PUBLIC TRANSPORT ANNOUNCEMENT DEVICE FOR THE BLIND USING RFID- AND ARM-BASED PLATFORM.

By

MOHAMMAD SHAFIQ BIN MOHAMMAD ASHRAF

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

PUBLIC TRANSPORT ANNOUNCEMENT DEVICE FOR THE BLIND USING RFID- AND ARM-BASED PLATFORM

by

Mohammad Shafiq Bin Mohammad Ashraf

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

_____

Dr. Zuki Yusoff
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2014

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

Mohammad Shafiq Bin Mohammad Ashraf

ABSTRACT

This report presents the development of a public transport announcement device for the blind using Radio Frequency Identification (RFID) and ARM-based platform. The device functions to notify the blind person of the arrival and determining the identity of the bus using voice announcement. The device also constantly notifies the blind person of the next station ones the blind has departed the desired bus. The RFID tag chosen for this project is of active RFID type since the requirement is to detect a bus at a few tens of meter range. The RFID is based on nRF24L01+ transceiver module. This RFID operate at 2.4GHz frequency which is a free ISM band. Upon testing, the receiver was able to detect the ID transmitted up to 150feet in open air and direct line of sight condition. A programmer device which functions to program the RFID tag with a desired ID is also designed. The ARM-based Platform used is Freescale Freedom FRDM-KL25Z board which is based on 32-bit ARM Cortex-M0+ Core Processor. The RFID is interfaced with FRDM-KL25Z board via SPI communication. As for sound module, WTV020-SD MODULE is chosen and this module communicates with the FRDM-KL25Z board using virtual serial port created using digital I/O`s..

ACKNOWLEDGEMENTS

First of all, I would like to thank Allah S.W.T. the all mighty for giving me strength, courage and wisdom to complete this project. Secondly, I would like to thank my parents for always being by my side and always remembering me in prayers. Next I would like to give my gratitude to Dr. Zuki Yusoff for being my supervisor and putting his trust on my abilities to perform this project. I would also like to thank for his guidance throughout the project. I consider myself lucky and honored to have been given a project by Dr. Zuki which might not only be resulting a product for a good cause but also achieved certain recognitions such as a Bronze medal in recently held Electrical and Electronics Engineering Exhibition (ELECTREX), and a Gold medal and Best Research Final Year Project Award in 33rd Science and Engineering Exhibition (SEDEX). I would also like to thank my good friend Tariq Aziz for helping me out numerous times whenever I needed. His help really help speed up the development of the project. Next, I would like to thank my other good friends for asking me programming related questions to which I did not know the answers. This made me study further related to those problems, and eventually helped smoothen the programming of this project literally.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RFID	:	Radio Frequency Identification.
ID	:	Identification
MCU	:	Microcontroller Unit
SPI	:	Serial Peripheral Interface
PC	:	Personal Computer
RF	:	Radio Frequency
IP	:	Internet Protocol
UART	:	Universal Asynchronous Receiver/Transmitter
LOS	:	Line-of-Sight
i/o	:	Input/Output
FRDM	:	Freescale Freedom Board
AI	:	Artificial Intelligence
LCD	:	Liquid Crystal Display

CHAPTER 1
INTRODUCTION

1.1 Background

Radio Frequency Identification (RFID) is a term commonly used for technologies employing radio waves in object detection mechanisms [2].This technology has been finding rapid popularity in all sort of industries, let it be asset tracking in manufacturing plants [2] or locating personnel on offshore oil and gas platforms [3]. The popularity gained by this technology is rather due to its simple operating concept of placing a microchip with an antenna on one item and reading the data off this microchip using a another device through RF [4]. The communication range between the reader and the transmitter can vary from a few cm to tens of meters [5], which widens its bandwidth of applications requiring non line of sight detection of further ranges.

ARM processor on its playground, is gaining just as much attention as the RFID. This is due to the fact it being based on 32-bit Reduced Instruction Set Computing (RISC) architecture, having high clock speed, highly competitive price and many other significant advantages over other 8-bit and 16-bit and even 32-bit devices of its competitors [6]. This makes it ideal in a lot of practical high speed applications just as would be interfacing it with any radio frequency module.

1.2 Problem Statement

The blind have hard time when they want to take a bus as a mode of public transport. Some of the problems/challenges they face are:

1. Difficulty in determining the arrival of a bus.
2. Difficulty in differentiating the bus they want to take to reach their desired destination.
3. Difficulty determining whether they have reached their destination once departing a desired bus.

1.3 Objectives

The objective of this project is to design, develop and implement a public transport announcement device for the blind using RFID- and ARM based platform. The ARM board used in this project is Freescale Freedom Platform which is based on ARM Cortex-M0+core [8]. The board is interfaced with an RFID module as well as a sound module. The board is able to extract certain information from the RFID module. Process this information and initiate the play of a sound file relative to the information received from the RF module.

1.4 Scope of Study

The Scope of study for this project shall include, but not limited to, the following:

- Design circuitry for RF module,
- Design the circuitry of Freescale Freedom board to interface with the RF module,
- Program the Freescale Freedom board to process the information provided by the RF module,
- Design the circuitry to interface the Freescale Freedom board with the sound module,
- Design the circuitry for the proper operation of the sound module,
- Program the Freescale to be able to play announcements based on the information received from the RF module,
- And/or other features if found feasible within relative to the time frame of the project, e.g. GPS module interface to identify the exact position of the bus during travel.

CHAPTER 2
LITERATURE REVIEW AND THEORY

An announcement system is a device which functions to make a certain announcement upon a certain type of trigger. The trigger can be of any type, let it be real time human voice generated or be predefined / prerecorded announcement initiated by a microcontroller process output. Announcement systems are very useful in emergency cases, such as, fire out break and so on. Or they can have a simple functionality of announcing daily school activities [9]. In simple words, the basic job of an announcement system is to notify a designated audience with various information.

Remote control has been among the most prevalent and convenient device ever invented [10]. RC has been used as a wireless control mechanism in controlling various devices, for example, TVs, DVD players, and so on , making our life easier [10]. Conventionally, Infrared (IR) has been used as a mode of communication between the controller and the controlled device. However, IR technology have some disadvantages, for instance it requires a direct line-of-sight for proper operation and has a range of approximately 10 meters only [11]. Nowadays, Radio Frequency (RF) based wireless communication has been getting higher attention as compared to IR based devices. This is due to some major advantages such as, higher range, and ability to penetrate through walls, eliminating the line-of-sight requirement [11].

2.1 RFID

One of the largest applications of RF technology is in Radio Frequency Identification (RFID). RFID or more commonly known as RFID tagging is used to track just as anything. In this mechanism, an item is tagged with an RFID tag, this tag contains a unique identification(ID) of the item, and this unique ID of the tag is read using an RFID reader and this ID is compared with the database to identify the description of the item [5].

Basically RDIF tag can be categorized into two, one, Passive Tags and two, Active Tags. In Passive RFID tag systems, the tag is powered by the signal energy transmitted by the RFID reader [12]. The tag captures the reader`s energy, powers itself on and then reflects the energy back to the reader. Even though passive tags can operate in multiple frequencies such as, low frequency (30Khz to 300Khz), high frequency (3Mhz to 30Mhz) and ultra-high frequency (300 MHz to 3 GHz), their readable range is limited to approximately 10m maximum [13]. Passive tags are best suite in applications that require short range readability, such as security pass, parking tag and so on. This is because the tag is only ON when it is in the vicinity of the reader and the reflected signal energy is also very low [12].

Active tags on the other hand are equipped with their own transmitter as well as power source [13]. Usually a battery is used to power the tag and normally they operate in ultra-high frequency band. Since active tags operate using their own power source and in ultra-high frequency band, they have higher range of up to 100m [13]. Due to these characteristics, these tags are used in applications that require longer readability range, such as outdoor campus personnel tracking, container tracking and so on [14].

Having numerous numbers of applications, many development modules are also developed to help consumers make customized RFID systems. These modules have the capability to be interfaced with microcontrollers and some module even offer the capability to perform desired modulation technique.

2.2 The Arm Processor

ARM based microcontrollers are the hot picks in the market at the moment. This is due to their high specs, unparalleled performance, power efficiency, ease of use and many other advantages compared to their competitors. Freescale Freedom board, FRDM-KL25Z is based on ARM Cortex-M0+ Core processor. With clock speed of up to 48MHz, 16KB RAM and 128 KB FLASH is a powerful 32-bit development board for many application [15]. It is also equipped with multiple serial ports that makes it ideal for interfacing modules that require serial protocols as a mode of communication [8]. To summarize the functionalities the pin out of the FRDM-KL25Z is as shown is figure below:

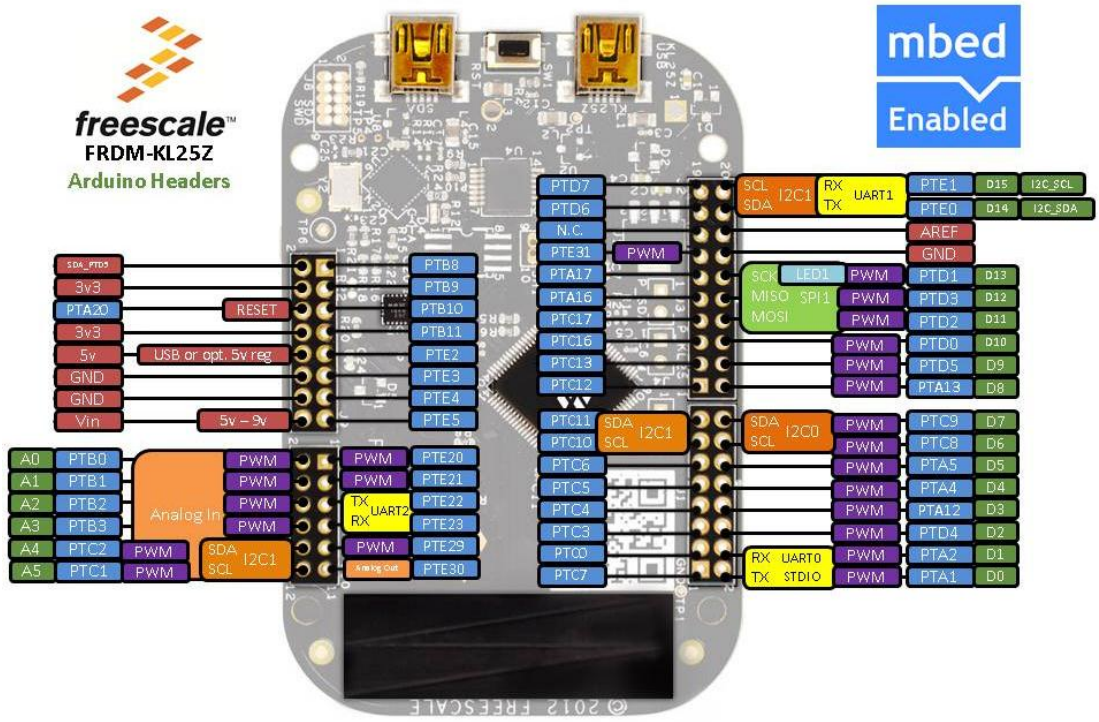

Figure 1 : Pinout of FRDM-KL25Z [1].

The key features of this board are as shown below.

- 2xSPI,
- 2xI2C,
- 3xUART,
- 6xPWM,
- 6xADC and
- General Purpose Input-Output (GPIO)

In addition to that, it is also equipped with an on-board 3-axis accelerometer (MMA8451Q), a PWM controlled RGB LED and a Capacitive touch sensor [1].

2.2 The Sound Module

A sound module is a device or an electronic instrument that functions to play a sound with or without a human-playable interface such as a keyboard. Sound modules can be "played" or triggered using an externally connected device. The external device can be a controller, a device that can provide the human-playable interface and may not have the ability to produce sounds of its own. Some sound module can synthesize the voice directly from the speaker, apply modifications to it and play it real time, while some modules can save the pre-recorded sound files and are triggered to play by many means. On other hand, some sound modules offer the ability to be interfaced

with microcontrollers and can be programmed to play sound files as desired according to the application.

Combining the capabilities of the FRDM-KL25Z board and considering the available options on RFID and sound modules related product, this report proposes to use this combination in the development public announcement device for blind.

CHAPTER 3
METHODOLOGY

3.1 Methodology and Project Activities

A waterfall methodology approach is also known as linear-sequential life cycle model [16]. Following this approach, each phase must be completed prior to the beginning of the next phase. This approach is chosen due to its simplicity to understand, can be managed easily since every phase has predetermined deliverables and a review process and is efficient in project where the requirements are well defined. This method can be illustrated as below,

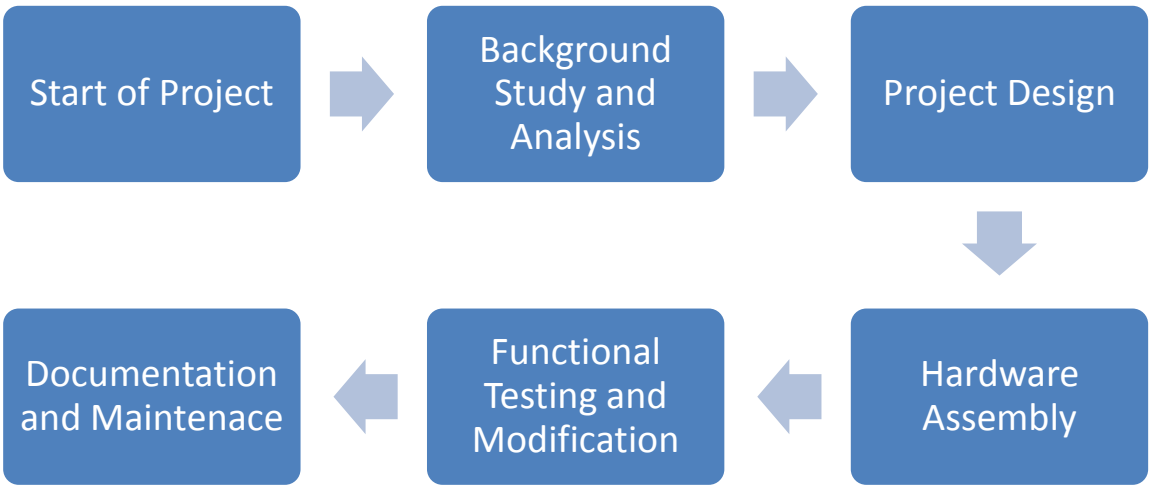

Figure 2: Overview of the waterfall Methodology approach.

3.1.1 Project Activities.

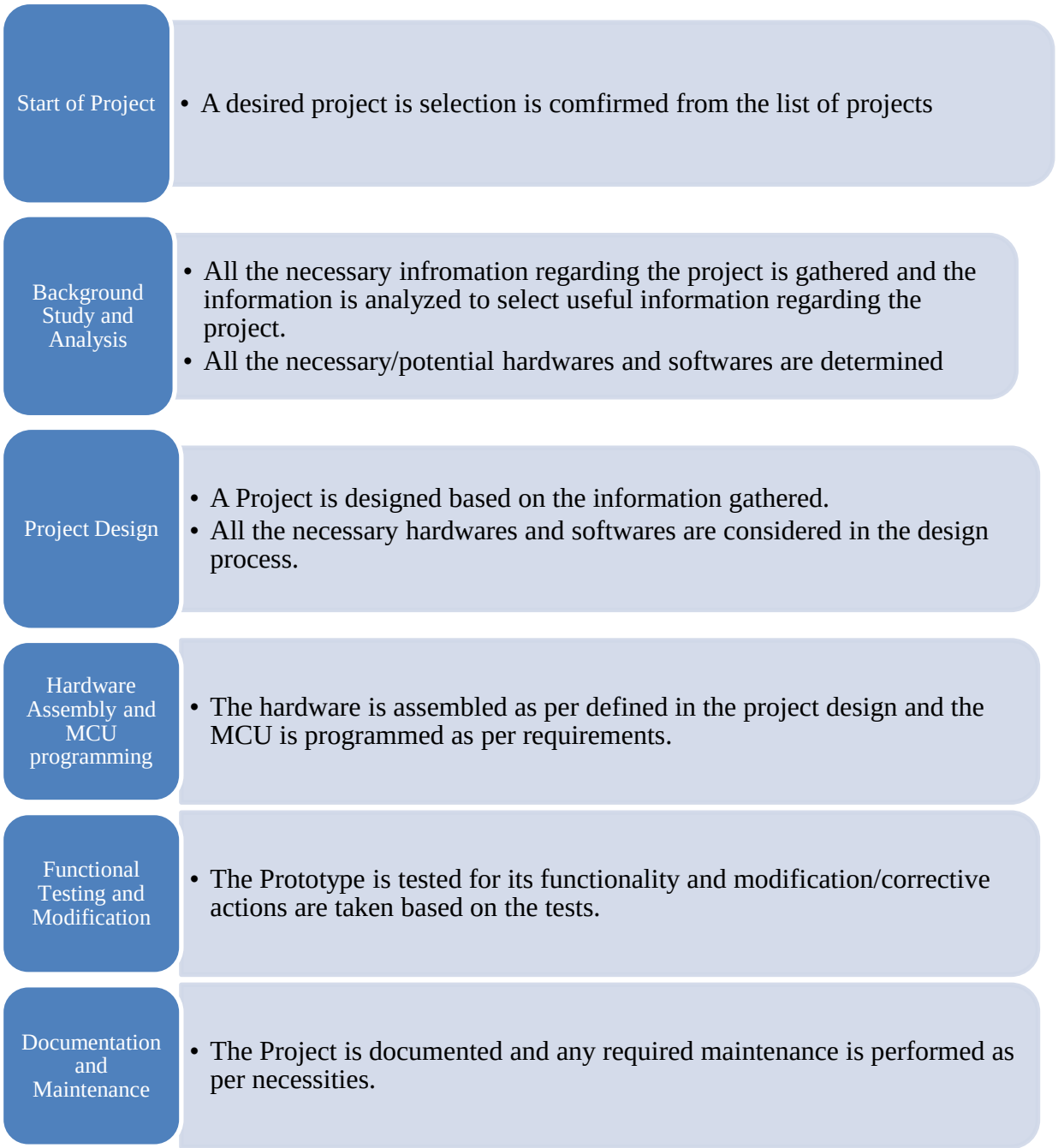

Figure 3: Project activities methodology approach.

3.2 Detailed Methodology and Project Activities

Applying the methodology approach defined earlier to this project, the flow of the project is divided into three phase, each being a major component of the device prototype and a final phase whereby all three phase are integrated. The three phases are, phase 1, RFID, phase 2, programming and phase 3, Sound module. The simplified overall view of the device architecture is shown in figure below. The three phases are indicated by the green arrow region relatively labeled as their phase number. While that the blue arrow region highlights the two different parts and location of the device, the RFID transmitter, which is located at the bus and RFID receiver together with announcement module handled by the user.

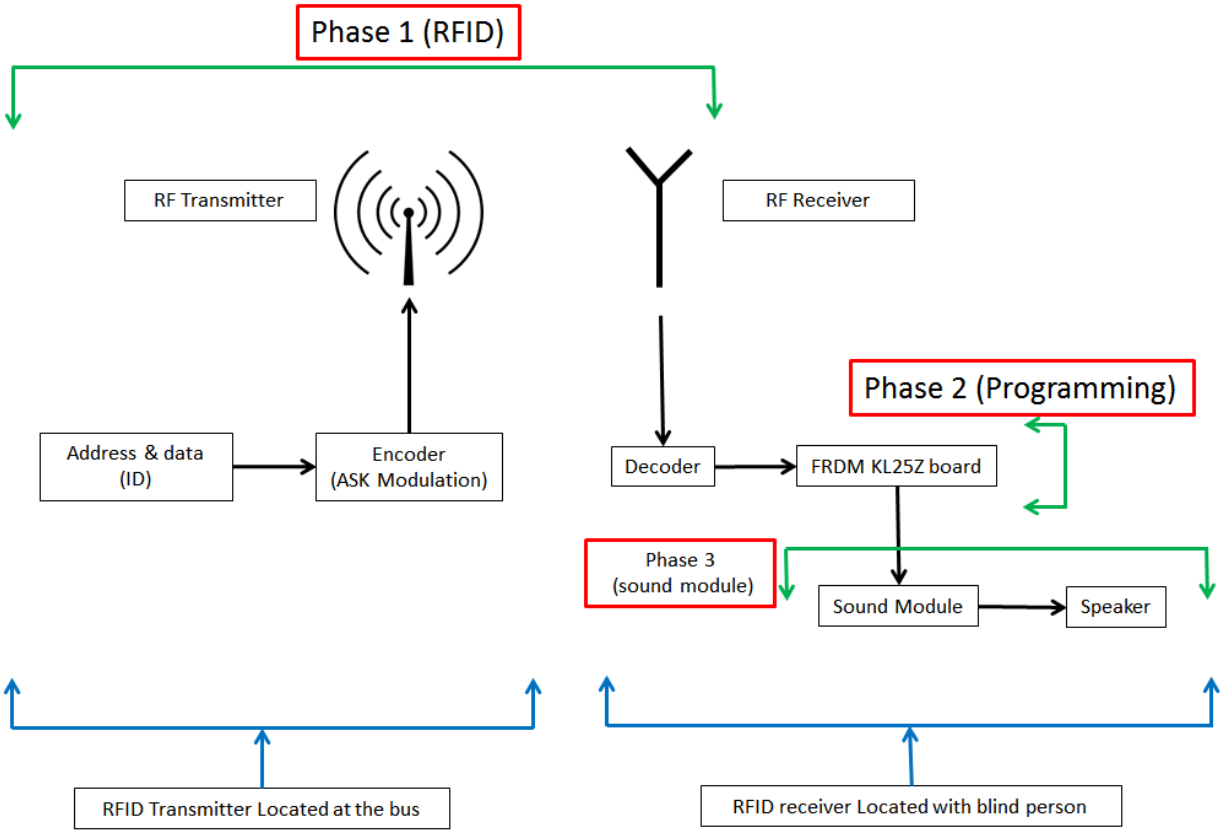

Figure 4: The overall view of the device architecture.

3.3 Phase 1: RFID

For the phase one which is related to the RFID transmitter and receiver, first of the all, the literature is reviewed to identify the potential candidates that can be used as RFID. The requirements for RFID module are such that it is able to;

i) Carry a unique ID.

ii) The tag can be read by the reader at a range of at least 100f, this is to identify the arrival of bus and the bus station earlier so is to give the user some time to stop the bus.

iii) Cheap.

Having known the best candidate, the next step is to find the supplier and buy the product. After receiving the item, the RFID is tested for its functionality to be complying with the requirements stated above. After testing the functionality is confirmed, the proper circuit to integrate RFID with other phases is designed. The whole process is as shown below.

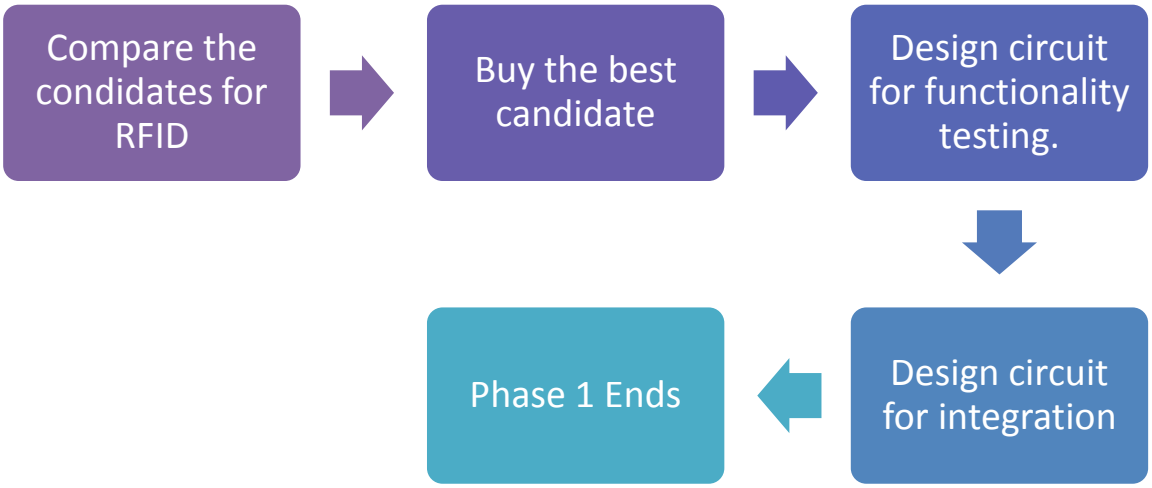

Figure 5: Phase 1 activity flow.

3.3.1 The RFID Hardware Selection.

First of all, a research was done to determine the suitable candidate as per requirement criteria stated in methodology above. A few potential candidates were identified and compared. These include the Xbee, nRF24L01 Tranceiver and RF-RX/TX-315 Module. These three modules are shown below.

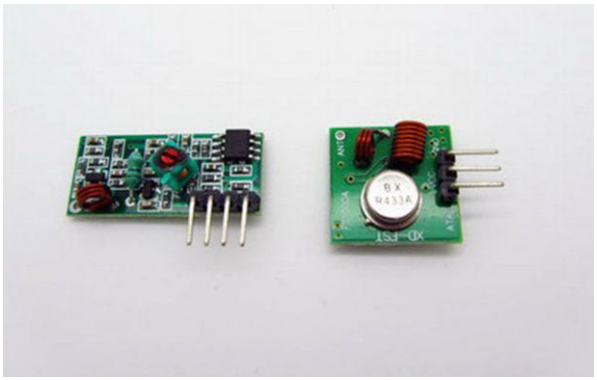

Figure 6: RF-RX/TX-315 Module.

Figure 7: XBee

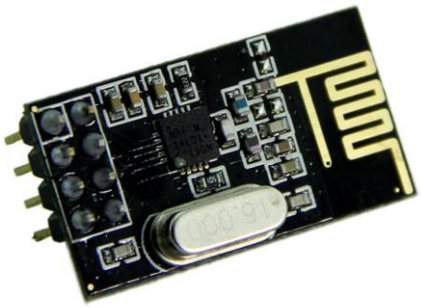

Figure 8: nRF24L01 Transceiver

Comparing the three items above, the brief description of the items is as per below.

- **nRF24L01+**

The nRF24L01+ is a single chip 2.4GHz transceiver with an embedded baseband protocol engine, designed for ultra-low power wireless applications. The nRF24L01 is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz. An MCU (microcontroller) and very few external passive components are needed to design a radio system with the nRF24L01. The nRF24L01 is configured and operated through a Serial Peripheral Interface (SPI). Some of its applications include Wireless PC Peripherals, RF remote controls for consumer electronics, Active RFID, Asset tracking systems and Home and commercial automation. Some of the key features of this module are as below;

I. Worldwide 2.4GHz ISM band operation
II. Up to 2Mbps on air data rate
III. Ultra-low power operation
IV. 11.3mA TX at 0dBm output power
V. 12.3mA RX at 2Mbps air data rate
VI. 900nA in power down
VII. 22μA in standby-I
VIII. On chip voltage regulator
IX. 1.9 to 3.6V supply range
X. Enhanced ShockBurst™
XI. Automatic packet handling
XII. Auto packet transaction handling
XIII. 6 data pipe MultiCeiver™
XIV. Low cost BOM
XV. ±60ppm 16MHz crystal
XVI. 5V tolerant inputs
XVII. Compact 20-pin 4x4mm QFN package

- **Xbee**

XBee Wi-Fi embedded RF modules provide simple serial to IEEE 802.11 connectivity. By bridging the low-power requirements of wireless device networking with the infrastructure of 802.11, the XBee Wi-Fi allows wireless application for energy management, process and factory automation, wireless sensor networks, intelligent asset management and more. Focused on the rigorous requirements of these wireless device networks, the XBee Wi-Fi gives developers IP-to-the-device capability. The Xbee uses SPI or UART interface to communicate with Microcontroller Unit (MCU). Same as nRF24L01, it operates at 2.4GHz frequency.

- **RF-RX/TX-315 Module**

The low cost RF Receiver can be used to receive RF signal from transmitter at the specific frequency. Super regeneration design ensures sensitive to weak signal. The transmitter of this module nominally operates at 315MHz frequency. With a detection range of up to 100m this module can be used as standalone RF transmitter/receiver and does not require MCU for normal operation. This module can be used with various encoders and decoders that allows it to transfer certain amount of information as per encoded by the encoders. The information can be decoded at the receiver side accordingly. Some other application that this modules offer are Industrial remote control, telemetry and remote sensing, Alarm systems and wireless reception for various types of low-rate digital signal and so on.

Based on the characteristics of each module analysed, RF-RX/TX-315 Module and nRF24L01+ are seemed to be practical with our application hence both of them were tested to determine the most feasible candidate.

3.3.2 Testing of RF-RX/TX-315 and nRF24L01+ module

To confirm RF-RX/TX-315 to be used in this project as RFID module, a circuit was designed to test its functionality and compliance to requirement specified. The following test circuit was designed.

→ To compare the functionality of the rf modules

- **The Transmitter circuit**

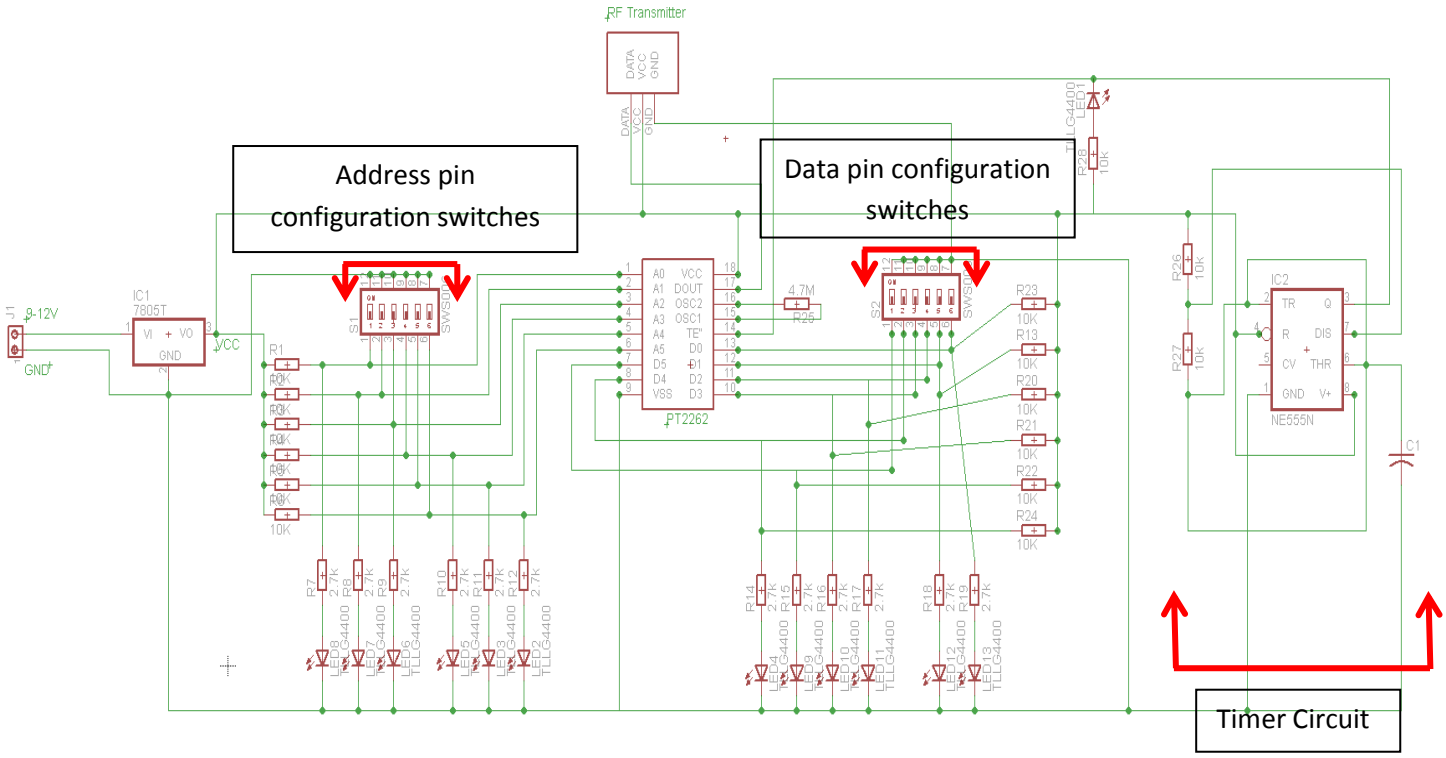

Figure 9: RFID Transmitter Test circuit.

I. Assemble the above circuit.

II. Address pins of the transmitter and the receiver need to be identical for the receiver to allow the decoding of the data sent.

III. Data pin configuration switches are used to send the binary data or the unique ID of the particular transmitter.

IV. The ID of the transmitter is manipulated and compared with the ID received at the receiver side.

XVIII. The purpose of the timer circuit is to allow transmission for a certain amount of time only rather than the transmitter being on continuously.

This reduces the power consumption and keeps refreshing the transmitted signal.

XIX. The timing of the timer it set based on considering both, the transmission delay as well as propagation delay.

- The Timer Circuit.

The transmission enable pin to be controlled by the timing circuit is active low. To control the duration of transmission, T2 as shown in figure 13 should be at least equal to transmission time + propagation time. Since radio waves propagate at the speed of light, the propagation delay time can be ignored.

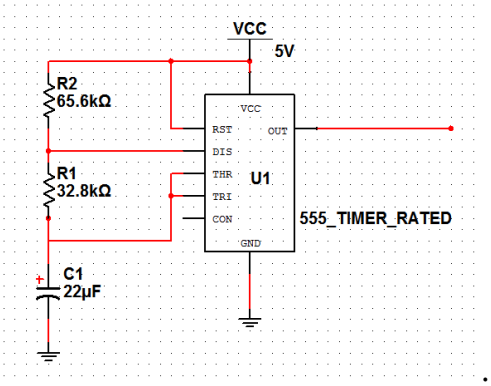

Figure 10: Timer circuit to control the Transmission duration.

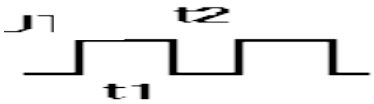

Figure 11: 555 timer timing profile.

The transmission time can be determined using timing diagram provided in the datasheet or simply scoping the Dout pin of transmitter and any data out pin of the receiver. Using the second method, the transmission time was found to be approximately 240ms, as shown in figure 12.

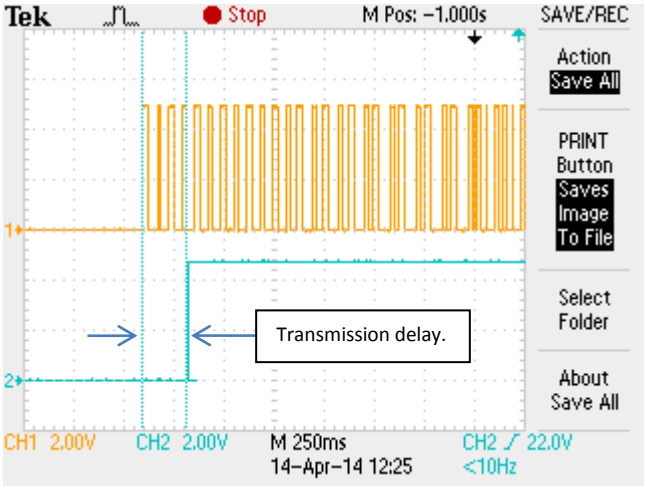

Figure 12: Transmission delay timing identification.

Giving some margin, T2 is set to be 0.5s, which is twice the transmission time and T1 is set to be 1.5s. Calculations for the circuit set up are as shown below.

T2= 0.5 sec, T1=1.5sec, C1= 22us (predefined).

T2= 0.5= 0.693(R1)(C1)

R1= 32.8kΩ

T1=1.5=0.693(R1+R2)(C1)

R2=65.6kΩ

- **The Receiver circuit**

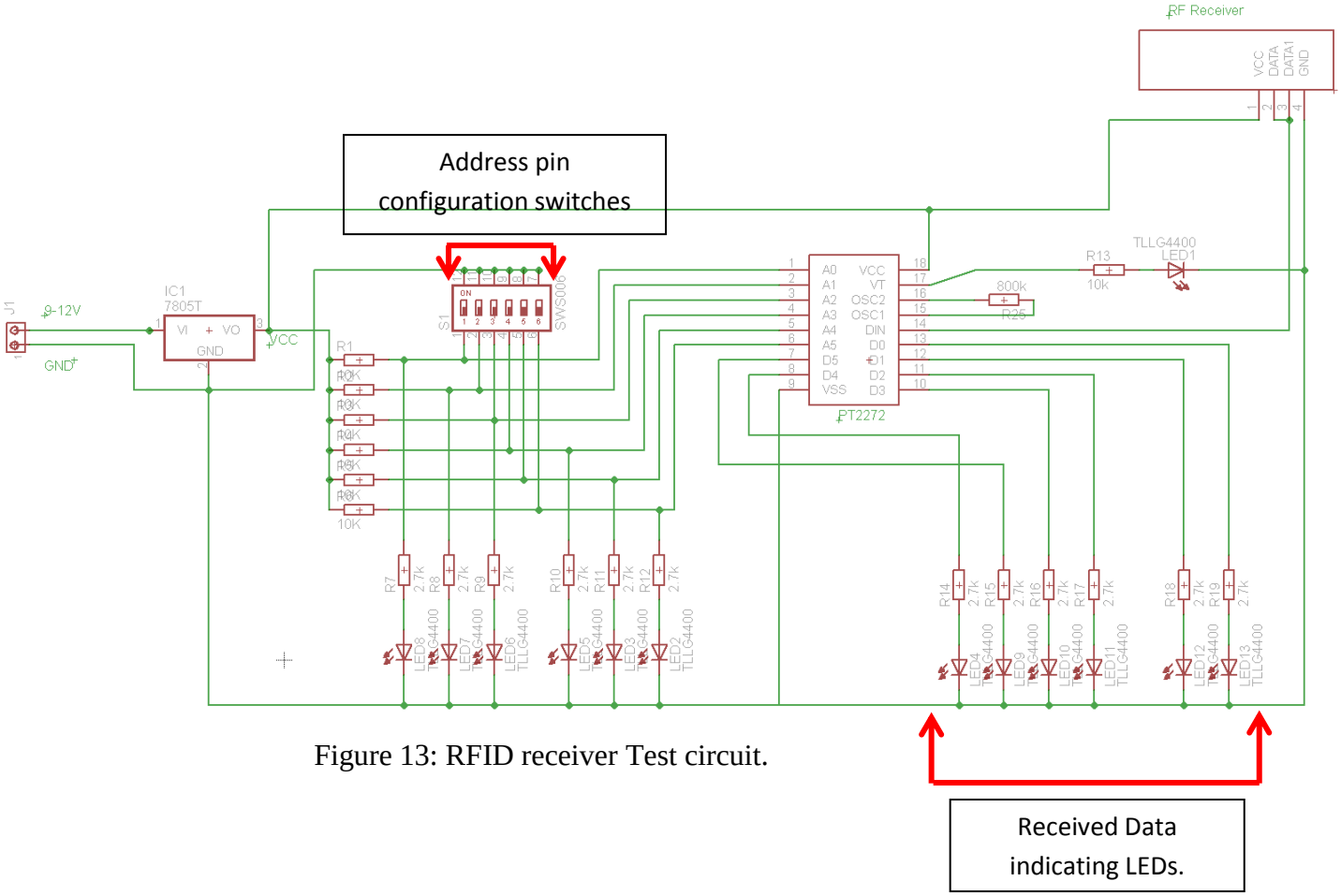

Figure 13: RFID receiver Test circuit.

I. The address pins of the receiver are made sure to be same as the transmitter; only then the data of the transmission is read by the receiver and output accordingly.

II. For testing purposes LEDs are used to indicate and confirm the data received is the same as transmitted. As for actual application the data pin are fed into the microcontroller as a unique ID to be processed.

3.3.3 To test the functionality of n24RFL01+ module.

As to test the functionality of nRF24L01+ not much of circuitry is required but, it requires a lot of programming to make it function. This module communicates with the microcontroller via SPI protocol, refer to appendices for the SPI initialization/configuration programming details. The schematic of the module is shown below.

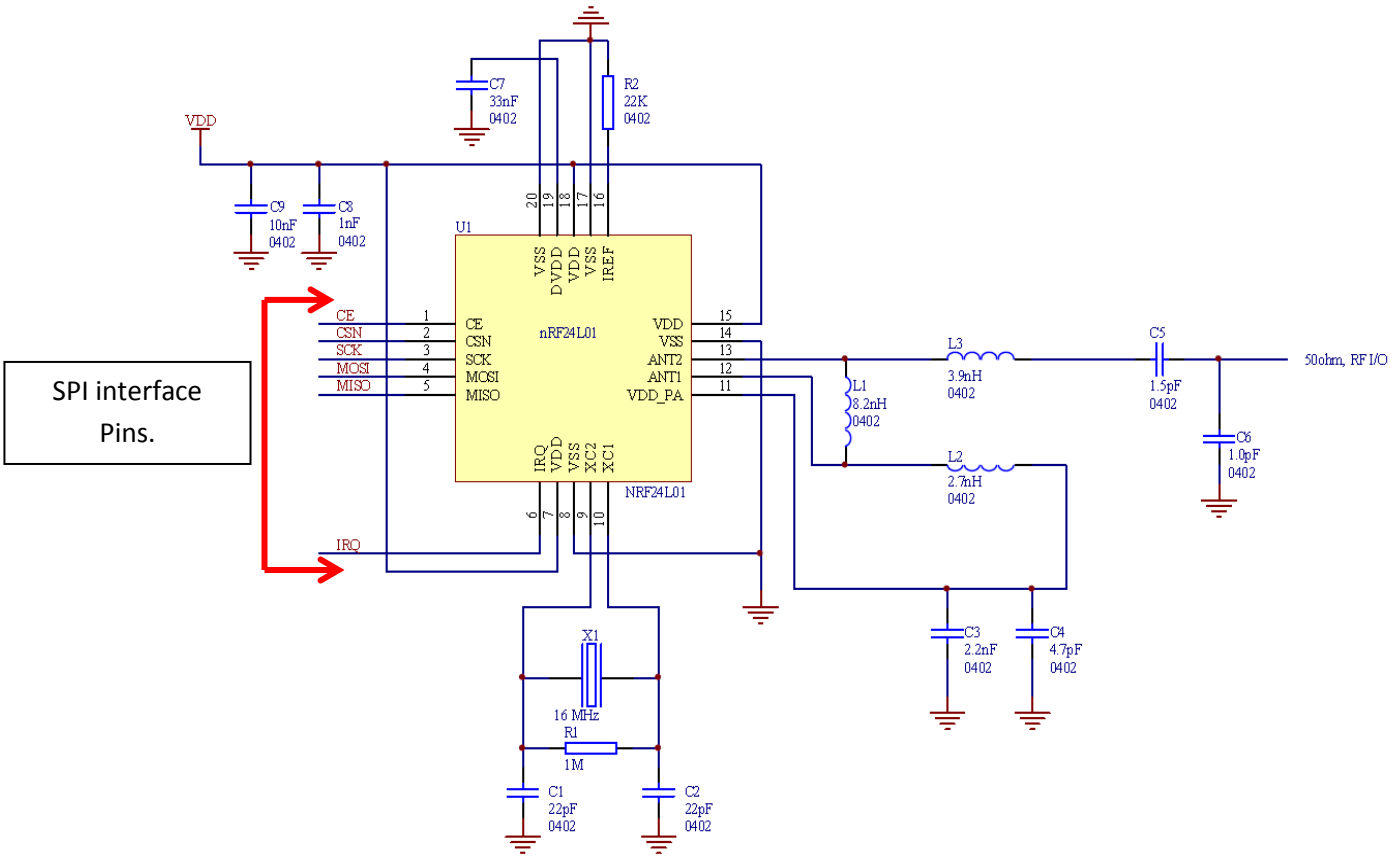

Figure 14: Schematics of nRF24L01+ module.

I. The pins of nRF24L01+ to be interfaced with the SPI pins of the microcontroller are shown in the figure above.

II. The following connection of pins is made.

TABLE 1: Interface connection between the FRDM-KL25Z and nRF24L01.

nRF24L01+	FRDM-KL25Z
CE	PTD0
CSN	PTD5
SCK	PTD1
MOSI	PTD2
MISO	PTD3
IRQ	PTA1

III. 2 of such setup are prepared, one to act as transmitter and another to act as receiver.

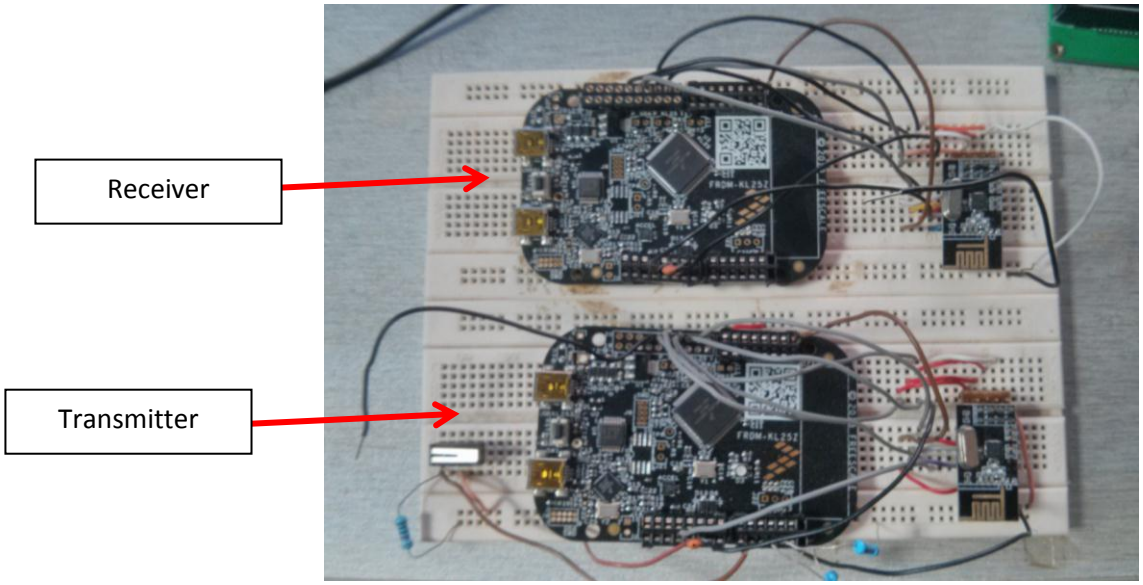

Figure 15: 2 sets of microcontroller and nRF24L01+. The top one acts as a receiver and the bottom one as a transmitter.

IV. The top set is programmed as receiver and the bottom one as transmitter.

V. The rf module needs to be configured properly to make it function. Refer to appendices for the configuration of rf module program.

VI. The transmitter test program macro is as shown below.

```
if(!sw1)
    {

      do
      {
      RED_LED_ON;                 //turn on red led
      transmitid(1,2,3);          //transmit 3 packet of integers, 1, 2 and 3.
      }
      while(!sw1);                //Continue transmitting as long as sw1 is
                                    pressed.
      TFC_Delay_10uS(100);        //delay 1ms
      }
```

VII. The program above transmits an id which is made up of 3 integers, 1, 2, and 3 continuously as long as sw1 is not released. And the red led on the freedom board acts as indicator.

VIII. The receiver test program macro is as shown below.

```
receiver();                                  //go to receiver mode

if ((x==1) && (y==2) && (z==3))              //compare the captured data, if same then
                    {
                            RED_LED_ON;          //turn on red led and turn off the rest
                            GREEN_LED_OFF;
                            BLUE_LED_OFF;

                    }
else if ((x==0) && (y==0) && (z==0))  //if not same then, do not turn on any led.
                    {
                             RED_LED_OFF;
                             GREEN_LED_OFF;
                             BLUE_LED_OFF;


                    }
NRFwrite_bit_write(7,7,1);                   // clear RX_DR flag.
```

The receiver receives the packet and compares if the received packet is the same as the sent packet, if it is, then to indicate it, the red led on the receiver board is turned on. Else, no led is turned on.

3.3.4 To Test for range of transmission accessibility.

I. Both modules should be able to cover a range of around 100ft, this is to identify the arrival of bus or station earlier so is to give the user some time to stop the bus.

II. To test for range, both modules were tested upon line-of-sight (LOS) area without any obstruction in between. This is to simulate the area bus station area which is open and with no obstructions.

3.4 Phase 2: The programming.

Since the Freescale Freedom (FRDM) Board is relatively new upon my grasp. Very little experience and knowledge about handling this board is in hand. Hence the methodology approach chosen is such that to enhance the knowledge of the board and the working environment of the board first. This could include to enhance the knowledge on e.g. Integrated Development Environment IDE to be used for programming, the programmer to load the program into MCU and the extra circuitry required to practically run and test the board.

Next, the programming is started by understanding and acquiring the basic skills that is to be able to control digital i/o pins (led blinking program). Prior to running this program, the i/o pins of the board are thoroughly understood and then the test circuit is designed. Next, how to use/program other peripherals are understood, most importantly the serial port. After that, the circuitry to interface the FRDM board with RFID Module and the sound module is designed and implemented. The project activity flow for phase 2 is as shown below.

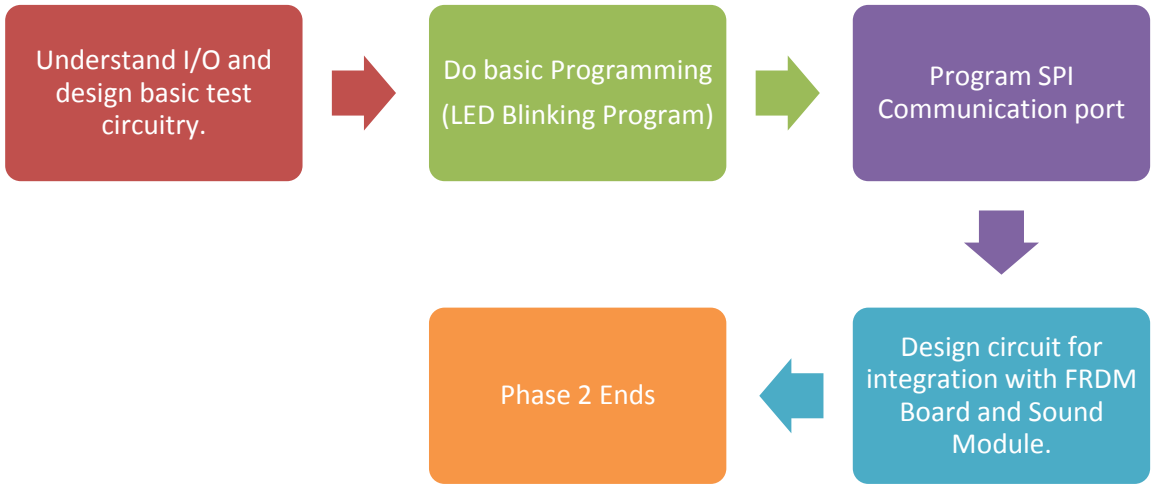

Figure 16: Project activity flow for Phase 2.

3.4.1 Testing the program functionality.

Before any of the modules are actually tested, it is made sure that the necessary skills to program them are in grasp. And the program responds to the commands as required. Most importantly the program to be tested is the SPI protocol. In true SPI communication protocol, it involve 4 pin mode, the pins are named as.

1. MOSI, which refers to “Master Out Slave In”, in this mode the microcontroller acts as the master. And this pin acts as the master data output pin.

2 MISO, which refers to “Master Input Slave Output”, this pin in master mode acts as the input.

3 CE, which refers to “Chip Enable” pin, is an output to enable the slave chip.

4 CSN, refers to chip Select Not, this is when pulled down is actually when the slave listens to the data sent by the master.

These pin must be configured correctly for the SPI port to function correctly as desired.

To test the SPI protocol communication,

1. The SPI port is configured.
2. The send some data using master output slave.
3. The output data is captured using oscilloscope to confirm its function.

To configure the SPI, the following code is used.

```
////////////////// SPI INIT START ////////////////////////////
void spi_init(void)
{
SIM_SCGC5 |= SIM_SCGC5_PORTD_MASK;              //Turn on clock to D module
 SIM_SCGC4 |= SIM_SCGC4_SPI0_MASK;              //Enable SPI0 clock

  SPI0_C1 = SPI_C1_MSTR_MASK | SPI_C1_SSOE_MASK;          //Set SPI0 to Master & SS pin
to auto SS
  SPI0_C2 &= ~SPI_C2_MODFEN_MASK;   //Master SS pin acts as slave select output
```

```
  SPI0_BR = (SPI_BR_SPPR(0x02) | SPI_BR_SPR(0x08));    //Set baud rate prescale divisor to 3 &
set baud rate divisor to 64 for baud rate of 15625 hz

  CE_pin_high;
          //RX mode
  CSN_pin_high;
          //SPI idle

  SPI0_C1 |= SPI_C1_SPE_MASK;    //Enable SPI0
}
////////////////////////// SPI INIT END ////////////////////////////
```

3.5 Phase 3: The Sound Module.

The Sound Module is used to play sound based on instructions received from the FRDM board. Hence, it requires the sound module to be compatible with the communication protocol peripheral available on the FRDM board. Based on our earlier research, it is known that there are a few types of communication protocol peripherals available that we can make use of, which include SPI, UART, I2C, and even PWM and ADC might be of some use.

So, first of all a sound module that is compatible to be interfaced with the FRDM board is identified. Next, the interfacing circuitry is designed and finally the module is programmed. It can be highlighted as below,

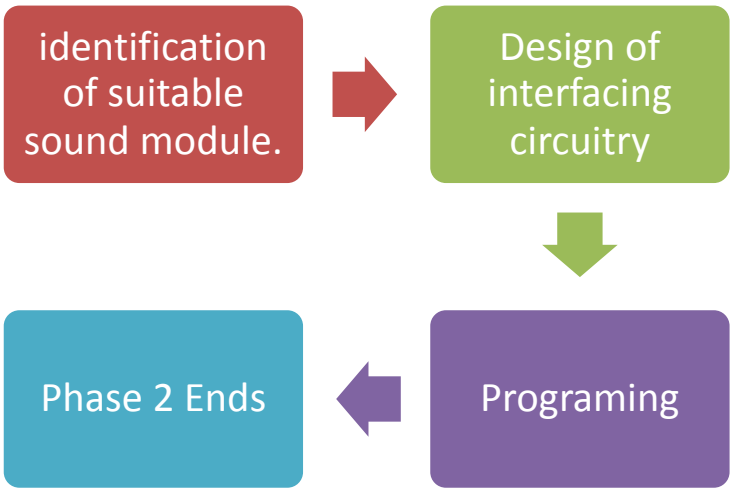

Figure 17: Project activity flow of phase 3.

3.5.1 Sound Module Selection.

The sound module chosen for this project is WTV020-SD module. This module can be used in many different modes such as MP3 mode, Key mode, Loop play mode as well as Two line serial mode. As for this project`s application, this module is used in Two line serial mode. In this mode the data is sent serially trough CLK/P04 and DIP05 pin as shown in pin out diagram of the module below.

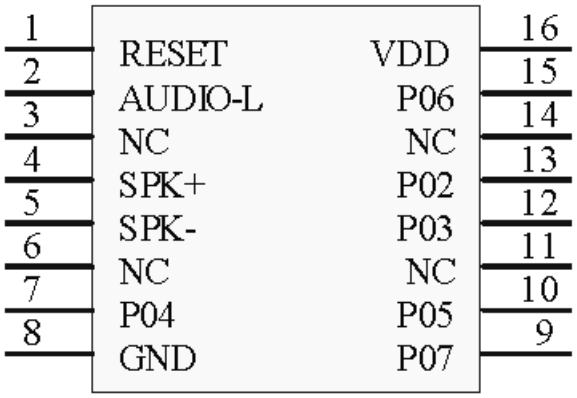

Figure 18: Pin out of WTV020-SD module.

This module finds its application in many situations, for instance, automobile, intelligent home system, learning tools, industrial controls toys and so on. Some of the key features are as below;

- Support 1GB SD card max. or SPI flash 64MB max.
- Support 4 Bit ADCPM format files.
- Sampling rate from 6KHz to 36KHz for AD4 voice format.
- Sampling rate from 6KHz～16KHz for WAV voice format
- 16 Bit DAC / PWM audio output.
- Key mode, MP3 mode and two line serial mode are optional .Can choose one of them
- Copy voice files to SD card by PC.
- Working voltage: DC2.7～3.5V
- Quiescent current:: 3uA

In serial mode, the voice is loaded in SD card and each voice file is saved as a 4 digit file, such as 0000.ad4, 0001.ad4 and so on. The maximum no. of sound files that this module support is 512; hence the name of the final file is 0512.ad4. In order to trigger a file to be played, the trigger data sent to the module is the binary of the corresponding file name. This is illustrated in table 3 below.

ADDRESSES	TRIGGER STATE	FILE NAME(.ad4)	TRIGGER DATA（BINARY）
ADDR 1	PLAY 1th GROUP VOICE	0000	0000000000000000
ADDR 2	2nd	0001	0000000000000001
ADDR 3	3rd	0002	0000000000000010
ADDR 4	4th	0003	0000000000000011
……	……	……	……
ADDR 509	509th	0508	0000000111111100
ADDR 510	510th	0509	0000000111111101
ADDR 511	511th	0510	0000000111111110
ADDR 512	512th	0511	0000000111111111

Table 2: Sound files naming and trigger address.

Since the data sent as trigger to the module is 16-bit long, while that the FRDM-KL25Z board is based on 32-bit ARM processor, “Masking” technique may be used in programming to reduce the 32-bit word to 16-bit word.

3.5.2 Application Circuit of the sound module.

The following circuit can be used when operating the module in Line serial mode. The P04 and P05 of the module are connected to the SPI Tx/Rx of the FRDM-KL25Z board. While that the clock of the module is made common with the clock of the FRDM-KL25Z board.

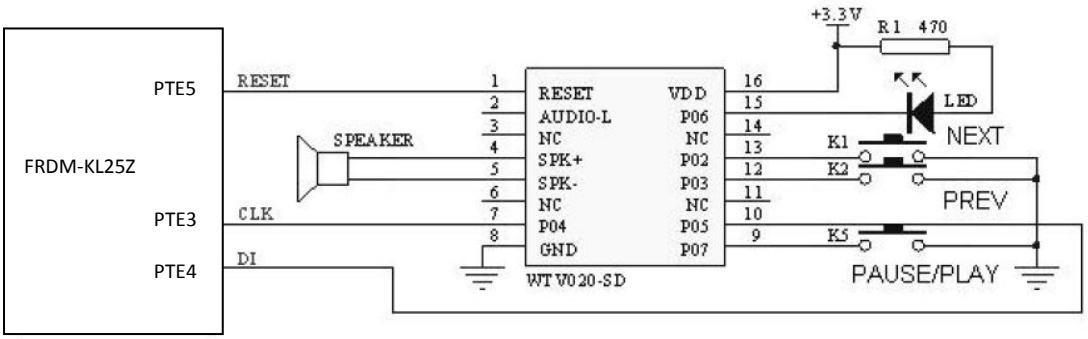

Figure 19: Application circuit diagram of sound module.

It can be noticed that this module operates at 3.3V but normally 5 volt DC is used, hence 2 serial diodes may be added to power input to lower down the voltage.

3.5.3 Testing the Sound Module.

Since this module operates in 2 line serial mode, it will be a waste of serial port to put this module on it. This is because a virtual 2 port communication port can also be created. And hence the saved SPI port can be utilized for other applications. The virtual SPI port should be able to depict the following timing diagram,

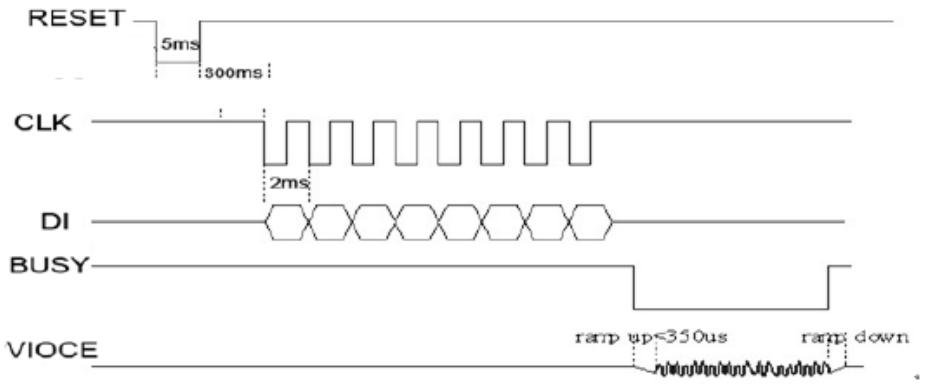

Figure 20: Timing diagram to send command to the sound module.

To test this module,

I. The circuit is constructed as shown in figure 19.
II. The pins are interfaced as below.

Table3: Interfacing Connection between Sound Module and FRDM-KL25Z.

SOUND MODULE	FRDM-KL25Z
CLK	PTE3
DI	PTE4
RESET	PTE5

III. The FRDM-KL25Z is programmed to as per timing diagram above.
IV. A certain name of the file is sent to play it.
V. Verify that the correct sound file is played based on the command sent.

The following code is used to produce the above timing diagram.

```
//////////////////////resetting the audio module start ////////////////////
void init_audio(void)
{

        sckon;                                          // clock on
        sdaoff;
///////////Reseting the module///////////
            rston;
            TFC_Delay_10uS(10000);
            rstoff;
            TFC_Delay_10uS(1000);
            rston;
            TFC_Delay_10uS(30000);

                    send_audio(VOLUME_7);       // to set to maximum sound



}
//////////////////////resetting the audio module END ////////////////////


///////////////////////////// send audio start//////////////////////////////
void send_audio(unsigned int value)
{

        unsigned int mask;
         sckoff;
         TFC_Delay_10uS(200);

          for (mask = 0x8000; mask > 0; mask >>= 1)
                    {
                    if (value & mask) sdaon;
                    else            sdaoff;

                    sckoff;

                    TFC_Delay_10uS(20);

                            sckon;

                            TFC_Delay_10uS(20);
                    }

         TFC_Delay_10uS(200);   //stop bit indication


}
//////////////////////////////// send audio end//////////////////////////////
```

3.6.1 The LCD and Keypad.

In case the nRF24L01+ transceiver is used as RFID, a programmer device to program this RFID is to be designed as well. This programmer can put onboard on each and every tag for the convenience of programming it any time or can be external to save cost. For the programmer, an LCD and a Keypad is used to perform the settings.

The LCD chosen is a 4X20 character LCD as shown below,

Figure21: 4X20 character LCD.

It's a 4 lines by 20 character LCD parallel interface LCD. It is based on Hitachi HD44780U controller. The LCD communicates with MCU using 11 pins, of which 8 pins are data pins used to send an 8 bit ASCII character. The pins and their functions are shown below.

Table4: LCD`s pins functions.

PIN NO.	SYMBOL	DESCRIPTION	FUNCTION
1	VSS	GROUND	0V (GND)
2	VCC	POWER SUPPLY FOR LOGIC CIRCUIT	+5V
3	VEE	LCD CONTRAST ADJUSTMENT	
4	RS	INSTRUCTION/DATA REGISTER SELECTION	RS = 0 : INSTRUCTION REGISTER RS = 1 : DATA REGISTER
5	R/W	READ/WRITE SELECTION	R/W = 0 : REGISTER WRITE R/W = 1 : REGISTER READ
6	E	ENABLE SIGNAL	
7	DB0	DATA INPUT/OUTPUT LINES	8 BIT: DB0-DB7
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	LED+	SUPPLY VOLTAGE FOR LED+	+5V
16	LED-	SUPPLY VOLTAGE FOR LED-	0V

3.6.2 The Keypad

The key pad to be used is the 4X4 numerical keypad as shown below.

Figure22: 4 X4 keypad.

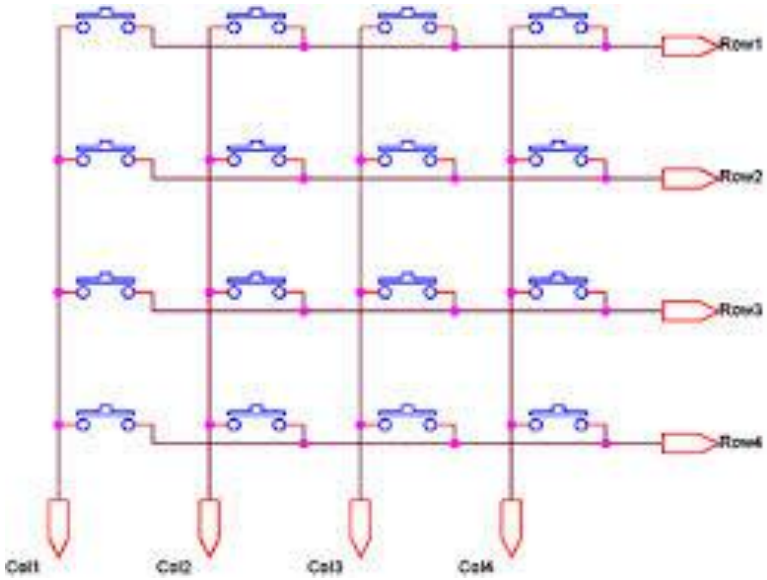

Figure23: interface connection of the keypad with MCU.

When interfacing the keypad and the MCU, the columns are configured as output while the rows act as input. The keypad is programmed in such a way that at a time only one col is turned on and the status of the row is checked to determine which key has been pressed.

3.6.3 Testing the Keypad and LCD

To test the LCD and keypad, both of them are connected to the MCU and any key on the keypad is pressed and the corresponding character should be displayed at the LCD.

I. The LCD is interfaced with the MCU using the following pins.

LCD	FRDM-KL25Z
DB0	PTE20
DB1	PTE21
DB2	PTE22
DB3	PTE23
DB4	PTE29
DB5	PTE30
DB6	PTC1
DB7	PTC2
RS	PTB1
R/W	PTB2
E	PTB3

Table5: Pin Assignment on MCU for the LCD.

II. Connect the Keypad to the MCU using the following pins.

Keypad	FRDM-KL25Z
Col1	PTC9
Col2	PTC8
Col3	PTA5
Col4	PTA4
Row1	PTA12
Row2	PTD4
Row3	PTA2
Row4	PTA1

Table6: Pin Assignment on MCU for the Keypad.

III. Program the MCU.

- For the program refer to appendices.
- Point 3 in the appendices explains the initialization and definition and other functions related to keypad and the LCD.

3.7 Final Phase.

The first 3 phases as a whole are to make sure all the necessary hardware are identified, complying to the specifications, and are configured correctly prior to proceeding to the final phase. This to simplify the development and configuration process as well as troubleshooting of the problematic component or program. Doing this also provides an effective way of determining problem at a lower level of development rather than troubleshooting at a higher level, which might pose higher level of difficulty in finding the solution.

The final phase combines all three phase. In the final phase all the modules are integrated hardware wise as well programming wise. The integrated hardware circuit is designed according to functionality required. While the firmware is designed to depict flow charts that program the MCU to have a required AI.

3.7.1 The Hardware Design

For this project 3 devices are designed. The first is the transmitter, second the receiver and notifier and the third is the programmer. The transmitter is the one that is programmed with a certain ID that corresponds to a certain bus or a bus station. From the testing done on the feasible type of RF modules to be used in this project, nRF24L01+ was identified to be the right candidate. The basic components required to develop the transmitter are a MCU and an nRF24L01+ transceiver unit. The setup of the component for the transmitter is as shown in the figure below.

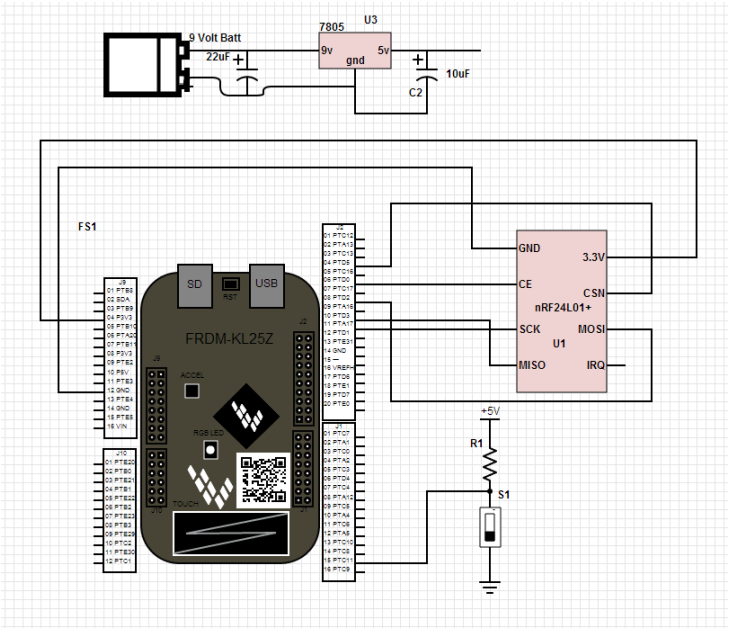

Figure 24: Schematics of Transmitter. (R1=10k).

The next device designed is the Notifier. The notifier functions to play a sound file upon receiving a certain information/command form the MCU. The notifier also consist of a MCU, an nRF24L01+ unit but in addition also has a sound module and relative integration circuitry to make run the sound module. The simplified schematics of the notifier is as shown in the figure below.

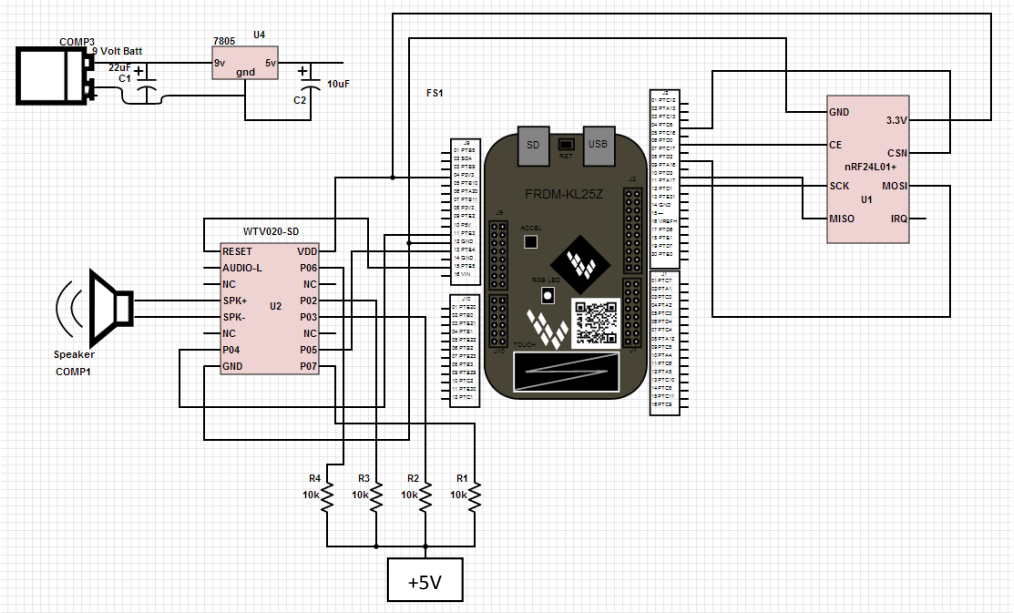

Figure 25: Schematics of Notifier.

The third device designed is the programmer. This device functions to program an id of tag. Assuming a tag is to be reprogrammed to another ID, it needed to be done by changing the ID in the program function and reprogramming the MCU using a PC. By having a programmer, this hassle can be avoided and the tag can be reprogrammed interactively. The schematic of the programmer circuit is as shown in the figure below.

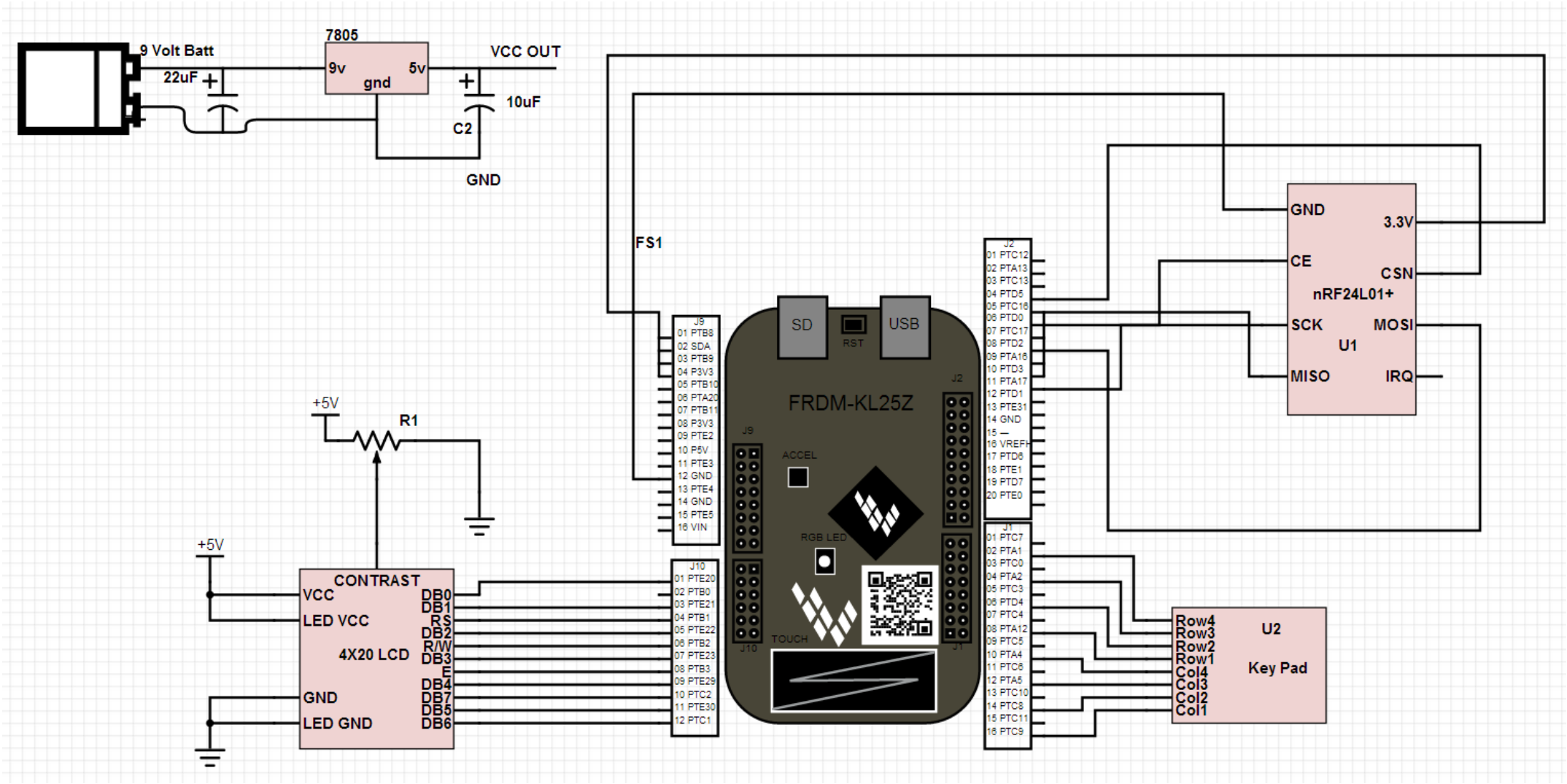

Figure 26: Schematics of Programmer.

3.7.2 The Firmware Design

In the earlier stages of the project, whereby the components are being configured and programmed and tested, the basic function are created as well. These functions are essential to ease the work during firmware programming, as the programmer only needs to “plug and play” those functions by calling them wherever necessary and need not to rewrite the functions again. Since there are three devices designed each device consists of its own firmware. The firmware flow charts are shown as below.

- Transmitter firmware.

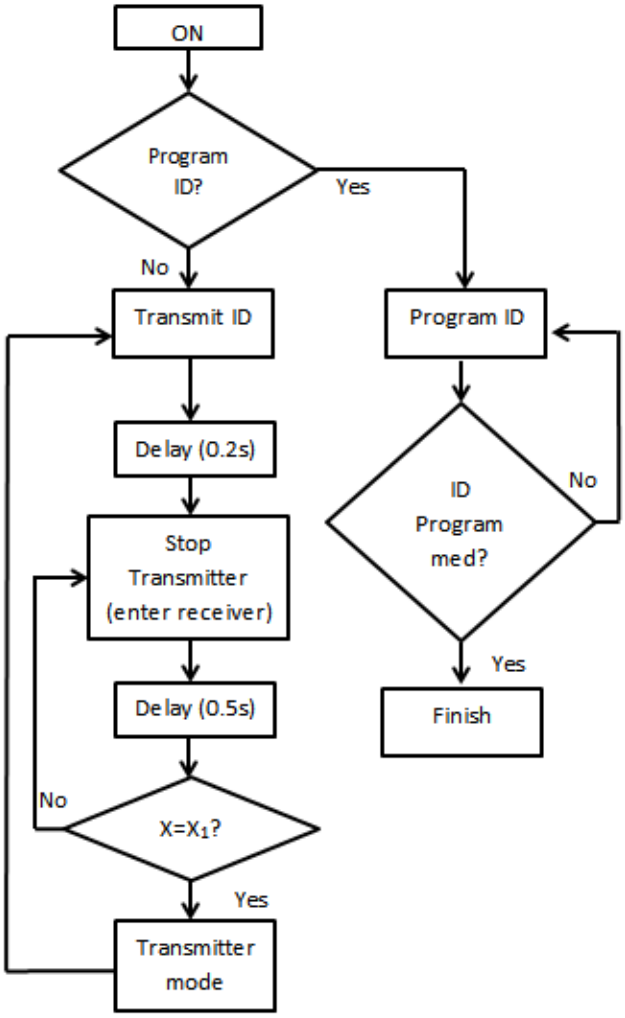

Figure 27: Firmware design of the transmitter.

The RFID tag runs in two modes, 1st the transmission mode, 2nd the program mode. When the tag is turned on, it checks which mode to enter. Assuming that it enters transmit mode, it transmits its ID for 0.2 seconds and then enters receiver function and start listening to instructions if there are any for 0.5seconds. If there are any

instructions, it performs the instructed task before continuing transmitting. Next, if it is in the program mode, it checks if the new ID is received, until the new ID is received or the tag is reset to transmit mode, it keeps on looping in this function.

For the program refer to “transmitter.c” code in the appendices.

- Programmer firmware.

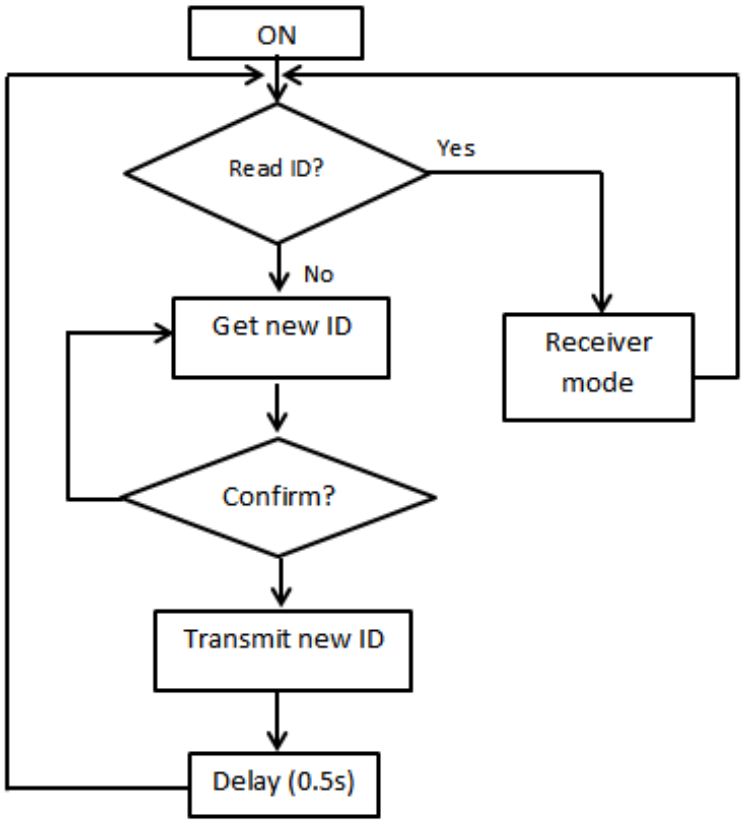

Figure 27: Firmware design of the RFID tag ID programmer.

The programmer has 2 functions, 1st to read the tag and the 2nd to program the tag`s ID. When turned on, it checks which operation to perform. When writing a tag, it gets the new ID and confirms it. Once confirmed, the new ID is transmitted and programmed to the Tag. While in the Tag Read function, the programmer enters the receiver mode, reads the tag`s id and displays it on the LCD. The “prog.c” as indicated in the appendices is loaded to the MCU. In the appendices “prog.c” file explains the coding details of the programmer firmware.

- Notifier firmware.

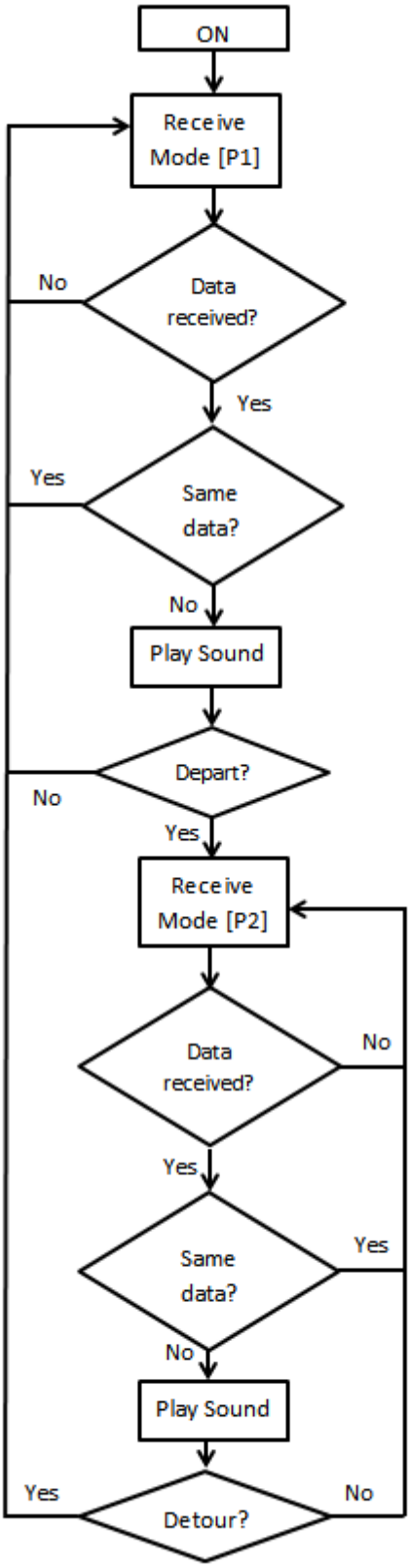

Figure 28: Firmware design of the Notifier.

The notifier when turned ON, enters receive mode and looks for new ID. When new ID is received, it compares the received ID with the previously received ID. This is to avoid the notifier from notifying the same ID again and again. If the ID received is not the same as previous data, the notifier compares the ID with the data base and notifies the user accordingly. Next, it asks the user whether the user would like to depart on the notified bus or not. If the user chooses to depart, the notifier enters the receive mode 2, whereby now it checks for the next upcoming bus station and notifies accordingly. If the user chooses not to take the notified bus, the notifier goes back to the receive mode and detects for new bus ID`s.

To program the MCU with above Flowchart, "Notifier.c" file as shown appendices is loaded.

- **Steps on loading a program file on MCU.**

For the flowcharts above, the program files specified and provided in the appendices are main program for that particular device firmware only. In fact, there exist other supporting files as well. But the initialization and configuration of all three devices is made common and program differs from each other by its main program only. This is to simplify the progress so as not requiring each device firmware to be initialized and programmed differently.

The list of main and supporting files is as shown in figure below.

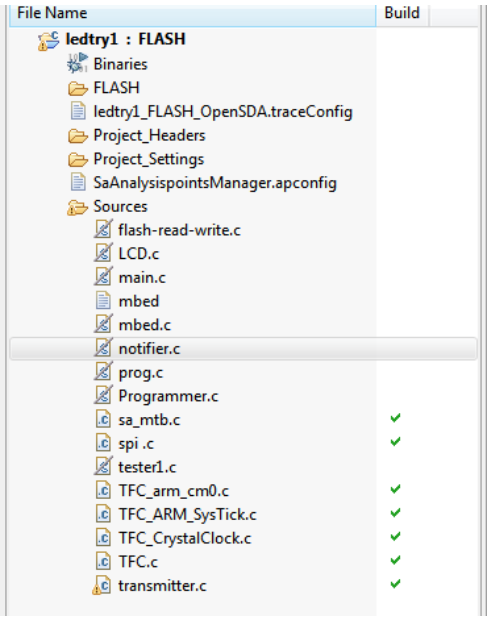

Figure 29: .c files in the project.

When loading a firmware to a device the following steps are taken.

I. Check the firmware file to be loaded.
II. Uncheck the other firmware files.
III. Load the program into the MCU.

In this project there exist 3 firmware files, “Notifier.c”, “transmitter.c” and prog.c. When loading for instance, “Notifier.c” file into the MCU, the “Notifier.c” should be checked and the “transmitter.c” and “prog.c” files should be unchecked. Same steps apply if the MCU is to be loaded with other firmware files.

3.8 Key Milestone and Project Gantt Chart

3.8.1 Key Milestones.

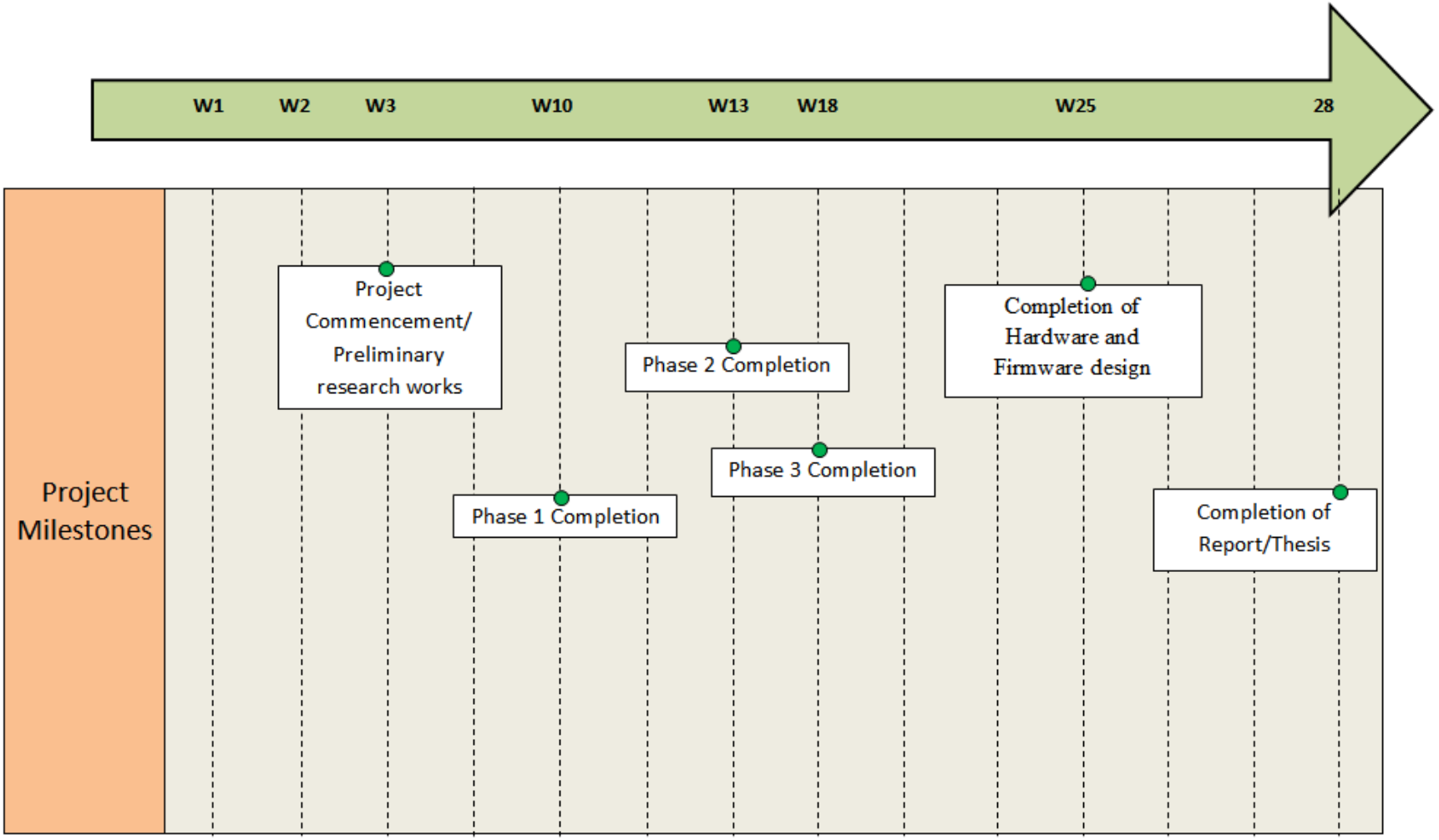

Figure 30: Key Milestones of the project.

3.8.2 Gantt Chart Report Submission Activities

No	Details \ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Selection of Project Title															
2	Preliminary Research Work and Literature Review															
3	Submission of Extended Proposal															
4	Oral Proposal Defense Presentation															
5	Continue on project work and prototyping															
6	Preparation of Interim Report															
7	Submission of Interim Draft Report															
8	Submission of Interim Final Report															

Suggested milestone

Process

No	Details / Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14	15	17
1	Project Work Continues																	
2	Submission of Progress Report																	
3	Project Work Continues																	
4	Pre-SEDEX																	
5	Submission of Draft Final Report																	
6	Submission of Dissertation (soft bound)																	
7	Submission of Technical Paper																	
8	Viva																	
9	Submission of Project Dissertation (Hard Bound)																	

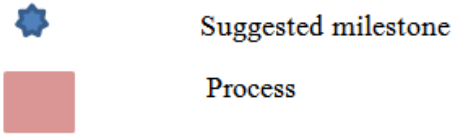

Figure 31: Gantt Chart Report Submission Activities

3.8.3 Gantt Chart Project Related Activities.

No	Details	1 (13-19 Jan)	2 (20- 26 Jan)	3 (27Jan- 2Feb)	4 (3 -9 Feb)	5 (10-16 Feb)	6 (17-23 Feb)	7 (24Feb- 2Mar)		8 (3-9 mac)	9 (10-16 mac)	10 (17-23 Mac)	11 (24-30 Mac)	12 (31 Mar - 6 Apr)	13 (7-13 Apr)	14 (14-21 Apr)	15 (19-25 may)
1	Selection of Project Title																
2	Preliminary Research Work and Literature Review																
	RFID related activities (Phase1)																
3	Selection of RFID module																
4	Supplier Search and Sourcing																
5	Design of RFID test Circuit																
6	Assembly and Testing of RFID circuit																
	Sound Module related activities (Phase 3)																
7	Selection of sound module																
8	Supplier search and Sourcing																
9	Determine interfacing circuit																
	FRDM-KL25Z related activities (Phase 2)																
9	Sourcing of FRDM-KL25Z board																
10	Installation of programming related software and circuitry preparation of freedom board																
11	Basic Programming understanding																
12	Preparation for final exams FYP Idle																

Figure 32: Gantt Chart project related activities for FYP1.

No	Details	1 (19-25May)	2 (26May-1June)	3 (2-8 June)	4 (9-15 June)	5 (16-22June)	6 (23-29June)	7 (30June-6 Jul)		8 (7-13Jul)	9 (14-20Jul)	10 (21-27Jul)	11 (28Jul-3Aug)	12 (4-10Aug)	13 (11-17Aug)	14 (18-24Aug)
	Project Prototype works.															
1	Integrating all three phase.															
2	Programming.															
3	Testing and troubleshooting.															
4	Circuit fabrication															
5	**Project Add-on.															

Figure 33: Gantt Chart project related activities for FYP2.

3.9 Tools

3.9.1 Software

Table7: Software used in the project.

No.	Name	Description
1	Microsoft Office	• For writing report • Other paperwork purposes
2	EndnoteX7	• References Purposes
3	CodeWarrior	• To write programming code
4	GCC compiler	• Code Compiler
5	EAGLE	• To draw Schematics
6	SchemeIt	• To draw Schematics

3.9.2 Hardware

Table8: Hardware used in the project.

No.	Name	Description
1	Freescale Freedom Platform FRDM-KL25Z	MCU to Processes appropriate I/O`s.
2	RF Module (RF_RX_315 ,315MHz RX/TX) / nRF24L01+	To make wireless communication.
3	Sound module (WTV020-SD Module)	To play sound announcement.
4	4x20 LCD	To display the ID for RFID programming.
5	Keypad	To interact with MCU during ID programming.
6	Oscilloscope	To troubleshoot and verify circuit/program.
7	Multimeter	To troubleshoot and verify circuit/program.
8	Miscellaneous	Soldering tools, other hardware.

CHAPTER 4
DISCUSSION AND RESULTS

4.1 Results

4.1.1 RF_RX_315, 315MHz RX/TX, RFID transmission testing.

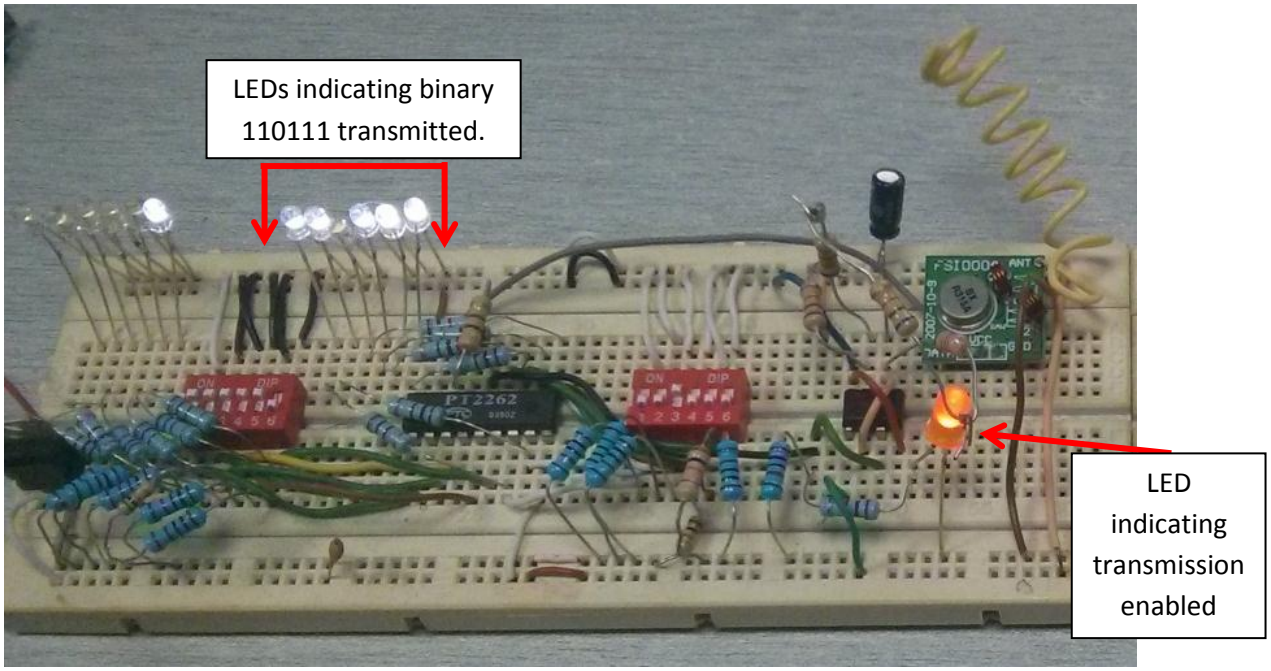

Figure 34: Actual Transmission Circuit.

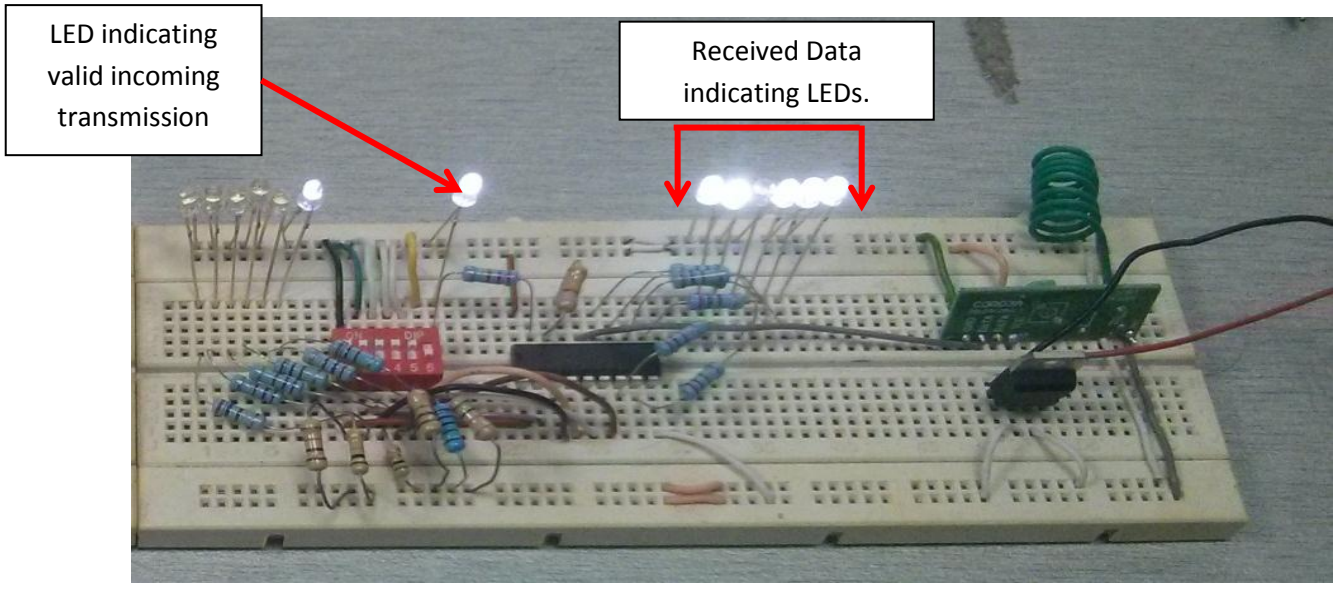

Figure 35: Actual Receiver Circuit.

From the figure above it can be seen that a binary of 110111 was transmitted and the same binary was received at the receiver. This confirms that effective ID transmission has occurred.

Then another set of binary was sent to simulate the arrival of another bus. The results are shown below.

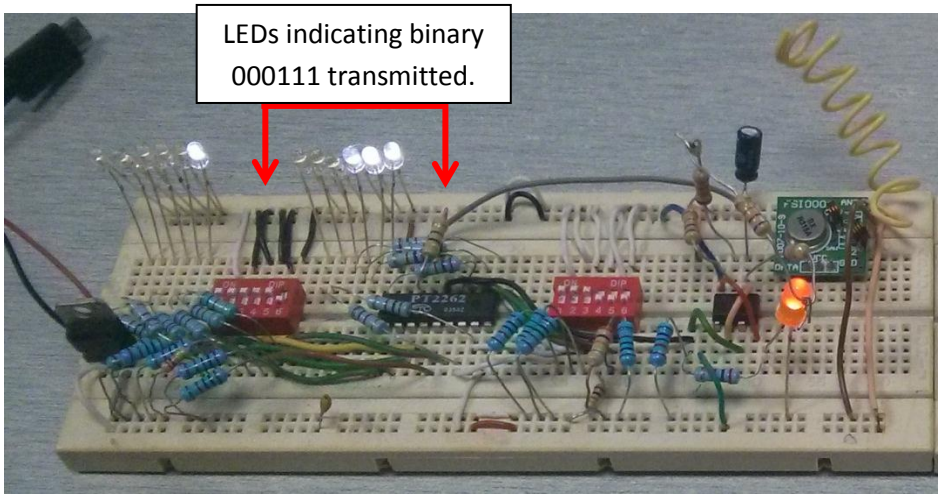

Figure 36: Transmission of 000111 binary over RF.

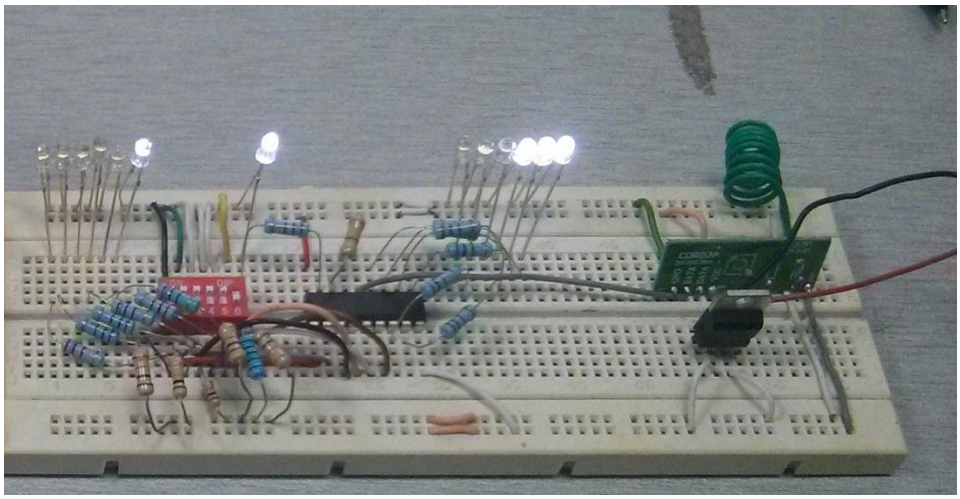

Figure 37: Receiving of 000111 binary at the receiver.

Similarly, different types of buses can have their own unique ID.

4.1.2 nRF24L01+ transmission testing.

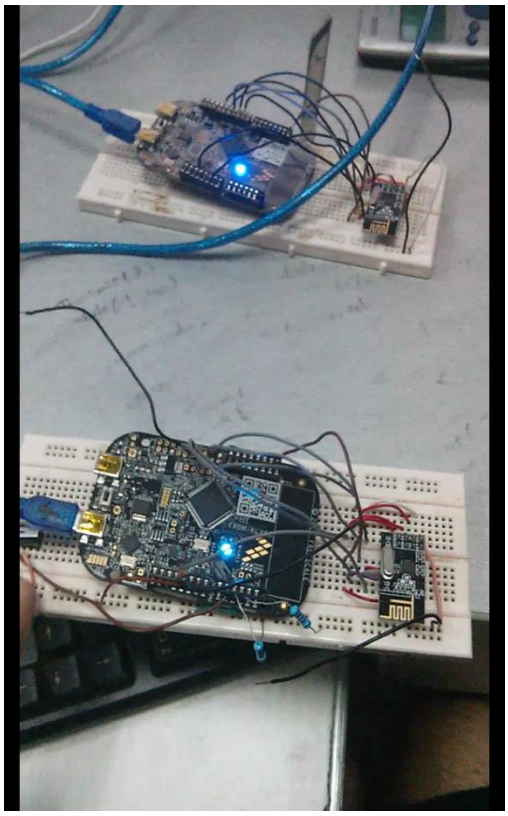

Figure38: No transmission, led =blue

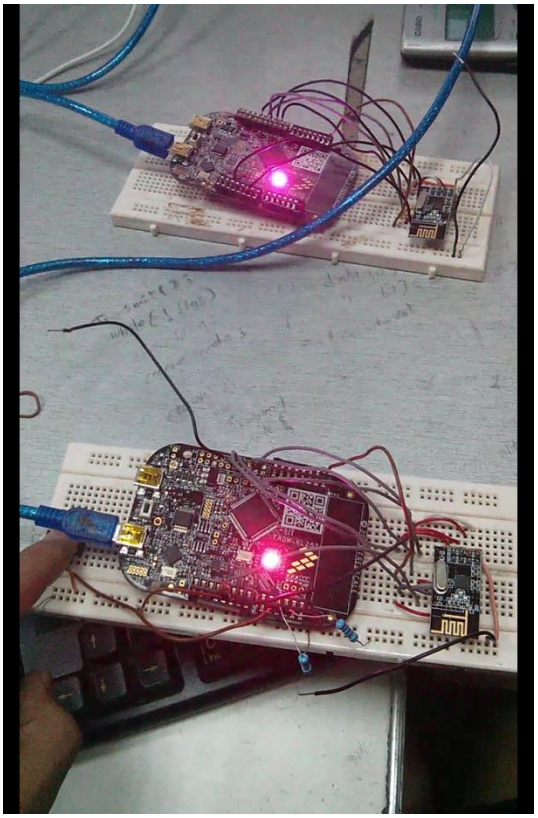

Figure39: Transmission successful, Led=pink.

Figure 1 show that the blue LED is turned on when the transmission button is not pressed. While that when the transmission button is pressed, the LED turns pink. This confirms that successful transmission has occurred and the id read by the receiver matched the id sent by the transmitter.

The id detected by the receiver can also be confirmed by the going into the debug mode of CodeWarrior and reading the payload of the receiver. This is shown in the figure below.

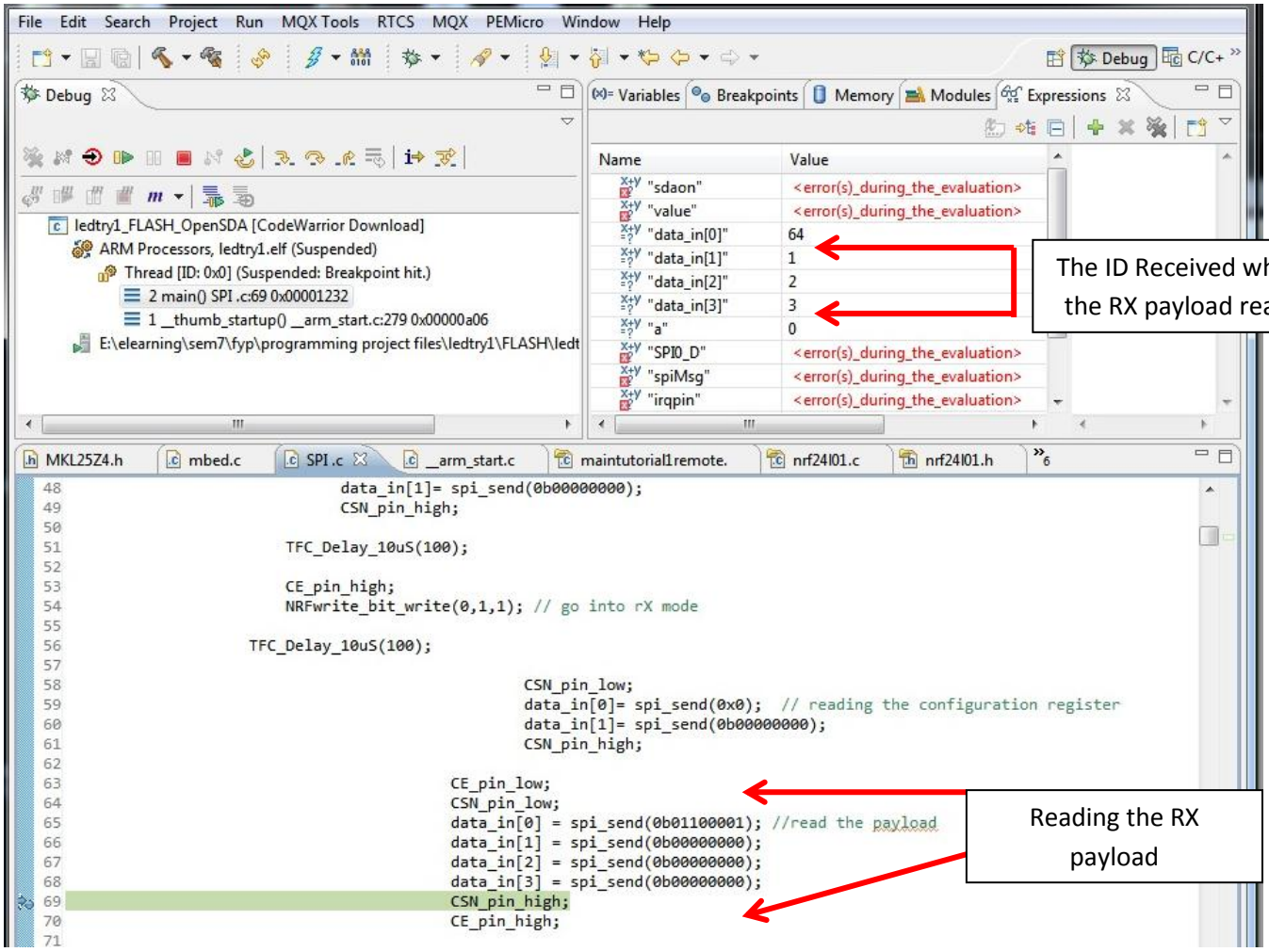

Figure40: reading the RX payload.

4.1.3. Test for Detection Range.

Both of the modules above were also tested to determine the range of detection. For RF_RX_315 ,315MHz RX/TX module the detection range depends on some variables such as the voltage supplied, the antenna length, physical obstructions and so on. The test was performed with variables fixed as below;

Voltage supplied to the transmitter : 5 V

Antenna length : 24cm (representing quarter wave length)

Physical obstructions : Line Of Sight (no obstruction).

The results showed that, the detection range was up to approximately 150feet. This is more than the detection range set in the requirements above, which is 100feet.

While as for nRF24L01+, it has an onboard antenna and fixed voltage. Under same environmental conditions, the nRF24L01 also has a detectability range of almost 150ft as well.

4.1.4 Discussion on RFID

As for the RF_RX_315,315MHz RX/TX later in the testing it was realized that it have some drawbacks, such as, limited number of ID`s available, most importantly it can give false information by detecting the bus on the other side of the road, not expandable and static design since no additional Artificial Intelligence can be added and so on.

While that these drawbacks can be overcome by using nRF24L01 based RFID. In addition to that, since it's a transceiver, a lot of other practical beneficial features can be achieved. Hence, RFID based on nRF24L01+ is chosen to be most feasible.

4.2 Testing results for SPI port functionality program.

The figure below shows the screen shot of the signal captured using oscilloscope. The signal on channel 2 represents the clock signal while channel 1 represents the data sent over MOSI pin. The figure below confirms that the SPI port is programmed correctly and the data sent is correct as well.

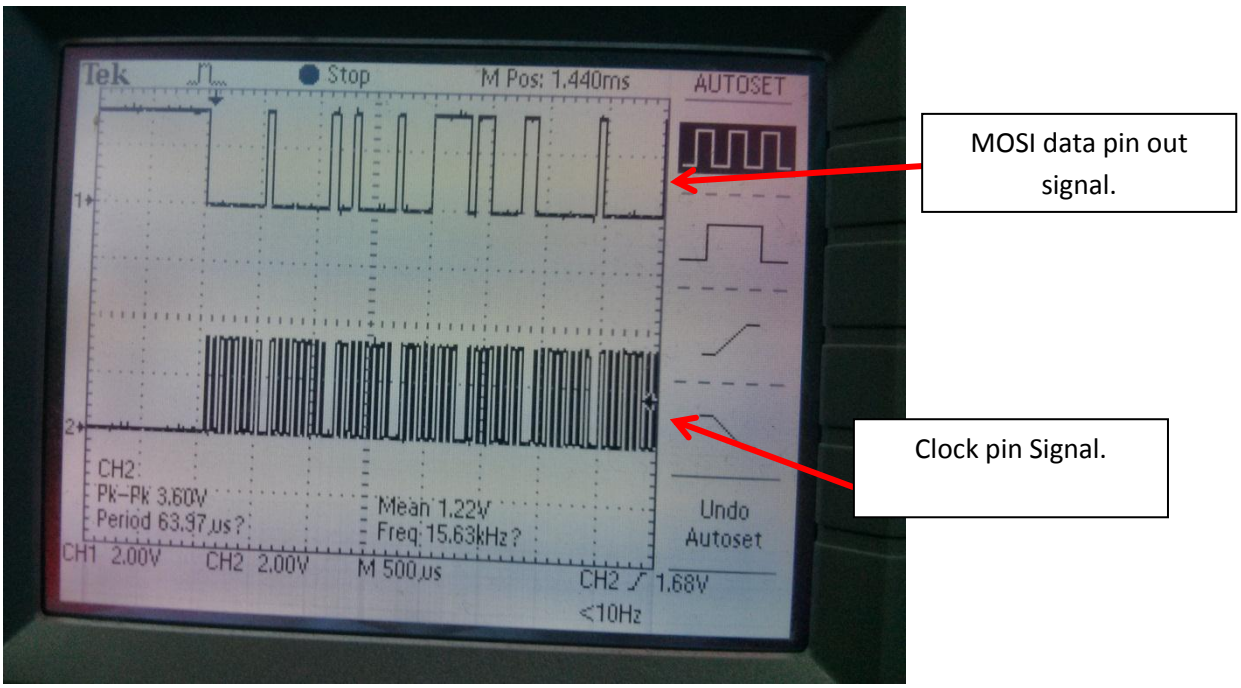

Figure41: signal captured at the clock and MOSI pin of the SPI port.

4.3 Testing for sound module

Running the code revealed the below signals, the top signal represent the signal captured at the clock pin, while the bottom signal represents the command or the name of the file sent to be played.

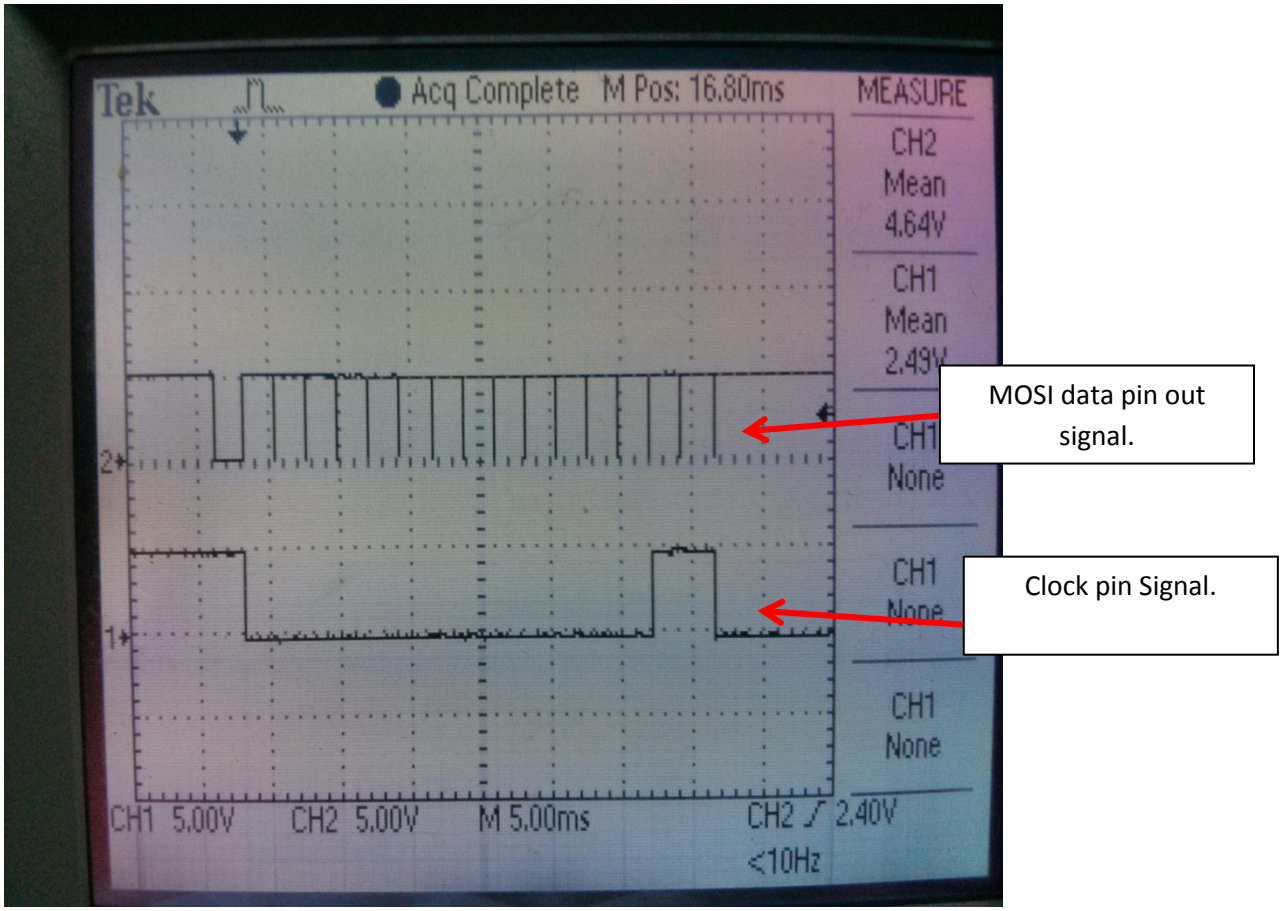

Figure 42: Timing diagram created by the code run for the test.

For the figure above it can be observed that each command is equivalent to 16 clock cycles, and the file name sent is 0b0000000000000111. The resulted sound played as a response to the command above was correct. Hence, confirming that the code works.

4.4 Results of testing the keypad and the LCD.

After connecting the pins and uploading the code, the pins 1→ * were pressed and the LCD displayed the characters accordingly. It can been seen as per photos below.

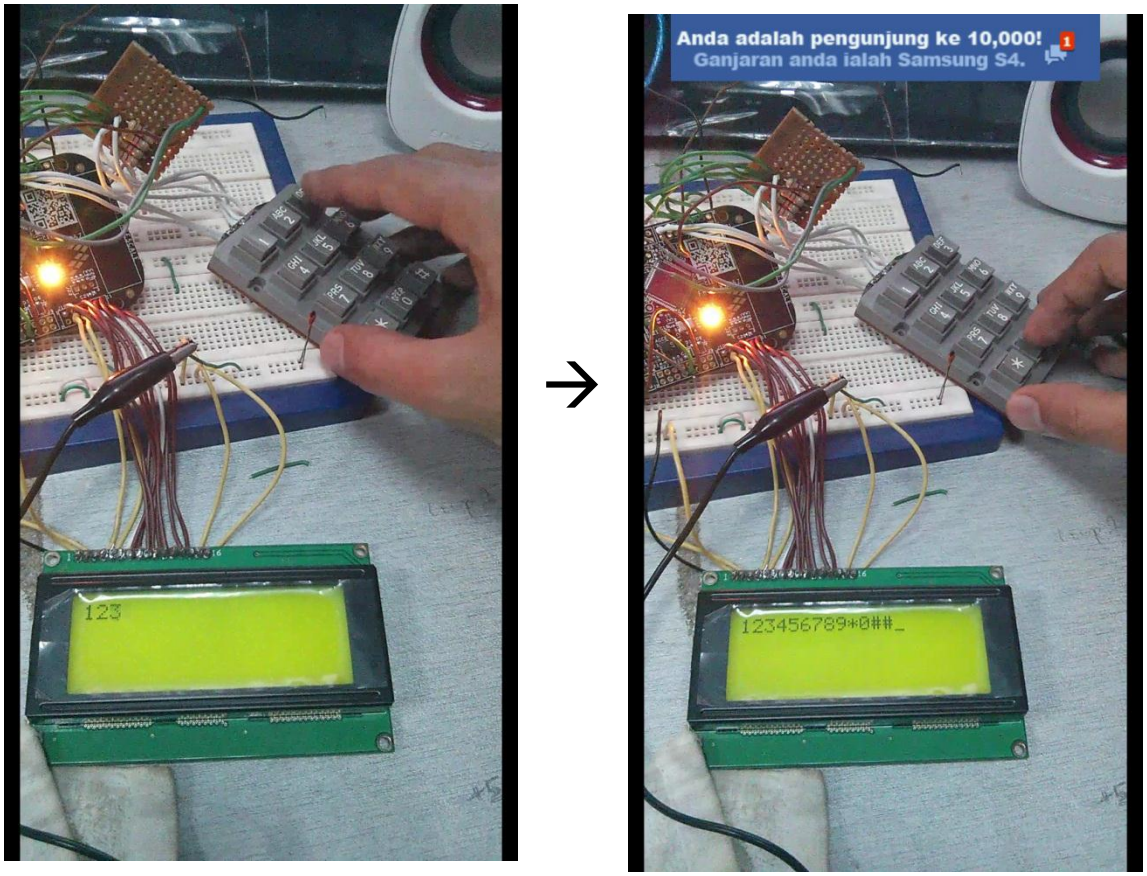

Figure43: The characters displayed on LCD as the keypad buttons are pressed.

The figures above confirm that both, the LCD and the keypad are working properly as commanded by the code.

4.5 Final Phase Results

4.5.1 The Hardware Design.

In this project, three devices are designed based on the schematics shown in Chapter 3. The figures below shows the resultant circuits

- Transmitter

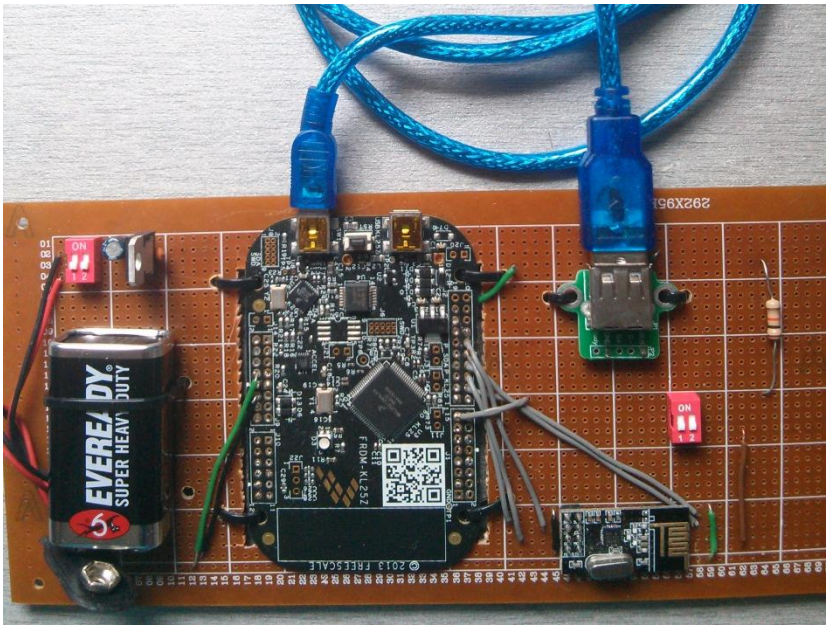

Figure 44: Hardware Prototype of Transmitter.

- The Notifier.

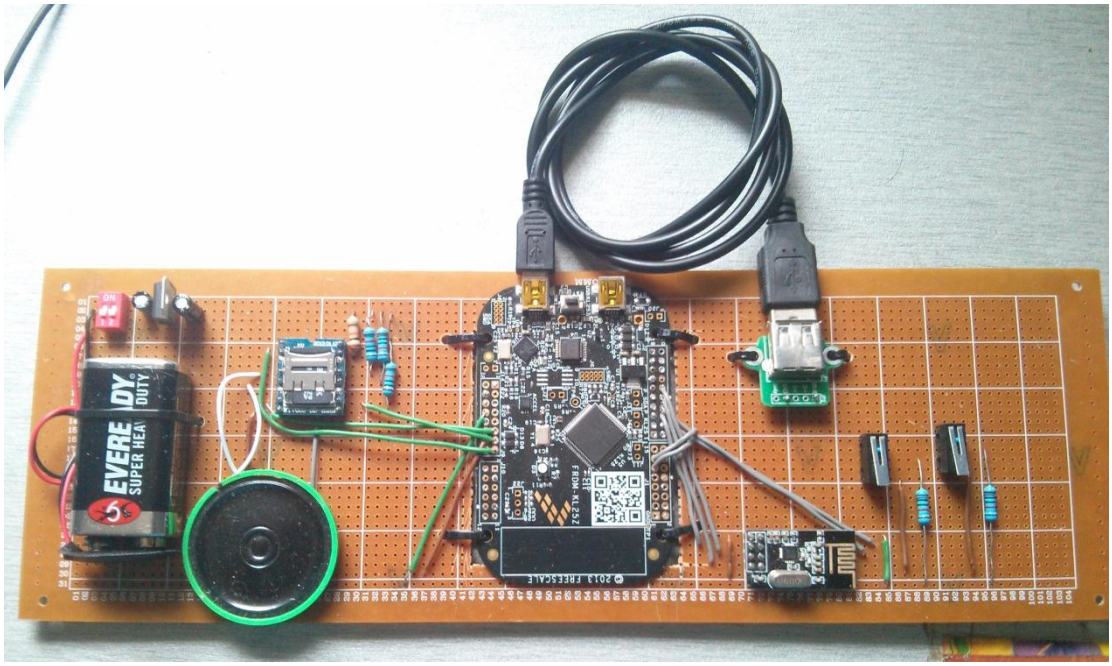

Figure 45: Hardware Prototype of the Notifier.

- The Programmer

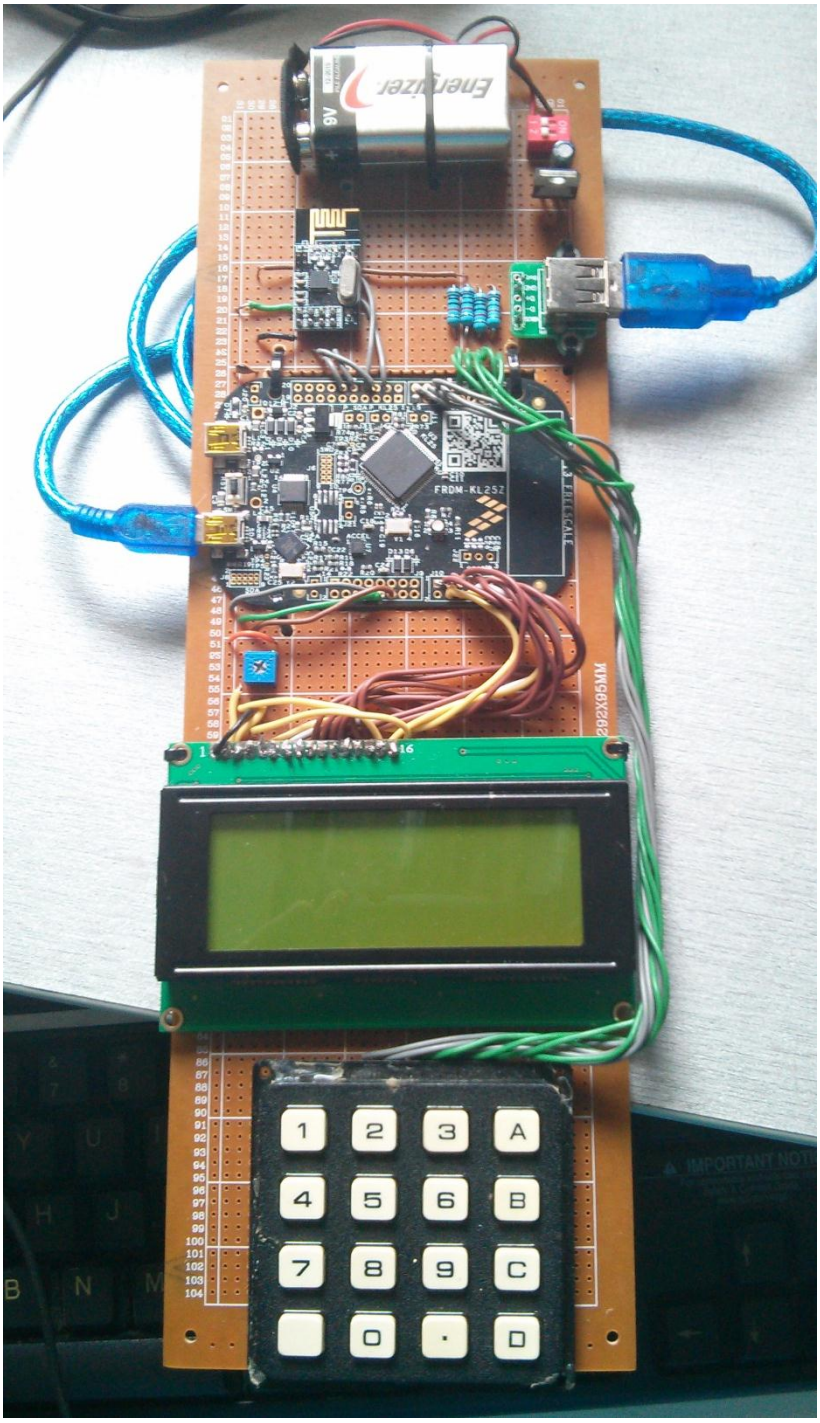

Figure 46: Hardware Prototype of the Programmer

.

4.5.2 The Firmware Design.

- **The Programmer**

  The read and write interface of the programmer were tested, and proved to be working as shown in figure below.

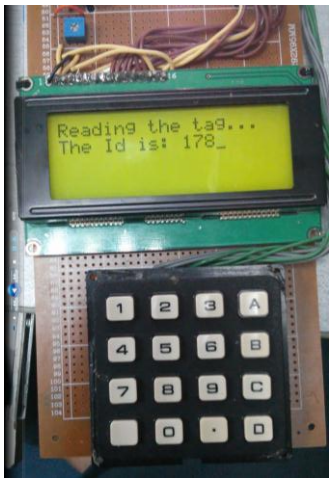

Figure 47: Initial ID of the tag read (178).

Figure 48: The tag is programmed with new ID, 456.

Figure 49: New tag ID read, 456.

- **Transmitter and the Notifier.**

  The transmitter and the notifier are tested together. The notifier was turned on first, then the transmitter. Once the transmitter was turned on, it was detected by the notifier and the relative sound files were played and it followed as per the flow chart. To confirm whether the ID sent by the transmitter is correct or not, the ID was read using the Programmer and it gave positive results.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

In this project, a system based on RFID and Arm processor was developed. The system consisted of 3 devices, a transmitter which carries a unique ID of a bus and a bus station, a Notifier, which detects the ID of Bus or bus stations and gives out a sound notification and finally a programmer, which functions to program RFID tag.

In this project, a system based on RFID and Arm processor was developed. The system consisted of 3 devices, a transmitter which carries a unique ID of a bus or a bus station, a Notifier, which detects the ID of Bus or bus stations and gives out a sound notification and finally a programmer, which functions to program RFID tag. The results as discussed in the previous chapter show that all the components are working fine when tested individually and after being integrated and programmed by following the firmware discussed in methodology.

One of the advantages of using a transceiver-MCU combination is that it is not only limited to its current functionality and can be expanded by integrating other modules as well. This combination is in fact having a wireless connection between microcontrollers operating at 2Mbps. Which means that one microcontroller can be used to control the functionality of another microcontroller wirelessly. This opens up the possibilities of making the public transport system much more intelligent as a whole, since more modules can be added to the current system. At the end of day, an intelligent public transport system whereby busses would be talking to each other, busses would be talking to bus stations and bus stations might be talking to other bus stations, transferring various information can be created.

REFERENCES

1. Grove, S., *mbed FRDM KL25Z.* 2013 September 19.
2. Pathak, R. and S. Joshi, *Recent Trends in RFID and a Java based Software Framework for its Integration in Mobile*. Acropolis Institute of Technology & Research; Shri Vaishnav Institute of Technology & Science: Indore, M.P., India.
3. Iverson, W. *Enhancing Worker Safety with Real-time Personnel Location Systems*. 2011 April 11 2014 February 16]; Available from: http://www.automationworld.com/safety/enhancing-worker-safety-real-time-personnel-location-systems.
4. Violino, B. *The Basics of RFID Technology*. 2005 Januray 16 2014 February 16]; Available from: http://www.rfidjournal.com/articles/view?1337.
5. Bonsor, K. and W. Fenion. *How RFID Works*. 2007 November 5 2014 February 16]; Available from: http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid3.htm.
6. Craske, S., *ARM Cortex-R architecture.* 2013 October.
7. Morrow, J. *Advances in Technology*. 2010 February 20 2014 February 16]; Available from: http://www.studymode.com/essays/Advances-In-Technology-Essay-347617.html.
8. Mouser. *Freescale FRDM-KL25Z Freedom Development Platform*. 2014 February 17]; Available from: http://www.mouser.com/new/Freescale-Semiconductor/freescale-freedom-kl25z/.
9. Ellis, J. *What is a PA System?* 2014 February 11 2014 February 19]; Available from: http://www.wisegeek.org/what-is-a-pa-system.htm.
10. Boaventura, A.J.S. and N.B. Carvalho, *A Batteryless RFID Remote Control System.* Microwave Theory and Techniques, IEEE Transactions, 2013. **61**(7): p. 2727-2736.
11. Layton, J. *How Remote Controls Work*. 2005 November 10 2014 February 18]; Available from: http://electronics.howstuffworks.com/remote-control2.htm.
12. Cisco, *Wi-Fi Location-Based Services 4.1 Design Guide*. RFID Tag Considerations. 13.
13. Impinj. *The Different Types of RFID Systems*. 2014 February 19]; RFID Frequencies]. Available from: https://www.impinj.com/guide-to-rfid/the-different-types-of-rfid-systems.aspx.
14. Griebenow, A., *Active Systems.* A Guide to Best-Fit Applications for Active RFID System Alternatives, 2008: p. 11.
15. mbed. *mbed FRDM KL25Z.* 2013 2014 February 19]; Available from: https://mbed.org/handbook/mbed-FRDM-KL25Z.
16. ISTQB. *What is Waterfall model- advantages, disadvantages and when to use it?* 2012 January 11 2014 February 19]; Available from: http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/.

APPENDICES

1. SPI Communication Code

```
int spi_send(int spiMsg)
{
  int i=0;                                    //FGPIOD_PCOR |= (1 << 0);  //Set SS low

 while(!(SPI_S_SPTEF_MASK & SPI0_S))
 {
   asm("nop");                        //While buffer is not empty do nothing
 }

 SPI0_D = spiMsg;                     //Write char to SPI


 while(!(SPI0_S & SPI_S_SPRF_MASK))
 {
   __asm("NOP");
 }                                           // Wait for receive flag to set

 i=SPI0_D;

 return i ;

}
```

2. nRF24L01+ Configuration Code.

```
//////////////////////NRF setup Start//////////////////////

void NRF_setup()
{
spi_init();                                    // sets up the pins the baud rate and the initial
state of CE_pin and CSN_pin.

CSN_pin_low;                                          //low down CSN pin to enable write on
the NRF register CONFIG register
data_in[0]= spi_send(0b00100000); //
data_in[1]= spi_send(0b00111111); // powering up and setting as receiver
CSN_pin_high;



CSN_pin_low;                                          //low down CSN pin to enable write on
the NRF register EN_AA .
data_in[0]= spi_send(0b00100001); //
data_in[1]= spi_send(0b00000001); // autoack enable on pipe0
CSN_pin_high;



CSN_pin_low;                                      //low down CSN pin to enable write on the
                                                    NRF register EN_RXADDR
```

```
data_in[0]= spi_send(0b00100010);
data_in[1]= spi_send(0b00000001); // enabling pipe0 only
CSN_pin_high;



CSN_pin_low;                                    //low down CSN pin to enable write on the
                                                  NRF register Setup_AW
data_in[0]= spi_send(0b00100011);
data_in[1]= spi_send(0b00000011);// Setting the Address Bytes to be 5
                                                  Bytes.
CSN_pin_high;



CSN_pin_low;                                    //low down CSN pin to enable write on the
                                                  NRF register SETUP_RETR
data_in[0]= spi_send(0b00100100);
data_in[1]= spi_send(0b00101111);// setting 750us restransmit time , and
                                                  no. of tranmsit ries to be 15
CSN_pin_high;



CSN_pin_low;                                    //low down CSN pin to enable write on the
                                                  NRF register RF_CH
data_in[0]= spi_send(0b00100101);
data_in[1]= spi_send(0b00001111);//setting 2.4015 GHZ as operating
                                                  Frequency
CSN_pin_high;



CSN_pin_low;                                    //low down CSN pin to enable write
                                                  on the NRF register RF_SETUP
data_in[0]= spi_send(0b00100110); //
data_in[1]= spi_send(0b00100110);// setting the data rate to be 250kbps
                                                  and setting the output power to be 0dbm.
CSN_pin_high;                                   //low down CSN pin to enable write on the
                                                  NRF register STATUS

NRFwrite_bit_write(7, 4, 1);            //clear RT interrupt
NRFwrite_bit_write(7, 5, 1);            //clear TX interrupt
NRFwrite_bit_write(7, 6, 1);            //clear RX interrupt
                                                  // clearing all the interrupts



CSN_pin_low;                                    //low down CSN pin to enable write
                                                  on the NRF register RX_ASSR_PO
data_in[0]= spi_send(0x2a);             // setting the Address length on
                                                  pipe 0.
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
CSN_pin_high;                           // setting the RX address to be 0x1212121212
```

```
CSN_pin_low;                                    //low down CSN pin to enable write on the NRF
                                                register TX_ASSR_PO
data_in[0]= spi_send(0x30);          //setting the Address length on pipe 0.
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
data_in[1]= spi_send(0x12);
CSN_pin_high;                                   // setting the TX address to be 0x1212121212



CSN_pin_low;                                    //low down CSN pin to enable write on the NRF
                                                  register RX_PW_P0
data_in[0]= spi_send(0x31);
data_in[1]= spi_send(0x03);          // setting the pipe0 rx payload to 3 bytes
CSN_pin_high;


CSN_pin_low;                                    //low down CSN pin to enable write on the NRF
                                                  register
data_in[0]= spi_send(0b00110001);
data_in[1]= spi_send(3);
CSN_pin_high;

}
////////////////////////////////NRF setup end//////////////////////////////
```

3. LCd and Keypad code.

```
void lcd_init(void)
{
        TFC_Delay_10uS(1500);
        lcd_send_command(0x01);                         //command to clear the lcd = 00000001
        TFC_Delay_10uS(200);                    //delay for 2ms
        lcd_send_command(0x38);                         //command to tell the lcd that we are
using 8 pin data mode.
        TFC_Delay_10uS(50);                             //delay for 50us
        lcd_send_command(0b00001111);  //0000"1"111 to start using this command, next "1" makes
the display turn on. next "1" turn the cursor on. the final "1" makes the cursor blinking.
        TFC_Delay_10uS(50);                             //delay for 50us
}


void lcd_busy_check()
{
        GPIOC_PDDR &= ~d7_loc;                  //setting the d7 pin`s direction to be input.
        rw_pin_high;                                    //telling mr. lcd that it is not a command
but read by setting the rw pin high.
        rs_pin_low;                                     //telling mr. lcd that it is not gonna be a
character but a command read by setting rs pin low.

        while(d7_busy)
        {
        en_pin_high;                            // telling mr. lcd to start listening by setting the
enable pin high.
        asm volatile ("nop");           // giving mr. lcd some time to listen to what we said .
        asm volatile ("nop");           // giving mr. lcd some more time to listen to what we said
.
```

```
        en_pin_low;                                     // telling mr. lcd to stop listening by
settig en pin low.
        }

        GPIOC_PDDR |= d7_loc;           //setting the d7 pin`s direction to be output back.

}

void lcd_goto(uint8_t line , uint8_t box)
{
        lcd_send_command(line_loc[line]+box+(0x80)-1);

}
int getkey()                                                    // get key function,
reads the status of the keypad and returns as x for further processing.
{
x=0;
row1_high;
row2_low;
row3_low;

if(col1_st)
{
                        //lcd_send_char(0x30+1);
                        //TFC_Delay_10uS(15000);
                        while(col1_st)
                        {}
                        x=1;
                        TFC_Delay_10uS(15000);
}
if(col2_st)
{
                        //lcd_send_char(0x30+4);
                        //TFC_Delay_10uS(15000);
                        while(col2_st)
                        {}
                        x=4;
                        TFC_Delay_10uS(15000);
}

if(col3_st)
{
                        //lcd_send_char(0x30+7);
                        //TFC_Delay_10uS(15000);
                        while(col3_st)
                        {}
                        x=7;
                        TFC_Delay_10uS(15000);
}

if(col4_st)
{
                        //lcd_send_char(0x30-6);
                        //TFC_Delay_10uS(15000);
                        while(col4_st)
                        {}
                        x=10;                                   //return 10 indicates
that * is pressed
                        TFC_Delay_10uS(15000);
}
```

```
TFC_Delay_10uS(10);
row1_low;
row2_high;
row3_low;

if(col1_st)
{
                        //lcd_send_char(0x30+2);
                        //TFC_Delay_10uS(15000);
                        while(col1_st)
                        {}
                        x=2;
                        TFC_Delay_10uS(15000);
}
if(col2_st)
{
                        //lcd_send_char(0x30+5);
                        //TFC_Delay_10uS(15000);
                        while(col2_st)
                        {}
                        x=5;
                        TFC_Delay_10uS(15000);
}

if(col3_st)
{
                        //lcd_send_char(0x30+8);
                        //TFC_Delay_10uS(15000);
                        while(col3_st)
                        {}
                        x=8;
                        TFC_Delay_10uS(15000);
}

if(col4_st)
{
                        //lcd_send_char(0x30);
                        //TFC_Delay_10uS(15000);
                        while(col4_st)
                        {}
                        x=11;                                   // return 11 indicates
that 0 is pressed.
                        TFC_Delay_10uS(15000);
}

TFC_Delay_10uS(10);

row1_low;
row2_low;
row3_high;

if(col1_st)
{
                        //lcd_send_char(0x30+3);
                        //TFC_Delay_10uS(15000);
                        while(col1_st)
                        {}
                        x=3;
                        TFC_Delay_10uS(15000);
}
if(col2_st)
```

```
{
                        //lcd_send_char(0x30+6);
                        //TFC_Delay_10uS(15000);
                        while(col2_st)
                        {}
                        x=6;
                        TFC_Delay_10uS(15000);
}

if(col3_st)
{
                        //lcd_send_char(0x30+9);
                        //TFC_Delay_10uS(15000);
                        while(col3_st)
                        {}
                        x=9;
                        TFC_Delay_10uS(15000);
}

if(col4_st)
{
                        //lcd_send_char(0x23);
                        //TFC_Delay_10uS(15000);
                        while(col4_st)
                        {}
                        x=12;                          // return 11 indicates that #
                                                       is pressed.
                        TFC_Delay_10uS(15000);
}



return x;
TFC_Delay_10uS(10);
}


void print_x(int x)          // print_x function, takes x returned by keypad
                             function and prints accordingly, suggested to be
                             // used with keypad function only, for other printing
                             use lcd_send_char() or lcd_send_string() function.
{

        if(x==1)
                        {lcd_send_char(0x30+1);
                        TFC_Delay_10uS(10000);}
        if(x==2)
                        {lcd_send_char(0x30+2);
                        TFC_Delay_10uS(10000);}
        if(x==3)
                        {lcd_send_char(0x30+3);
                        TFC_Delay_10uS(10000);}
        if(x==4)
                        {lcd_send_char(0x30+4);
                        TFC_Delay_10uS(10000);}
        if(x==5)
                        {lcd_send_char(0x30+5);
                        TFC_Delay_10uS(10000);}
        if(x==6)
                        {lcd_send_char(0x30+6);
```

```
                    TFC_Delay_10uS(10000);}
        if(x==7)
                    {lcd_send_char(0x30+7);
                    TFC_Delay_10uS(10000);}
        if(x==8)
                    {lcd_send_char(0x30+8);
                    TFC_Delay_10uS(10000);}
        if(x==9)
                    {lcd_send_char(0x30+9);
                    TFC_Delay_10uS(10000);}
        if(x==10)
                    {lcd_send_char(0x2A);
                    TFC_Delay_10uS(10000);}
        if(x==11)
                    {lcd_send_char(0x30);
                    TFC_Delay_10uS(10000);}
        if(x==12)
                    {lcd_send_char(0x23);
                    TFC_Delay_10uS(10000);}


}

void lcd_send_command(int command)
{
        lcd_busy_check();                   //check if mr. lcd is busy, if he is then
                                            dont proceed .

        setting_lcd_pin_state(command);
        rw_pin_low;                         // telling mr lcd to listen by setting rw
                                            pin low.
        rs_pin_low;                         // telling mr lcd that its gonna be a
                                            command by setting register select, rs pin
                                            low.
        en_pin_high;                        // telling mr. lcd to start listening by
                                            setting the enable pin high.
        asm volatile ("nop");               // giving mr. lcd some time to listen to
                                            what we said .
        asm volatile ("nop");               // giving mr. lcd some more time to listen to
                                            what we said .
        TFC_Delay_10uS(500);                //delay for 50us
        en_pin_low;                         // telling mr. lcd to stop listening by settig en pin
                                            low.
}


void lcd_send_char(int character)
{
        lcd_busy_check();                   //check if mr. lcd is busy, if he is then
                                            dont proceed .

        setting_lcd_pin_state(character);
        rw_pin_low;                         // telling mr lcd to listen by setting rw
                                            pin low.
        rs_pin_high;                        // telling mr lcd that its gonna be a
                                            character by setting register select, rs
                                            pin high.
        en_pin_high;                        // telling mr. lcd to start listening by
                                            setting the enable pin high.
        asm volatile ("nop");           // giving mr. lcd some time to listen to
                                            what we said .
        asm volatile ("nop");           // giving mr. lcd some more time to listen
                                            to what we said .
```

```
        TFC_Delay_10uS(500);            //delay for 50us
        en_pin_low;                             // telling mr. lcd to stop listening by
                                                settig en pin low.

}

void lcd_send_string(char *thestring)
{
while(*thestring>0)
{
        lcd_send_char(*thestring++);
}



}

void setting_lcd_pin_state(int data)
{
        if((1<<0)&data)
        {d0_high;}

        if(!((1<<0)&data))
        {d0_low;}

        if((1<<1)&data)
        {d1_high;}
        if(!((1<<1)&data))
        {d1_low;}

        if((1<<2)&data)
        {d2_high;}
        if(!((1<<2)&data))
        {d2_low;}

        if((1<<3)&data)
        {d3_high;}
        if(!((1<<3)&data))
        {d3_low;}

        if((1<<4)&data)
        {d4_high;}
        if(!((1<<4)&data))
        {d4_low;}

        if((1<<5)&data)
        {d5_high;}
        if(!((1<<5)&data))
        {d5_low;}

        if((1<<6)&data)
        {d6_high;}
        if(!((1<<6)&data))
        {d6_low;}

        if((1<<7)&data)
        {d7_high;}
        if(!((1<<7)&data))
        {d7_low;}
}
```

4)Configuration of Port Data Control Regiters and Port Data Direction Register

```
void TFC_InitGPIO()
{
SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK |
SIM_SCGC5_PORTC_MASK | SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK; //
ENABLE CLOCK FOR ALL PINS AT PORT B AND D


PORTA_PCR1   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // GPIO for
Keypad col4 input
PORTA_PCR4   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // GPIO for
Keypad row4 output
PORTA_PCR5   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // GPIO for
Keypad row3 output
PORTA_PCR12 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // GPIO for
Keypad col1 input
PORTA_PCR2   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // GPIO for
Keypad col3 input

PORTB_PCR18 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;               //GPIO
FOR on board Led
PORTB_PCR19 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;               //GPIO
FOR on board Led
PORTB_PCR1  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;               //Setting
LCD rs pin as GPIO
PORTB_PCR2  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;               //Setting
LCD rw pin as GPIO
PORTB_PCR3  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;               //Setting
LCD en pin as GPIO


PORTC_PCR1  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D6
PORTC_PCR2  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D7
PORTC_PCR9  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // GPIO for
Keypad row1 output
PORTC_PCR8  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // GPIO for
Keypad row2 output
PORTC_PCR11 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // always 0 ,
used as gnd for keypad.
PORTC_PCR6  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // input pin for
"YES" button
PORTC_PCR10 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // input pin for
"no" button
PORTC_PCR11 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // input pin for
"pgrm" button

PORTE_PCR3   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // MUX THE PINS
AS GPIO for CLK pin of sound module.
PORTE_PCR4   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // MUX THE PINS
AS GPIO for SDA pin of sound module.
PORTE_PCR5   = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // MUX THE PINS
AS GPIO for RST pin of sound module.
PORTE_PCR20 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D0
PORTE_PCR21 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D1
PORTE_PCR22  = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;        // Setting Data
Pins For LCD as GPIO. D2
```

```
PORTE_PCR23 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D3
PORTE_PCR29 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D4
PORTE_PCR30 = PORT_PCR_MUX(1) | PORT_PCR_DSE_MASK;         // Setting Data
Pins For LCD as GPIO. D5  ; d6 and D7 located in port C1 and C2. go find
there!


PORTD_PCR5 = PORT_PCR_MUX(1)| PORT_PCR_DSE_MASK;      //Set PTD5 as GPIO
and as CSN_pin
PORTD_PCR0 = PORT_PCR_MUX(1)| PORT_PCR_DSE_MASK;      //Set PTD0 to mux 2
[SPI0_PCS0]     CE_pin
PORTD_PCR1 = PORT_PCR_MUX(2)| PORT_PCR_DSE_MASK;      //Set PTD1 to mux 2
[SPI0_SCK]
PORTD_PCR2 = PORT_PCR_MUX(2)| PORT_PCR_DSE_MASK;      //Set PTD2 to mux 2
[SPI0_MOSI]
PORTD_PCR3 = PORT_PCR_MUX(2)| PORT_PCR_DSE_MASK;      //Set PTD3 to mux 2
[SPIO_MISO]
PORTD_PCR4 = PORT_PCR_MUX(1)| PORT_PCR_DSE_MASK;      //GPIO for Keypad
col2 input

GPIOA_PDDR |= row3_loc;    //Setting row3 as output
GPIOA_PDDR |= row4_loc;    //Setting row3 as output
GPIOA_PDDR &= ~col1_loc;   //Setting col1, col2, col3 and col4 as input.
GPIOD_PDDR &= ~col2_loc;
GPIOA_PDDR &= ~col4_loc;
GPIOA_PDDR &= ~col3_loc ; // previously this pin was used for irqpin, refer
below.


GPIOB_PDDR |= RED_LED_LOC | GREEN_LED_LOC | rw_loc | rs_loc | en_loc ;  //
setting rw, rs and en pin for LCD as output. // SETTING PIN 18 AND 19 OF
PORT B AND PIN1 OF PORT D AS OUTPUT DEFINED BY "O" .

GPIOD_PDDR |= BLUE_LED_LOC;                                              //
Setting pin1 of portD as output for blue led.

GPIOE_PDDR |= CLK_LOC | SDA_LOC | RST_LOC | d0_loc | d1_loc | d2_loc |
d3_loc | d4_loc | d5_loc ;       // Setting PIN 3 4 AND 5 OF PORT e AS gpio
output ; setting d0 --> d5 as output for LCd.

GPIOD_PDDR |= CE_pin | CSN_pin ;
      // setting CE and CSN pin as output for RF module.
      //GPIOD_PDDR &= ~col3_loc;
      //Setting col3 of keypad as input.

GPIOC_PDDR |= d6_loc;                                                    //
setting d6 of LCD as out, D7`s direction is not set since we might need to
read to check for busy status or write to this pin.
GPIOC_PDDR |= row1_loc | row2_loc | key_gnd_loc ;   // setting row 1 and 2
as output for keypad ; keypad gnd as output but is always gnded.
GPIOC_PDDR &= ~yes_loc;          // setting "yes" button as input

GPIOC_PDDR &= ~no_loc;           // setting "no" button as input

GPIOC_PDDR &= ~pgrm_loc;         // setting pgrm pin as input


}
```

5. Transmitter.c

```
while(1)
	{

if(pgrmbt)
{
	while(pgrmbt) //checks if the program button is ON or OFF if its ON
it enters program mode.
	{
	receiver();
////////////////////////////////////////////////////////////////////////
	if((data_in[1]==8)&&(data_in[2]==8)&&(data_in[3]==8))
	{
		for(w=0;w<100;w++)
		{
		transmitid(m,n,p); //transmit 1,2 and 3.
		}

		TFC_Delay_10uS(20000);		//delay 0.2s.


	}
////////////////////////////////////////////////////////////////////////
	w=0;
	if((data_in[1]==9)&&(data_in[2]==9)&&(data_in[3]==9))
	{		//TFC_Delay_10uS(100000); //delay2 s
		do
		{
		receiver();
		w++;

		if((data_in[1]!=data_in[2])||(data_in[2]!=data_in[3]))
		{break;}

		}while((w<100));

				m=data_in[1];
				n=data_in[2];
				p=data_in[3];

				}
////////////////////////////////////////////////////////////////////////
			}
		}
if(!pgrmbt) //checks if the program button is ON or OFF if its ON it enters
transmit mode.
{
	while(!pgrmbt)
	{	for(w=0;w<100;w++)
		   {
		       transmitid(m,n,p);		//transmit 1,2 and 3.
		   }

		TFC_Delay_10uS(20000);		//delay 0.2s.
		receiver();			//go to receiver mode.
		TFC_Delay_10uS(50000);		//delay 0.5s.
			}

	}
}
```

5. prog.c

```
{
            TFC_Delay_10uS(1500);
            lcd_send_command(0x01);                 //command to clear the
lcd = 00000001
            TFC_Delay_10uS(200);                    //delay for 2ms

            lcd_goto(1,1);
            lcd_send_string("Read Tag?");

            lcd_goto(1,17);
            lcd_send_string("(A)");


            lcd_goto(2,1);
            lcd_send_string("Program Tag?");

            lcd_goto(2,17);
            lcd_send_string("(B)");


            while(1)
            {
            x=getkey();


            if(x==13)                          // if button A is pressed.
            {

                  TFC_Delay_10uS(1500);
                  lcd_send_command(0x01);               //command to
clear the lcd = 00000001
                  TFC_Delay_10uS(200);                  //delay for 2ms

                  lcd_goto(1,1);
                  lcd_send_string("Reading the tag");

                  for(t=0;t<50;t++)
                  {
                        transmitid(8,8,8);

                  }

                  /*for(w=0;w<=300;w++)
                  {
                  receiver();
                  if((data_in[1])!=(data_in[2])!=(data_in[3]))
                  {break;}

                  }
                  */
                  w=0;
                  do
                  {
                  receiver();


                  w++;
```

```
}while((data_in[1]==data_in[2])&&(data_in[2]==data_in[3])&&(w<300));


              if(w<300)
              {
              TFC_Delay_10uS(50000);                  //delay for 1s
              lcd_send_string(".");
              TFC_Delay_10uS(50000);                  //delay for 1s
              lcd_send_string(".");
              TFC_Delay_10uS(50000);                  //delay for 1s
              lcd_send_string(".");

              lcd_goto(2,1);
              lcd_send_string("The Id is: ");

              x=data_in[1];
              print_x(x);
              x=data_in[2];
              print_x(x);
              x=data_in[3];
              print_x(x);
              }

              if(w>=300)
              {
              lcd_goto(2,1);
              lcd_send_string("Time Out, No Tag");
              lcd_goto(3,1);
              lcd_send_string("Press A to Retry");

              }




       }

       if(x==15)
         {
          break;
       }

       if(x==14)
                    {
                                  TFC_Delay_10uS(1500);
                                  lcd_send_command(0x01);
//command to clear the lcd = 00000001
                                  TFC_Delay_10uS(200);
//delay for 2ms

                           lcd_goto(1,1);
                           lcd_send_string("Enter new ID:");
                           lcd_goto(2,1);

              ////////////////////////////////////
                           l=0;
                           do
```

```
                        {
                              l=getkey();
                              print_x(l);

                        }while(l==0);
                        x1=l;
                  ////////////////////////////////////
                        l=0;
                        do
                        {
                        l=getkey();
                        print_x(l);

                        }while(l==0);
                        x2=l;
                  ////////////////////////////////
                        l=0;
                        do
                        {
                        l=getkey();
                        print_x(l);

                        }while(l==0);
                        x3=l;

                  ////////////////////////////

                        lcd_goto(3,1);
                        lcd_send_string("Confirm ID?");

                        lcd_goto(4,1);
                        lcd_send_string("Yes(C)");
                        lcd_goto(4,11);
                        lcd_send_string("No(D)");

                        l=0;
                        do
                        {
                        l=getkey();
                        print_x(l);


                        }while(l==0);

                        if(l==15)
                        {     TFC_Delay_10uS(1500);
                              lcd_send_command(0x01);
//command to clear the lcd = 00000001
                              TFC_Delay_10uS(200);
//delay for 2ms

                              lcd_goto(1,1);
                              lcd_send_string("Programming");
                              for(w=0;w<50;w++)
                                    {
                                    transmitid(9,9,9);
                                    }
                              for(w=0;w<100;w++)
                                    {
                                    transmitid(x1,x2,x3);
```

```
                                        }
                                lcd_goto(2,1);
                                lcd_send_string("Done!!!");

                        }


                        if(l==16)
                        {
                                break;
                        }


                }

        }

}
```

6. Notifier.c

```
int main(void)
{
        TFC_Init();

////initialising sound module ////
        init_audio();

//// initialising NRF module ////
        NRF_setup(); // INITIALISING THE PORT THE PINS , THE INITIAL
REGISTERS,
        TFC_Delay_10uS(13);


        while(1)
        {
                brk=0;


                receiver();

                if((x==1 && y==2 && z==3)&&(x!=a && y!=b && z!=c))
                {
                        send_audio(0x0001);TFC_Delay_10uS(500000);
        // play: sound , next bus is no. 28.
                        send_audio(0x0000);TFC_Delay_10uS(400000);
        // play: this bus travels enroute on jalan ampang to kota raya.
                        send_audio(0x0002);TFC_Delay_10uS(300000);
        // play: do you want to take this bus?

                        a=x;b=y;c=z;

                        for(w=0;w<4000000;w++)          // for loop detecting
the yes or no bt. and status of brk from stationrx function.
                                                                // breaks
if, w>10000  OR nobt is pressed OR brk=1.
                        {
                                if(yesbt)
```

```
                        {

      send_audio(0x0003);TFC_Delay_10uS(400000);// play: now, you will be
travelling on jalan ampang.
                              stationrx();
                        }

                        if(nobt)
                        {

      send_audio(0x0007);TFC_Delay_10uS(300000);//play: continuing
travelling on jalan ampang.}//play: continuing waiting for another bus.
                              break;
                        }

                        if(brk==1)
                        {
                              break;
                        }

                  }


                  if(w>3999000)
                        {



      send_audio(0x0007);TFC_Delay_10uS(400000);//play: continuing
travelling on jalan ampang.}

                        }


            }


      }


}

void stationrx()
{

      while(1)
      {

      receiver();

      if((x==1 && y==2 && z==3))//&&(x!=a && y!=b && z!=c))
        {


            send_audio(0x0004);TFC_Delay_10uS(500000);          //play:
next station is KLCC do you want to depart?


            a=x;b=y;c=z;
```

```
            for(w=0;w<4000000;w++)
               {
                   if(yesbt)
                          {
                                 brk=1;

      send_audio(0x0005);TFC_Delay_10uS(500000);// play: you are choosing
to depart at station KLCC.

      send_audio(0x0006);TFC_Delay_10uS(500000);// play: thank you for
using this device, have a safe journey ahead.
                          }

                   if(nobt)
                          {

      send_audio(0x0008);TFC_Delay_10uS(400000);//play: continuing
travelling on jalan ampang.
                                 break;
                          }

                   }

            if(w>3999900)
            {

                   send_audio(0x0008);TFC_Delay_10uS(400000);//play:
continuing travelling on jalan ampang.

            }



        }


      if(brk==1)                                    // final break to get
out of the stationrx function.
      {
            break;
      }



      }

}
```