

FM Radio Technical Report

Box Number 3

Laboratory, Design and Professional Studies
University of Surrey
Department of Electrical and Electronic Engineering

Jordan | Scarlett | Samuel | Rafal



Abstract

The semester two group project for the LDPS IV module in 2020 construct and design an FM receiver based on the specification outlined by the department. The task required constructing the hardware for the FM receiver also the software required to run the receiver, the construction of the functioning was cut short as a result of the COVID-19 pandemic from 23 April 2020. The report details the technical descriptions of the project and also outlines our approach to the task

Contents

1. Introduction to project	3
1.1. Project Objectives.....	3
1.2. Overall Schematic of Project.....	4
1.3. Project teams	4
1.4. Gannt Chart	5
1.5. Explanation of the FM Radio Receiver	6
2. Hardware	7
2.1. Hardware Overview.....	7
2.2. Input Power AND VOLTAGE Regulator	7
2.3. I2C Converter Protocol.....	8
2.4. Microcontroller connections.....	9
2.5. Audio Amplifier	10
2.5.1. <i>Physical implementation of Audio Amplifier Circuit</i>	10
3. Programming	11
3.1. Design.....	11
3.2. Implementation.....	12
LCD Screen	12
3.1.1. VOLUME Buttons.....	13
3.1.2. Channels Buttons.....	13
3.2. Overall schematic	14
Full List Of Function List.....	14
4. Box Design	16
5. PARTS LIST	18
6. Price list.....	19
Intended User Manuel	19
Conclusion.....	19
APPENDIX	20
References	21

1. INTRODUCTION TO PROJECT

1.1. PROJECT OBJECTIVES

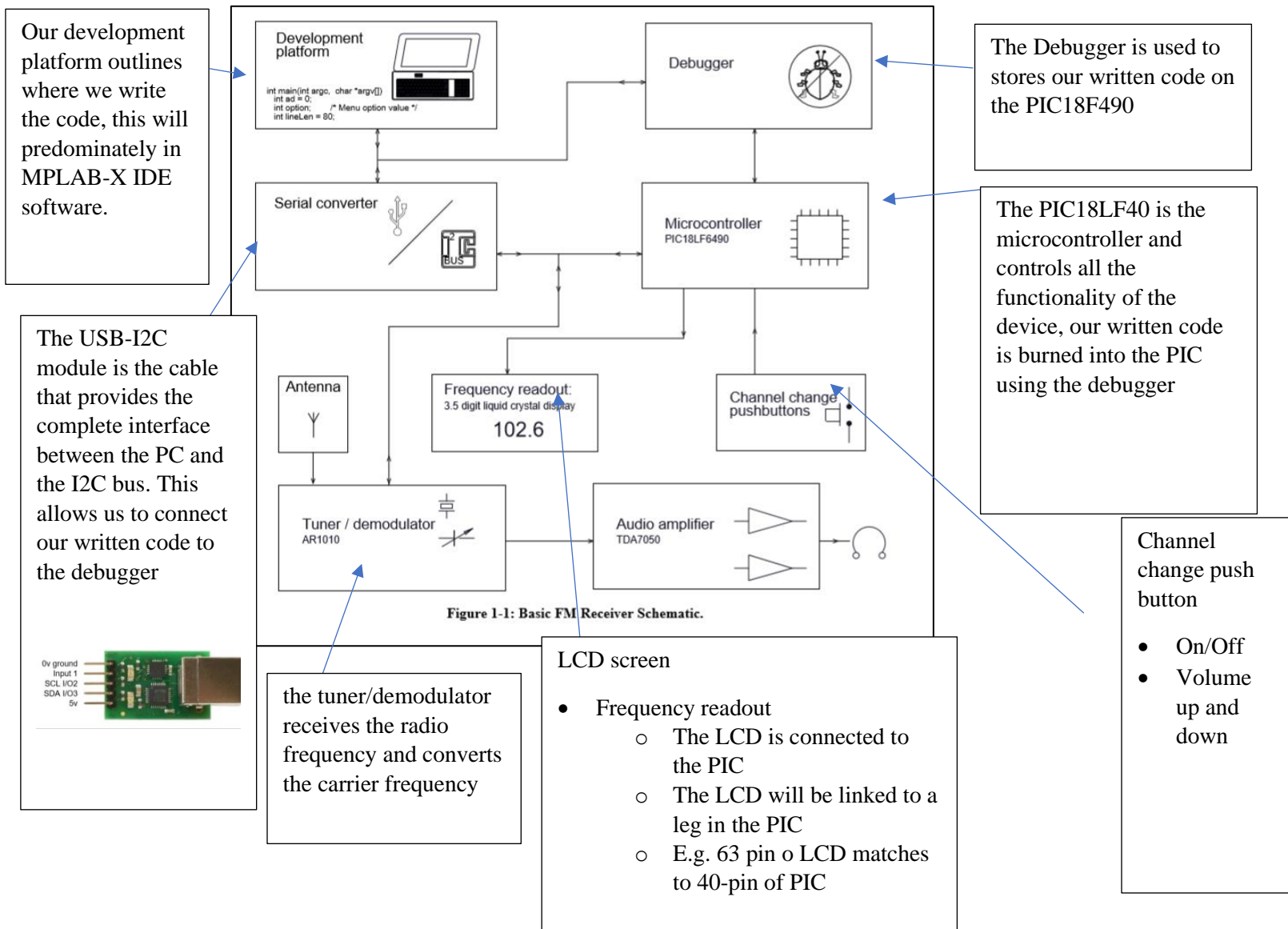
The main objective of the project was to produce a prototype FM receiver. The University provided us with the following components outlined in Table 1. In order to meet the main objectives, the project went through 3 main phases: planning, designing and implementation.

Specification	
Input	Reasonable sized batteries (1.5V AA x3)
Output	3.5 mm headphone jack output, capable of driving a normal set of 35 ohms impedance headphones; and able to give stereo (L & R channels) audio outputs.
Frequency range	Tunable approximately over the UK broadcast band (87.5MHz to 108 MHz)
Provided Hardware Component	
AIROA AR1010 adaptor board	
LE33CZ voltage regulator	
Microcontroller PIC18LF6490	
Viatronix LCD display to show the frequency of the channel	
Push buttons	
Antenna	
Power Amplifier TDA2822	
Matrix Board	
Batteries (1.5V AA x3)	
Headphones	
Circuits built by Hardware team	
Voltage regulator circuit	
Amplifier circuit	
Push button circuit	
Final circuit connecting all the sub circuits and components	
Voltage regulator circuit	
Main functionalities	
<ul style="list-style-type: none"> • Tune up or down a frequency and FM radio should be able to tune to pre-set frequencies • Switch channels manually by pressing a channel change button or by tuning • Volume increases or decreases upon pressing the volume buttons. • Viatronix LCD should be able to display the frequency of the radio station 	

Table 1. showing the parts of the FM radio receiver. (1)

1.2. OVERALL SCHEMATIC OF PROJECT

Figure 1. showing the parts of the FM radio receiver. (1)



1.3. PROJECT TEAMS

Before beginning the project, we outlined each of the constituent software and hardware component provided by the University to aid our understanding of the overall task. Figure 1 depicts how the team mapped out most of the overall components of the tasks to ensure each team member had a good understanding of the overall project. Following this we then split each team member into two teams: hardware and software based on each team member strengths. Each of our roles are described in table 2 below

Table 2 each team members allocated role for the FM Receiver Project

Rafal	Jordan	Scarlett	Samuel
Primarily worked on the software for the PIC18LF6490 and AR1010 including, creating most the code required to work on the PIC18LF6490, error checking	Working on the hardware, including building the test board, working on the audio amplifier, and also managing the general project	Working on the hardware ,including soldering the correct components and error checking with the software team	Working on the software for the PIC18LF6490 and AR1010 chip and designing and assembling the enclosure.

1.4. GANTT CHART

Before beginning the task, our group held a meeting to outline the main components for the hardware, testing and software requirements. The following Gantt Chart below depicts our expectation for the length of the tasks based on each team member skills in relation to their allocated role. We regularly referred to the Gantt Chart, and our progress can be seen in the progress report, which mostly matched our Gantt Chart planning

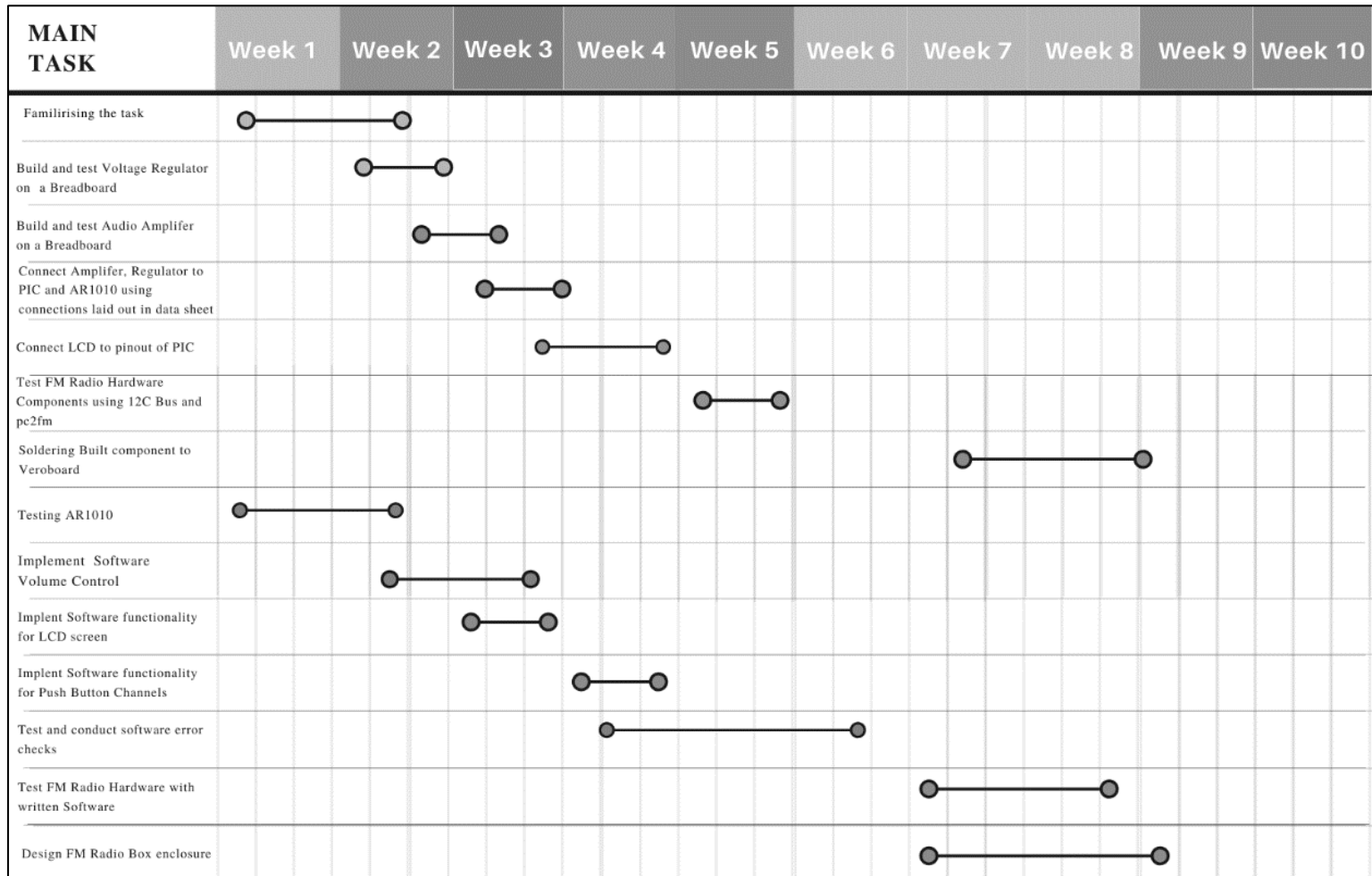


Figure 2 depicting the designed Gantt Chart of most of the outlined stages of the project

1.5. EXPLANATION OF THE FM RADIO RECEIVER

An FM receiver receives radio signals broadcasted on a carrier frequency within the range of 87.5MHz to 108.0MHz using an antenna. An FM receiver then uses electronic filters to separate the desired radio frequency signal from all the other signals picked up by the antenna, following this, an amplifier increases the power of the signal for further processing. The incoming signal needs to first be demodulated, which involves multiple stages including, RF Amplifier, Frequency Mixer, and other hardware processes. We used an AIROHA AR1010 Single Chip FM Radio Receiver to perform these preprocessing stages for us.

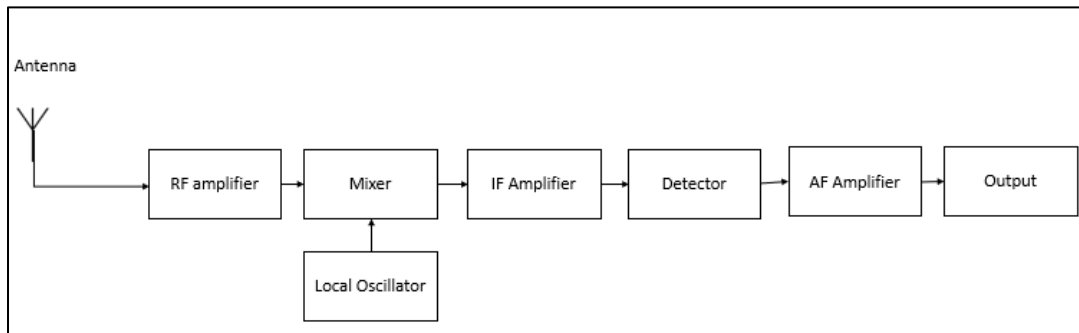


Figure 3. showing the parts of the FM radio receiver. (1)

The FM radio receiver building blocks are shown in the figure above. The working principle of the individual parts can be explained as follows.

The RF Front end → This stage consists of the circuitry between the receiver's antenna input and mixer. This is needed to process the modulated signals received at the antenna into signals suitable for input in the Intermediate Frequency Amplifier. The RF front end includes all the filters to remove undesired signals and a local oscillator which is used with a mixer to change the frequency of a signal. This new frequency is the Intermediate Frequency

Mixer → The mixer is essentially a product modulator, that multiplies the received signal with a local oscillator (LO) signal. A mixer has 3 terminals: RF (Radio Frequency), LO (Local Oscillator) and IF (Intermediate Frequency). The block diagram symbol of a mixer is shown in figure 3b.

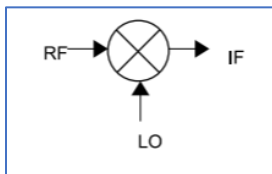


Figure 3b Symbol of a 3-terminal mixer

The Intermediate frequency → The incoming RF is shifted to an IF for amplification before final detection is done.

The Foster–Seeley discriminator → This is a common type of FM detector circuit; it is used to convert frequency changes into amplitude changes. Foster–Seeley discriminators are sensitive to both frequency and amplitude variations, unlike some detectors.

An audio power amplifier (or power amp) → amplifies low-power electronic audio signals to a level that is high enough for driving loudspeakers or headphones.

2. HARDWARE

2.1. HARDWARE OVERVIEW

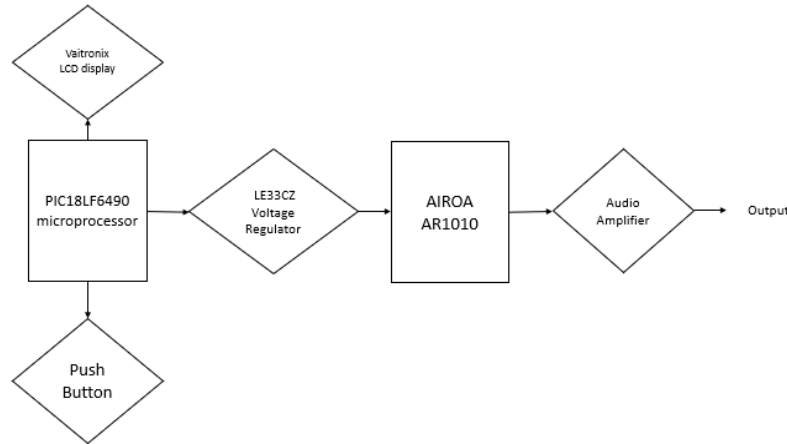


Figure 4 showing the high-level diagram of the hardware components

Figure 4 below outlines the high-level diagram of the circuit components. The square shape represents circuits built by the hardware team, and the kite depicts circuits provided by the University

The hardware team approached this section in a modular manner, meaning that the main components were broken down into three main sections:

1. Connecting Microcontroller to required inputs such as push buttons and LCD pin out
2. Voltage regulator to regulate voltage supply for the AR1010
3. Amplifier output stage, for sufficient power for loudspeaker/headphone

The University primarily provided us with existing circuit designs and pin out layouts for the task. Following the modular approach meant that each individual stage could be first built on a breadboard as reference and then tested individually before soldering on a Veroboard and connecting them together

2.2. INPUT POWER AND VOLTAGE REGULATOR

3 x AA 1.5Volt batteries are used to power the PIC18LF6490 microprocessor, thus resulting in an overall input voltage of 4.5Volts. The data provided to us specifies that the LCD screen achieves good contrast around approximately 5 Volts. However, a problem with this is the 5V required for the microcontroller and LED is too powerful for the AR1010, as this operates at around 3.3V. The AR1010 FM receiver chip is used later in the circuit, thus voltage regulation is required. The LE33CZ regulator IC is used to keep the **voltage** relatively close to a desired value for the LCD and AR1010. Figure 5a outlines the circuit we used to build the voltage regulator.

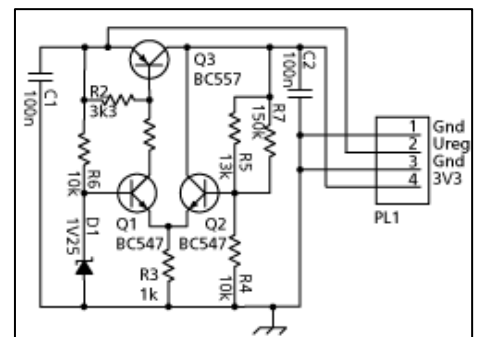


Figure 5a showing the provided Voltage Regulator Circuit

2.3. I2C CONVERTER PROTOCOL

For this project, an Inter-integrate Circuit (I2C) was used to communicate with AR1010. The I2C uses a 2-wire interface because only 2 signals (SDA and SCL) are used for communication.

- SCL is the clock line. This is used to synchronise all data transfers over the I2C bus.
- SDA is the data line. This is used to transmit and receives all data.

There are two devices required in an I2C, a “master” and a “slave”. The device which initiates the data transfer and provides the required clock signal on the I2C bus is the master. The target device that is being addressed to is the slave. In this project PIC **microprocessor** was the **master** and the **AR1010** FM radio chip was the **slave**.

Once the hardware team finished building the regulator, in order to test that the voltage was enough for the AR1010, we used the USB to I2C converter.

During this stage, the software team was still programming the PIC Microcontroller functions, so we used the pc2fm program rather than the PIC. Figure 5b depicts the I2C bus connections we used to test

Our test demonstrated that the regulator was sufficient enough to reduce the voltage to approximately 3.3V for the AR1010.

This meant that we were then able to make the connection between the regulator and the PIC, figure 5c illustrates this

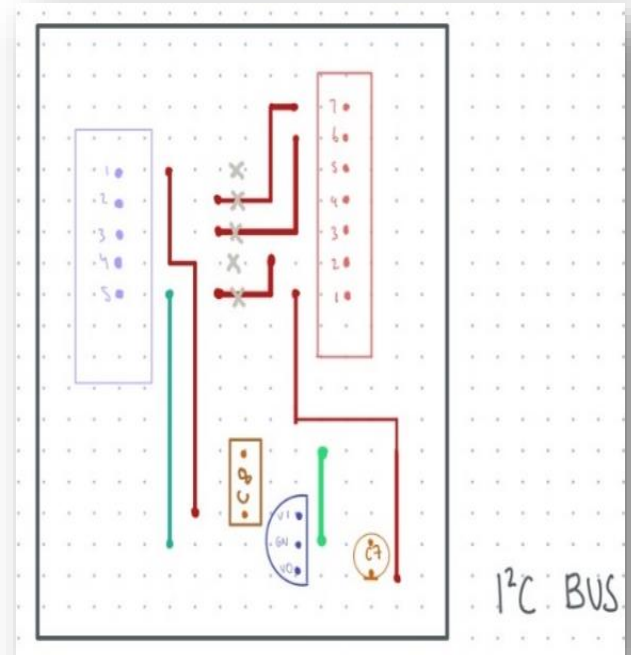


Figure 5b showing the Veroboard layout of the I²C Bus connections

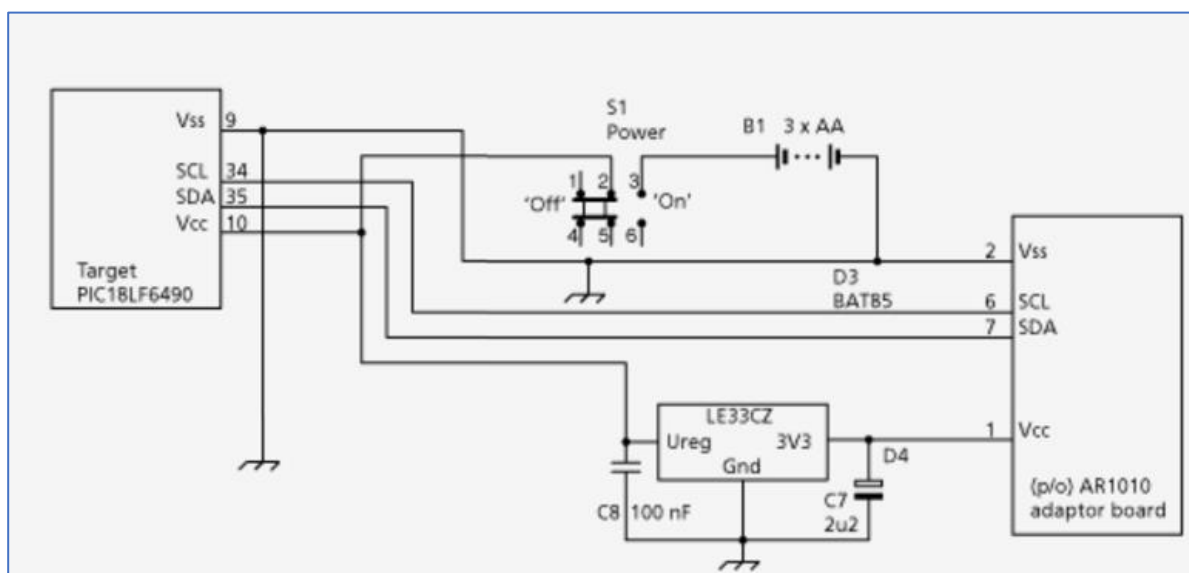


Figure 5c showing the connections for the PIC Microcontroller to the AR1010 using the voltage regulator to reduce the voltage for the AR1010

2.4. MICROCONTROLLER CONNECTIONS

The PIC18LF6490 Microcontroller provided by the University is used to control the overall functionalities of the circuit. This means that all external buttons, switches, and inputs must be connected to the PIC.

The PIC18LF6490 is required to be connected to both the push buttons and LCD in order to successfully control the functionalities. The hardware team sketched out the connections for the LCD's pinout out to the pinout of the PIC using female-female jumper wires. The wires were purchased and listed in our price list. Figure 6a shows the connections required for the LCD and push buttons; this is tabulated in the following page.

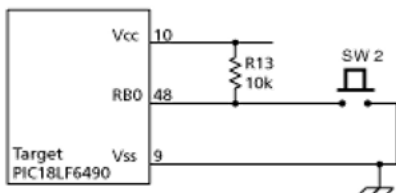


Figure 6b showing the switch input

The schematic for the push button switch was given to us, as seen in figure 6b. The voltage input pin from the PIC18LF6490 to the switch is initially set to high (or 1) as it is connected to a positive Vcc rail (pin 10) across a 10kΩ resistor. If a button is pressed, the voltage would be set to Low as it is instead being connected to the ground rail (Vss). This results in a change of voltage, which is detected by the PIC Microprocessor. The software team, thus process the functionalities dealing the high and low outcomes, which allows us to process how we want to switch buttons to operate

LCD Pin Out

Appendix C outlines the LCD's IDC pinout in tabular form. Figure 6c are the female-to-female jumper wires purchased. We purchased a large set of 120 wires in order to cover all the connections between the PIC, AR1010 and all the other essential hardware components. Once these connections have been made, our software written can thus operate on the segment ordinals.

Figure 6a. showing the written connections for the PIC

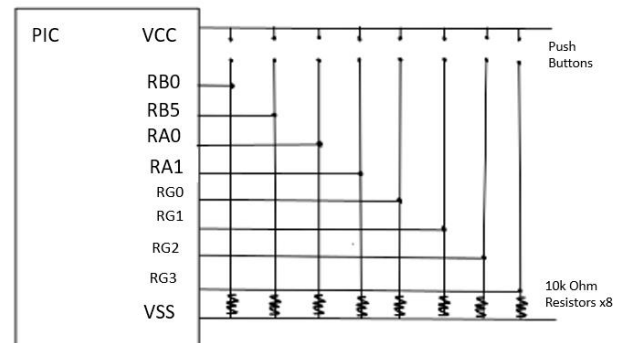
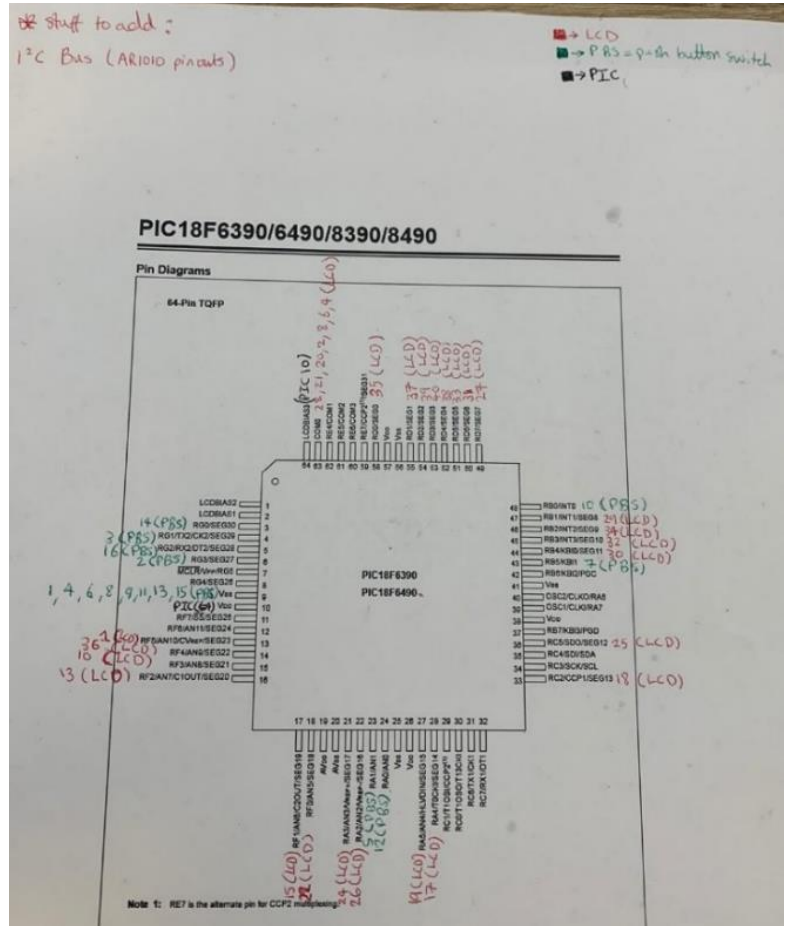


Figure 6c showing the Veroboard layout of the I2C Bus connections

2.5. AUDIO AMPLIFIER

The tasks outline that audio amplification is required to provide a sufficient output gain. Prior to amplification, the provided output signal does not have enough power to drive audio hardware such as speakers. The University provided us with the circuit outlined in Figure 7a for amplifying the output. The amplifier receives the input signal from the AR1010, this input is DC biased. The 10uF coupling capacitors C6 and C7 are used to keep circuit stability and block the DC voltage. A TDA2822 power amplifier is chosen for the circuit, this is a low power stereo amplifier that is also used in Walkman players¹. Also, C8 and C9 are the output capacitors for coupling the strong signals to the speakers/headphone.

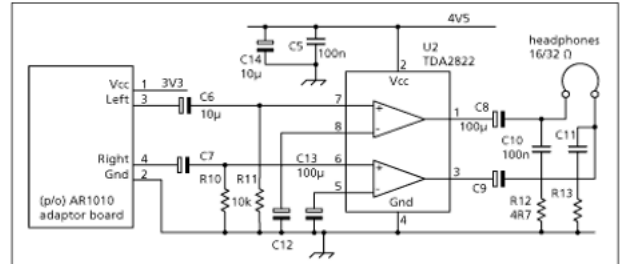


Figure 7a. showing the Audio Amplifier Output

2.5.1. Physical implementation of Audio Amplifier Circuit

Figure 7b demonstrates our Veroboard layout for the audio amplification. This following schematic was implemented on the breadboard on and through testing, successfully produced sufficient output gain around 60-100dB. The above circuit is built on breadboard and tested with the signal generator and oscilloscope. The gain was measured to be 98dB which is in the confined limits mentioned above. This circuit is then built on Veroboard.

The circuit below demonstrates the voltage regulator block connected to the amplification block. This overall was tested with the provide pc2fm code and when connected to frequency 96.4 MHz we could hear a radio station

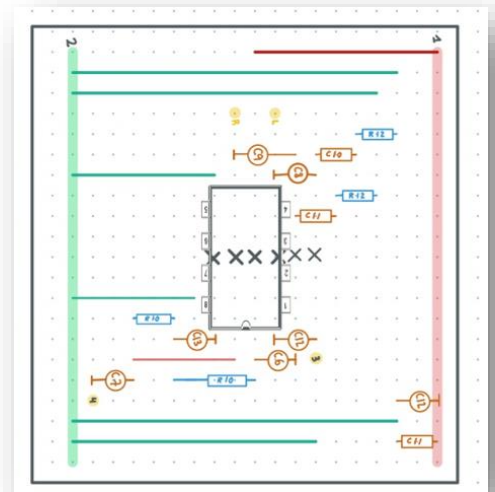


Figure 7b. showing the Veroboard Audio Amplifier Schematic

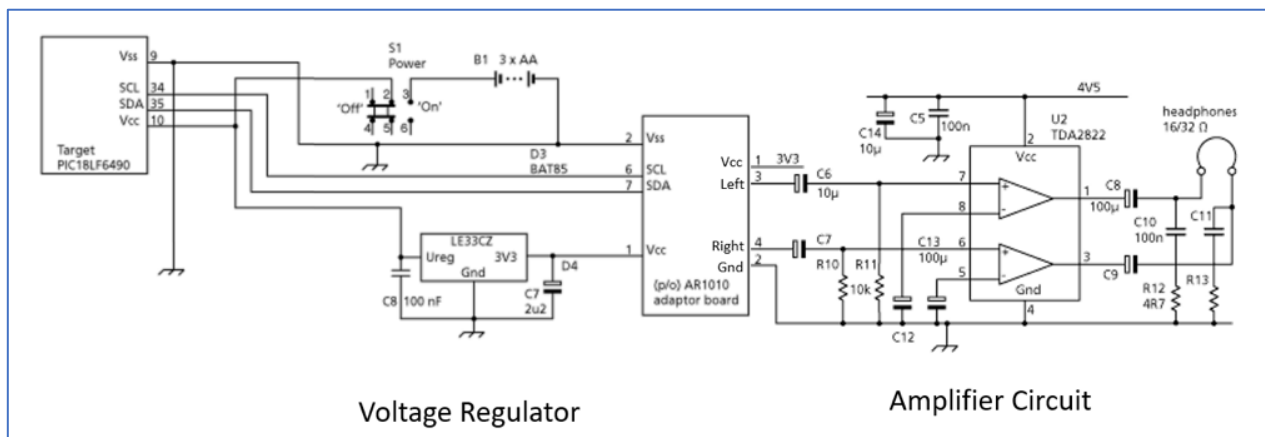


Figure 7c. showing the Circuit schematic with components connected to the PIC

¹ Page 5 of Electronic Project Vol. 14 Book, by EFY Enterprises Pvt Ltd, 2009
<https://books.google.co.uk/books?id=AnUTQ4FT25sC&printsec=frontcover#v=onepage&q&f=false>

3. PROGRAMMING

3.1. DESIGN

The development platform used for the project was MPLAB-X, which is an IDE by Microchip Technology Inc for developing embedded applications on a PIC microcontroller. The programming language of the given file was C Programming. In order to program the PIC18LF6490 microcontroller, MPLAB PICKIT 3 In-Circuit Debugger is used to connect it to a computer with MPLAB-X IDE installed.²

Before we began developing, we studied the register map outlined in Appendix A which describe the bit operations for particular functions. The software team also created a top-down structure chart of the pre-written function calls were made in order for us to fully understand the logic of the overall software. From this we could split the main pre-written functionalities into 3 parts.

1. Obtaining AR1010 Chip version (Blue Region in figure 8b)
2. Initialising the AR1010 and writing empty values in the AR1010 (Yellow Region in figure 8b)
3. Calculating and displaying the current frequency (Green Region in figure 8b)

Top down diagram

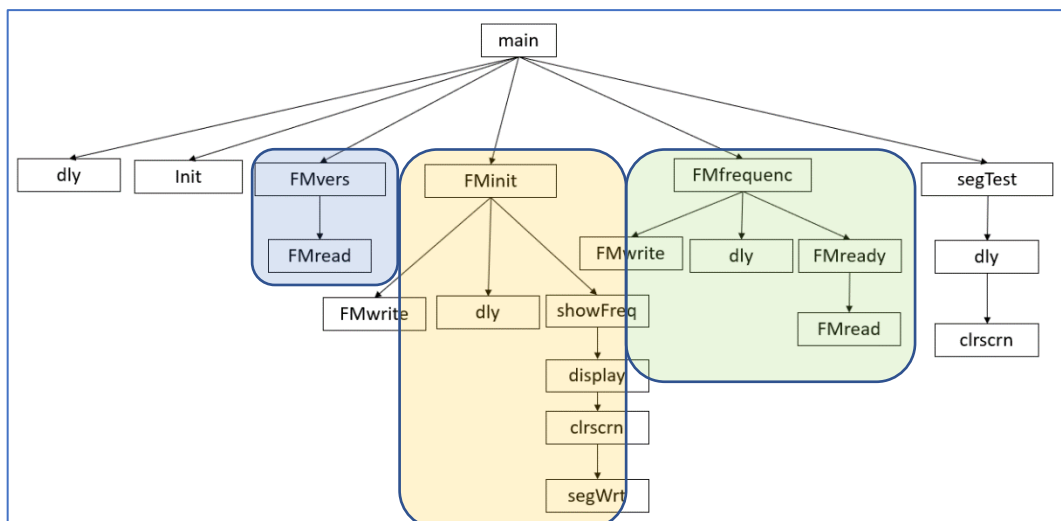
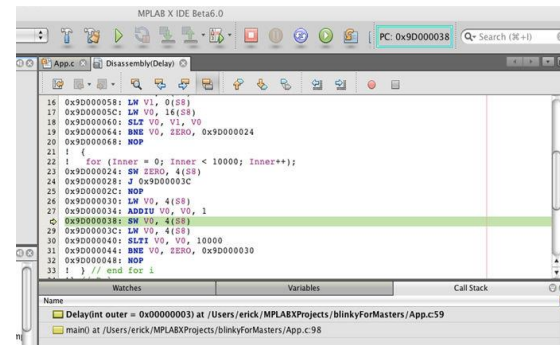


Figure 8b showing Top-down structure chart of pre-written functions

Following this, we outlined the main features required for the software team to program as

1. Displaying the frequency values on the LCD screen
2. Channel control to change the frequency channel
3. Volume control to increment or decrement the frequency

Figure 8. MPLABX IDE



² <http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf>

3.2. IMPLEMENTATION

LCD Screen

To display the frequency channel from the AR1010 FM chip via the PIC microcontroller, a 3.5 digit seven-segment display was used. The controller for the LCD screen is mainly implemented in the `display()` and `showFrq()` functions. The display function takes an argument which is the given frequency intended to be displayed, and using the pre-written function, changes the state of the segment ordinals. Figure 9 outlines the segment ordinals schematic we made for the LCD display.

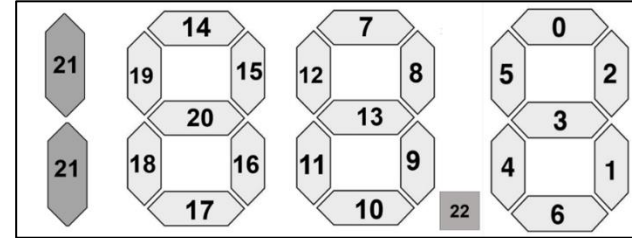


Figure 9 showing LCD Segment Ordinals

Implementation of LCD screen

The `showFrq()` function handles the current frequency channel. In this function, we first check if the current channel is equal to the frequency we want to display, if it is not, we pass the frequency as an argument to the `display()` function.

In the display function, the frequency is split into each individual digit, and each separate digit is stored in a 4-element array. Once we have split each individual digit, all the current values on the screen are cleared and the value we want is displayed. This is done using case statements. As each value is stored in the array, the case statement would switch through each digit in the array, and if the digit meets a case, the corresponding segment ordinals are displayed. Below is an example of the segment ordinals being displayed for 6 and 7.

The `segWrt` function is used to write to a given segment ordinal, to do this hexadecimal signal encoding is used to communicate to with the LCD module. We used a bit wise shift to read the hexadecimal encoding in the code. When the program calls `Segwrt()` with a given ordinal, it sends that hexadecimal signal through a pin to the LCD module as binary data serially, (one bit at a time). Table 3 highlights the hexadecimal encoding for the LCD display

LCD Display	Hexadecimal Encoding
0	0x3F
1	0x07
2	0x5B
3	0x4F
4	0x66
5	0x6D
6	0x7D
7	0x07
8	0x7F
9	0x6F

```

case 6:
    segWrt(0 + pos*7, 1);
    segWrt(2 + pos*7, 1);
    segWrt(3 + pos*7, 1);
    segWrt(4 + pos*7, 1);
    segWrt(5 + pos*7, 1);
    segWrt(6 + pos*7, 1);
case 7:
    segWrt(0 + pos*7, 1);
    segWrt(1 + pos*7, 1);
    segWrt(2 + pos*7, 1);

```

Figure 9b Implementation for writing to segment ordinals

Table 3 showing hexadecimal encoding for LCD display

3.1.1. VOLUME BUTTONS

Implementation of the volume is mostly completed in the VolUp function. This function has arguments which can either be set to “TRUE” for incrementing the volume, and “FALSE” for decrementing volume. All the register bits for the volume is stored in arr[44] as seen in figure 10.

The global variable “volume” is used to store the current volume. In our code we created an array of the registers at each volume level, and the positions in the array corresponds to a given volume level. The first ‘n’ volume level is indicated in the array using arr[n] and the second volume is indicated by arr[2n].

```
unsigned char arr[44] = {
    0xf, 0xf, 0xf, 0xf, 0xf, 0xe, 0xe, 0xd, 0xd, 0xb, 0xa,
    0x9, 0x7, 0x6, 0x6, 0x5, 0x5, 0x4, 0x4, 0x2, 0x1, 0x0,
    0x0, 0xc, 0xd, 0xe, 0xf, 0xe, 0xf, 0xe, 0xf, 0xf, 0xf,
    0xf, 0xf, 0xe, 0xf, 0xe, 0xf, 0xe, 0xf, 0xf, 0xf
};
```

Figure 10. Array of volumes levels stored

In Appendix A the register map outlines that there are two volume control fields in the AR1010 register, Volume in register 3 (D7-D10) and Volume2 in register 14 (D12-D15). The definition ‘FMASKNOVOL’ is used to indicate register 3 bits 7-10 for the first volume, and ‘FMASKNOVOL2’ is used to indicate register 14 bits 12-15. Once the register bit values are obtained for the corresponding volume, we used the variable ‘temp’ to store the binary value of the current volume, and we can then shift the volume by the appropriate number of bits to the left to increment or decrement

3.1.2. CHANNELS BUTTONS

Before Changing the channel was implemented using nextChen() function, which takes a Boolean argument. Depending on it, it goes to the next or previous channel. It uses FMfrequenc() (already implemented) and showFreq() functions.

We defined ‘FMASKRDCHAN’ in the header file to indicate the register bits 2, which should indicate the channel number bits as specified in the register map in Appendix A. From this we then assigned the “chan” variable to the current register channel in bits.

Increment or decrementing the channel is decided based on the argument to the function. Before the channel is incremented or decremented, the program would for overflow, to ensure the channel is within the specified range (87.5MHz to 108 MHz). The frequency is then changed by calling the FMfrequenc(), which first performs the operation below so the frequency can be fed into the AR1010 chip

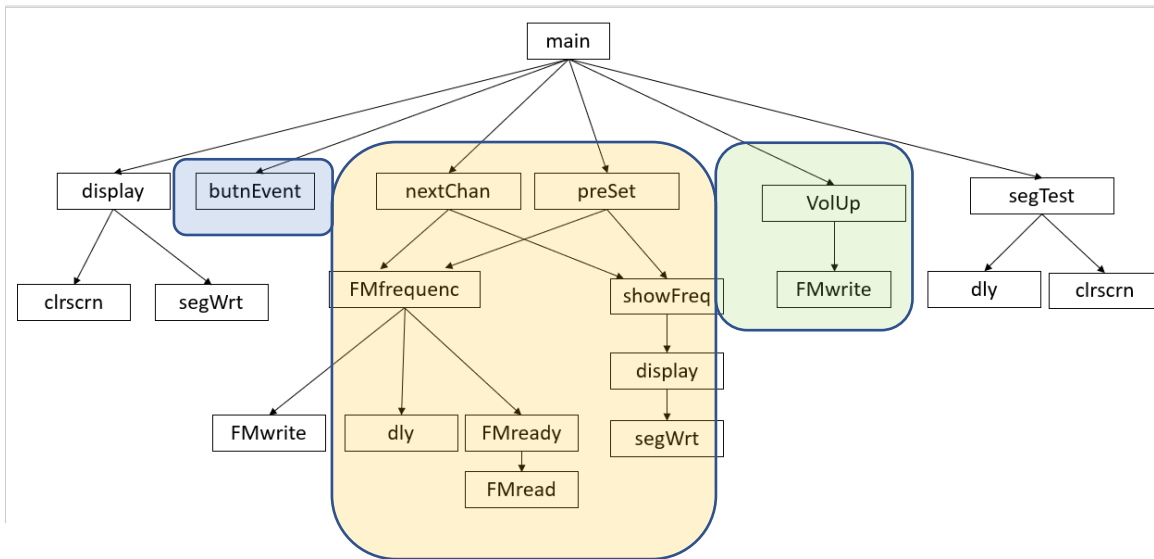
$$Freq(MHz) = (690 HAN)/10$$

Finally, calling showFreq() function is then used to show this frequency on the LCD display.

3.2. OVERALL SCHEMATIC

Figure 11 illustrates another top down diagram of our user written functions in the main. The blue highlight indicates our written function for the buttons, the orange indicates the channel button and the green indicates the volume button. The chart helps us highlight the logic our written code

Figure 11 showing the top down level diagram of our written functions from the main



FULL LIST OF FUNCTION LIST

Function	Explanation
<code>unsigned char butnEvent(unsigned char *butn)</code>	<p>This function is used to obtain the latest change in state for the pushbutton set. The ‘butn’ parameter is used to define which button has changed</p> <p>The function should return</p> <ul style="list-style-type: none"> • 0 if no button has changed state, • 1 if button is pushed, • 2 if button is released.
<code>void dly(int d)</code>	Purpose of the function is to define the clock. The function takes an argument ‘d’ and delays the operation of the program for d milliseconds using a loop
<code>void clrscrn()</code>	Purpose of the function is to clear the LCD screen by setting all LCD segments to 0.
<code>void Init()</code>	Purpose of this function is to initialize the PIC184LF Microcontroller with designated register bits.

<code>void segWrt(unsigned char segOrd, unsigned char state)</code>	Purpose of this function is to write an individual LCD segment. The first argument is the segment ordinal between 0-22. The second argument, 'state', is used to determine if the segment is ON (true) or OFF (false)
<code>unsigned int display(unsigned int freq)</code>	This function is used to set which given segment Ordinals are to be set to ON or OFF based on a given digit and position.
<code>unsigned char FMwrite(unsigned char adr)</code>	Purpose of function is to write to the AR1010 register and return XS, '1', on success of XF, '0', on error
<code>unsigned char FMread(unsigned char regAddr, unsigned int *data)</code>	Purpose is to read a two byte of the AR1010 register specified by the regAddr parameter and store the read bits into the pointer *data
<code>unsigned char FMready(unsigned int *rdy)</code>	This function is used to see if the AR1010M FM module is ready. The rdy parameter is used to store the busy/ready status. The parameter will become non-zero if the chip is ready, zero if busy.
<code>unsigned char FMinitt()</code>	This function initializes the AR1010 register with empty values
<code>unsigned char FMfrequenc(unsigned int f)</code>	Purpose of this function is to tune the AR1010A to a new frequency. The function takes an argument 'f' which would be the new frequency as a multiply of 100kHz. To obtain the channel, the function follows the operations specified in the data sheet
<code>unsigned char FMvers(unsigned int *vsn)</code>	The purpose of this function is to obtain the AR1010 chip version.
<code>unsigned char FMid(unsigned int *id)</code>	This function obtains the FM chip ID. With the parameter id being the ID number. This is 0x1010 for AR1010 devices.
<code>void errfm()</code>	This is the e function called if there is a firmware error firmware. Called mostly if XF is returned in a function called. Content of the function is it calls an infinite for loop so
<code>unsigned char showFreq()</code>	Display the frequency that the receiver chip is set to. @return XS if successful, else XF on failure.
<code>unsigned char nextChan(unsigned char up)</code>	The nextChan() function is to tune to the next channel. Function returns XS on success XF on error.
<code>unsigned char volUp(unsigned char vup)</code>	This function function is used to implement the volume control, here there is an array of values of the VOLUME register bits for each particular volume level.
<code>unsigned char segTest(/*unsigned char test_arg*/) </code>	This is a pre-written code that is used to-used to test the LCD screen
<code>unsigned char preSet(unsigned char next)</code>	Purpose of this function is to enable the user to choose the station from the predefined set.

4. BOX DESIGN

As a group we decided to attempt a retro design for our FM radio box, to do this we decided to opt for a wooden chassis and design. Below is an example of inspirations³, the two boxes depict the look we would hope to achieve from our box design⁴ Following this we broke down the designs the designs into the following components: front view, back view, side view, top view, and bottom view

Mechanical Design of enclosure

The enclosure was designed using the functionalities of Microsoft Office. For our first design, we designed the box with the main measurements. Several materials can be used for constructing the case of an FM transmitter, but wooden sheets were adopted for this project, we hoped to glue these wooden sheets above a metal enclosure.



Figure 12 showing box inspiration

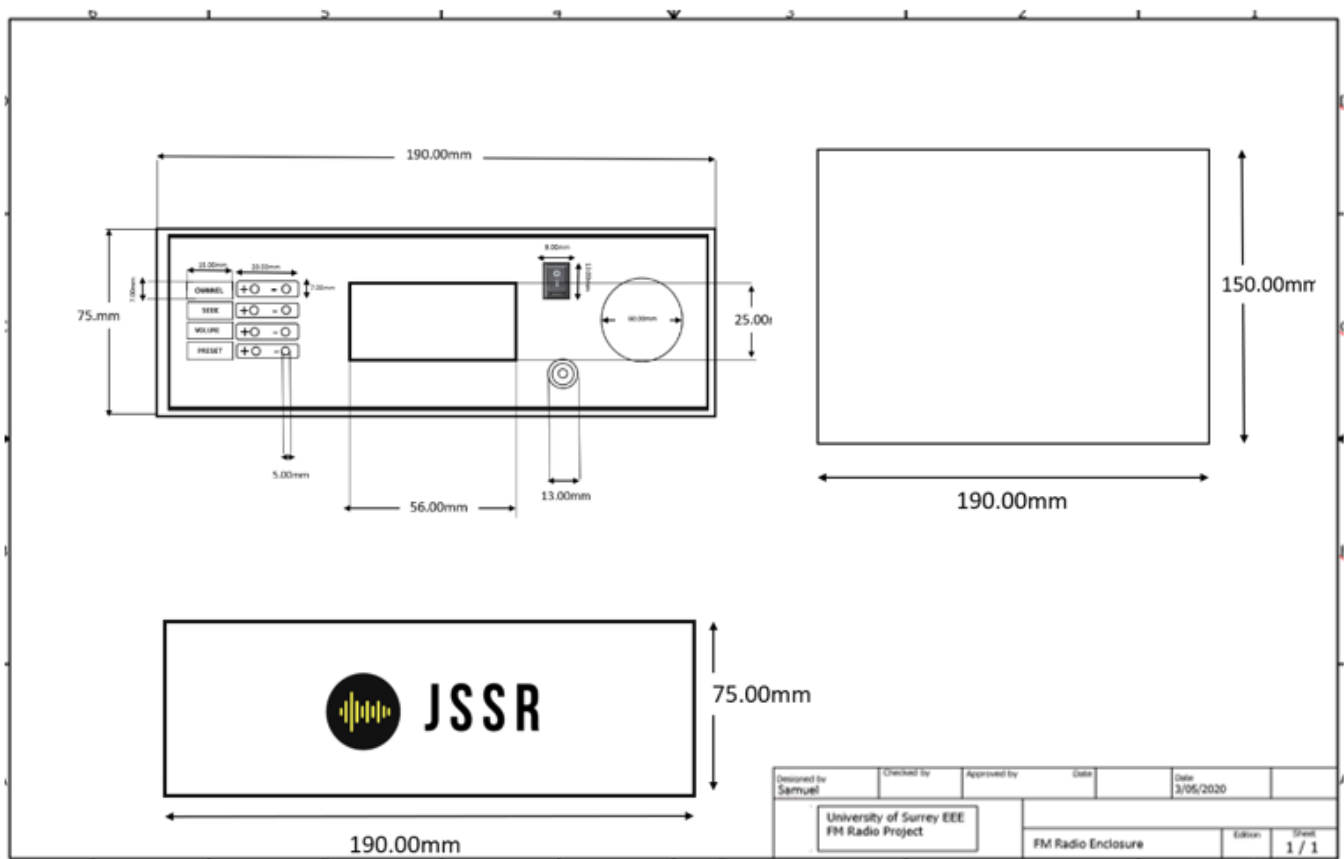


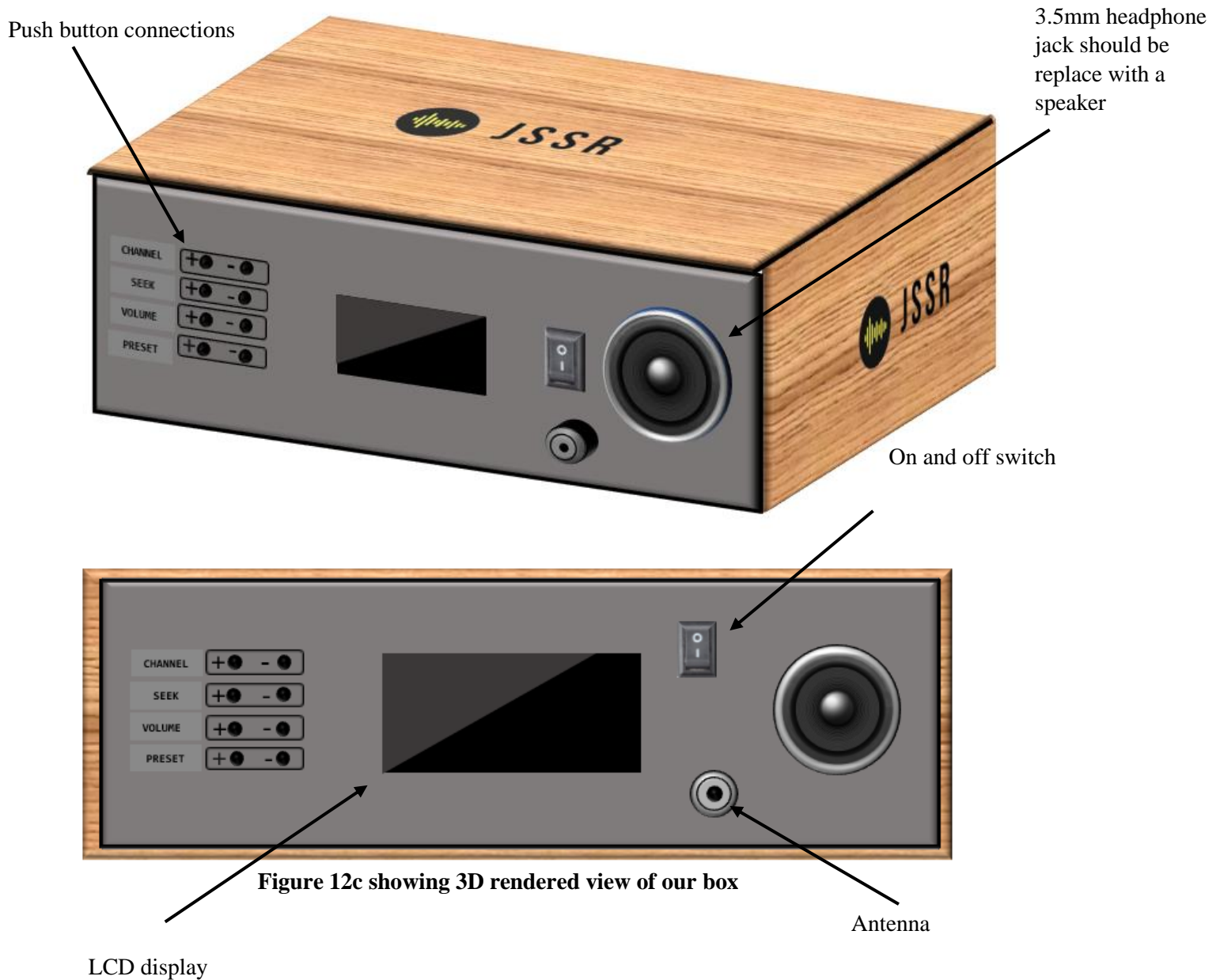
Figure 12b showing the measurements of our box

³ <https://www.amazon.co.uk/JYL-Vintage-Multimedia-Classical-Receiver/dp/B07W4PYFQQ>

⁴ http://www.imi.eu/promotional-items/am-fm-radio-with-elegant-wooden-design-packed-in-a-coloured-gift-box-colour-silver-brown_74629/

Rendered Box

Figure 12c illustrates the aesthetic we would like our box to follow with the wooden chassis. After final consideration, we hope our rendered box we hope follows this given look.

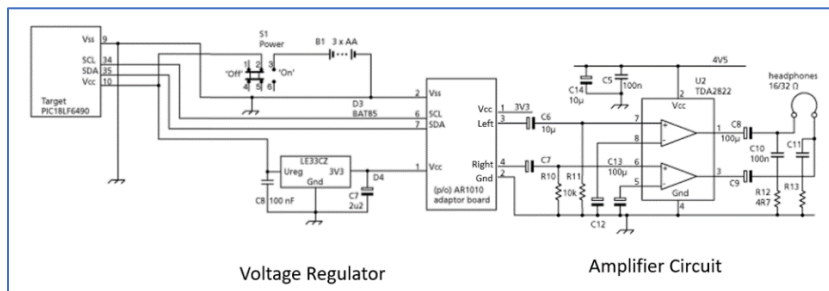


5. PARTS LIST

Microcontroller and Push Button listing

Component	Notation	Value	Quantity
Resistor	R0	10k Ω	X8
Push buttons	P1	N/A	X8
LCD Screen with Jumper Wire Connector	N/A		X200

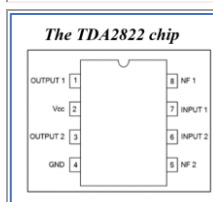
Full circuit for reference depicted below again



Voltage Regulator and Power Block

Component	Notation in power block	Value	Quantity
Capacitor	C7	2.2u	X1
	C8	100n	X1
Battery	B1	1.5V AA	X3
SPST Power Switch	S1	NA	X1
Voltage Regulator	LE33CZ		X1

Audio Amplifier Component listing

Component	Notation in amplifier block	Value	Quantity
Capacitor	C5, C10, C11	100n	X3
	C6, C7	10u	X2
	C8, C9	100u	X2
Resistor	R10, R11,	10k	X2
	R12, R13	4R7	X2
Power Amplifier	TDA2822D Dual Voltage Power Amp	 <p>The TDA2822 chip</p>	X1

6. PRICE LIST

The associated with the wooden sheets are listed below

Item	Wooden sheets 22400 x 1000cm
Spec	The wooden strips are used for the design of the box to match the rendered design
Quantity	The sheets are cut to size, so quantity is x1
Source	https://www.woodsheets.com/plywood-sheet-cut-to-size/

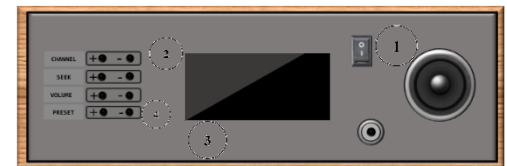
The associated with the f2f jumper wires are listed below for connecting modulus to the PIC

Item	Jumper Wire Cable 40 pcs F2F20 cm (8 inches)
Spec	Used for connecting to LCD screen
Quantity	In total x3 (120 Jumper wires)
Source	https://www.amazon.co.uk/AZDelivery-Jumper-Wire-Cable-Parent/dp/B0813X4ZPR

INTENDED USER MANUEL

To switch on the FM Radio, press the on and off switch indicated by (1)

1. To tune to a channel, press the buttons indicated by (2) till your desired frequency is displayed on the LCD screen (3)
2. To increase the volume, press the volume button indicated by (4)
3. The seek button is used for searching for station to station frequencies. Pressing the + button or – is used to search up or down station. Once the tuner finds a station a station whilst tune, it stops the seek function



No sound	Adjust volume controls by pressing + or – in the volume button (10)
Poor signal	Possibly due to low signal reception. Consider adjusting the Aerial or moving the FM Box till better signal is picked up

CONCLUSION

Comment on task

Overall, we believe we successfully built each individual hardware and software component required to meet our objectives specified in Table 1, we had also planned to include some additional novelties in our project. The Technical Report outlines the objectives set for the project and our approach. As specified above the tasks was divided into software and hardware, but these did not limit our roles in helping each other out when challenges occurred. We did plan to move to rigorous testing once we returned from the Easter break, however, our progress was cut short due to the COVID-19 Pandemic in 2020. We believe had this had not occurred, we would have had a working FM Receiver with a designed enclosure and our intended progress is outlined in the Progress Report. Overall, through the task did also build on our programming knowledge through the use of embedded C

Programming to control the PIC and AR1010 receiver and also encouraged the group to think critically. A significant amount of planning was required in order to meet the objectives before the deadline, and the Gannt chart illustrates this.

Technical challenges

There were some technical challenges that arose from the project, each specific to the individual hardware and software teams. These challenges included

1. Familiarising with embedded coding
 - a. Properly understanding how the registers, bits and high-level C code were connected did take some time
2. Debugging
 - a. Each individual part did at some point result in errors, for example the LCD screen code. This did require a lot of time debugging and addressing the issues

However, through working together and successful debugging we managed to address all challenges.

APPENDIX

Appendix A -AR1010 FM Radio Register Map

ADDR/Alias	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
00H R0									no_en							EN
01H R1			rds_en							rds_int_en	stc_int_en	deemp	mono	smute	hmute	
02H R2										TUNE						CHAN<8:0>
03H R3	SEEKUP	SEEK	SPACE	BAND<1:0>				VOLUME<3:0>								SEEKTH<6:0>
04H R4																
05H R4																
06H R6																
07H R7																
08H R8																
09H R9																
0AH R10														seek_wrap		
0BH R11	hilo_side													hiloctr_b1		hiloctr_b2
0CH R12																
0DH R13												GPI03<1:0>		GPI02<1:0>		GPI01<1:0>
0EH R14				VOLUME2<3:0>												
0FH R15													rds_sta_en	rds_mecc<1:0>		rds_ctrl
10H R16																
11H R17																
12H RSSI																IF_CNT<8:0>
13H STAT																READCHAN<8:0>
14H RBS																RDSR STC SF ST
15H RDS1																RBS1<1:0> RBS2<1:0> RBS3<1:0> RBS4<1:0>
16H RDS2																RDS1<15:0>
17H RDS3																RDS2<15:0>
18H RDS4																RDS3<15:0>
19H RDS5																RDS4<15:0>
1AH RDS6																RDS5<15:0>
1BH DEVID																rds_dsc<15:0>
1CH CHIPID																rds_dfc<15:0>
																VERSION<3:0>
																MFID<11:0>
																CHIPNO<15:0>

Appendix B -Push Button Pin Out in tabular form

PIC18F6390 Pin Name	PIC18F6390 Pin Number	Pull-up pin	IDC Pin	Button
RBO/NT0	48	2	10	Chan+
RB5/KB11	43	3	7	Chan-
RA0/AN0	24	4	12	PreSet+
RA1/AN1	23	5	5	PreSet-
RG0//SEG30	3	6	14	Vol+
RG1/ TX2/CK2/SEG29	4	7	3	Vol-
RG2/RX2/DT2/SEG28	5	8	16	SegTest
RG3/SEG27	6	9	2	Error
Vss	9	-	1,4,6,8,9,11,13,15	Switch return

Appendix C -LCD Pin Out in tabular form

PIC name	PIC pin	IDC pin	LCD pin	Varitronix name
RD0_SEG0	58	35	21	3A
RD1_SEG1	55	37	20	3B
RD2_SEG2	54	39	19	3C
RD3_SEG3	53	40	18	3D
RD4_SEG4	52	38	17	3E
RD5_SEG5	51	33	22	3F
RD6_SEG6	50	31	23	3G
RD7_SEG7	49	27	25	2A
RB1_SEG8	47	29	24	2B
RB2_SEG9	46	34	15	2C
RB3_SEG10	45	32	14	2D
RB4_SEG11	44	30	13	2E
RC5_SEG12	36	25	26	2F
RC2_SEG13	33	18	27	2G
RA4_SEG14	28	17	30	1A
RA5_SEG15	27	19	29	1B
RA2_SEG16	22	26	11	1C
RA3_SEG17	21	24	10	1D
RF0_SEG18	18	22	09	1E
RF1_SEG19	17	15	31	1F
RF2_SEG20	16	13	32	1G
RF3_SEG21	15	10	03	K
RF4_SEG22	14	36	16	DP3
RF4_SEG23	13	01	38	Z
COM0	63	28	12	DP2
COM0	63	21	08	DP1
COM0	63	20	28	COL
COM0	63	02	39	X
COM0	63	08	02	Y
COM0	63	06	01	COM
COM0	63	04	40	COM

REFERENCES

1. Appendix A - Supplementary External Documentation PIC18F6390/6490/8390/8490 Datasheet <http://ww1.microchip.com/downloads/en/DeviceDoc/39629c.pdf>
2. AR1010 Datasheet http://www.rockbox.org/wiki/pub/Main/ArchosVision24c/AR1010_brief.pdf
3. TDA2882D Power Amplifier Datasheet <http://pdf.datasheetcatalog.com/datasheet/stmicroelectronics/1463.pdf>
4. LE33CZ 3.3V Voltage Regulator Datasheet <http://www.st.com/content/ccc/resource/technical/document/datasheet/98/09/12/54/6e/d1/45/36/CD00000545.pdf/files/CD00000545.pdf/jcr:content/translations/en.CD00000545.pdf>