

University of Southern Queensland
Faculty of Health, Engineering & Sciences

Wireless Grain Silo Monitoring and Control

A dissertation submitted by

Christopher White

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Electrical & Electronic)

Submitted: October, 2013

Abstract

'Wireless Grain Silo Monitoring and Control' was a project put forward by Agridry International to USQ. The scope detailed that a wireless system had to be designed to communicate from the office to the grain silos 100m away, with cost being a factor.

The purpose of making a wireless system for the grain silo environment was to add convenience by reducing the time spent travelling to and from the silos and to have a lower cost than the current system

Several combinations of wireless communication were researched to select the optimum design for the system and 434MHz ASK RF modules with an Arduino microprocessor were selected for the task.

A number of prototypes have been built throughout the project, with new features added in every version. The current version is able to send packets using half-duplex communication across a distance of 100m, using dipole antennas.

The polling system has been implemented communicate between the Master module and each of the silos, sending setting information for the silo fans. Also, errors are reported via the Serial Monitor when the Master can't achieve communication.

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean

Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Chris White,

0061004803

Signature

Date

Acknowledgements

Firstly I would like to thank my supervisor Dr John Leis and Agridry manager Mr Murray Fenner for their guidance during the project. Their advice has been insightful and expanded my engineering knowledge.

I would like to thank my friends, David Dobson, Newman Sana, Robert Skillington and John Sutcliff for volunteering when testing the first prototype. Also, I would like to thank them for being well-mannered while working in the laboratory.

I would like to thank the members of Darling Downs Toastmasters for their help in improving my public speaking skills. Presenting my project to a large audience was made easier with their help.

Furthermore, I would like to thank my family and friends for their support during the project.

Table of Contents

| | |
|---------------------------------------|------|
| Abstract..... | i |
| Certification of Dissertation | iii |
| Acknowledgements..... | iv |
| Glossary..... | vii |
| List of Figures | viii |
| 1. Introduction | 1 |
| 1.1 Aim | 1 |
| 2 Literature Review..... | 2 |
| 2.1 Grain Silos | 2 |
| 2.2 Controller | 2 |
| 2.3 Normal Setup | 3 |
| 2.4 Scope..... | 3 |
| 2.5 Communication Options..... | 4 |
| 2.6 Microcontroller Options | 5 |
| 2.7 Wave propagation..... | 5 |
| 3 Methodology..... | 6 |
| 3.1 Selection of equipment..... | 6 |
| 3.2 Prototype Design | 6 |
| 3.3 Programming | 6 |
| 3.4 Circuit design..... | 6 |
| 3.5 Testing..... | 7 |
| 3.6 Consequential Effects | 7 |
| 4 Design & Implementation | 9 |
| 4.1 Software..... | 9 |
| 4.1.1 Packet Structure..... | 9 |
| 4.1.2 Timing..... | 10 |
| 4.1.3 Half-duplex Communication | 10 |
| 4.1.4 Polling System | 11 |
| 4.1.5 Address Assignment..... | 11 |
| 4.1.6 Routing..... | 12 |
| 4.1.7 EEPROM | 12 |
| 4.1.8 Security | 13 |

| | | |
|-------|--|----|
| 4.2 | Hardware | 13 |
| 4.2.1 | Voltage Regulators | 13 |
| 4.2.2 | Transistor Circuit | 13 |
| 4.2.3 | Antenna Requirements and Design | 14 |
| 4.2.4 | Regulations..... | 15 |
| 5 | Evaluation | 16 |
| 5.1 | Simplex Communication | 16 |
| 5.2 | Half-duplex Communication | 16 |
| 5.3 | Dipole construction..... | 17 |
| 5.4 | Code structure | 17 |
| 5.5 | Polling..... | 18 |
| 6 | Conclusion..... | 19 |
| 6.1 | Further Work and Recommendations | 19 |
| 6.1.1 | Antennas | 19 |
| 6.1.2 | Circuit Construction | 20 |
| 6.1.3 | Address Assignment..... | 20 |
| 6.1.4 | Security | 20 |
| | References | 21 |
| | Appendix A – Project Specification | 22 |
| | Appendix B - Circuit schematics..... | 23 |
| | Appendix C – Office code..... | 24 |
| | Appendix D – Master Code | 27 |
| | Appendix E – Sensor code..... | 32 |
| | Appendix F – Risk Assessment | 37 |

Glossary

ACMA - Australian Communications and Media Authority

AGC – automatic gain control

CRC – cyclic redundancy check

Dry-bulb temperature – air temperature with 0% humidity

EEPROM - Electrically Erasable Programmable Read Only Memory

FFD – Full Function Device, a device that can designate other nodes with addresses

LED- Light Emitting Diode

MHz- Megahertz

PCB- Printed Circuit Board

PLC- Power Line Communication

RF- Radio Frequency

TTL – Transistor-transistor logic

USQ- University of Southern Queensland

Wet-bulb temperature – combined measurement of temperature and humidity

λ - Wavelength

List of Figures

| | |
|---|----|
| Figure 1: Explanation of controller temperature regulation (Fusae, 2001)..... | 3 |
| Figure 2: Normal setup of silos and controller | 3 |
| Figure 3: The proposed setup for project | 4 |
| Figure 4: Illustration of Huygen’s Principle..... | 5 |
| Figure 5: Serial Monitor showing packets with buffer overrun effects | 10 |
| Figure 6: The oscilloscope readings detailing the Half-duplex communication | 11 |
| Figure 7: BAAM application for addressing silos. | 12 |
| Figure 8: Configurations of the Monopole (left) and Dipole (right). (Jacobsen 2008) | 14 |
| Figure 9: Radiation pattern of a dipole antenna (Beasley, Miller 2008)..... | 15 |
| Figure 10: Configuration of dipole antennas | 17 |
| Figure 11: Results from polling test (Sensors module)..... | 18 |

1. Introduction

Wireless technology has changed the world that we live in as it is convenient and can be tailored to many applications. It is widely used in telecommunications such as television, radio, internet, remotes, etc. However, wireless has not been applied to controlling and monitoring grain silos.

1.1 Aim

Agridry International had proposed the topic 'Wireless Grain Silo Monitoring and Control' to the Engineering Faculty at University of Southern Queensland (USQ). Agridry specialises in the manufacture of grain silos, dryers and control equipment. The project required a wireless system to communicate control signals from the office to turn on the silo fans and acquire temperature and humidity information from the silo sensors. The system requires the communication to be reliable and have a range of 100m. Making the system wireless will provide convenience and reduce the time spent travelling to the silos. Also, the total cost of the system is a major factor and has to be minimised.

2 Literature Review

This chapter will look into the current system that control grain silos, the proposed solution, options that are available to solve the problem and relevant wireless communication theory.

2.1 Grain Silos

The primary purpose of the grain silo is to store grain and to keep it at a suitable temperature and humidity level. If this level is too high, there are some problems that can occur, such as mould and pests, which would result in poor quality grain (Fusae, 2001). To resolve this, the grain is aerated by fans at the bottom of the silo, which dehumidifies it.

There are four settings that fans can run at:

- Manual
- Rapid
- Economy
- Off

Manual setting is for when grain is just been added into the silo and runs for 24 hours to remove harvest heat. After the Manual setting has run, it automatically switches to Rapid.

Rapid is for grain that has been recently added and will turn the fan on full for about 12 hours a day. This will quickly bring the grain to the lowest dry-bulb temperature which removes negative effects.

Economy is for grain that has been in storage for some time and slowly brings it to its wet-bulb temperature. This temperature prevents negative effects, is suited for long-term storage and saves power.

Off is when the silo is empty and no longer in use. These settings are configured by switches on each of the silos or by a large switchboard.

2.2 Controller

Agridry has designed a controller that will manage the settings and temperature information. The controller will read the varying outside temperature levels of the day with a sensor, as it fluctuates during the day, as represented in Figure 1. The controller will have a set temperature at which the grain has to be kept at, normally around 16 °C and regulate it by switching the silo fans on and off. The controller switches on all of the fans, with a particular setting (Rapid or Economy), when the outside air temperature goes below a calculated point. The fans will run during the night and switch off in the morning, as it is more efficient. The Rapid setting is first to switch on and has a linear cooling rate, while Economy switches on a short duration later and has a slower response.

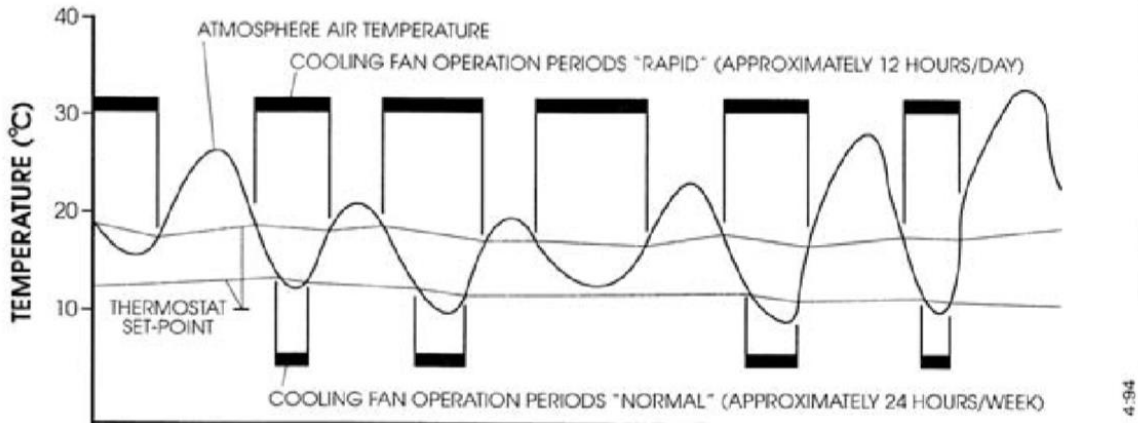


Figure 1: Explanation of controller temperature regulation (Fusae, 2001).

2.3 Normal Setup

The current system is shown in Figure 2, the switches and the controller are placed near the silos. However, they are far from the office, up to 100m away, and these settings need to be regularly changed and the temperatures monitored. Also, there are a large number of wires going to the silos and if the complex is large, multiple controllers have to be purchased which are expensive. The idea of making this system wireless is to get rid of travelling to and from the grain silos and the reduction of cost.

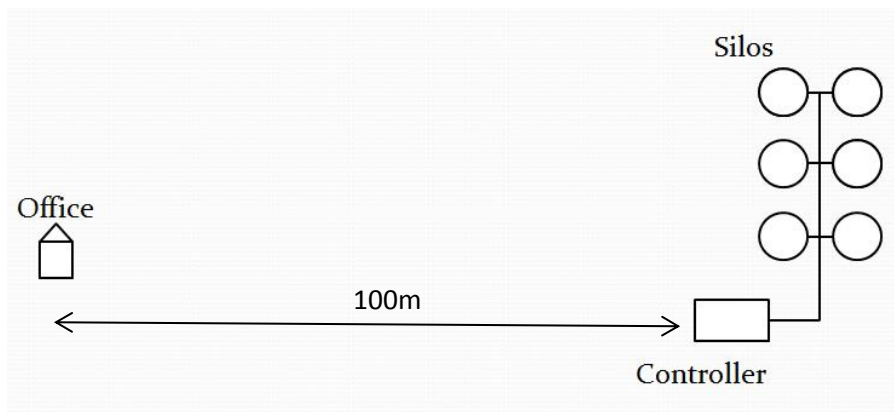


Figure 2: Normal setup of silos and controller

2.4 Scope

The current plan is to design three modules; office, master and sensor modules (Figure 3). The Office module will be placed at the office and receive information from the controller, to switch on Rapid and/or Economy silos. The information will be transmitted using a communication method, up to 100m to the Master module. From there, the signal will be transmitted by a polling or pseudo-random system to each of the Sensor modules, switching the appropriate fans on. The system will have to resend the settings to the silos every 15-30 mins to ensure that the setting is correct.

Also, the Sensor modules will have the silos temperature and humidity information which needs to be sent back to the office. This information needs to be sent at regular intervals as well, about one minute apart.

This concept is new, as there haven't been any development of wireless monitoring and control in the grain silo application. Although, by using wireless communication theory and designing, implementing and testing a prototype will state if a solution is feasible.

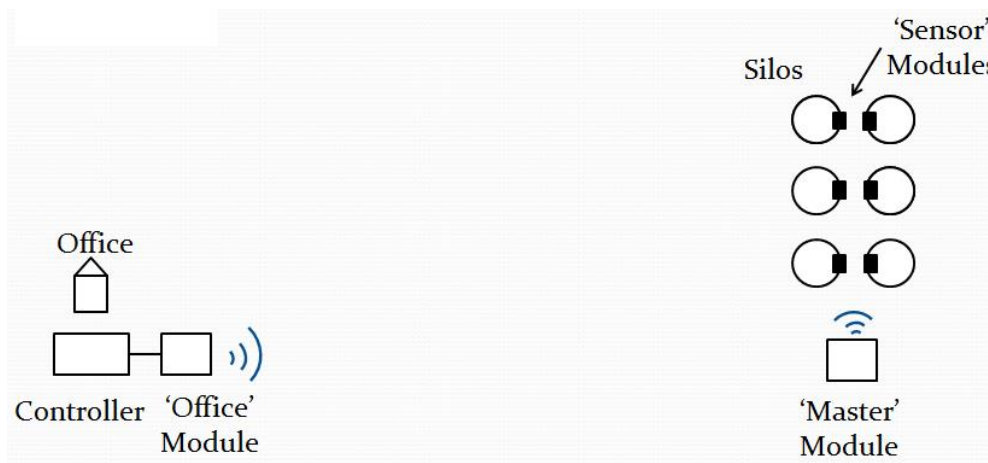


Figure 3: The proposed setup for project

2.5 Communication Options

There are several methods of communicating information between the office and the silos. The methods that are available are Power Line Communication, Radio Frequency and Zigbee.

Power Line Communication (PLC) is achieved by sending data through a power line, using the three phase voltage as a carrier. The modulator superimposes the data onto the three phase voltage and is demodulated at the receiving end to retrieve the information. The advantages of using this system are that no additional wires are required as it uses the existing power lines connected to the silos and it can transmit over long distances. However, the disadvantages are that the system doesn't have much versatility compared to others, most systems are expensive and there are some safety issues dealing with 240V.

The next method is Radio Frequency (RF). An online electronics distributor, Little Bird Electronics sell small 434 MHz ASK (Amplitude Shift Keying) RF modules, which uses an unlicensed frequency. These are very cheap, around \$5, and only require 5V power supply and a serial input from a microcontroller. The disadvantages are that ASK is susceptible to noise and can cause errors. Also, when other devices near the frequency band are present, slight errors can occur.

Zigbee is another method of sending information and it is known to have higher reliability and transmission distance. However, the disadvantages are that some of the functions on the Zigbee

aren't needed, the cost is moderate and operates at 2.4GHz which will reflect more off conductive surfaces.

2.6 Microcontroller Options

With using the above communications methods, a microcontroller has to be used to read, write and manipulate information. The microcontrollers that have been researched are the PIC and Arduino.

The PIC is a microcontroller manufactured by Microchip, which are fairly cheap (\$5) and come in various shapes, sizes and functionalities. The disadvantages are that it requires a separate programmer (\$60) and external components.

The next microcontroller is the Arduino Uno. The Arduino uses an atMEGA328 from Atmel and has the required external components soldered onto a printed circuit board. The programming of the board is done with its own firmware and is relatively easy to use. Some of the disadvantages are that the cost is moderate (\$30/board) and the project only uses 5 pins out of its total 18.

2.7 Wave propagation

When sending RF signals, the main concern is that they will most likely diffract and reflect off the metal of the silos making it hard to detect the information being sent.

In electromagnetic wave propagation, diffraction follows Huygen's Principle; a point on the primary wavefront can act as a source for primary waves (Beasley, Miller 2008). As shown in Figure 4, when the primary wavefront meets an obstacle it radiates from a point. On the opposite side of the obstacle, there are shadow zones where the signal can't be received. At lower frequencies, it is known that the shadow zone is smaller and is less affected by obstacles (Beasley, Miller 2008).

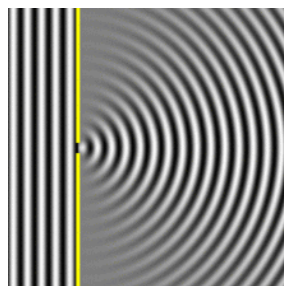


Figure 4: Illustration of Huygen's Principle

For the design, the reflection has to be minimised so that the signal isn't distorted by any reflected waves. At lower frequencies, the attenuation of the reflected signal when using steel or other metals is large. Also at lower frequencies, the attenuation in free space is small and can travel further. (Parsons, Hancock, 2012)

3 Methodology

This chapter details the method of selecting the equipment, designing the prototypes, programming and testing. Also the ethics and consequential effects involved in this project are discussed.

3.1 Selection of equipment

The 434MHz modules were selected for the task, due to the effects of reflection and diffraction being low. Arduino was the selected microcontroller because the simplicity of uploading the program onto the board.

3.2 Prototype Design

Once the equipment had been selected, the prototype had to be designed and implemented. The prototyping had several stages:

- Prototype 1 – Simplex communication
- Prototype 2 – Half-duplex communication
- Prototype 3 – Dipole construction
- Prototype 4 – Polling system

The first prototype was designed with minimal components, one RF transmitter on one Arduino and the RF receiver on the other Arduino. Tests were conducted to see what solutions worked and identify any problems with the design. Once the system worked, the prototype was updated to the next stage with a new feature.

The second prototype had implemented half-duplex communication to provide a reliable channel. A RF pair was added to the Arduinos, so each board had one transmitter and one receiver.

The third prototype had added a dipole to the RF modules to provide better range.

The fourth prototype had implemented a polling system to go through each of the silo addresses and retrieve information.

3.3 Programming

The programming of the Arduino is done through a USB cable that is attached to the board and computer. The Arduino board has its own firmware that was downloaded to the computer, which uses its own language that is based off C and Processing.

3.4 Circuit design

The project had required a circuit to work with the Arduinos. Arduino ProtoShields are connected on top to give a surface to solder components onto. The 434 MHz modules are connected to the

ProtoShield via a female headers, this is to enable the ability to connect and remove the modules if they require modification.

The 434 MHz RF transmitters have an operating voltage of 3-12V, a higher voltage produces a greater transmitting distance. To produce the highest transmitting power, the 9V battery that powers the circuit is converted by a 12V step-up regulator. The Arduino and the rest of the circuit need to operate at 5V; therefore a step-down regulator is implemented.

The antennas are made of thin solid core wire and are connected to the RF modules. The remaining resources of the project are basic electronic components such as wire, LEDs, push buttons and battery clips. The circuit is arranged in the manner of Appendix 1.

3.5 Testing

There are many methods of testing if the system is working properly or locating a problem. The Arduino has the functionality of displaying information on the Serial Monitor located on the computer. The USB cable that is used for programming the microcontroller is connected to the serial pins D0 and D1 (RX and TX respectively). Using the command `Serial.print()` in the code will display values on the screen that can be used for debugging.

The multimeter was used for measuring voltage between two points and resistor values. The voltage was measured across the battery and voltage regulator output was taken to ensure the Arduino had received the appropriate 5V power supply. Also, the multimeter was used to find if any components weren't soldered on correctly.

Another item of equipment that was used during testing is the Oscilloscope. The Oscilloscope was mainly used for reading the signals on the serial ports connected to the 434MHz transmitter and receiver (D3 and D12 of Appendix 1).

The testing of the prototypes was initially done in the electronics laboratory for the short distance tests. In the case of prototypes 1, 2 and 3, the long distance tests were conducted in an outside area located near a building.

3.6 Consequential Effects

There are a number of ethical issues to take into consideration when designing the wireless system. This is required to reduce risks occurring and improve satisfaction of users operating the system.

First of all, the silo fans have to be switched on one at a time as each fan draws a lot of current when starting. If a large number of fans are started together, there will be high current draw on the lines and a voltage drop will occur with the possibility of a brownout.

The modules have to be designed to resend the information. There is a small chance that the silos will switch on by random noise and resending the signal will ensure that each silo has the correct setting. Otherwise, having the silos switched on at the wrong time or with the wrong setting will ruin the quality of the grain. The scope detailed a resending time of 15 – 30 mins would be sufficient.

The RF receiver has an AGC (automatic gain control) which converts the incoming signal to the appropriate TTL voltage level, so that it can communicate with the Arduino serial port. While there is no 434MHz signal, the receiver amplifies the noise and produces a random sequence of levels on the receiving line. If the data is transmitted without a proper structure, it could be hard to detect the information from the noise.

With any wireless system, security has to be implemented into this system so that it prevents any hacker infiltrating the network. The worst case scenario would be if the hacker was able to change the settings of the fans and ruin the grain.

4 Design & Implementation

This chapter details the how the software and hardware were designed and implemented into the system.

4.1 Software

4.1.1 Packet Structure

The transmission of data has to be done in a packet system, which is widely used in wireless systems. Each packet will include several sync bytes, a header, address, data and checksum bytes, respectively in order (Beasley, Miller 2008).

While sending information, the transmitter and receiver timing may not be synchronised with each other and produce wrong information. Sending Sync bytes of 01010101 at the start of the packet synchronises the modules. The header byte is placed after the sync bytes and indicates the start of the packet. The address byte specifies where the data needs to be sent; examples are the master module or silo number. The Data byte is the important information that needs to be sent, such as the setting or silo temperature.

After each packet, the checksum is calculated and placed to enable error detection. The checksum is calculated by doing an exclusive-or operation on each of the previous bytes, in this case, the header, address and data bytes. The packet is sent to the receiver with the checksum on the end. The receiver looks at the received bytes, calculates the checksum again and compares the two checksum bytes. If they are the same the packet is accepted, otherwise it is discarded.

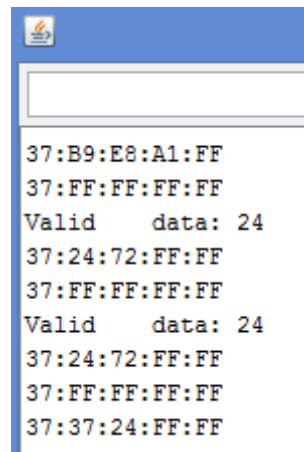
The last prototype completed in the project had the packet structure of one header byte, one byte to indicate the destination address, one byte to indicate the source address, one byte for data and one byte for the checksum. The number of bytes per section could be increased to provide greater detail. However, a longer packet will be more susceptible to errors and increase the number of retries before getting the correct information. The number of bits per section can't be decreased, as a byte provides a range of 256 values which is well suited to amount of addresses that can be distributed to the silos. Reducing the range would cause restrictions on the design. Also the temperature and humidity values from the sensors will be placed into the data section, which needs to have a high enough resolution to provide accurate readings. The temperatures will range from -10 to 50 °C and need a resolution of 0.5 °C.

The software serial library had to be used, as the normal Serial port was being used for displaying debugging information to the computer. The software serial is able to create additional serial ports that are simulated by the software on the microcontroller. The functions used are `SoftwareSerial(RX,TX)` which defines the pins that the new serial port is set, `SerialA.begin()` sets the baud rate and `SerialA.write("xyz")` writes information onto the port.

The RF modules operate with a baud rate between 300-2400 bits per second. The rate of 1200 bits was selected because operating at a slower speed will increase the number of symbols per bit and make it less susceptible to errors.

4.1.2 Timing

After each byte, the Arduino requires the command `Serial.flush()` to ensure that all of the information in the transmit buffer is sent. Without this command, the Arduino has the tendency to overwrite the byte in the buffer before it is sent to the transmitter. In the example of Figure 5, it is shown that some complete packets (37:24:72) are received and others only show the first few bytes, as a result of the bytes being overwritten.



```
37:B9:E8:A1:FF
37:FF:FF:FF:FF
Valid data: 24
37:24:72:FF:FF
37:FF:FF:FF:FF
Valid data: 24
37:24:72:FF:FF
37:FF:FF:FF:FF
37:37:24:FF:FF
-----
```

Figure 5: Serial Monitor showing packets with buffer overrun effects

4.1.3 Half-duplex Communication

To be able to achieve reliable communication between both of the Arduinos, half-duplex method had to be implemented. Half-duplex is two-way communication where only one device can send information to the other, at any one time. In the system that has been built, the first module sends information to the second, as shown on channel 2 (middle) in Figure 6. After a short delay, the second module transmits an acknowledgement packet back to the original module to confirm the information, shown on channel 3 (bottom) in Figure 6. Channel 1 (top) is the reading taken from one of the receivers and shows the summation of the two. The modules will repeat sending information both ways until the acknowledgement is received by the first module. This method takes an amount of time, but ensures accuracy of the information being sent.

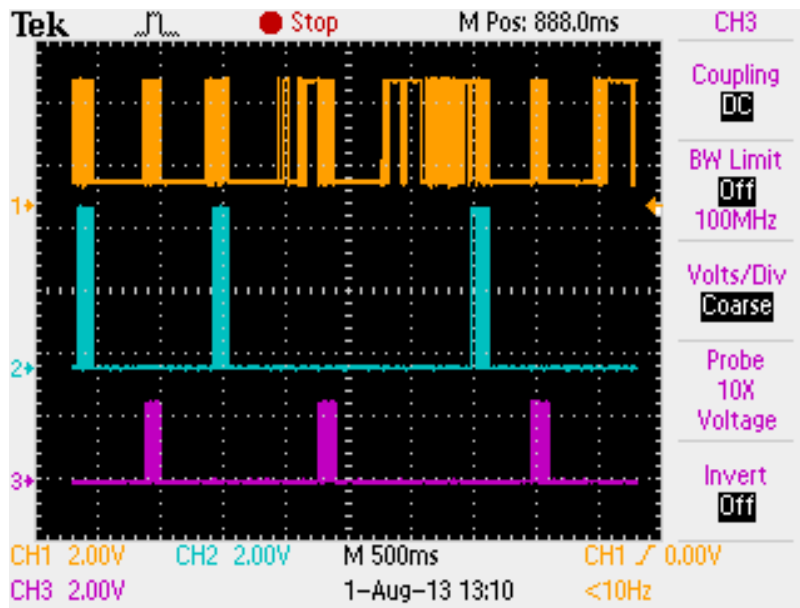


Figure 6: The oscilloscope readings detailing the Half-duplex communication

4.1.4 Polling System

In the case of the Master module communicating to the Sensor modules, there has to be a polling system implemented. The master module will go through a list of all of the silo addresses, poll each sensor and receive back a packet, still using half-duplex communication.

The other method is to use pseudo-random communication. The Master module will broadcast a message to all of the silos, asking to report back information of its current state. Each Sensor module will have a random time slot allocated, to transmit back to the master. If a collision between the data occurs, new random time slots are generated and the system tries again.

The polling system has advantages over pseudo-random communication in this scenario. Polling the information will ensure that each silo has the correct information, but at the cost of time. The timing of this system is not important, as there is a 15 min gap between transmitting the setting from the controller. Also, the amount of modules transmitting is reduced with the polling system, as only one signal is sent at a time. With pseudo-random, many modules would be transmitting over a short period of time which could prove difficult in an environment with reflective surfaces.

4.1.5 Address Assignment

In the situation of the system is being first setup, the all of the modules won't have addresses and they need to be assigned. El Rachkidy, N et al. (2010) have detailed methods of assigning addresses for wireless systems in mines, DAAM, SAAM and BAAM. These methods, especially BAAM (Binary Address Assignment Mechanism), would be useful in the silo environment because some of the silos won't be reachable if the communication was only between the Master and Sensors. In figure 7, the Master might have difficulty communicating normally with silo 7. With BAAM, the Sensors are given

the ability of a FFD (full function device) (El Rachkidy, N et al. 2010) to be able to communicate with other nodes in the network and assign addresses to its two children.

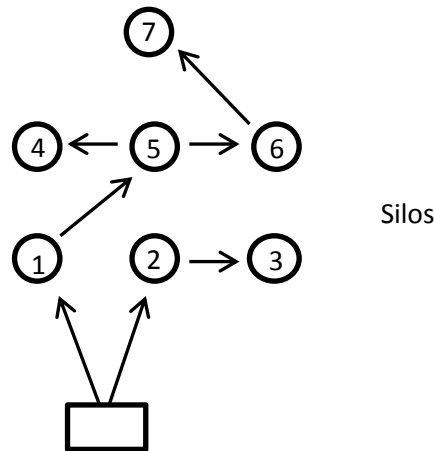


Figure 7: BAAM application for addressing silos.

The method of assigning address will be based off El Rachkidy, N et al. (2010) and Schungers, C et al. (2002). The Master module will start up first and has two vacant spaces for children. Each of the Sensors will start up randomly, send 'Hello' messages to other devices and listens for any 'Info' packets (Schungers, C et al. 2002). The 'Info' messages contain the information of the neighbouring devices which details if it is also requesting an address, if it is a FFD and whether it has any vacancies. If there is any conflict between the devices, the device waits and stores information of its neighbours. When there is no conflict, the device selects a FFD and requests an address. The FFD will give the device half of available addresses and designate the first address to that device.

4.1.6 Routing

If the above addressing scheme is used, the messages have to be routed when the Master is polling the Sensors. Each FFD will contain the addresses of its children and the devices below them. When a device is selected to be polled (example of silo 7 in Figure 7), the route is calculated and sends the packet to the first address (silo 1). The devices repeatedly pass down the packet until it reaches the desired address (1 → 5 → 6 → 7). The same path is taken when sending back the acknowledgement.

4.1.7 EEPROM

EEPROM is non-volatile memory space on the Arduino that can store 8-bit values. The memory won't be deleted after the power is turned off and can be retrieved later. If power failures occur, the EEPROM will be useful as it can protect against loss of important data, such as the address and routing information. Although, the EEPROM has a slow access time and will have to be used rarely for the program to run smoothly. Also, Arduino has the limit of storing 1 Kbyte of EEPROM on to the microcontroller.

4.1.8 Security

As the wireless signal can be received by any 433MHz receiver, some form of encryption has to be implemented into the system to prevent any security threats. Encryption is achieved by generating a value (key) and performing XOR with the packet being sent. The receiver will have the same key and can decrypt the packet to get the original information. Anyone without the key won't be able to read the information and hack the device.

4.2 Hardware

4.2.1 Voltage Regulators

To be able to operate effectively, the circuit needs a 12V power supply for the transmitter and a 5V supply for the Arduino and the receiver. The device is connected to a 9V battery which is converted to 12V with a step up regulator. This 12V is then converted down with a 5V regulator to power the rest of the circuit.

The 9V battery wouldn't be suitable for the final prototype, as the battery would deplete over a few days and need replacing. However, silo fans are connected to 24V and this line can be used to power the modules, via a 12V and 5V step down regulator. Also the office module could be designed to use a power supply from a computer USB port or 240V power port. Using the 9V battery in the current setup provides the ease of use during testing.

4.2.2 Transistor Circuit

After sending the packet, the transmitter module had the tendency to return to logic level 1 and continue to send the 433 MHz signal. This is not ideal, as it wastes power and can interfere with other devices. The solution was to place a transistor on the power line so that it can be turned on and off when needed. The transistor acts as a switch, as enables current to flow from the collector to the emitter once there is a current in the transistor base (Appendix 1, D4). The transistor collector current needs to be fully saturated to work effectively and requires two resistors to set the current. The calculations of resistor values connecting to the transistor are shown below.

$$\begin{aligned} I_{c_{sat}} &= \frac{V_{cc}}{R_c} & I_B &= \frac{I_{c_{sat}}}{\beta} & R_b &= \frac{V}{I} \\ &= \frac{12}{1000} & &= \frac{12mA}{150} & &= \frac{5V}{80\mu A} \\ &= 12mA & &= 80\mu A & &= 62500 \Omega \end{aligned}$$

4.2.3 Antenna Requirements and Design

The detail on the antenna length required was not specified by the seller. In antenna design, the monopole antenna is known to be $\lambda/4$, λ being wavelength. To find the wavelength of the 434 MHz signal Equation 1 is used:

$$\lambda_0 = \frac{c}{f} \quad (\text{Equation 1})$$

C is the speed of light (3×10^8) and f is the frequency (434 MHz), which comes to a distance of 17.28 cm for the $\lambda/4$. The actual wavelength required is smaller, as radio waves travel slightly slower than the speed of light (about 95%) and the antenna has to be trimmed accordingly.

The monopole antenna works with a ground plane to create a mirror image of the original antenna, and acts like a dipole antenna, being $\lambda/2$ long. The RF modules have a ground plane on the printed circuit board, however they are small compared to the size of the antenna and are inadequate for the task.

The dipole was constructed to provide a better quality antenna. The dipole antenna consists of two wires being $\lambda/4$ long, which are connected to the antenna and ground lines of the module and are pointed in opposite directions (Figure 8). The radiation of a dipole antenna in Figure 9, shows that the maximum power is perpendicular to the antenna wires and no power (null) when inline.

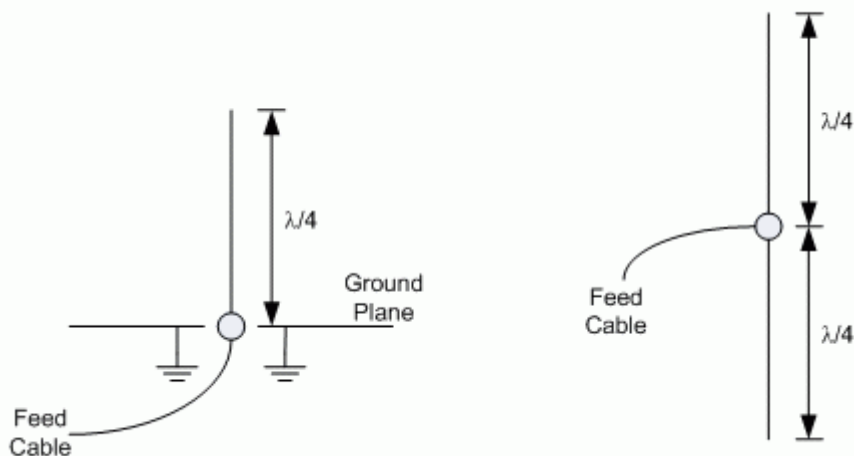


Figure 8: Configurations of the Monopole (left) and Dipole (right). (Jacobsen 2008)

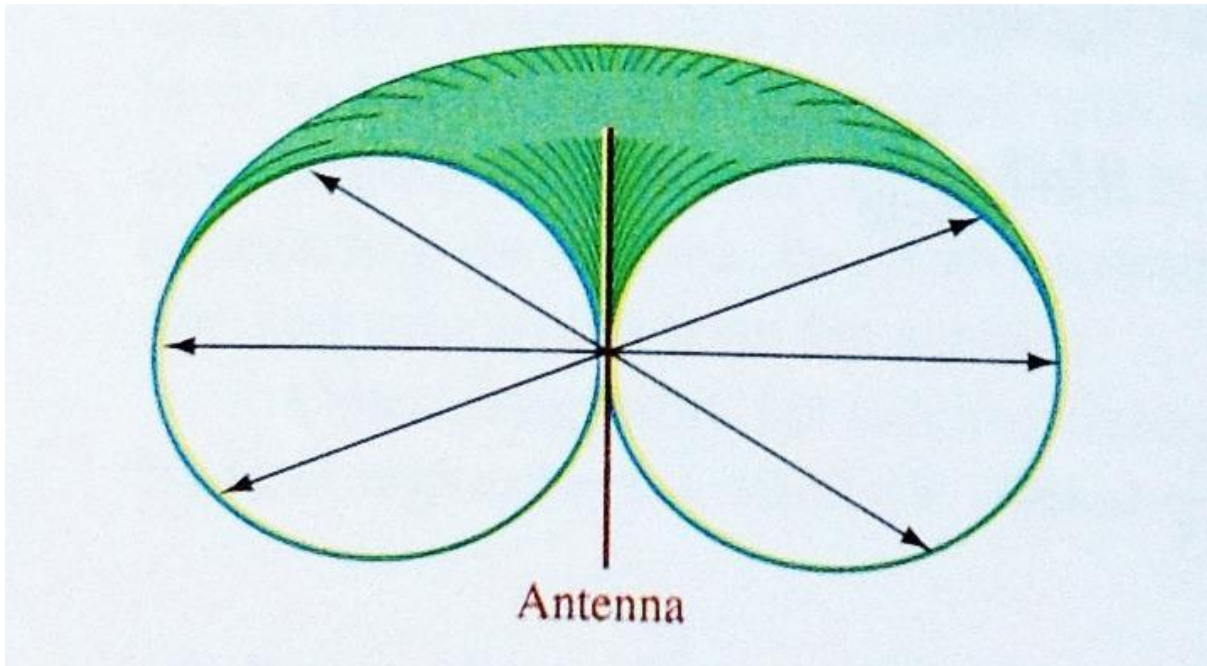


Figure 9: Radiation pattern of a dipole antenna (Beasley, Miller 2008)

4.2.4 Regulations

No regulations have been found detailing the time limit of transmitting the 434 MHz signal. However, AMCA (Australian Media and Communication Authority) states the requirement of maximum equivalent isotropically radiated power (EIRP) of 25 mW (ACMA 2013). The EIRP is shown in Equation 2, D_t being the directivity of the antenna and W_t the power of the antenna.

$$EIRP = D_t W_t \quad (\text{Equation 2})$$

Substituting known values, directivity of a dipole antenna is 1.64.

$$25mW \geq 1.64 * W_t$$

$$W_t \leq 15.24mW$$

As shown the input power of the antenna has to be less than 15.24mW to comply with regulations. The power of the input of the antenna has not been found out, as there is no datasheet for the RF modules and the current can't be calculated.

5 Evaluation

During the project, there have been many stages due to the fact that there are numerous variables to consider. The project started with getting the 434MHz modules to operate and then improve the design with altered prototypes.

5.1 Simplex Communication

The first prototype was constructed and a simple experiment of sending data with simplex or one-way communication. The modules at this stage only consisted of one RF transmitter and receiver pair connected to the Arduino. The two Arduinos had separate code uploaded, one for the transmitter circuit and one for the receiver.

The transmitter Arduino was implemented to send a packet every second. The data byte changed with every transmission and counted from one to fifty. The receiver was placed at incremental distances of 1m, 5m, 10m, 20m and 30m from the transmitter and printed the received packets to the Serial Monitor on the computer.

The results from the computer had produced an indication of reliability of the system. If the signal was affected by noise or synchronisation errors there would be a gap in the number sequence. After the counting loop had finished, the amount of packets received is shown. Table 1 shows the results of the experiment at different distances. As seen, the results are not ideal as the signal drops off at 30m.

| Distance (m) | Valid packets/50 | % |
|--------------|------------------|----|
| 1m | 22 | 44 |
| 5m | 18 | 36 |
| 10m | 14 | 28 |
| 20m | 6 | 12 |
| 30m | 4 | 8 |

Table 1 : Experiment results of the first prototype

5.2 Half-duplex Communication

The second prototype had improved by implementing half-duplex communication, the transistor circuit and voltage regulators. The half-duplex communication was setup with the Office module sending a packet to the Master module. If the Master receives this packet correctly, it would send an acknowledgement packet containing the same data back to the Office. The Office would compare the sent and received data and would turn on an LED if it was correct.

The experiment of obtaining the maximum distance was conducted again. Instead of depending on a computer to retrieve information, the Arduinos were placed separately with their own power supply

and illuminated LEDs if the system was working. In this experiment, the signal of this system had dropped out around 70m.

5.3 Dipole construction

The next phase of the prototype was to increase the range and to design the dipole antenna to replace the monopole. As explained before, the dipole were soldered onto the RF modules via the antenna and ground lines and pointed in opposite directions.

In the first configuration, the dipoles were arranged parallel to each other. The spacing between the dipoles was 2-3 cm and when testing, the maximum distance reached 50m. As a consequence of the incorrect spacing, one of the dipoles was acting as a reflector and distorting the signal. The appropriate spacing between the antennas should be $\lambda/2$ or 34.6cm to remove the reflective effect.

To confirm that the signal was affected by the distance between the dipoles and not any other effects, the dipoles were arranged right angles to each other (Figure 10). In wave propagation, the electric (E) and magnetic fields (H) travel right angles to each other. The polarization relates to the positioning of the antenna, a vertical antenna will have a vertical polarization. A signal from an antenna with vertical polarization will not be received by an antenna with horizontal polarization because of the fields are at right angles and not aligned (Beasley, Miller 2008). The experiment was run with this configuration and produced a reliable signal at a range of 100m.

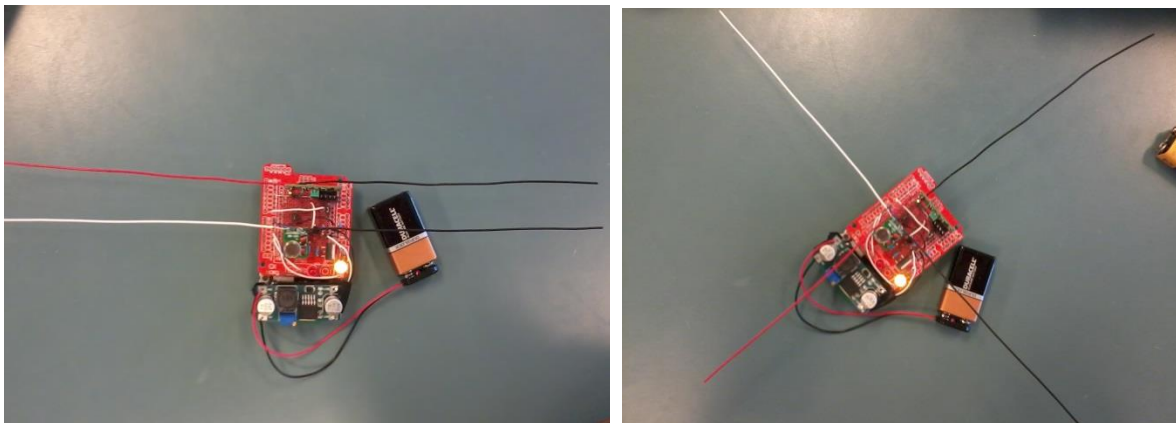


Figure 10: Configuration of dipole antennas

5.4 Code structure

The code in the previous stage was unorganised and only consisted of a single loop. The code had to be organised into function blocks, as it provided large sections of code that could be reused and copied to other Arduinos. The functions could have been organised into libraries and included into each of the codes. However, the pin assignments for the two boards are different and all of them would have to be declared through the library function, which would appear unordered.

The code had implemented into states to organise what code executed in particular circumstances, e.g. sending data from Office to Master, acknowledgement received from Master, etc. Also it proved useful when the polling system was tested (see 5.5). Instead of the Master waiting to receive a signal from the Office, the state was changed to 1 to skip the requirement.

5.5 Polling

The next prototype includes a polling system, as explained above. In the first stage, one Arduino will be loaded with the Office code and transmit to the Master normally. The Master will be implemented to wait for a period of time before transmitting to the sensors. This is to avoid the case where the Office doesn't receive back the acknowledgement and keeps transmitting.

As there are only two Arduinos available, one Arduino acted as the Master and the other acted as the Sensor modules. The Master is loaded with a range of addresses to indicate each of the Sensors and poll each one. The Sensors Arduino will act as many sensors (addresses 101 to 105) and have different settings allocated (Figure 11). Once polled, the silo changes its setting and sends back an acknowledgement containing its current state.

After each address is polled, the Master checks the silos that haven't reported back an acknowledgement, and then re-polls those addresses. The program will retry a number of times, about 10, before continuing. If there is still no acknowledgement from an address, an error message is given to indicate that a Sensor can't communicate with the Master or it is not switched on.

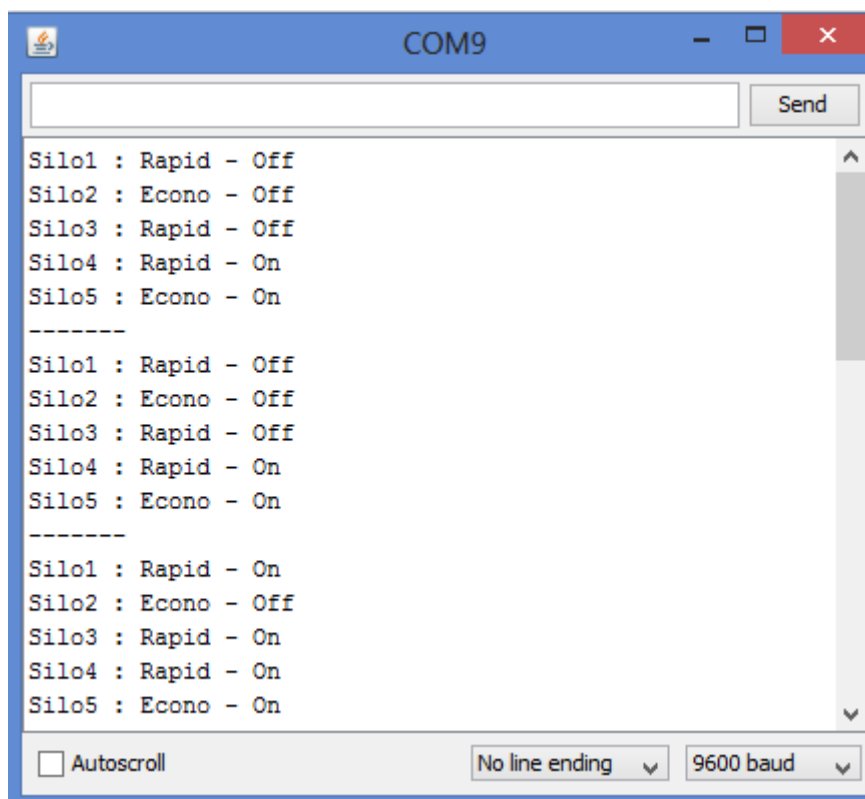


Figure 11: Polling test turning Rapid on (from Sensors module)

6 Conclusion

This project has taken relevant literature and researched possible solutions for the problem. There was research done on the existing grain silo setup, the available options that are available and wireless communication theory. The communication theory had spanned different areas such as packet structure, communication methods, antenna design, address assignment and security.

A method of testing the system has been detailed which involves various types of equipment such as the oscilloscope, multimeter and computer. The ethical issues have been carefully considered implemented with the design

There were several prototypes designed and implemented in the duration of the project. Firstly, the prototype had achieved simplex communication, where the problems of timing and packet structure were identified and solved. The second prototype had added half-duplex communication, which made a reliable channel for communication. However, there was a problem with inadequate transmission distance and a dipole antenna had to be constructed. In the third prototype, the arrangement of the antennas had presented a problem and was solved. The final prototype constructed in this project had implemented the polling system and achieved communication between the Master and the Sensors modules and displayed errors to the Serial Monitor.

6.1 Further Work and Recommendations

The time given for the project was limited and there is still further work that can be done to complete the system. There are a few recommendations, detailed below, for future research into this project.

6.1.1 Antennas

As stated before, dipole antennas with a spacing of $\lambda/2$ has to be constructed to avoid reflection. Otherwise, the circuit can be designed to use only one antenna for the transmitter and receiver.

The thickness of the antenna could be increased to give a more rigid design and protect against environmental factors such as wildlife damaging the antennas. If the size of the antenna is increased, the total length has to be trimmed by a few percent to make it resonant and have no imaginary component to its impedance (Stutzman W, 1998).

Also, the feed lines to the antennas have to be considered, a coaxial cable would be selected in this situation. The impedance of the line has to be matched to 73Ω , to provide a good VSWR (Voltage Standing Wave Ratio) and match impedance of the dipole antenna.

The prototype could be designed to use only one antenna per module, by connecting the transmitter and receiver lines together. This wasn't tested during the project as there was a risk that the receiver wouldn't have protection from the current coming from the transmitter. If this is the case, the receiver would be damaged and not operate. As there are no datasheets or schematics for the receiver module, testing would have to be conducted to determine if it is capable of using one antenna.

6.1.2 Circuit Construction

The design of the circuit could be improved by manufacturing a PCB (Printed Circuit Board) and sorting the layout of the components. The neater design would reduce poor solder joints, bridging of tracks and make it easier to test. The PCB could be designed have a ground plane which will act with a monopole antenna.

6.1.3 Address Assignment

The addressing system and routing, as stated above, could have been implemented into the design. Simple polling and BAAM could be tested in the silo environment. Simple polling is directly from the Master to the Sensors without any routing. This technique would be faster but might have sensor that are unreachable. On the other hand, BAAM would be slower due to the routing, but will be more reliable when finding Sensor modules. These methods, or others, would have to be tested to determine the optimal solution.

6.1.4 Security

The encryption process has not been implemented as it is one of the final stages of the project. The method of using a key can be done, however transferring the key to other devices and other encryption methods have not been researched.

References

- AMCA 2013, 'Spectrum at 434 MHz for low powered devices', *TheAMCA*, 30 July, viewed 11/09/13, < <http://www.acma.gov.au/theACMA/spectrum-at-434-mhz-for-low-powered-devices> >
- Beasley J, Miller G 2008, *Modern Electronic Communication 9th edn*, Pearson, Upper Saddle River, New Jersey.
- Fusae, T 2001, *Storage Aeration & Fumigation - Grain Management No 1 - V1.3*, RFM Australia Pty Ltd 2004, Toowoomba, viewed 16/03/2013, <<http://www.agridry.com.au/pdf/no1storagev1.4.2.pdf>>
- Jacobsen, Eric 2008, 'Frequency Dependence in Free Space Propagation', viewed 11/09/13, <<http://www.dsprelated.com/showarticle/62.php>>
- Parsons, D & Hancock, N 2012, *ELE3506 Electronic Measurement*, University of Southern Queensland, Toowoomba. p 2.42
- El Rachkidy, N.; Guitton, A.; Bakhache, B.; Misson, M., "Address assignment for wireless sensor networks in mines," *Wireless Communications in Unusual and Confined Areas (ICWCUCA), 2012 International Conference on* , vol., no., pp.1,4, 28-30 Aug. 2012
URL: <http://ieeexplore.ieee.org.ezproxy.usq.edu.au/stamp/stamp.jsp?tp=&arnumber=6402494&isnumber=6402322>
- Stutzman, W & Thiele G, 1998, *Antenna Theory and Design*, John Wiley and Sons, Inc, New York, USA.
- Schurgers, C.; Kulkarni, G.; Srivastava, M.B., "Distributed on-demand address assignment in wireless sensor networks," *Parallel and Distributed Systems, IEEE Transactions on* , vol.13, no.10, pp.1056,1065, Oct 2002,
URL: <http://ieeexplore.ieee.org.ezproxy.usq.edu.au/stamp/stamp.jsp?tp=&arnumber=1041881&isnumber=22330>

Appendix A – Project Specification

University of Southern Queensland
Facility of Engineering and Surveying

ENG 4111/2 Research Project
Project Specification

For: **Christopher White**
Topic: Wireless Grain Silo
Supervisors: Dr John Leis
Sponsorship: Agridry International Ltd.

Project Aim: To design a module to communicate data to and from a grain silo approximately 100 meters away. Transmission of the data will be either RF or powerline carrier, which will send the controller signals to the silos and send back the temperature and humidity information.

Program: (Issue B, 21/03/13)

1. Design the 'office' module at the controller end
2. Design the 'master' module at the silo end
3. Create a protocol to allow the 'office' and 'master' modules to communicate
4. Implement error checking to avoid collision or interference
5. Test to ensure they work and test limits
6. Design 'sensor' modules for each of the silos
7. Research into ways of communicating from 'master' to 'sensor' (RF or powerline carrier)
8. Create a protocol for 'master' and 'sensor' communication
9. Implement polling or pseudo-random methods to obtain information from silos

As time and resources permit:

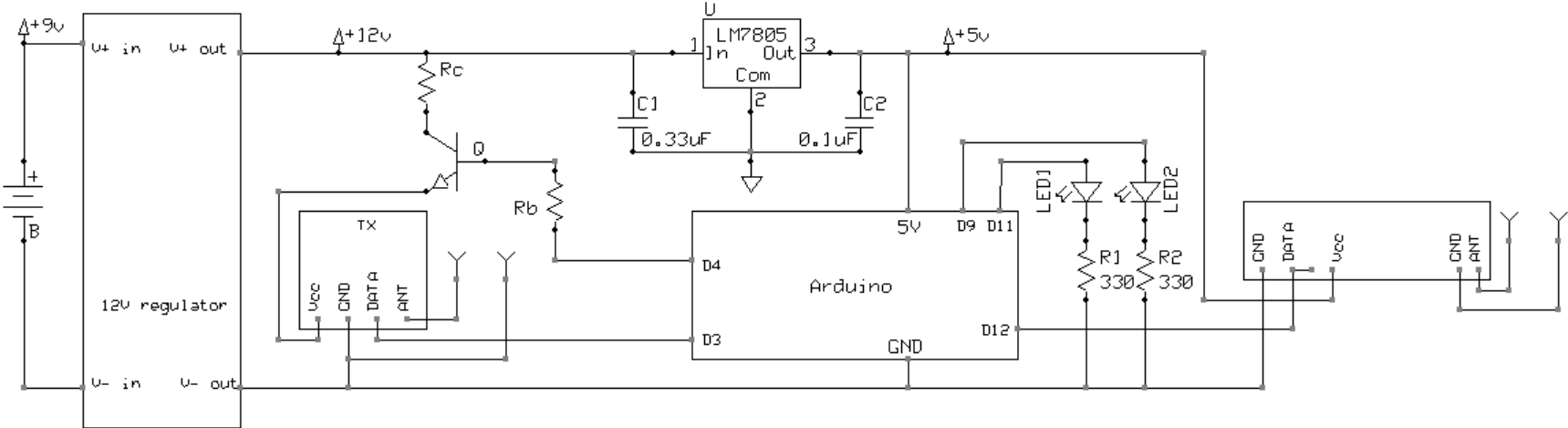
10. Evaluate the performance of the prototype, if necessary alter the communication protocols and error checking mechanisms
11. Design the system so that it can be installed and maintained easily.
12. Design a user-friendly interface that displays the silo information.

Student Name: Christopher White

Supervisor Name: Dr John Leis

Examiner/Co-Examiner: Chris Snook

Appendix B - Circuit schematics



Appendix C – Office code

```
#include <SoftwareSerial.h>    // use SoftwareSerial library

byte data = 0;
byte ack = 0;

int state = 0;           //starting state

long time1 = 0L;
long time2 = 0L;

const byte rapid = B01000001;    //possible settings
const byte econ = B01000010;
const byte both = B01000011;
const byte off = B01000000;

byte officeA = B00100100; //24 in HEX  Arduino addresses
byte masterA = B00011000; //18

const byte header = B00110100; // 34

const int TXLED = 11;    // Pin assignments for the board
const int RXLED = 9;
const int Trans = 4;
const int RX = 12;
const int TX = 3;

SoftwareSerial SerialA(RX,TX); //Use Software serial library and define RX and TX pins

void setup()
{
  SerialA.begin(1200);
  Serial.begin(9600);
  digitalWrite(Trans,LOW);
  pinMode(TX,OUTPUT);
  pinMode(Trans,OUTPUT);
  pinMode(RXLED,OUTPUT);
  pinMode(TXLED,OUTPUT);
  pinMode(RX,INPUT);
}

void loop()
{
  data = rapid;

  switch (state)
  {
  case 0:
    Send(masterA,officeA,data); //using Send function, send data to Master from Office
    time1 = millis();           //time1 is the number of milliseconds since the program started

    while (millis()-time1 <= 1000); //read data for 1000ms
```

```

{
ack = Rec(masterA);      // check for acknowledgement packet from Master
if (ack == data)        // check against the data sent
{
time2 = millis();      // read current time
state = 1;
}
}
break;

case 1:

digitalWrite(TXLED,HIGH);          //Indicate with LED

if (millis()-time2 > 10*1000 && time2 != 0) // wait for 10 seconds
{
digitalWrite(TXLED,LOW);
delay(500);
time2 = 0;                      //reset variables
state = 0;
}
break;
}
}

```

//=====

```

void Send(byte addTo,byte addFrom,byte data1) //create Send function
{
byte sync = B01010101;
byte chsum = 0;
chsum = (((header ^ addTo) ^ addFrom) ^ data1);

digitalWrite(TXLED,HIGH);          //turn on LED
digitalWrite(Trans,HIGH);         //enable power for transmitter
SerialA.write(sync); // Send bytes
SerialA.flush();
delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(header);
SerialA.flush();
delay(3);
SerialA.write(addTo);
SerialA.flush();
delay(3);
SerialA.write(addFrom);
SerialA.flush();
delay(3);
SerialA.write(data1);
SerialA.flush();
}

```

```

delay(3);
SerialA.write(chsum);
digitalWrite(TX,LOW);           //set level low for serial
digitalWrite(Trans,LOW);       //disable power for transmitter
digitalWrite(TXLED,LOW);       //turn off LED
}

byte Rec(byte addFrom)         //create Receive function
{
  int bufL=15;                 //buffer length
  byte buf[bufL-1];           //buffer.
  int hstart = -1;            //header position

  if (SerialA.available() >= bufL) // wait for serial to have enough values
  {
    digitalWrite(RXLED,HIGH);    // turn on LED
    for(int i=1;i<bufL;i++)
    {
      buf[i]=0;                  //set all of buffer to zero
      buf[i] = SerialA.read();   //put values in buffer
      if((buf[i] == header) && (hstart == -1)) //find header
      {hstart = (i);}
    }
    digitalWrite(RXLED,LOW);    // turn off LED

    // read checksum byte, check if address is for Office and where it is from.

    if (buf[hstart + 4] == (((buf[hstart] ^ buf[hstart+1]) ^ buf[hstart+2]) ^ buf[hstart+3]) &&
(buf[hstart + 1] == officeA) && (buf[hstart + 2] == addFrom))
    {
      return buf[hstart+3];     //return the data packet
    }
  }
}
}

```

Appendix D – Master Code

```
#include <SoftwareSerial.h>

SoftwareSerial SerialA(7,8); //Use Software serial library and define RX and TX pins

int state = 1;
int k = 0;
byte header = B00110100; // 34

byte data = 0; //to store the data
byte data2 = 0; //to store the data
byte temp_data = 0; //to store the data
byte silo[5];

const byte rapid = B01000001; //possible settings
const byte econ = B01000010;
const byte both = B01000011;
const byte off = B01000000;

const byte rapid_off = B00100001;
const byte econ_off = B00100010;

long time2 = 0L;

byte officeA = B00100100; //24 Arduino addresses
byte masterA = B00011000; //18 in HEX

const int TXLED = 12; // Pin assignments
const int RXLED = 11;
const int Trans = 3;
const int RX = 7;
const int TX = 8;

void setup()
{
  SerialA.begin(1200); //set baud rates
  Serial.begin(9600);
  digitalWrite(Trans,LOW); //set transistor low
  pinMode(TX,OUTPUT); //set pin modes
  pinMode(Trans,OUTPUT);
  pinMode(RXLED,OUTPUT);
  pinMode(TXLED,OUTPUT);
  pinMode(RX,INPUT);

  for (int i=0;i<6;i++)
  {
    silo[i]= 0;
  }
}
```

```

void loop()
{
  switch (state)
  {
  case 0:          //sending acknowledgement to Office

    data = Rec(officeA);    // check for data from Office

    if (data == rapid || data == econ || data == both || data == off) //check setting
    {
      delay(50);    // short delay
      Send(officeA, masterA, data); //sent data to Office from Master
      temp_data = data; //store data into tempory variable
      data = 0; //reset data
      time2 = millis(); //set time2 to milliseconds since program started
    }

    if (millis() - time2 >= 5000 && time2 != 0) //if no signal arrives after 5 sec
    {
      state = 1;
    } // change state

    break;

  case 1:          //sending data to silos
    k++;
    temp_data = rapid; //when not using Office in testing

    for (byte j=101; j<106; j++) // go through silos addresses
    {
      Send(j, masterA, temp_data); //send data to silos from Master with 'temp' data

      Serial.print("%");
      long time1 = millis(); //read time

      while (millis()-time1 <= 1000); //read data for 1000ms
      {
        Serial.println("*");
        delay(50);
        data2 = Rec(j); // read acknowledgement

        silo[j-100] = data2; //set that is has been acknowledged
        Serial.println(silo[j-100]);
        data2 = 0;
      }
    }
    //check if all acknowledged or 3 attempts made

    if ((silo[0] != 0 && silo[1] != 0 && silo[2] != 0 && silo[3] != 0 && silo[4] != 0) || k == 5)
    {
      k = 0;
    }
  }
}

```

```

    temp_data = 0;
    state = 2;
}
time2 = 0;

break;

case 2:

for (int i=1;i<=5;i++) //Report settings
{
    Serial.print("Silo");
    Serial.print(i);
    Serial.print(" : ");
    Serial.println(silo[i]);

    /*
    switch (silo[i])
    {
    case rapid_off:
        Serial.println("Rapid - Off");
        break;

    case econ_off:
        Serial.println("Econo - Off");
        break;

    case rapid:
        Serial.println("Rapid - On");
        break;

    case econ:
        Serial.println("Econo - On");
        break;

    default:
        Serial.println("None");
        break;
    }

    */
}
state = 0;
break;
}
}

//=====

void Send(byte addTo,byte addFrom,byte data1)
{

```

```

byte sync = B01010101;
byte chsum = 0;
chsum = (((header ^ addTo) ^ addFrom) ^ data1);

digitalWrite(TXLED,HIGH);
digitalWrite(Trans,HIGH);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(header);
SerialA.flush();
delay(3);
SerialA.write(addTo);
SerialA.flush();
delay(3);
SerialA.write(addFrom);
SerialA.flush();
delay(3);
SerialA.write(data1);
SerialA.flush();
delay(3);
SerialA.write(chsum);
digitalWrite(TX,LOW);
digitalWrite(Trans,LOW);
digitalWrite(TXLED,LOW);
}

```

```

byte Rec(byte addFromR)
{
  int bufL=15; //buffer length
  byte buf[bufL-1]; //buffer.
  int hstart = -1; //header position

  if (SerialA.available() >= bufL)
  {
    digitalWrite(RXLED,HIGH);
    for(int i=1;i<bufL;i++)
    {
      buf[i] = SerialA.read(); //put values in buffer
      if((buf[i] == header) && (hstart == -1)) //find header
      {
        hstart = (i);
      }
    }
  }
}

```

```
digitalWrite(RXLED,LOW);

if (buf[hstart + 4] == (((buf[hstart] ^ buf[hstart+1]) ^ buf[hstart+2]) ^ buf[hstart+3]) && (buf[hstart
+ 1] == masterA) && (buf[hstart + 2] == addFromR)) //read the checksum byte
{
    return buf[hstart+3];
}
}
```


Appendix E – Sensor code

```
#include <SoftwareSerial.h>

int state = 0;
byte header = B00110100; // 34

byte data = 0;      //to store the data
byte data2 = 0;     //to store the data
byte temp = 0;      //to store the data
byte silo_state[10];

const byte rapid = B01000001;//65
const byte econ = B01000010;//66
const byte both = B01000011;//67
const byte manu = B01000100;//68
const byte off = B01000000;//64

const byte rapid_off = B00100001;//33
const byte econ_off = B00100010;//34

long time1 = 0L;
long time2 = 0L;

byte officeA = B00100100; //24
byte masterA = B00011000; //18 in HEX
byte silo = 0;

const int TXLED = 11;
const int RXLED = 9;
const int Trans = 4;
const int RX = 12;
const int TX = 3;

SoftwareSerial SerialA(RX,TX); //RX,TX for RF module

void setup()
{
  SerialA.begin(1200);
  Serial.begin(9600);
  digitalWrite(Trans,LOW);
  pinMode(TX,OUTPUT);
  pinMode(Trans,OUTPUT);
  pinMode(RXLED,OUTPUT);
  pinMode(TXLED,OUTPUT);
  pinMode(RX,INPUT);
}

void loop()
{
  switch (state)
  {
    case 0:      //set silo states
      silo_state[1] = rapid_off;
      silo_state[2] = econ_off;
```

```

silo_state[3] = rapid_off;
silo_state[4] = rapid;
silo_state[5] = econ;

state = 1;
break;

case 1:          //report silo states

for (int i=1;i<=5;i++)
{

    Serial.print("Silo");
    Serial.print(i);
    Serial.print(" : ");

    switch (silo_state[i])
    {
    case rapid_off:
        Serial.println("Rapid - Off");
        break;

    case econ_off:
        Serial.println("Econo - Off");
        break;

    case rapid:
        Serial.println("Rapid - On");
        break;

    case econ:
        Serial.println("Econo - On");
        break;
    }
};
state = 2;
break;

case 2:
data = Rec(masterA,silo);          //receive data from Master
delay(50);
//-----
if (data == rapid)
{
    switch (silo_state[silo -100])
    {
    case econ:
        Send(masterA,silo,econ_off);    //send acknowledgement to master
        silo_state[silo-100] = econ_off;
        break;

    case rapid:
        Send(masterA,silo,rapid);      //send acknowledgement to master
        break;
    }
}

```

```

case econ_off:
    Send(masterA,silo,econ_off);    //send acknowledgement to master
    break;

case rapid_off:
    Send(masterA,silo,rapid);    //send acknowledgement to master
    silo_state[silo-100] = rapid;
    break;
}
}

//-----

if (data == econ)
{
    switch (silo_state[silo - 100])
    {
    case econ:
        Send(masterA,silo,econ);    //send acknowledgement to master
        break;

    case rapid:
        Send(masterA,silo,rapid_off);
        silo_state[silo-100] = rapid_off;
        break;

    case econ_off:
        Send(masterA,silo,econ);
        silo_state[silo-100] = econ;
        break;

    case rapid_off:
        Send(masterA,silo,rapid_off);
        break;
    }
}

//-----

if (data == both)
{
    switch (silo_state[silo -100])
    {
    case econ:
        Send(masterA,silo,econ);
        break;

    case rapid:
        Send(masterA,silo,rapid);
        break;

    case econ_off:
        Send(masterA,silo,econ);
        silo_state[silo-100] = econ;
        break;

```

```

    case rapid_off:
        Send(masterA,silo,rapid);
        silo_state[silo-100] = rapid;
        break;
    }
}
//-----

if (data == off)
{
    switch (silo_state[silo -100])
    {
        case econ:
            Send(masterA,silo,econ_off);
            silo_state[silo-100] = econ_off;
            break;

        case rapid:
            Send(masterA,silo,rapid_off);
            silo_state[silo-100] = rapid_off;
            break;

        case econ_off:
            Send(masterA,silo,econ_off);
            break;

        case rapid_off:
            Send(masterA,silo,rapid_off);
            break;
    }
}
//-----

if (millis()-time1 >= 5*1000) //every 5 seconds
{
    Serial.println("-----");
    time1 = millis();
    state = 1; //change state (report silo states)
}
break;
}
}

//=====

void Send(byte addTo,byte addFrom,byte data1)
{
    byte sync = B01010101;
    byte chsum = 0;
    chsum = (((header ^ addTo) ^ addFrom) ^ data1);

    digitalWrite(TXLED,HIGH);
    digitalWrite(Trans,HIGH);
    SerialA.write(sync);
    SerialA.flush();
}

```

```

delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(sync);
SerialA.flush();
delay(3);
SerialA.write(header);
SerialA.flush();
delay(3);
SerialA.write(addTo);
SerialA.flush();
delay(3);
SerialA.write(addFrom);
SerialA.flush();
delay(3);
SerialA.write(data1);
SerialA.flush();
delay(3);
SerialA.write(chsum);
digitalWrite(TX,LOW);
digitalWrite(Trans,LOW);
digitalWrite(TXLED,LOW);
}

```

```

byte Rec(byte addFrom, byte &addTo)

```

```

{
  int buf1=15;    //buffer length
  byte buf[buf1-1]; //buffer.
  int hstart = -1; //header position

  if (SerialA.available() >= buf1)
  {
    digitalWrite(RXLED,HIGH);
    for(int i=1;i<buf1;i++)
    {
      buf[i]=0;      //set all of buffer to zero
      buf[i] = SerialA.read(); //put values in buffer
      if((buf[i] == header) && (hstart == -1)) //find header
      {
        hstart = (i);
      }
    }
    digitalWrite(RXLED,LOW);

    if (buf[hstart + 4] == (((buf[hstart] ^ buf[hstart+1]) ^ buf[hstart+2]) ^ buf[hstart+3]) &&
(buf[hstart + 2] == addFrom)) //read the checksum byte
    {
      addTo = buf[hstart+1];
      return buf[hstart+3];
    }
  }
}
}

```

Appendix F – Risk Assessment

| Hazard | At risk | Risk level | Exposure | Consequences | Control |
|--|-------------------|------------------|--------------|--|---|
| During Project | | | | | |
| Damaging Lab equipment (Oscilloscope & Power supply) | Equipment, Myself | Very slight | Occasionally | Minor/ major equipment damage Minor/ major injury | Be mindful of connecting instruments to circuits. |
| Testing outside - trip hazard of measuring tape | Other people | Slight | Rarely | Minor injury | Test in a wide open space (area outside Z block). Look out for pedestrians and direct them. |
| Testing outside – sun damage | Myself | Slight | Occasionally | Minor – Major injury | Wear sunscreen when going outside for long periods |
| Damaging Arduino | Equipment | Very Slight | Frequently | Minor – major equipment damage | Be wary of connections. Use low voltages if possible. |
| Electrocution (Lab) | Myself | Extremely Slight | Occasionally | Major injury/death | Be careful of high voltage devices, ensure they are connected properly |
| Soldering - burns | Myself | Slight | Occasionally | Minor injury | Have a tidy workspace and use appropriate equipment when soldering |
| Soldering - fumes | Myself | Slight | Occasionally | Minor injury | Make sure exhaust fan is turned on when soldering |

| | | | | | |
|---|-----------------------------|------------------|--------------|--|--|
| Soldering - eyes | Myself | Slight | Occasionally | Minor – Major injury | Wear eye protection |
| After Project | | | | | |
| Electrocution (placed in Office/ Silos) | Others, Myself | Extremely Slight | Occasionally | Major injury/death | Make the design fool – proof. Ensure wires are connected properly |
| Water damage to equipment | Equipment | Slight | Occasionally | Minor/ Major equipment damage | Incorporate water-proof cases in the design of the modules |
| Physical damage | Equipment | Slight | Occasionally | Minor/ Major equipment damage | Make case durable and good solder joints. |
| EMI (Electromagnetic Interference) | Equipment, other equipment. | Slight | Frequently | Minor equipment malfunction/interference | Design circuit to follow regulations. |