IERG4999 Final Year Project II

Using STM32 and Si473x to Build a Network Radio

By
CHEUNG Wing Ting
1155110523

A FINAL YEAR PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF INFORMATION ENGINEERING DEPARTMENT OF INFORMATION ENGINEERING
THE CHINESE UNIVERSITY OF HONG KONG

May,2021

ABSTRACT

Traditional radio's ability is limited, namely transferring radio signal in a long distance is unstable. Also, multiple devices are required if we want to listen to the podcast and pre-recorded music. In this project, a small-scale radio station which is a FM transmitter will be built. Variable audio source such as real-time podcasting and pre-recorded music can be played using the transmitter. The real-time signals will be transferred to a radio receiver. In additional, an infrared (IR) remote will also be built for convenience. The remote can control not only the FM transmitter but also the receiver. Users can adjust the frequency of the transmitter and the receiver. Moreover, the volume of the receiver can be controlled by the remote. The ultimate goal is to build a system that transfers audio from different sources, such as network streaming devices, through the radio station built in this project with an IR remote function.

INTRODUCTION

Traditional radio signal cannot be transferred through a long distance and noises always occur. The service is unstable. For example, when we are in a house, we need to be close to the window in order to capture the signals, which is inconvenient. Moreover, a long-distance radio signal transfer will lower the quality of the audio.

Traditional radio can only receive FM signals and AM signals transmitted by radio station. We need to change the device when we want to listen music. Switching between different devices are inconvenient.

To maintain the traditional radio style and improve radio experiences, this new network radio system is introduced. This radio system is in analogy domain and uses air as a medium. First, the transmitter will be connected to an audio source. Then the audio signals will be transferred to the receiver. Users can access the signal transferred by the transmitter using a website or an application at anytime and anywhere in the house. User can listen to different type of audio by tuning the frequency of the receiver instead of switching different devices. Different audio sources such as network streaming audio and pre-recorded music can all be transmitted though this network radio system.

BACKGROUND

Radio system

Radio is a wireless system which provides real-time communication. Signals are transferred through electromagnetic (EM) wave [1]. The signal from the source will first pass through an amplifier, then a modulator [1] (Figure 1). There are three ways of modulation including amplitude modulation (AM), frequency modulation (FM) or phase modulation (PM) [1]. After the modulation, the signal will become an EM wave and be transferred from the transmitter to the receiver.

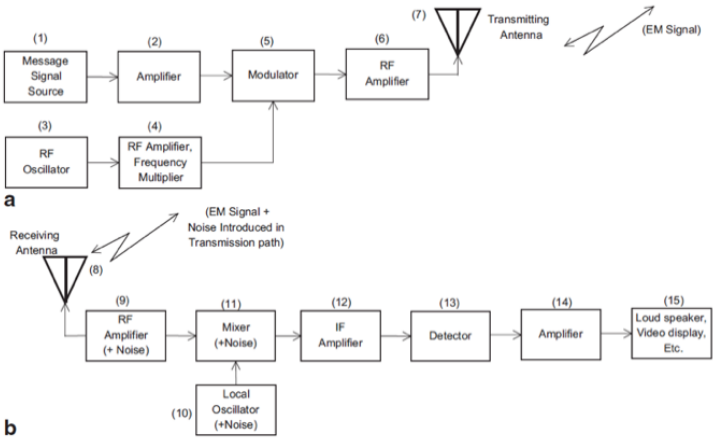

Figure 1. Block diagram of transmitter (a), and a receiver (b). [1]

Infrared

Infrared is widely applied on remote controlling, for example, the remote controller for television and air conditioner. It is efficient and low-cost. There are different protocols used for infrared transmission, such as PWM of NEC protocol, PPM of Philips RC-5 Protocol and so on. However, the IR signal cannot be transmitted if there are some barriers between the transmitter and the sensor. With the rapid development of Internet of things (IoT) technology, there are some potential risks of leaking sensitive information by attacking IR remote control device since it is difficult for single purpose IR remote control device to detect whether the IR signal source is malicious [2].

Related work

In order to increase the quality of the radio, some internet-based network systems are introduced. It is in digital domain and uses Internet as a medium. The receiver of the internet-based network radio needs to have access to mobile networks or WIFI connections in order to connect to the radio server. After connecting to the radio server, the network radio services will be provided. For example, RTHK is one of the network radio station in Hong Kong (Figure 2) [3]. RTHK provides different channels at different frequencies. People can access the radio by visiting RTHK website. With network radio, although the distance problem of traditional radio is solved, the traditional radio style is lost. It is more similar to a music player rather than a radio.

Figure 2. RTHK Network Radio[2].

METHODOLOGY

Hardware

Microcontroller STM32F103C8, an ARM 32-bit Cortex-M3 CPU operating at 72 MHz [4], is used in this project. It has several communication interfaces which includes two I^2C interfaces and three USARTs [4].

A Monolithic Digital Stereo FM Transmitter from Elechouse with a IC KT0803K is used to build the transmitter. It includes Industry standard 2-wire I^2C MCU [5]. I^2C interface is used for the inner boards communication with STM32F103C8. A 3.5mm audio cable is used for connecting the audio source to the FM transmitter. Pre-recorded music is used in this project. The FM frequency range is from 70 to 108 MHz .

A FM/AM receive module, PA102BA-S with microcontroller SI473x, is used in this project for implementing the remote control function for the receiver. SI473x family provides a high consumer experience with a low cost [6]. This module is connected to the STM32F103C8 by using I^2C communication interface. No speaker will be connected to the receiver module in this project since the main focuses of this project are the ability of FM transmission and IR remote controlling. Therefore, printing the statues of the receiver module in the Serial Monitor from Arduino is sufficient enough for testing the ability of IR remote controlling for the receiver module.

An IR remote is used to transmit the IR light. The number buttons are going to be used for entering the exact frequency. The IR receiver module will be connected to the PA1 on STM32F103C8. Electrical signal will be sent when it receives IR light [7].

For the antenna, any metal could serve [8]. 75cm of Dupont Line are used as the antenna. Dupont Line is chosen because it has a hard pin which can make the connection of the antenna and the board more durable. The approximated length of the antenna is calculated by dividing 468 by the frequency [9].

$$\text{Antenna length (feet)} \approx \frac{468}{\text{Frequency(MHz)}}$$

Equation 1. The formula for the approximated length of antenna .

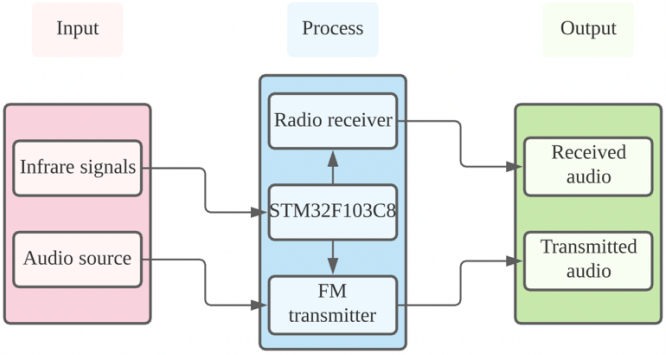

Figure 3. The system block diagram.

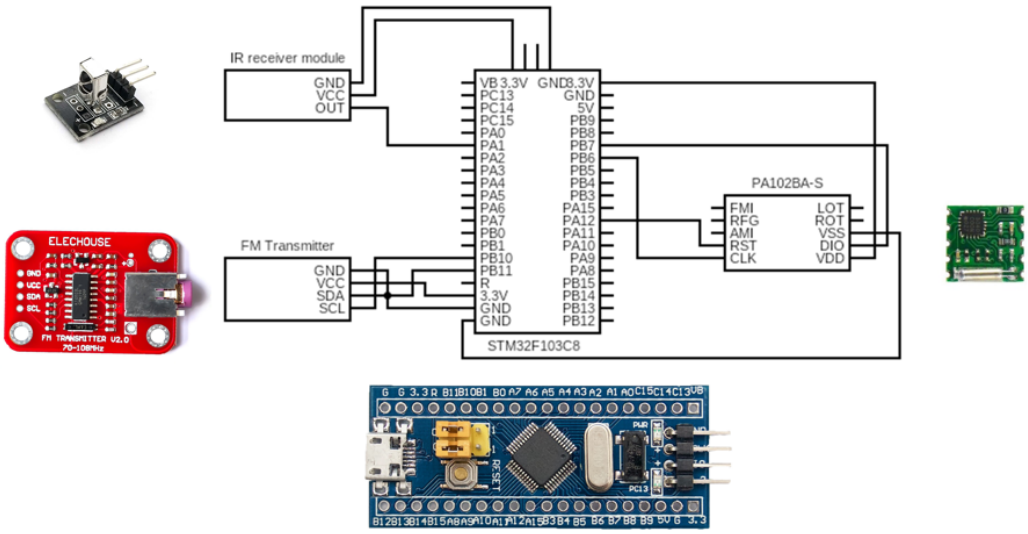

Figure 4. Microcontroller communication.

Software

To develop the device, Arduino Software Integrated-Development-Environment (IDE) is used in this project [10]. Library used for developing STM32 series microcontrollers in Arduino IDE from MAKER.PRO is used [11]. Coding and uploading can both be done in Arduino IDE.

Codes

Codes for the FM transmitter is developed based on the FMTX library from Elechouse [12] [13].In this project, the Wire library is applied for the transmitter part [13] [14], IRMP library is modified for the IR module and SI4735 library is used for the radio receiver [15] [16]. The description of the codes used are shown in Table 1.

Filename	Description
`FMTX.h`	It is a header file of the library for the FM transmission part in this project.
`FMTX.cpp`	It handles the I^2C communications between STM32F103C8 and FM transmitter.
`PinDefinitionsAndMore.h`	It is a header file which included the pin definitions for IR signal receiving.
`SimpleReceiver.ino`	It executes on Arduino IDE which functions as the main file. It decodes the electronical signals of each key on the remote.

Table 1. Code description.

Remote buttons

A special input format is used since there is no “fullstop” button. User needs to input the frequency in 4-digits format by using the number buttons. For example, inputting “0900” to adjust the transmitting frequency to 90.0MHz, inputting “1000” to adjust the transmitting frequency to 100.0MHz. Key “*” is assigned for cleaning the input while “#” key is assigned for submitting the input.

“Left” and “Right” buttons are used to adjusting the frequency of the receiver. “Up” and “Down” buttons are used for turning the volume of the receiver up and down respectively.

Key	Decoded result
1	0x45
2	0x46
3	0x47
4	0x44
5	0x40
6	0x43
7	0x7
8	0x15
9	0x9
0	0x19
*	0x16
#	0xD
UP	0x18
DOWN	0x52
LEFT	0x8
RIGHT	0x5A
OK	0x1C

Table 2. The decoded result of the buttons of the IR remote.

FMTX.cpp [12]

To use the I^{2}C communication, library Wire is used since it includes the basic operations for I^{2}C communication [14].

```
void i2c_init(void)
{
    Wire.begin();
    Wire.setClock(((F_CPU / I2C_FREQ) - 16) / 2);
}

void i2c_start(void)
{
    Wire.beginTransmission(0x3E);
#ifdef __FM_DEBUG
    Serial.print("I2C Start\r\n");
#endif
}

void i2c_stop(void)
{
    Wire.endTransmission();
#ifdef __FM_DEBUG
    Serial.print("I2C Stop\r\n");
#endif
}

void i2c_write(u8 dt)
{
    Wire.write(dt);
#ifdef __FM_DEBUG
    Serial.print("I2C Write\r\n");
#endif
}
u8 i2c_read_nack(void)
{
    Wire.read();
#ifdef __FM_DEBUG
    Serial.print("I2C Read Nack\n");
#endif
}
```

PinDefinitionsAndMore.h [15]

The pin for receiving IR signal is set to be PA1 in this file.

```
#define IRMP_INPUT_PIN   PA1
#define IRMP_INPUT_PIN_STRING   "PA1"
```

SimpleReceiver.ino [15]

Different proposes are assigned to each key by switching cases. For example, the number keys will be assigned for entering the exact frequency, the "*" is assigned for cleaning the input while "#" key is assigned for submitting the input. To control the FM transmission. This file is modified base on `fmtx_demo.ino`, `SI4735_01_STM32_POC.ino` and `SimpleReceiver.ino` from Elechouse, SI4735 library and IRMP library respectively [16] [15]. The calculation of the frequency adjustment for the transmitter is from `fmtx_demo.ino`. Built-in functions si4735.volumeUp(), si4735.volumeDown(), si4735.frequencyUp() and si4735.frequencyDown() are extracted from `SI4735_01_STM32_POC.ino` after studying the original code. The delay() function is used to avoid duplicate signals caused by the pressing time. The architecture of the is modified base on `SimpleReceiver.ino`.

The showStatus() function is from `SI4735_01_STM32_POC.ino`. Its function is to show the current frequency of the receiver.

```
void showStatus()
{
  si4735.getStatus();
  si4735.getCurrentReceivedSignalQuality();
  Serial.print("Receiver: You are tuned on ");
  if (si4735.isCurrentTuneFM())
  {
    Serial.print(String(currentFrequency / 100.0, 2));
    Serial.print("MHz ");
    Serial.println((si4735.getCurrentPilot()) ? "STEREO" : "MONO");
  }
  else
  {
    Serial.print(currentFrequency);
    Serial.print("kHz");
}
```

This is the main of the program. First, it checks if the new data is available and get them. Then, it skips repetitions of command. After that, it evaluates IR command. Different cases represent different IR commands. For the numbers buttons (case 0x45, 0x46, 0x47, 0x40, 0x44, 0x43, 0x7, 0x15, 0x9 and 0x19), the corresponding value which is 0 to 9 will be stored in an array `fm_adj[]`. When the "#" or "OK" (case 0xD and 0x1C) button is pressed, `fm_adj[]` will be sent to function `fm_set_freq()` to change the frequency of the transmitter. In the case of "*" (case 0x16) is pressed,0 is assigned to each elements in `fm_adj[]` so as to clear the array. When "UP" or "DOWN" (case0x18 and 0x52) is pressed, it will process functions `si4735.frequencyUp()` and `si4735.frequencyDown()` respectively to turn up and down the volume of the receiver. If "LEFT" or "RIGHT" (case 0x8 and 0x5A) is pressed, it will adjust the frequency of the receiver by running functions `si4735.frequencyUp()` and `si4735.frequencyDown()`.

```
void loop() {
    // Check if new data available and get them
    if (irmp_get_data(&irmp_data)) {
    // Skip repetitions of command
        if (!(irmp_data.flags & IRMP_FLAG_REPETITION)) {
        // Here data is available and is no repetition -> evaluate IR command
            if (irmp_data.command == 0x45 || irmp_data.command == 0x46 ||
irmp_data.command == 0x47 || irmp_data.command == 0x44 ||
                irmp_data.command == 0x40 || irmp_data.command == 0x43 ||
irmp_data.command == 0x7 || irmp_data.command == 0x15 ||
                irmp_data.command == 0x9 || irmp_data.command == 0x19
```

```
                || irmp_data.command == 0x18|| irmp_data.command == 0x52||
irmp_data.command == 0x8|| irmp_data.command == 0x5A){

              switch (irmp_data.command) {
              case 0x45: // 1
                fm_adj[i] = '1';
                Serial.println("1");
                delay(100);
                break;
              case 0x46: // 2
                fm_adj[i] = '2';
                Serial.println("2");
                delay(100);
                break;
              case 0x47: // 3
                fm_adj[i] = '3';
                Serial.println("3");
                delay(100);
                break;
              case 0x44: // 4
                fm_adj[i] = '4';
                Serial.println("4");
                delay(100);
                break;
              case 0x40: // 5
                fm_adj[i] = '5';
                Serial.println("5");
                delay(100);
                break;
              case 0x43: // 6
                fm_adj[i] = '6';
                Serial.println("6");
                delay(100);
                break;
              case 0x7: // 7
                fm_adj[i] = '7';
                Serial.println("7");
                delay(100);
                break;
              case 0x15: // 8
                fm_adj[i] = '8';
                Serial.println("8");
                delay(100);
                break;
              case 0x9: // 9
                fm_adj[i] = '9';
                Serial.println("9");
                delay(100);
                break;
              case 0x19: // 0
                fm_adj[i] = '0';
```

```
                Serial.println("0");
                delay(100);
                break;
              case 0x18: // UP
                si4735.volumeUp();
                Serial.print("Volume turned up: Current volume:");
                Serial.println(si4735.getVolume());
                break;
              case 0x52: // DOWN
                si4735.volumeDown();
                Serial.print("Volume turned down:Current volume:");
                Serial.println(si4735.getVolume());
                break;
              case 0x8: // LEFT
                si4735.frequencyUp();
//                showStatus();
                break;
              case 0x5A: // RIGHT
                si4735.frequencyDown();
//                showStatus();
                break;
              }
              i++;
            }else if (irmp_data.command == 0x16 ||irmp_data.command == 0xD ||
irmp_data.command == 0x1C){
              switch (irmp_data.command) {
              case 0x16: // *
                for (int j=0; j<4; j++)
                fm_adj[j] = 0;
                i = 0;
                Serial.println("Input cleared!");
                break;
              case 0xD: // #
              case 0x1C:
                float ch;
                if (i==4){
                  ch = (fm_adj[0]-'0')*100+(fm_adj[1]-'0')*10+(fm_adj[2]-
'0')*1+0.1*(fm_adj[3]-'0');
                  if(ch>=70&&ch<=108){
                    fmtx_set_freq(ch);
                    Serial.print("Transmitter: New Channel:");
                    Serial.print(ch, 1);
                    Serial.println("MHz");
                  }else{
                    Serial.println("ERROR:Channel must be range from 70Mhz to
108Mhz.");
                  }
                }else{
                  Serial.println("ERROR:Input Format Error.");
                }
                for (int j=0; j<4; j++)
```

```
                fm_adj[j] = 0;
                i = 0;
                break;
              }

            }
        }
        //irmp_result_print(&irmp_data);
    }
    currentFrequency = si4735.getCurrentFrequency();
  if (currentFrequency != previousFrequency)
  {
    previousFrequency = currentFrequency;
    showStatus();
    delay(100);
  }
}
```

TESTING AND EXPEXTED RESULT

FM transmitter

To test the FM transmitter, an audio sources will be connected to the transmitter part. The audio is processed by the microcontrollers which are STM32F103C8 and KT0803K. A radio receiver is used to receive the signals. The audio source is expected to be played by the receiver.

Range

The range of transmission is measured in centimeter. 30-40 cm of transmitting length is expected for the FM transmitter module from Elechouse.

Noise

Some background noises are expected since it is related to the power source which is not adjustable in this project. However, the content of the audio source is expected to be heard clearly.

IR remote controller

To test the IR remote-control function, a remote will be used for transmitting the signal to the IR receiver module. The frequency of transmission is expected to be adjusted by the IR remote.

Adjusting of the transmitting frequency

Audio source will be played in the new frequency with the controlling by the IR remote. The frequency of transmitting is expected to be adjusted after pressing the “#” key, which is for submitting the transmission frequency input. User can clear the input by pressing the “*” key. The process should be shown in the Serial Monitor from Arduino for tracking.

Adjusting of the receiving frequency

The receiving frequency of the FM/AM receive module will be adjusted by pressing the “Left” and “Right” key. The frequency will increase when the “Left” key is pressed and decrease when the “Right” key is pressed. The process should be shown in the Serial Monitor from Arduino for tracking.

Adjusting the volume of receiver

The volume of the FM/AM receive module will be turned up and down by pressing the “Up” and “Down” key respectively. The variation of the sound should be shown in the Serial Monitor from Arduino for tracking.

RESULT

Range

The FM transmitter built in this project can transmit the signal in a range of 40cm which is within the expected transmitting length for the FM transmitter module.

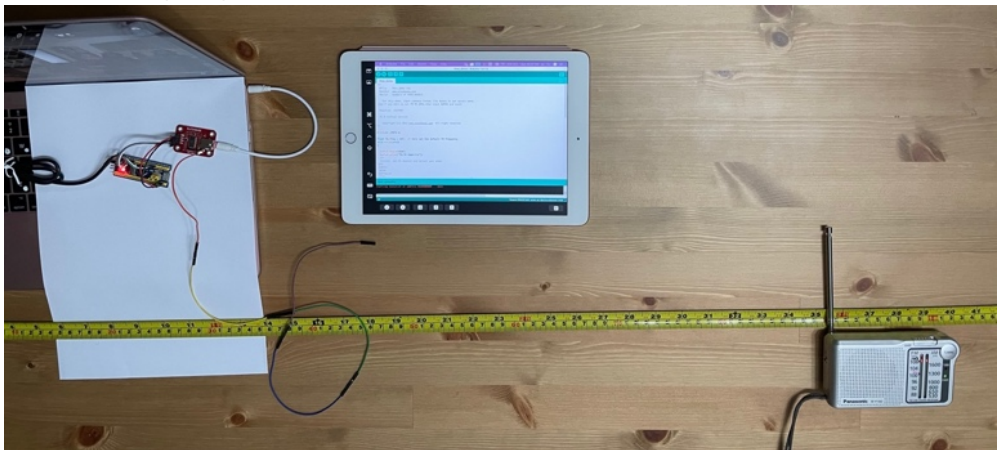

Figure 5. Length of the transmission.

Noise

There are some background noises which is not adjustable in this project since it is related to the power source. The content of the audio source can still be heard clearly. The quality of the transmitted audio is in an acceptable level.

Adjusting of the transmitting frequency

Audio source is played in the new frequency after pressing the "#" key which is the submit button. The process of inputting the frequency and the new frequency after submission are shown in the Serial Monitor for tracking.

```
16:41:51.155 -> Input cleared!
16:41:55.151 -> 1
16:41:55.758 -> 0
16:41:59.235 -> 0
16:42:00.514 -> 0
16:42:04.712 -> Transmitter: New Channel:100.0MHz
```

Figure 6. Transmitting frequency adjustment result.

Adjusting of the receiving frequency

The frequency is increased 0.1MHz when the "Left" key is pressed and decreased 0.1MHz when the "Right" key is pressed. The new receiving frequency is shown for each adjustment which is each press.

```
16:42:24.610 -> Receiver: You are tuned on 106.90MHz MONO
16:42:26.899 -> Receiver: You are tuned on 106.80MHz MONO
16:42:28.473 -> Receiver: You are tuned on 106.70MHz MONO
16:42:30.047 -> Receiver: You are tuned on 106.80MHz MONO
16:42:31.463 -> Receiver: You are tuned on 106.90MHz MONO
16:42:33.055 -> Receiver: You are tuned on 107.00MHz MONO
```

Figure 7. Receiving frequency adjustment result.

Adjusting the volume of receiver

The volume of the FM/AM receive module is turned up and down by pressing the "Up" and "Down" key respectively. The variation of the sound is shown in the Serial Monitor for tracking.

```
16:42:12.062 -> Volume turned up: Current volume:46
16:42:13.676 -> Volume turned up: Current volume:47
16:42:16.179 -> Volume turned down:Current volume:46
16:42:17.646 -> Volume turned down:Current volume:45
```

Figure 8. Volume adjustment result.

Conclusion

In this work, A FM transmitter and receiver with IR controlling is built. The audio is transmitted by using this device. The range and the quality of transmission both achieved the expectation. The noises in the transmitting process cannot be eliminated since it is related to the power source. The quality of the transmission can be improved by tuning the frequency to avoid overlapping other radio channels that will influence the signals. The IR remote is hugely improved convenience of this device. User can use the remote to control the device within certain range.

Future direction

In the future, more advanced functions could be added to this device. Since most people have a smartphone, an application for smart phone can be developed to control both transmitter and receiver thought WIFI or Bluetooth connection. However, using application in smartphone might be convenient, the cost to build this device will be higher.

REFERENCE

[1] H. J. De Los Santos, C. Sturm and J. Pontes, Radio Systems Engineering, Springer International Publishing, 2015.

[2] Z. Zhou, W. Zhang, S. Li and N. Yu, "Potential risk of IoT device supporting IR remote control," *Computer Networks,* vol. 148, pp. 307-317, 2019.

[3] "rthk.hk:radio," RTHK, [Online]. Available: https://www.rthk.hk/radio. [Accessed 28 October 2020].

[4] STMicroelectronics, "Datasheet - STMicroelectronics," [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf. [Accessed 28 October 2020].

[5] KTMicro. [Online]. Available: https://www.elechouse.com/elechouse/images/product/FM%20Transmitter%20Module/KT0803K.pdf. [Accessed 28 October 2020].

[6] Anonymous, "Silicon Labs Introduces Industry's First Fully- Integrated AM/FM Radio Receivers with Short- Wave Band Coverage," *Business Wire,* 12 November 2007.

[7] "TECH EXPLAINED: INFRARED SENSORS," SPRING WISE, 10 April 2019. [Online]. Available: https://www.springwise.com/tech-explained-infrared-sensors/. [Accessed 9 February 2021].

[8] P. Kanakaraja, "Short-Range FM Radio Station," *Electronics for You,* 13 July 2017.

[9] C. Walter, F. James, L. David and A. Michael, Modern Cable Television Technology (Second Edition), Morgan Kaufmann, 2004.

[10] Arduino, "Software," [Online]. Available: https://www.arduino.cc/en/main/software. [Accessed 28 October 2020].

[11] R. Watson, "How to Program the STM32 "Blue Pill" with Arduino IDE," 6 July 2019. [Online]. Available: https://maker.pro/arduino/tutorial/how-to-program-the-stm32-blue-pill-with-arduino-ide. [Accessed 28 October 2020].

[12] Wilson, "FM Radio Transmitter Module Manual - Elechouse," Elechouse, 31 October 2012. [Online]. Available: http://www.elechouse.com/elechouse/images/product/FM%20Transmitter%20Module/FM%20Radio%20Modulator%20Module.pdf. [Accessed 1 December 2020].

[13] M. Islam, R. Mahmud, Kaimujjaman, M. Sultana and M. Hossain, "Digital FM Transceiver Design and Construction using Microcontroller and I2C Interfacing Techniques," *International Journal of Recent Technology and Engineering,* vol. 8, no. 5, 2020.

[14] SM, "Arduino - Wire," Arduino, 24 December 2019. [Online]. Available: https://www.arduino.cc/en/reference/wire. [Accessed 8 February 2021].

[15] ukw100, "Github - ukw100/IRMP - Infrared Multi Protocol Decoder," [Online]. Available: https://github.com/ukw100/IRMP. [Accessed 8 March 2021].

[16] pu2clr, "pu2clr / SI4735: SI4735 Library for Arduino," [Online]. Available: https://github.com/pu2clr/SI4735. [Accessed 20 April 2021].