19th May 2020

FM Radio Progress Report

Box Number 3

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

Jordan | Scarlett | Samuel | Rafal

Hardware

Current PROGRESS OF Hardware

We used a modular method to approach the hardware section of the project, this was done by splitting the main circuitry components required to be built or connected between Scarlett and Jordan. Our initial task included: familiarizing with the hardware components, setting up the circuit configurations as well as recognizing the relevant connections between the PIC and the other provided circuits.

1.1. POWER REGULATOR

Whilst we were waiting for the software team to finalize the code required of the PIC functionalities, we began working on the power regulator. By following the schematic illustrated in Figure 1 and using the LE33CZ voltage regulators that were provided, we connected each component to the AR1010 adaptor board. As specified before, the software functionalities of the PIC were not finalized yet, so we thus decided to connect it to the USB to I2C converter which uses a pre-complied pc2fm program as opposed to the PIC. This therefore allowed us to power our circuit using a PC as a power generator and download the pc2fm software directly into the circuit for testing purposes. Figure 2 outlines the connections for connecting the AR1010 to the USB to I2C as opposed to the PIC18LF6490

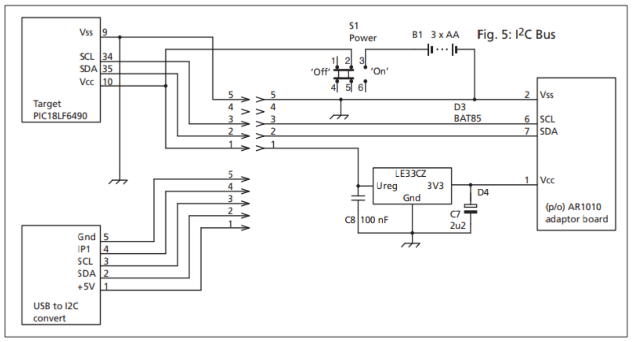


Figure 1 – Connections for either the PIC Microcontroller or USB to I2C converter to the voltage regulator and AR1010

1.2. TESTING THE POWER REGULATOR.

After checking all the connections were correct, to ensure the voltage regulator does not burn, we decided to plug in a USB cable from the computer to our circuit. Using a DVM, we measured the output voltages across the ground and the voltage regulator component; thus, we got the output we expected: 3.3 Volt.

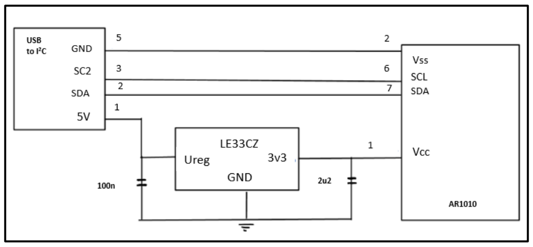


Figure 2 -Voltage Regulator Circuit with USB to I2C

2.1. AUDIO AMPLIFER

After setting up the first part of the circuit. We moved onto the audio amplifier part of the FM radio. Using the provided circuit for the sake of simplicity, the audio amplifier was built on the breadboard as well for testing purposes.

The audio amplifier was fairly easy to set up and build. By using a built-in audio amplifier that was given to us (TDA2822), and soldering it, we were able to use it in our circuit and thus saving a lot of time. Since we were only testing if the audio amplifier worked, we did connect it to the AR1010 board during this stage.

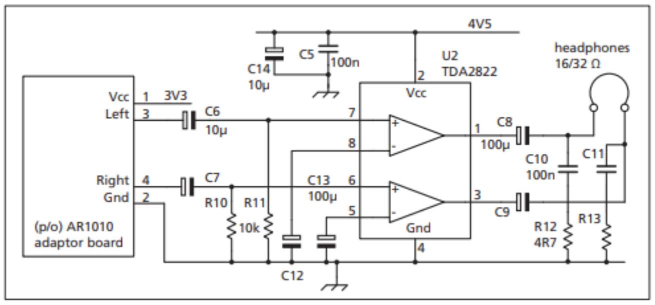


Figure 3 -Audio Amplifier Circuit Schematic Provided by the University

2.2. AUDIO AMPLIFIER TESTING.

With the circuit built, we substituted the adaptor board with a 3.5mm socket. After soldering a socket and connecting it into the circuit, where the adaptor board was supposed to be, we tested that the audio was working.
In order to find out the gain of our circuit, we also decided to connect a function generator, and check the output and input gain. The gain we got was: 98 dB.

2.3. FM RADIO TESTING

With each part of our FM radio working now, we decided to move on by implementing these two circuits together and downloading provided software to test if our circuit was working. We connected the Left(3) and Right(2) inputs of our audio amplifier to the corresponding part of the AR1010 adaptor board (3) and (2) pins respectively.

Now it was time to power our whole circuit up, to check if the final prototype was working.

We connected a USB cable to our I2Cbus circuit. Then we opened the pc2fm.exe file to run the program. By checking that the pc2fm program was working with the corresponding COM ports, we downloaded the software into our circuit and ran the radio program.

The FM radio channel that was pre-set was working properly thus our breadboard prototype was working.

During this stage, the software team had nearly completed the main functionalities for the PIC18LF6490, so we thus created the schematic in figure 4 to connect our working components to the PIC18LF6490 as opposed to the USB I2C

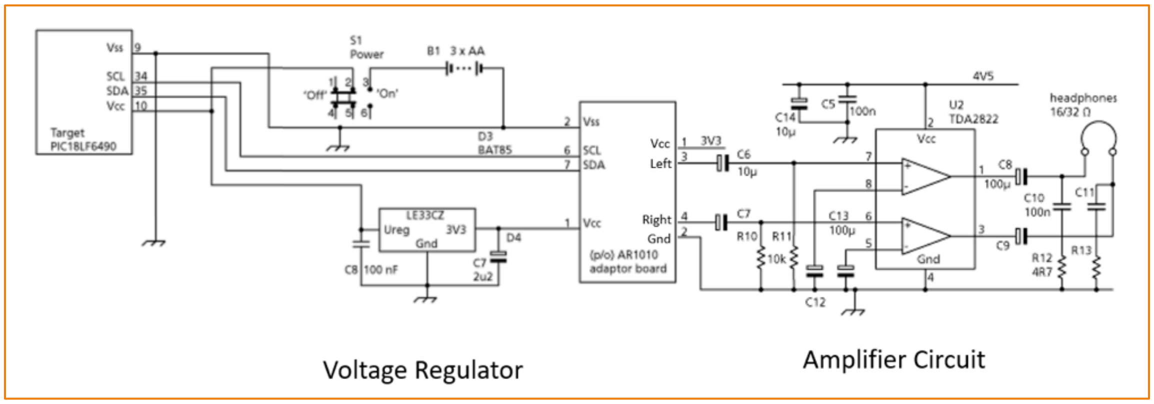


Figure 4 – Full Circuit Schematic, showing the PIC Microcontroller connected to the Power Regulator, AR1010 FM Receiver, and Audio Amplifier

3. VEROBOARD

After the final breadboard prototype was working, we decided to design a Veroboard layout for the final project and test if our prototype would work on Veroboard as well. Below are the Veroboard designs.

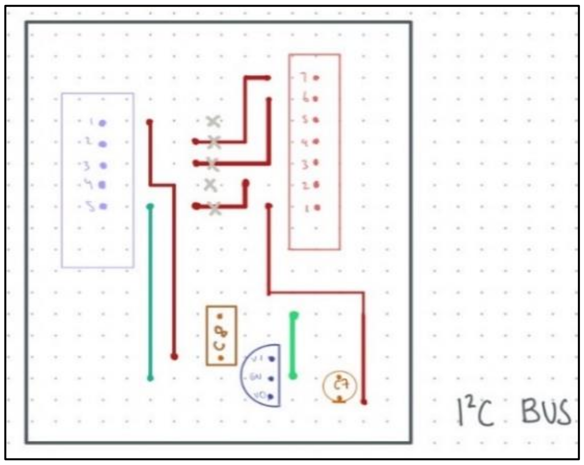


Figure 5 – Veroboard design for the I2C Bus Connections

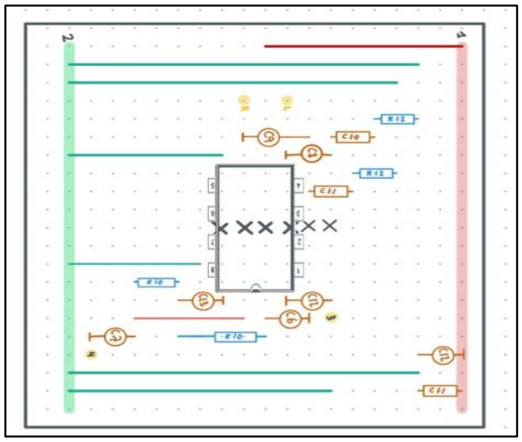


Figure 6 – Veroboard design for the Audio Amplifier Schematic

PLAN TO COMPLETE OF HARDWARE

Although we made great progress overall on the hardware components, the COVID-19 halted the continuation of physical implementation of the FM Receiver Project. We had built and tested the audio amplifier, voltage regulator on the breadboard and the Veroboard. We planned to move on to connecting the components onto the matrix board using dot matrix connections.

1.1. Week 8

For week 8, we planned to test the connects of the LCD's IDC pinout to the pinout of the PIC using female-to-female jumper wires. We purchased a large set of 120 wires which would have covered all the connections between the PIC, AR1010 and all the other essential hardware components.
After connecting the LCD to the PIC, we would test the input software code provided by the software team before thinking about soldering the components down on the matrix board. Also, we would make more progress on improving the box.

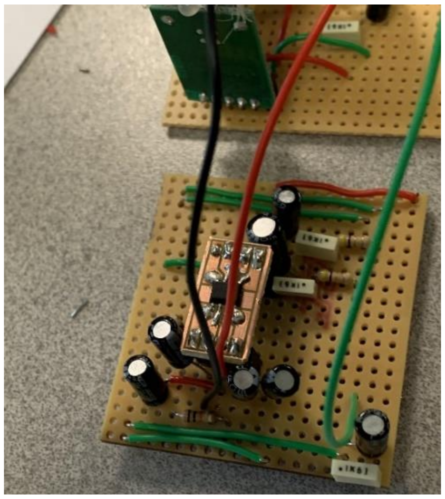


Figure 7- Audio amplifier built on Veroboard connecting to I2C circuit

Figure 8- The purchased f2f jump wires.

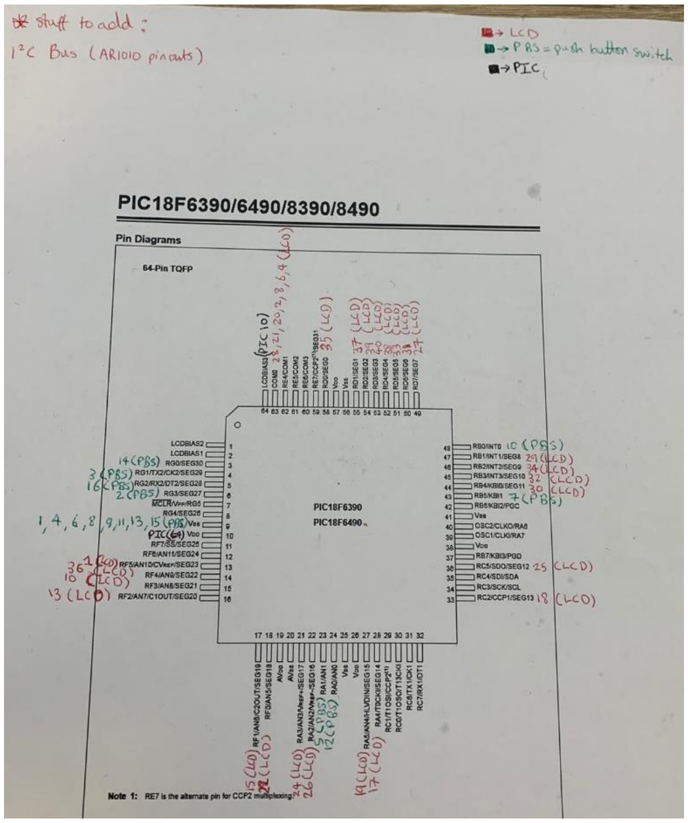


Figure 9- The ICD pinout connections of the push buttons and LCD to the PIC.

1.2. Week 9

By week 9, we would have proceeded to test the pushbuttons code on the PIC also using the female-to female cables. Provided the code works after any necessary troubleshooting, we would solder this down onto the matrix board. More enhanced improvements would also be made on the box within that time period. Once all the soldering of the hardware connections to the PIC are complete, we will move in to fit all the parts together on the matrix board such as the AR1010 and input switch. All components would be efficiently placed in the box and tested with a battery supply to see if further troubleshooting is required for the hardware or software part of the project.

1.3. Week 10

Week 10, we would look to finalise the box by adding labels and our group logo onto the box. Further necessary trouble shooting would be done if needed.

Figure 9- Group Logo.

Software

The work was split between Rafal and Samuel. Our first task was familiarizing with the software. And setting up the development platform as well as recognizing the relevant functions required to run the software such as required buttons and LCD channels. Our complied code is also submitted for reference to see our current progress

CURRENT PROGRESS OF SOFTWARE

1.1. VOLUME

We have managed to implement volume buttons. They are calling function volUp(), with appropriate argument, which is "true" for incrementing the volume, and "false" for decrementing it. For code tidiness, inside the function, there is an array of values of the VOLUME register bits, for each volume level. Additionally, there is a global "volume" variable, which indicates the current volume level

The working process of that function can be described in several stages:

1. "Volume" variable is incrementing or decrementing, depending on the input argument. It is checked for a proper band (0-22) as well.
2. Mute bit is cleared and set to "true".
3. Changes on two VOLUME register bits:
    a. VOLUME bits are cleared.
    b. Value for current volume level for this register bit is written from the array and assigned into the temporary variable "temp".
    c. Binary value from "temp" is shifted by the appropriate number of bits to the left (to match VOLUME register bits).
    d. To the proper register bits "temp" value is added.
    e. Whole register is written to AR1010.
    f. Tune bit is cleared and set to "true"

It should result in a short moment (too short to hear) of silence and then return to normal functionality with a different volume level.

1.2. <u>LCD Screen</u>

We have implemented controller for LCD screen as well. It is done by display() function, which takes frequency (in 10.000Hz) or volume level as an input. There are 2 already written functions used - segWrt(), which change state of a segment ordinal, and clrscn(), which clears the screen.

The idea of working can be described in stages:

1. Dividing frequency input into 4 separated digits.
2. Clearing the screen.
3. Depending on whether the argument was a volume level or frequency - lighting up the decimal point and changing limit of the loop or not.
4. Looping through digits and calling appropriate segWrt() functions.
5. If frequency – checking the most significant digit – in case of 0 it does nothing

It should result in a proper frequency or volume level shown on the LCD panel.

1.3. <u>CHANNEL CHANGE</u>

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.

There are a few stages of this function:

1. Assigning "chan" variable to the current channel.
2. Switching to the proper part of code depending on the input argument.
3. Increasing or decreasing the "chan" variable.
4. Checking for overflow of this variable and fixing if needed.
5. Calling FMfrequenc() function to change the current frequency.
6. Calling showFreq() function to show this frequency on the LCD display.

It should result in changing the station currently received.

1.4. <u>OTHER FUNCTIONS</u>

1. butnEvent() – handles button interactions; returns button index
2. showFreq() – calls display() function to update frequency displayed
3. segTest() – iteratively changes state of segments (with delays) to ensure proper working
4. preSet() – enables user to choose the station from the predefined set

<u>PLAN TO COMPLETE REMAINING SOFTWARE COMPONENT</u>

Overall, we believe our progress with the software side has been good and the main tasks left was testing the functionalities with the hardware and planning out any novel features. Considering the current situation, we would like to present our plan for completion of the project assuming 24 lab hours available for each member basing on the Report specifications. In case of additional features, we wanted to focus primarily on User Interface (UI). We planned to implement simple additions, like On/Off LED, as well as more complex ones. One of the examples is a two-sided menu of the FM receiver, this would enable us greatly enhance UI. Even with the simple LCD display, it would allow us to expand the functionality of pre-set stations, including adding and deleting them on fly. Implementation of a simple LED, together with ease to code function, could notify if the station is added to pre-set ones or not.

Mechanical Design of FM RADIO Box

Current Progress

We planned on focusing on the main functionalities of the FM Radio first before commencing on the box, however, to aid us in the future, we have mapped out the main components required for our box and outlined the measurements and design approach we would like between week 3-4. More detail about our box inspiration is expressed in the technical report, but below outlines are first measurements of the box followed by our rendered box

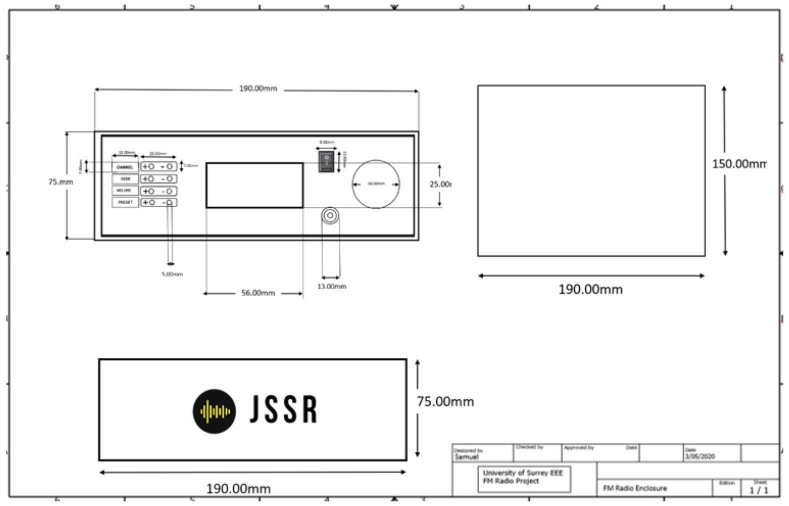
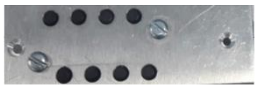


Figure 10- Measurement of planned FM Radio Enclosure

Below show all the measurements required for the box, using these measurements, we decided that we wanted to follow a wooden chassis design to match our rendered box illustrated below. During week 4 the enclosure was designed using the functionalities of Microsoft Office. On week 6, we found wooden sheets available online that can be cut to length[1] using the equipment in the electronics labs such as the scroll saw.

Buttons for the box should follow the schematic below



LCD screen provided

Output should be connected to speaker which replaces the headphone

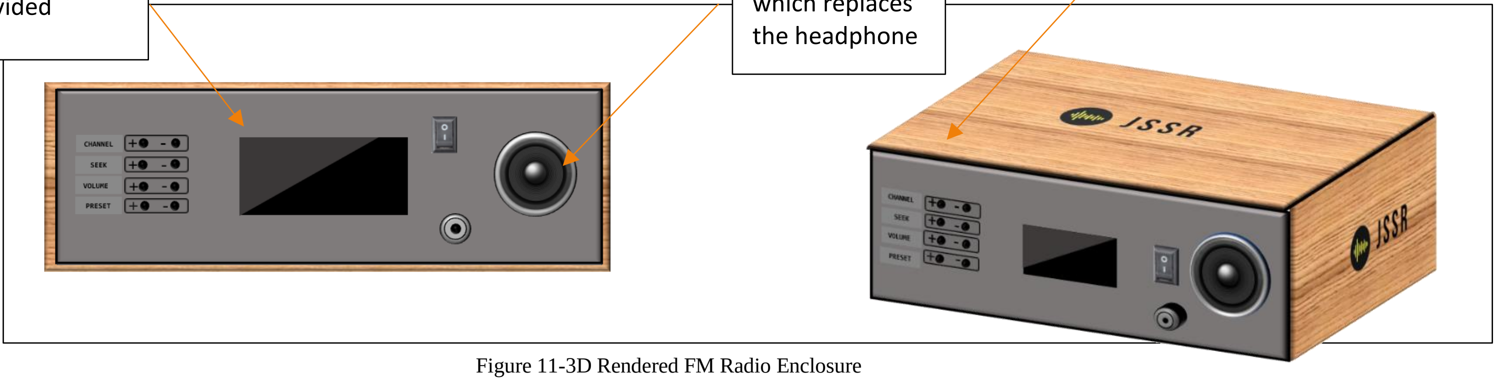


Figure 11-3D Rendered FM Radio Enclosure

Plan to complete Box

Although our box had been designed, given the COVID-19 situation, we were unable to finish the physical implementation of the enclosure. We initially expected to complete the box after the Veroboard implementation, so we can be more certain about the measurements required. We also planned to obtain the wooden sheets during the Easter break, and begin ready to design our box by week 8.Week 8 would have entailed mainly design the measurement and basic box structure, and following this in week 9 we had hoped to obtain the wooden sheets and begin gluing these on the enclosure. Week 10 we had hope to include any additional box designs such as add our logo

[1]Wooden sheets 22400 x 1000cm used for the design on enclosure https://www.woodsheets.com/plywood-sheet-cut-to-size/?

Summary of Week by Week Progress

Week	*Hardware (Scarlett and Jordan)*	*Software (Rafal and Samuel)*
1	▪ Familiarising with the hardware components ▪ Making first circuit configuration	• Learning Embedded C and familiarise with MPLAB • Doing small projects to familiarise with the software – from SurreyLearn • Filling functions: • showFreq(), btnEvent() - Sam • nextChan(), volUp(), preSet(), – Raf al • Description of functions in main.c – for familiarising
2	• Preparing case for the FM Receiver • Recognising the relevant connections between the PIC and the other provided circuits. • Testing these connections on the breadboard • Test that a visual LCD output can be viewed for the software code provided using the breadboard	• Familiarising with functions for: • Output digits to the LCD display - • Detect and act in response to channell button pushes (e.g. on/off, tune up/down) –
3	• Using data from the breadboard tests, begin to plot a dot matrix for the connections between the PIC, AR1010 and the other given circuits • Plot the dot matrix considering the size of components on the matrix board • Solder connections with the pen • Constantly check, with the software code that you get an LCD output. If not, then troubleshoot the matrix connections	• Finalising the code required for outputting digits to the LCD display • Begin programming function required for the volume button
4	• Make more improvements to the box design • Take over Scarlett's breadboard testing, matrix plots and ensure that the whole circuit is correctly built. • Troubleshoot any inconveniences that arise between hardware and software code • Carry on soldering appropriate connections	• Fixing all error checks the arose in the software code mostly being functional error that prevent full code from compiling. • Also revisiting the logic of each software component to ensure that the code worked as expected
7	• Make heavy improvements to the box • Carry on troubleshooting and checking the LCD output of the input software code	• Test code on the PIC Microcontroller to ensure that the code worked as expected

Conclusion

Overall, we believe given the shorter time scale we had to complete the project we made good progress. Although the overall project could not be tested together due to us not having access to the labs, we still confident that the tests should mostly be success as each individual modular component successfully operated as expected.

The software team were looking to prepare the code to produce the LCD display as a dot matrix instead of the regular segment display. Prior to the COVID-19 pandemic, we took out extensive research on how the working of a dot matrix. An example of a dot matrix grid is shown in figure 12. Here multiple LEDs are wired together in rows and columns which means that we do not wire all the individual LEDs, but instead can wire all the anodes in rows and cathodes in columns. This thus enables LEDs to be addressed by rows and column, enabling the implementation of the software to be more approachable. In the figure 12, for example, if R4 is pulled high and C3 is pulled low, the LED in the fourth row and third column will be turned on. Characters can be displayed by fast scanning of either rows or columns. [2]

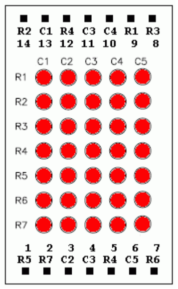


Figure 12, row and column arrangement of a dot matrix

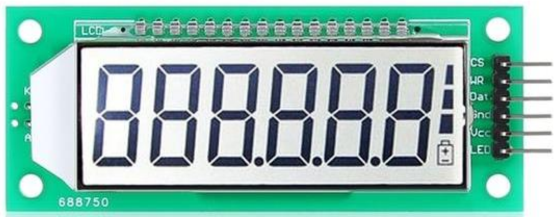
- 

Fig.13 - Segment style LCD display

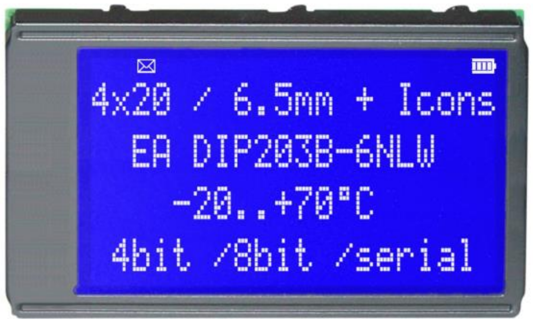
- 

Fig.14 - Dot matrix style LCD display

We would also had intended to implement an ON/OFF LED module that indicates when the radio was on or off. This again should have not been too difficult to implement and could have been implemented using the main components available in the Electronic Laboratory.

During the project we gained a significant number of skills such as: planning the tasks within groups, to enhanced knowledge of embedded programming and hardware debugging skills. We believe if we had more times, we would have been able to present a working FM Radio Receiver with a unique box enclosure.

[2] https://reviseomatic.org/help/2-picaxe/Picaxe%20LED%20Matrix.php