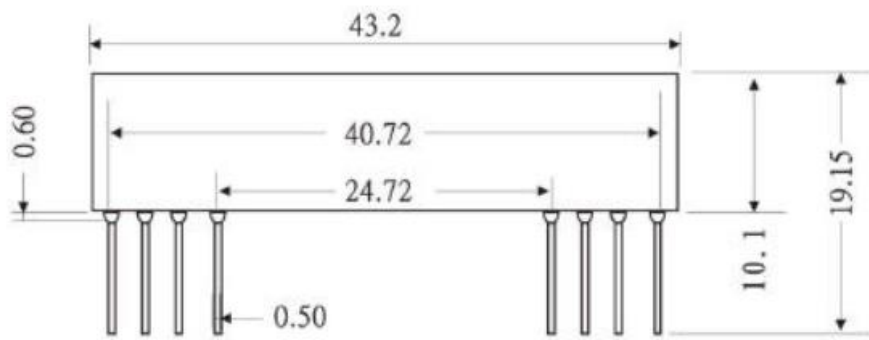
### Application Note:




pin 1 : Gnd  
 pin 2 : Digitak Output  
 pin 3 : Linear Output  
 pin 4 : Vcc  
 pin 5 : Vcc  
 pin 6 : Gnd  
 pin 7 : Gnd  
 pin 8 :ANT ( About 30 - 35 cm )

Modulation : AM  
 Supply Voltage : 5v dc

1



<p>*Frequency Range: 433.92 MHz          *Modulate Mode: ASK          *Circuit Shape: SAW          *Data Rate: 8kbps          *Supply Voltage: 3~ 12 V          *Power Supply and All Input / Output Pins: -0.3 to +12.0 V          *Non-Operating Case Temperature: -20 to +85 ℃          *Soldering Temperature ( 10 Seconds ) : 230 ℃ ( 10 Seconds )</p>	
---	---

Absolute Maximum Ratings

Rating	Value	Unit
Power Supply and All Input/Output Pins	-0.3 to +12.0	V
Non-Operating Case Temperature	-20 to +85	℃
Soldering Temperature(10 seconds)	230	℃

Electrical Characteristics, T=25℃, Vcc=3.6v, Freq=433.92MHz

Characteristic	Sym	Min.	Typ.	Max.	Unit
Operating Frequency (200KHz)	Vcc		433.92		MHz
Data Rate	ASK			8K	Kbps
Transmitter Performance(OOK@2.4kbps)					
Peak Input Current,12 Vdc Supply	ITP			45	mA
Peak Output Power	PO		10		mW
Turn On/ Turn Off Time	T ON/T OFF			1	US
Power Supply Voltage Range	Vcc	3		12	VDC
Operating Ambient Temperature	TA	-20		+85	℃
Tx Antenna Out (3V) +2.4dB	Vcc				mA

Application Note:



pin 1 : GND  
 pin 2 : Data in  
 pin 3 : VCC  
 pin 4 : ANT





## 12.0 Appendix (C) – Line Coding Style Comparison

The table below shows every possible 8-bit combination and then calculates the various parameters for each of the coding styles researched.

Data Possibility								NRTZ-L longest On Time	NRTZ-L Ave On Time %	NRTZ-M longest On Time	NRTZ-M Ave On Time %	RTZ longest On Time	RTZ Ave On Time %	Diff Man longest On Time	Diff Man Ave On Time %	Length On/Off longest On Time	Length On/Off Ave On Time %
B1	B2	B3	B4	B5	B6	B7	B8										
1	1	1	1	1	1	1	1	8	100	1	40	0.5	50	0.5	50	6.4	80
1	1	0	1	1	1	1	0	7	75	1	50	0.5	37.5	1	50	5.9	67.5
1	1	1	1	1	1	0	1	6	88	1	40	0.5	44	1	50	5.4	74
1	1	1	1	1	1	0	0	6	75	1	30	0.5	37.5	1	50	5.4	67.5
1	1	1	1	1	0	1	1	5	88	1	50	0.5	44	1	50	4.9	74
1	1	1	1	1	0	1	0	5	75	2	40	0.5	37.5	1	50	4.9	67.5
1	1	1	1	1	0	0	1	5	75	3	50	0.5	37.5	1	50	4.9	67.5
1	1	1	1	1	0	0	0	5	62	4	60	0.5	31	1	50	4.9	61
1	1	1	1	0	1	1	1	4	88	1	40	0.5	44	1	50	4.4	74
1	1	1	1	0	1	1	0	4	75	1	30	0.5	37.5	1	50	4.4	67.5
1	1	1	1	0	1	0	1	4	75	3	40	0.5	37.5	1	50	4.4	67.5
1	1	1	1	0	1	0	0	4	62	3	50	0.5	31	1	50	4.4	61
1	1	1	1	0	0	1	1	4	75	1	30	0.5	37.5	1	50	4.4	67.5
1	1	1	1	0	0	1	0	4	62	2	40	0.5	31	1	50	4.4	61
1	1	1	1	0	0	0	1	4	62	1	30	0.5	31	1	50	4.4	61
1	1	1	1	0	0	0	0	4	50	1	20	0.5	25	1	50	4.4	55
1	1	1	0	1	1	1	1	4	88	2	50	0.5	44	1	50	4.4	74
1	1	1	0	1	1	1	0	3	75	2	40	0.5	37.5	1	50	3.9	67.5
1	1	1	0	1	1	0	1	3	75	2	50	0.5	37.5	1	50	3.9	67.5
1	1	1	0	1	1	0	0	3	62	3	60	0.5	31	1	50	3.9	61
1	1	1	0	1	0	1	1	3	75	2	40	0.5	37.5	1	50	3.9	67.5
1	1	1	0	1	0	1	0	3	62	2	50	0.5	31	1	50	3.9	61
1	1	1	0	1	0	0	1	3	62	2	40	0.5	31	1	50	3.9	61
1	1	1	0	1	0	0	0	3	50	2	30	0.5	25	1	50	3.9	55
1	1	1	0	0	1	1	1	3	75	3	50	0.5	37.5	1	50	3.9	67.5
1	1	1	0	0	1	1	0	3	62	3	60	0.5	31	1	50	3.9	61
1	1	1	0	0	1	0	1	3	62	3	50	0.5	31	1	50	3.9	61
1	1	1	0	0	1	0	0	3	50	3	40	0.5	25	1	50	3.9	55
1	1	1	0	0	0	1	1	3	62	4	60	0.5	31	1	50	3.9	61
1	1	1	0	0	0	1	0	3	50	4	50	0.5	25	1	50	3.9	55
1	1	1	0	0	0	0	1	3	50	5	60	0.5	25	1	50	3.9	55
1	1	1	0	0	0	0	0	3	38	6	70	0.5	19	1	50	3.9	49
1	1	0	1	1	1	1	1	5	88	1	40	0.5	44	1	50	4.9	74
1	1	0	1	1	1	1	0	4	75	1	30	0.5	37.5	1	50	4.4	67.5
1	1	0	1	1	1	0	1	3	75	2	40	0.5	37.5	1	50	3.9	67.5
1	1	0	1	1	1	0	0	3	62	3	50	0.5	31	1	50	3.9	61
1	1	0	1	1	0	1	1	2	75	1	30	0.5	37.5	1	50	3.4	67.5
1	1	0	1	1	0	1	0	2	62	2	40	0.5	31	1	50	3.4	61
1	1	0	1	1	0	0	1	2	62	1	30	0.5	31	1	50	3.4	61
1	1	0	1	1	0	0	0	2	50	1	20	0.5	25	1	50	3.4	55
1	1	0	1	0	1	1	1	2	75	2	40	0.5	37.5	1	50	3.9	67.5
1	1	0	1	0	1	1	0	2	62	2	50	0.5	31	1	50	3.4	61
1	1	0	1	0	1	0	1	2	62	2	40	0.5	31	1	50	3.4	61
1	1	0	1	0	1	0	0	2	50	2	30	0.5	25	1	50	3.4	55
1	1	0	1	0	0	1	1	2	62	3	50	0.5	31	1	50	3.4	61
1	1	0	1	0	0	1	0	2	50	3	40	0.5	25	1	50	3.4	55
1	1	0	1	0	0	0	0	2	38	5	60	0.5	19	1	50	3.4	49
1	1	0	0	1	1	1	1	4	75	1	30	0.5	37.5	1	50	4.4	67.5
1	1	0	0	1	1	1	0	3	62	2	40	0.5	31	1	50	3.9	61
1	1	0	0	1	1	0	1	2	62	1	30	0.5	31	1	50	3.4	61
1	1	0	0	1	1	0	0	2	50	1	20	0.5	25	1	50	3.4	55
1	1	0	0	1	0	1	1	2	62	2	40	0.5	31	1	50	3.4	61
1	1	0	0	1	0	1	0	2	50	2	30	0.5	25	1	50	3.4	55
1	1	0	0	1	0	0	1	2	50	3	40	0.5	25	1	50	3.4	55
1	1	0	0	1	0	0	0	2	38	4	50	0.5	19	1	50	3.4	49
1	1	0	0	0	1	1	1	2	62	1	30	0.5	31	1	50	3.4	61
1	1	0	0	0	1	1	0	2	50	1	20	0.5	25	1	50	3.4	55
1	1	0	0	0	1	0	1	2	50	2	30	0.5	25	1	50	3.4	55
1	1	0	0	0	1	0	0	2	38	3	40	0.5	19	1	50	3.4	49
1	1	0	0	0	0	1	1	2	50	1	20	0.5	25	1	50	3.4	55
1	1	0	0	0	0	1	0	2	50	2	30	0.5	25	1	50	3.4	55
1	1	0	0	0	0	1	0	2	38	3	40	0.5	19	1	50	3.4	49
1	1	0	0	0	0	1	1	2	50	1	20	0.5	25	1	50	3.4	55
1	1	0	0	0	0	1	0	2	38	2	30	0.5	19	1	50	3.4	49



1	1	0	0	0	0	0	1	2	38	1	20	0.5	19	1	50	3.4	49	
1	1	0	0	0	0	0	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
1	0	1	1	1	1	1	1	1	88	2	50	0.5	44	1	50	2.9	74	
1	0	1	1	1	1	1	1	0	1	75	2	40	0.5	37.5	1	50	2.9	67.5
1	0	1	1	1	1	1	0	1	4	75	2	50	0.5	37.5	1	50	4.4	67.5
1	0	1	1	1	1	0	0	4	62	3	60	0.5	31	1	50	4.4	61	
1	0	1	1	1	0	1	1	3	75	2	40	0.5	37.5	1	50	3.9	67.5	
1	0	1	1	1	0	1	0	3	62	2	50	0.5	31	1	50	3.9	61	
1	0	1	1	1	0	0	1	3	62	2	40	0.5	31	1	50	3.9	61	
1	0	1	1	1	0	0	0	3	50	2	30	0.5	25	1	50	3.9	55	
1	0	1	1	0	1	1	1	2	75	2	50	0.5	37.5	1	50	3.4	67.5	
1	0	1	1	0	1	1	0	2	62	2	60	0.5	31	1	50	3.4	61	
1	0	1	1	0	1	0	1	2	62	2	50	0.5	31	1	50	3.4	61	
1	0	1	1	0	1	0	0	2	50	2	40	0.5	25	1	50	3.4	55	
1	0	1	1	0	0	1	1	2	62	3	60	0.5	31	1	50	3.4	61	
1	0	1	1	0	0	1	0	2	50	3	50	0.5	25	1	50	3.4	55	
1	0	1	1	0	0	0	1	2	50	4	60	0.5	25	1	50	3.4	55	
1	0	1	1	0	0	0	0	2	38	5	70	0.5	19	1	50	3.4	49	
1	0	1	0	1	1	1	1	1	75	2	40	0.5	37.5	1	50	2.9	67.5	
1	0	1	0	1	1	1	0	1	62	2	50	0.5	31	1	50	2.9	61	
1	0	1	0	1	1	0	1	2	62	2	40	0.5	31	1	50	3.4	61	
1	0	1	0	1	1	0	0	2	50	2	30	0.5	25	1	50	3.4	55	
1	0	1	0	1	0	1	1	1	62	2	50	0.5	31	1	50	2.9	61	
1	0	1	0	1	0	1	0	1	50	2	40	0.5	25	1	50	2.9	55	
1	0	1	0	1	0	0	1	1	50	3	50	0.5	25	1	50	2.9	55	
1	0	1	0	1	0	0	0	1	38	4	60	0.5	19	1	50	2.9	49	
1	0	1	0	0	1	1	1	1	62	2	40	0.5	31	1	50	2.9	61	
1	0	1	0	0	1	1	0	1	50	2	30	0.5	25	1	50	2.9	55	
1	0	1	0	0	1	0	1	1	50	2	40	0.5	25	1	50	2.9	55	
1	0	1	0	0	1	0	0	1	38	3	50	0.5	19	1	50	2.9	49	
1	0	1	0	0	1	0	0	1	50	2	30	0.5	25	1	50	2.9	55	
1	0	1	0	0	0	1	1	1	50	2	30	0.5	25	1	50	2.9	49	
1	0	1	0	0	0	0	0	1	25	2	20	0.5	12.5	1	50	2.9	42.5	
1	0	0	1	1	1	1	1	1	75	3	50	0.5	37.5	1	50	2.9	67.5	
1	0	0	1	1	1	1	0	1	62	3	60	0.5	31	1	50	2.9	61	
1	0	0	1	1	1	0	1	3	62	3	50	0.5	31	1	50	3.9	61	
1	0	0	1	1	1	0	0	3	50	3	40	0.5	25	1	50	3.9	55	
1	0	0	1	1	0	1	1	2	62	3	60	0.5	31	1	50	3.4	61	
1	0	0	1	1	0	1	0	2	50	3	50	0.5	25	1	50	3.4	55	
1	0	0	1	1	0	0	1	2	50	3	60	0.5	25	1	50	3.4	55	
1	0	0	1	1	0	0	0	2	38	4	70	0.5	19	1	50	3.4	49	
1	0	0	1	0	1	1	1	3	62	4	50	0.5	31	1	50	3.9	61	
1	0	0	1	0	1	1	0	2	50	3	40	0.5	25	1	50	3.4	55	
1	0	0	1	0	1	0	1	1	50	3	50	0.5	25	1	50	2.9	55	
1	0	0	1	0	1	0	0	1	38	3	60	0.5	19	1	50	2.9	49	
1	0	0	1	0	0	1	1	2	50	3	40	0.5	25	1	50	3.4	55	
1	0	0	1	0	0	1	0	1	38	3	50	0.5	19	1	50	2.9	49	
1	0	0	1	0	0	0	1	1	38	3	40	0.5	19	1	50	2.9	49	
1	0	0	1	0	0	0	0	1	25	4	40	0.5	12.5	1	50	2.9	42.5	
1	0	0	0	1	1	1	1	3	50	5	60	0.5	25	1	50	3.9	55	
1	0	0	0	0	1	1	0	2	38	5	70	0.5	19	1	50	3.4	49	
1	0	0	0	0	1	0	1	1	38	5	60	0.5	19	1	50	2.9	49	
1	0	0	0	0	1	0	0	1	25	5	50	0.5	12.5	1	50	2.9	42.5	
1	0	0	0	0	0	1	1	2	38	6	70	0.5	19	1	50	3.4	49	
1	0	0	0	0	0	1	0	1	25	6	60	0.5	12.5	1	50	2.9	42.5	
1	0	0	0	0	0	0	1	1	25	7	70	0.5	12.5	1	50	2.9	42.5	
1	0	0	0	0	0	0	0	1	12	8	80	0.5	6	1	50	2.9	36	
0	1	1	1	1	1	1	1	1	7	88	1	40	0.5	44	1	50	5.9	74
0	1	1	1	1	1	1	1	0	6	75	1	30	0.5	37.5	1	50	5.4	67.5
0	1	1	1	1	1	1	0	1	5	75	2	40	0.5	37.5	1	50	4.9	67.5
0	1	1	1	1	1	1	0	0	5	62	3	50	0.5	31	1	50	4.9	61
0	1	1	1	1	1	0	1	1	4	75	1	30	0.5	37.5	1	50	4.4	67.5

0	1	1	1	1	0	1	0	4	62	2	40	0.5	31	1	50	4.4	61
0	1	1	1	1	0	0	1	4	62	1	30	0.5	31	1	50	4.4	61
0	1	1	1	1	0	0	0	4	50	1	20	0.5	25	1	50	4.4	55
0	1	1	1	0	1	1	1	3	75	2	40	0.5	37.5	1	50	3.9	67.5
0	1	1	1	0	1	1	0	3	62	2	50	0.5	31	1	50	3.9	61
0	1	1	1	0	1	0	1	3	62	2	40	0.5	31	1	50	3.9	61
0	1	1	1	0	1	0	0	3	50	2	30	0.5	25	1	50	3.9	55
0	1	1	1	0	0	1	1	3	62	3	50	0.5	31	1	50	3.9	61
0	1	1	1	0	0	1	0	3	50	3	40	0.5	25	1	50	3.9	55
0	1	1	1	0	0	0	1	3	50	4	50	0.5	25	1	50	3.9	55
0	1	1	1	0	0	0	0	3	38	5	60	0.5	19	1	50	3.9	49
0	1	1	0	1	1	1	1	4	75	1	30	0.5	37.5	1	50	4.4	67.5
0	1	1	0	1	1	1	0	3	62	2	40	0.5	31	1	50	3.9	61
0	1	1	0	1	1	0	1	2	62	1	30	0.5	31	1	50	3.4	61
0	1	1	0	1	1	0	0	2	50	1	20	0.5	25	1	50	3.4	55
0	1	1	0	1	0	1	1	2	62	2	40	0.5	31	1	50	3.4	61
0	1	1	0	1	0	1	0	2	50	2	30	0.5	25	1	50	3.4	55
0	1	1	0	1	0	0	1	2	50	3	40	0.5	25	1	50	3.4	55
0	1	1	0	1	0	0	0	2	38	4	50	0.5	19	1	50	3.4	49
0	1	1	0	0	1	1	1	3	62	1	30	0.5	31	1	50	3.9	61
0	1	1	0	0	1	1	0	2	50	1	20	0.5	25	1	50	3.4	55
0	1	1	0	0	1	0	1	2	50	2	30	0.5	25	1	50	3.4	55
0	1	1	0	0	1	0	0	2	38	3	40	0.5	19	1	50	3.4	49
0	1	1	0	0	0	1	1	2	50	1	20	0.5	25	1	50	3.4	55
0	1	1	0	0	0	1	0	2	38	2	30	0.5	19	1	50	3.4	49
0	1	1	0	0	0	0	1	2	38	1	20	0.5	19	1	50	3.4	49
0	1	1	0	0	0	0	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5
0	1	0	1	1	1	1	1	5	75	2	40	0.5	37.5	1	50	4.9	67.5
0	1	0	1	1	1	1	0	4	62	2	50	0.5	31	1	50	4.4	61
0	1	0	1	1	1	0	1	3	62	2	40	0.5	31	1	50	3.9	61
0	1	0	1	1	1	0	0	3	50	2	30	0.5	25	1	50	3.9	55
0	1	0	1	1	0	1	1	2	62	2	50	0.5	31	1	50	3.4	61
0	1	0	1	1	0	1	0	2	50	2	40	0.5	25	1	50	3.4	55
0	1	0	1	1	0	0	1	2	50	3	50	0.5	25	1	50	3.4	55
0	1	0	1	1	0	0	0	2	38	4	60	0.5	19	1	50	3.4	49
0	1	0	1	0	1	1	1	3	62	2	40	0.5	31	1	50	3.9	61
0	1	0	1	0	1	1	0	2	50	2	30	0.5	25	1	50	3.4	55
0	1	0	1	0	1	0	1	1	50	2	40	0.5	25	1	50	2.9	55
0	1	0	1	0	1	0	0	1	38	3	50	0.5	19	1	50	2.9	49
0	1	0	1	0	0	1	1	2	50	2	30	0.5	25	1	50	3.4	55
0	1	0	1	0	0	1	0	1	38	2	40	0.5	19	1	50	2.9	49
0	1	0	1	0	0	0	1	1	38	2	30	0.5	19	1	50	2.9	49
0	1	0	1	0	0	0	0	1	25	2	20	0.5	12.5	1	50	2.9	42.5
0	1	0	0	1	1	1	1	4	62	3	50	0.5	31	1	50	4.4	61
0	1	0	0	1	1	1	0	3	50	3	40	0.5	25	1	50	3.9	55
0	1	0	0	1	1	0	1	2	50	3	50	0.5	25	1	50	3.4	55
0	1	0	0	1	1	0	0	2	38	3	60	0.5	19	1	50	3.4	49
0	1	0	0	1	0	1	1	2	50	3	40	0.5	25	1	50	3.4	55
0	1	0	0	1	0	1	0	1	38	3	50	0.5	19	1	50	2.9	49
0	1	0	0	1	0	0	1	1	38	3	40	0.5	19	1	50	2.9	49
0	1	0	0	1	0	0	0	1	25	3	30	0.5	12.5	1	50	2.9	42.5
0	1	0	0	1	0	0	1	3	50	4	50	0.5	25	1	50	3.9	55
0	1	0	0	0	1	1	0	2	38	4	60	0.5	19	1	50	3.4	49
0	1	0	0	0	1	0	1	1	38	4	50	0.5	19	1	50	2.9	49
0	1	0	0	0	1	0	0	1	25	4	40	0.5	12.5	1	50	2.9	42.5
0	1	0	0	0	0	1	1	2	38	5	60	0.5	19	1	50	3.4	49
0	1	0	0	0	0	1	0	1	25	5	50	0.5	12.5	1	50	2.9	42.5
0	1	0	0	0	0	0	1	1	25	6	60	0.5	12.5	1	50	2.9	42.5
0	1	0	0	0	0	0	0	1	12	7	70	0.5	6	1	50	2.9	36
0	0	1	1	1	1	1	1	6	75	1	30	0.5	37.5	1	50	5.4	67.5
0	0	1	1	1	1	1	0	5	62	2	40	0.5	31	1	50	4.9	61
0	0	1	1	1	1	0	1	4	62	1	30	0.5	31	1	50	4.4	61
0	0	1	1	1	1	0	0	4	50	1	20	0.5	25	1	50	4.4	55
0	0	1	1	1	0	1	1	3	62	2	40	0.5	31	1	50	3.9	61
0	0	1	1	1	0	1	0	3	50	2	30	0.5	25	1	50	3.9	55
0	0	1	1	1	0	0	1	3	50	3	40	0.5	25	1	50	3.9	55
0	0	1	1	1	0	0	0	3	38	4	50	0.5	19	1	50	3.9	49
0	0	1	1	0	1	1	1	3	62	1	30	0.5	31	1	50	3.9	61
0	0	1	1	0	1	1	0	2	50	1	20	0.5	25	1	50	3.4	55
0	0	1	1	0	1	0	1	2	50	2	30	0.5	25	1	50	3.4	55
0	0	1	1	0	1	0	0	2	38	3	40	0.5	19	1	50	3.4	49

0	0	1	1	0	0	1	1	2	50	1	20	0.5	25	1	50	3.4	55	
0	0	1	1	0	0	1	0	2	38	2	30	0.5	19	1	50	3.4	49	
0	0	1	1	0	0	0	1	2	38	1	20	0.5	19	1	50	3.4	49	
0	0	1	1	0	0	0	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
0	0	1	0	1	1	1	1	4	62	2	40	0.5	31	1	50	4.4	61	
0	0	1	0	1	1	1	0	3	50	2	30	0.5	25	1	50	3.9	55	
0	0	1	0	1	1	0	1	2	50	2	40	0.5	25	1	50	3.4	55	
0	0	1	0	1	1	0	0	2	38	3	50	0.5	19	1	50	3.4	49	
0	0	1	0	1	0	1	1	2	50	2	30	0.5	25	1	50	3.4	55	
0	0	1	0	1	0	1	0	1	38	2	40	0.5	19	1	50	2.9	49	
0	0	1	0	1	0	0	1	1	38	2	30	0.5	19	1	50	2.9	49	
0	0	1	0	1	0	0	0	1	25	2	20	0.5	12.5	1	50	2.9	42.5	
0	0	1	0	0	1	1	1	3	50	3	40	0.5	25	1	50	3.9	55	
0	0	1	0	0	1	1	0	2	38	3	50	0.5	19	1	50	3.4	49	
0	0	1	0	0	1	0	1	1	38	3	40	0.5	19	1	50	2.9	49	
0	0	1	0	0	1	0	0	1	25	3	30	0.5	12.5	1	50	2.9	42.5	
0	0	1	0	0	0	1	1	2	38	4	50	0.5	19	1	50	3.4	49	
0	0	1	0	0	0	1	0	1	25	4	40	0.5	12.5	1	50	2.9	42.5	
0	0	1	0	0	0	0	1	1	25	5	50	0.5	12.5	1	50	2.9	42.5	
0	0	1	0	0	0	0	0	1	12	6	60	0.5	6	1	50	2.9	36	
0	0	0	1	1	1	1	1	5	62	1	30	0.5	31	1	50	4.9	61	
0	0	0	1	1	1	1	0	4	50	1	20	0.5	25	1	50	4.4	55	
0	0	0	1	1	1	0	1	3	50	2	30	0.5	25	1	50	3.9	55	
0	0	0	1	1	1	0	0	3	38	3	40	0.5	19	1	50	3.9	49	
0	0	0	1	1	0	1	1	2	50	1	20	0.5	25	1	50	3.4	55	
0	0	0	1	1	0	1	0	2	38	2	30	0.5	19	1	50	3.4	49	
0	0	0	1	1	0	0	1	2	38	1	20	0.5	19	1	50	3.4	49	
0	0	0	1	1	0	0	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
0	0	0	1	0	1	1	1	3	50	2	30	0.5	25	1	50	3.9	55	
0	0	0	1	0	1	1	0	2	38	2	40	0.5	19	1	50	3.4	49	
0	0	0	1	0	1	0	1	1	38	2	30	0.5	19	1	50	2.9	49	
0	0	0	1	0	1	0	0	1	25	2	20	0.5	12.5	1	50	2.9	42.5	
0	0	0	1	0	0	1	1	2	38	3	40	0.5	19	1	50	3.4	49	
0	0	0	1	0	0	1	0	1	25	3	30	0.5	12.5	1	50	2.9	42.5	
0	0	0	1	0	0	0	1	1	12	5	50	0.5	6	1	50	2.9	36	
0	0	0	0	1	1	1	1	4	50	1	20	0.5	25	1	50	4.4	55	
0	0	0	0	1	1	1	0	3	38	2	30	0.5	19	1	50	3.9	49	
0	0	0	0	1	1	0	1	2	38	1	20	0.5	19	1	50	3.4	49	
0	0	0	0	1	1	0	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
0	0	0	0	1	0	1	1	2	38	2	30	0.5	19	1	50	3.4	49	
0	0	0	0	1	0	1	0	1	25	2	20	0.5	12.5	1	50	2.9	42.5	
0	0	0	0	1	0	0	1	1	25	3	30	0.5	12.5	1	50	2.9	42.5	
0	0	0	0	1	0	0	0	1	12	4	40	0.5	6	1	50	2.9	36	
0	0	0	0	0	1	1	1	3	38	1	20	0.5	19	1	50	3.9	49	
0	0	0	0	0	1	1	0	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
0	0	0	0	0	1	0	1	1	25	2	20	0.5	12.5	1	50	2.9	42.5	
0	0	0	0	0	1	0	0	1	12	3	30	0.5	6	1	50	2.9	36	
0	0	0	0	0	0	1	1	2	25	1	10	0.5	12.5	1	50	3.4	42.5	
0	0	0	0	0	0	1	0	1	12	2	20	0.5	6	1	50	2.9	36	
0	0	0	0	0	0	0	1	1	12	1	10	0.5	6	1	50	2.9	36	
0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0.5	50	2.4	30	
<b>Averages</b>									2	50	3	40	0.5	25	1	50	4	55

## 13.0 Appendix (D) – Code Alterations

\* Copyright Henry Minsky (hqm@alum.mit.edu) 1991-2009

\*

\* This software library is licensed under terms of the GNU GENERAL

\* PUBLIC LICENSE

\*

```
unsigned int byte_number = 0;
unsigned char msg[] = "ENG470 Murdoch Uni";
```

```
unsigned char codeword[128];
char sbuf[128];
int count = sizeof(msg+NPAR);
```

```
int Sending = 13;
int delay1 = 80;
unsigned int wait;
```

```
void setup () {
int erasures;
int nerasures = 0;
```

```
/* Initialization the ECC library */
```

```
initialize_ecc ();
```

```
/* ***** */
```

```
/* Encode data into codeword, adding NPAR parity bytes */
```

```
encode_data(msg, sizeof(msg), codeword);
```

```
//-----
```

```
/* Add one error and two erasures */
```

```
//byte_err(0x23, 1, codeword);
```

```
//byte_err(0x34, 7, codeword);
```

```
//byte_err(0x45, 14, codeword);
```

```
//byte_err(0x50, 5, codeword);
```

```
//byte_err(0x56, 9, codeword);
```

```
//-----
```

```
unsigned char codeword1[sizeof(msg)+NPAR];
sprintf(sbuf, "Encoded data is: \"%s\"\n", codeword);
Serial.print(sbuf);
for(int i =0; i< 50; i++){ codeword1[i+1] = codeword[i];}
#define ML (sizeof (msg) + NPAR)
sprintf(sbuf, "with some errors: \"%s\"\n", codeword);
Serial.print(sbuf);
```

```
/* we need to indicate the position of the erasures. Erasure
```

```

    Positions are indexed (1 based) from the end of the message... */
//erasures[nerasures++] = ML-24;
//erasures[nerasures++] = ML-29;

/* Now decode -- encoded codeword size must be passed */
//decode_data(codeword, ML);

/* check if syndrome is all zeros */
//if (check_syndrome () != 0) {
//correct_errors_erasures (codeword,
// ML,
// nerasures,
// erasures);

    sprintf(sbuf, "Corrected codeword: \"%s\"\n", codeword);
    Serial.print(sbuf);
//}

#include <stdio.h>
#include <stdlib.h>
#include "ecc.h"
//-----
#include <Adafruit_GFX.h> // Hardware-specific library
#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;
#include <stdint.h>

```

### Alterations Made to the Above Code

The above code was changed from the original so that it was able to send on one Arduino and Receive on another. The changes made to the transmitting coder were;

1. Change the data sent to a string that could be input but the user. In the code above the string being sent is Murdoch Uni ENG 470.
  2. The next change was this string was saved into an array and put through Henry Minsky's Function to add the Reed Solomon Code words.
  3. Once the array had the code words added to it was this array was converted to a binary array. This Binary Array was sent to the Transmission Loop Code So that it could then be sent.
- 
1. The receiving code used the exact same code except the data being transferred in binary form was saved in an array from the least significant bit to the most significant bit.

2. This array was then converted to character data type so that the code Henry Minsky wrote could apply the Galois theory to error detect and correct the data in the received array.
3. This data was then printed on the LCD screen using a 'for' loop to print each character in each position of the array.

Brandon Butler wrote the following code, it is the code used in the transmitting apparatus.

```
#define YP A2 // must be an analog pin, use "An" notation!
#define XM A3 // must be an analog pin, use "An" notation!
#define YM 8 // can be a digital pin
#define XP 9 // can be a digital pin

#define BLACK 0x0000
#define BLUE 0x001F
#define RED 0xF800
#define GREEN 0x07E0
#define CYAN 0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE 0xFFFF
#define ORANGE 0xFD20
#define GREENISH 0xFFE0

//-----
int TransmissionSetup = 0;
int menu = 0;
int Enter = 0;
int Parity = 0;
int Error = 0;
int counter3 = 0;
//-----

unsigned int byte_number = 0;
unsigned char msg[] = "ENG470 Murdoch Uni";

unsigned char codeword[128];
char sbuf[128];
int count = sizeof(msg+NPAR);

/* Some debugging routines to introduce errors or erasures
   into a codeword.
   */
/* Introduce a byte error at LOC */
void byte_err (int err, int loc, unsigned char *dst)
```

```

{
  sprintf(sbuf, "Adding Error at loc %d, data %#x\n", loc, dst[loc-1]);
  Serial.print(sbuf);
  dst[loc-1] ^= err;
}

/* Pass in location of error (first byte position is
   labeled starting at 1, not 0), and the codeword.
*/
void byte_erasure (int loc, unsigned char dst[], int cwsiz, int erasures[])
{
  sprintf(sbuf, "Erasure at loc %d, data %#x\n", loc, dst[loc-1]);
  Serial.print(sbuf);
  dst[loc-1] = 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////

int Sending = 13;
int delay1 = 80;
unsigned int wait;

void setup() {
  Serial.begin(9600); // open the serial port at 9600 bps:
  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  Serial.begin(9600); //
  tft.reset(); //
  uint16_t id = tft.readID(); //
  tft.begin(id); //
  tft.setRotation(3); // Screen (LCD) settings
  tft.fillRect(BLACK); //
  tft.setTextColor(GREEN, BLACK); //
  tft.setCursor(40,0); //
  tft.setTextSize(2); //
  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  pinMode(12,INPUT);
  pinMode(11,INPUT);
  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  /////
  int erasures[16];
  int nerasures = 0;

  /* Initialization the ECC library */

  initialize_ecc ();

  /* ***** */

```

```

/* Encode data into codeword, adding NPAR parity bytes */
encode_data(msg, sizeof(msg), codeword);

//-----

/* Add one error and two erasures */
//byte_err(0x23, 1, codeword);
//byte_err(0x34, 7, codeword);
//byte_err(0x45, 14, codeword);
//byte_err(0x50, 5, codeword);
//byte_err(0x56, 9, codeword);

//-----

unsigned char codeword1[sizeof(msg)+NPAR];

sprintf(sbuf, "Encoded data is: \"%s\"\n", codeword);
Serial.print(sbuf);
for(int i=0; i< 50; i++){ codeword1[i+1] = codeword[i];}

#define ML (sizeof (msg) + NPAR)
Serial.print(ML);

sprintf(sbuf, "with some errors: \"%s\"\n", codeword);
Serial.print(sbuf);
for(int i=0; i<50; i++){Serial.print(char(codeword[i]));}
Serial.println("");

/* We need to indicate the position of the erasures. Erasure
positions are indexed (1 based) from the end of the message... */

//erasures[nerasures++] = ML-24;
//erasures[nerasures++] = ML-29;

/* Now decode -- encoded codeword size must be passed */
//decode_data(codeword, ML);

/* check if syndrome is all zeros */
//if (check_syndrome () != 0) {
//correct_errors_erasures (codeword,
// ML,
// nerasures,
// erasures);

```



```

    sprintf(sbuf, "Corrected codeword: \"%s\\n\"", codeword);
    Serial.print(sbuf);
    //}
    //////////////////////////////////////
    ////
    while(TransmissionSetup < 1){

    if(menu <1){
    tft.setCursor(0,0);
    tft.print("Frequency Test With");
    tft.println("    Reed Soloman Error    correction");
    tft.println("");
    tft.println("Press the Red Button To Cycle Through 1 to 5");
    tft.println("");
    tft.print("Press the Black Button To Enter");
    menu=menu+1;}

    if(digitalRead(11)==HIGH && Enter <= 1){
    tft.setCursor(10,40);
    delay(200);
    Enter= Enter +1;
    }

    while(Enter ==1){
    if(Enter ==1 && counter3 < 1){tft.fillScreen(BLACK);
    tft.setCursor(50,80);
    tft.print("Number of Parity ");
    counter3 = counter3 + 1;}

    if(digitalRead(12)==HIGH){
    tft.fillScreen(BLACK);
    tft.setCursor(50,80);
    delay(800);
    Parity=Parity+1;
    tft.print("Number of Parity ");
    tft.print(Parity);
    delay(3000);
    if(Parity==5){Parity = 0;}
    }

    if(digitalRead(11)==HIGH && Parity >=1 && Enter < 2){
    tft.fillScreen(BLACK);
    tft.setCursor(10,40);
    Enter= 2;
    delay(300);
    }}

```

```

while (Enter == 2){
if(Enter == 2 && counter3 < 3){ tft.setCursor(0,80);
tft.print("Number of Artificial Error      ");
counter3 = counter3 + 1;
}

if(digitalRead(12)==HIGH && Enter ==2){
tft.fillScreen(BLACK);
tft.setCursor(0,80);
delay(800);
Error=Error+1;
tft.println("Number of Artificial Error      ");
tft.setCursor(170,120);
tft.println(Error);
delay(3000);
if(Error==5){Error = 0;}
}
  if(digitalRead(11)==HIGH && Error >=1 && Enter <3 ){
tft.fillScreen(BLACK);
tft.setCursor(10,40);
Enter= Enter +1;
delay(300);
Enter=3;
}
}
if(Enter == 3 && counter3 <= 4){
  tft.fillScreen(BLACK);
  tft.setCursor(10,40);
  tft.print("Transmitting");
  counter3 = 4;
}
if(counter3 == 4){TransmissionSetup = 1;}
}
//-----

if(TransmissionSetup == 1){

pinMode(Sending,OUTPUT);
digitalWrite(Sending,LOW);
delay(10000);

String message = "i love macca's "; //-----Your message to send
byte bytes [message.length()]; //-----Builds array to store message

```

```
message.getBytes(bytes, message.length()); //-----Stores message in byte format into array message
```

```
wait = millis()+delay1/5;
digitalWrite(Sending,HIGH); //-----HandShake HIGH, LOW, HIGH, LOW,
while(millis() <= wait){}
digitalWrite(Sending,LOW);
wait = millis()+delay1*0.8;
while(millis() <= wait){}
```

```
wait = millis()+delay1;
digitalWrite(Sending,LOW);
while(millis() <= wait){}
wait = millis()+delay1/5;
```

```
digitalWrite(Sending,HIGH);
while(millis() <= wait){}
digitalWrite(Sending,LOW);
wait = millis()+delay1*0.8;
while(millis() <= wait){}
```

```
wait = millis()+delay1;
digitalWrite(Sending,LOW);
while(millis() <= wait){}
wait = millis()+delay1;
```

```
codeword1[0]=int(sizeof(msg)+NPAR);
```

```
for (int i = 0; i < sizeof (codeword1); i++){ //-----Loops around x amount of times equal to the number of bytes in message
```

```
int bitvalue = (codeword1[i]); //-----Stores byte temporarily in bitvalue so bits can be determined by later loop
```

```
for(int j=0; j<8; j++){ //-----Loop to 8 so the following steps address each bit in each byte
```

```
if (bitvalue & byte(1)){
Serial.print("1");
wait = millis()+delay1/5;
digitalWrite(Sending,HIGH);
while(millis() <= wait){}
digitalWrite(Sending,LOW);
wait = millis()+delay1*0.8;
while(millis() <= wait){};
}
```

```

else{
  wait = millis()+delay1;
  Serial.print("0");
  digitalWrite(Sending,LOW);
  while(millis()<=wait){}
  digitalWrite(Sending,LOW);
  }

bitvalue = bitvalue >> 1;//-----shifts bits left one
wait = millis()+delay1;

}

Serial.print("");      //-----Used to test coding function
}}

```

Brandon Butler wrote the following code, it is the code used in the receiving apparatus.

```

int Receiving = 12;
int delay1 = 82;
int delay2 = delay1/10000;
unsigned int wait;
int x = 0;
int pos = 0;
byte message[40];
byte a = 0;
byte b = 1;
int handshake_accepted = 1;
int c = 40;
int counter1=0;
int set=0;
int counter3 = 0;
//-----
int erasures[16];
int nerasures = 0;

#include <stdio.h>
#include <stdlib.h>
#include "ecc.h"

unsigned char msg[] = "Nervously I loaded the twin ducks aboard the revolving platform.";
unsigned char codeword[256];
char sbuf[128];

```

```

/* Some debugging routines to introduce errors or erasures
into a codeword.
*/

/* Introduce a byte error at LOC */
void byte_err (int err, int loc, unsigned char *dst)
{
  sprintf(sbuf, "Adding Error at loc %d, data %#x\n", loc, dst[loc-1]);
  Serial.print(sbuf);
  dst[loc-1] ^= err;
}

/* Pass in location of error (first byte position is
labeled starting at 1, not 0), and the codeword.
*/
void byte_erasure (int loc, unsigned char dst[], int cwsiz, int erasures[])
{
  sprintf(sbuf, "Erasure at loc %d, data %#x\n", loc, dst[loc-1]);
  Serial.print(sbuf);
  dst[loc-1] = 0;
}

//-----

void setup() {

  pinMode(Recieving,INPUT);
  pinMode(13,OUTPUT);
  pinMode(11,INPUT);
  Serial.begin(9600);// open the serial port at 9600 bps:

  for(int i = 0; i < 8; i++){
    bitClear(a,i);
  }

  //////////////////////////////////////
  Serial.begin(9600);      //
  tft.reset();            //
  uint16_t id = tft.readID(); //
  tft.begin(id);          //
  tft.setRotation(3);     // Screen (LCD) settings
  tft.fillScreen(BLACK);  //
  tft.setTextColor(GREEN, BLACK); //
  tft.setCursor(40,0);    //
  tft.setTextSize(2);     //

```

```
////////////////////////////////////
```

```
tft.setCursor(0,0);  
tft.print("Frequency Test With");  
tft.println("   Reed Soloman Error   correction");  
tft.println("");  
tft.println("");  
tft.println("");
```

```
}
```

```
void loop() {  
  // put your main code here, to run repeatedly:
```

```
if(counter3 < 1 && digitalRead(11) == HIGH){  
  tft.fillScreen(BLACK);  
  tft.setCursor(0,0);  
  tft.print("waiting for response");  
  delay(5000);  
  tft.fillScreen(BLACK);  
  tft.setCursor(0,0);  
  counter3= counter3 +2; }
```

```
if(x<1){  
if(digitalRead(Recieving)==HIGH){  
  wait = millis()+delay1+delay2;  
  digitalWrite(13,HIGH);  
  delay(2);  
  digitalWrite(13,LOW);  
  while(millis(<=wait){}  
    if(digitalRead(Recieving)==LOW){  
      wait = millis()+delay1;  
      digitalWrite(13,HIGH);  
      delay(2);  
      digitalWrite(13,LOW);  
      while(millis(<=wait){}  
        if(digitalRead(Recieving)==HIGH){  
          wait = millis()+delay1;  
          digitalWrite(13,HIGH);  
          delay(2);  
          digitalWrite(13,LOW);  
          while(millis(<=wait){}  
            if(digitalRead(Recieving)==LOW){  
              wait = millis()+delay1-20;  
              while(millis(<=wait){}
```

```
x=x+1;  
handshake_accepted=handshake_accepted+1;
```

```

//.....
while(handshake_accepted > 0 && pos <= c ){
  b=a;
  for(int i = 0; i<=7; i++){
    wait = millis()+delay1;
    //digitalWrite(13,HIGH);
    //delay(2);
    //digitalWrite(13,LOW);
    while(millis())<wait){
      if(digitalRead(Receiving)==HIGH &&set==0){
        set = 5;
        Serial.print("1");
      }
      if(digitalRead(Receiving)==LOW && set ==5){
        set = 1;
        //Serial.print("0");
        digitalWrite(13,HIGH);
        delay(2);
        digitalWrite(13,LOW);
        delay(1);
        wait=wait-3;
      }

    }//-----
    //digitalWrite(13,HIGH);
    //delay(2);
    //digitalWrite(13,LOW);
    //delay(1);

    if(set==1){bitSet(b,i);}
    else{bitClear(b,i);}
    set = 0;

  }
  Serial.println(" ");
  Serial.println(char(b));
  Serial.print(" ");
  message[pos] = char(b);
  pos = pos + 1;
  c = message[0];
}

byte b= 0;

while(handshake_accepted > 0 && pos <= c ){

```

```

b = a; // stores a zero in all 8 bits of b during the initialize stage;
for(int i = 0; i <= 7; i++){

    if(digitalRead(Receiving)==HIGH){
        bitSet(b,i);
        Serial.print("1");
    }
    else{ Serial.print("0");
        bitClear(b,i);
    }
    wait = millis()+delay1;
    while(millis()<=wait){}
}
Serial.print(" ");
Serial.println(char(b));
Serial.print(" ");
message[pos] = char(b);
pos = pos + 1;
int(c)=int(message[0]);

} // while loop close Bracket

Serial.print(c);
Serial.println("");

byte message1[c];

for(int i=0; i < c; i++){
    message1[i]=message[i+1];
    Serial.print(char(message[i]));}
Serial.println("");
Serial.println("");

//-----
/* Initialization the ECC library */

initialize_ecc ();
#define ML (c)

tft.setCursor(0,0);

sprintf(sbuf, "with some errors: \"%s\"\n", message1);
tft.print(sbuf);

decode_data(message1, ML);

/* check if syndrome is all zeros */

```



```
if (check_syndrome () != 0) {
    correct_errors_erasures (message1, ML, nerasures, erasures);

    sprintf(sbuf, "Corrected Message: \"%s\"\n", message1);

    tft.print(sbuf);
}
//-----
}}}}// handshake brackets
```

# 14.0 Appendix (E) – Apparatus Diagrams and Wiring Schematic

The following diagrams show the wiring and physical apparatus that was built. During the experimental testing.

