James R. Di Re, Abdulrahman Kikia, Claudio Scione

May 31st, 2025

MakerSPACE Internship

# Arduino Radio - Final Report

## **Introduction**

In this report, we will detail the brainstorming, designing, coding and building of our Arduino internship project: an Arduino powered AM/FM radio. Over the course of the past three months, we have worked to put together a working radio while documenting the entire process to serve as an inspiration and a guide to anyone else who may want to embark on a similar project. Our goal when choosing this as our project was not only to improve our coding and debugging skills through a hands-on approach, but also to gain an understanding of radio waves, audio processing, and speakers. All this, in addition to enjoying the satisfaction of listening to music through a device we made from scratch.

## **Project Description**

### **Objective**

The goal of this project was to build a working AM/FM radio, using the Arduino Uno as the motherboard and the Arduino IDE to write the code. We aimed to not only be able to play music through speakers, but as display information about the frequency and station on a digital display. Finally, we wanted to design and build a casing to insert all the hardware into that was practical and portable. While this was the base goal of this project, we hoped to add as many features as possible to the radio within the time frame we had to complete this project. We hoped to add a function to set pre-set radio stations that you could choose and change as you wished, shortwave compatibility, radio station scanning and even Bluetooth connectivity.

### **Inspiration**

It was James who originally came up with the idea of making a radio using Arduino. He had watched a few videos on YouTube of people building using Arduino to make working radios. The videos he watched used FM only radio modules, and only detailed the wiring process. We decided it would be interesting to expand on those projects James had seen online to make a radio with more features and a case to enclose it in. After some research on the Arduino forums and other build project sites, we saw other examples of radios, each with their own unique construction and design. That made us realize that this was something we could realistically build and design on our own in the months we had to work on this project.

### **Design and Planning**

For the design, we wanted to build a radio that is reminiscent of old-school radio devices. We wanted to use knobs instead of buttons. We also wanted it to be unique with mismatching sizes for the different buttons, all while having some symmetry. We shared different sketches of the design and some concept 3D renders to see where our imagination could go. In the end, we couldn’t settle on a specific design until we got all parts working and tested.

Designing a case to house all the parts proved to be a challenge. The setup out of the box functioned well, but was fragile in the initial phases. The speaker wires kept disconnecting from the antenna with the slightest movement. Whenever we would tighten the screws to hold the wires in place, the wires would go out, or the speaker could stop playing. There were also pieces that had to be mounted to the front, so that the user can interact with the radio as intended. We made a prototype using cardboard to see how the pieces would go together. The speakers were problematic as they don’t screw holes to mount them in place. Their wires also interfered with the screen if the positioning wasn’t optimal. So we had to figure out the perfect position for them and add a shelf that comes out of the sides, such that they sit on top of them.

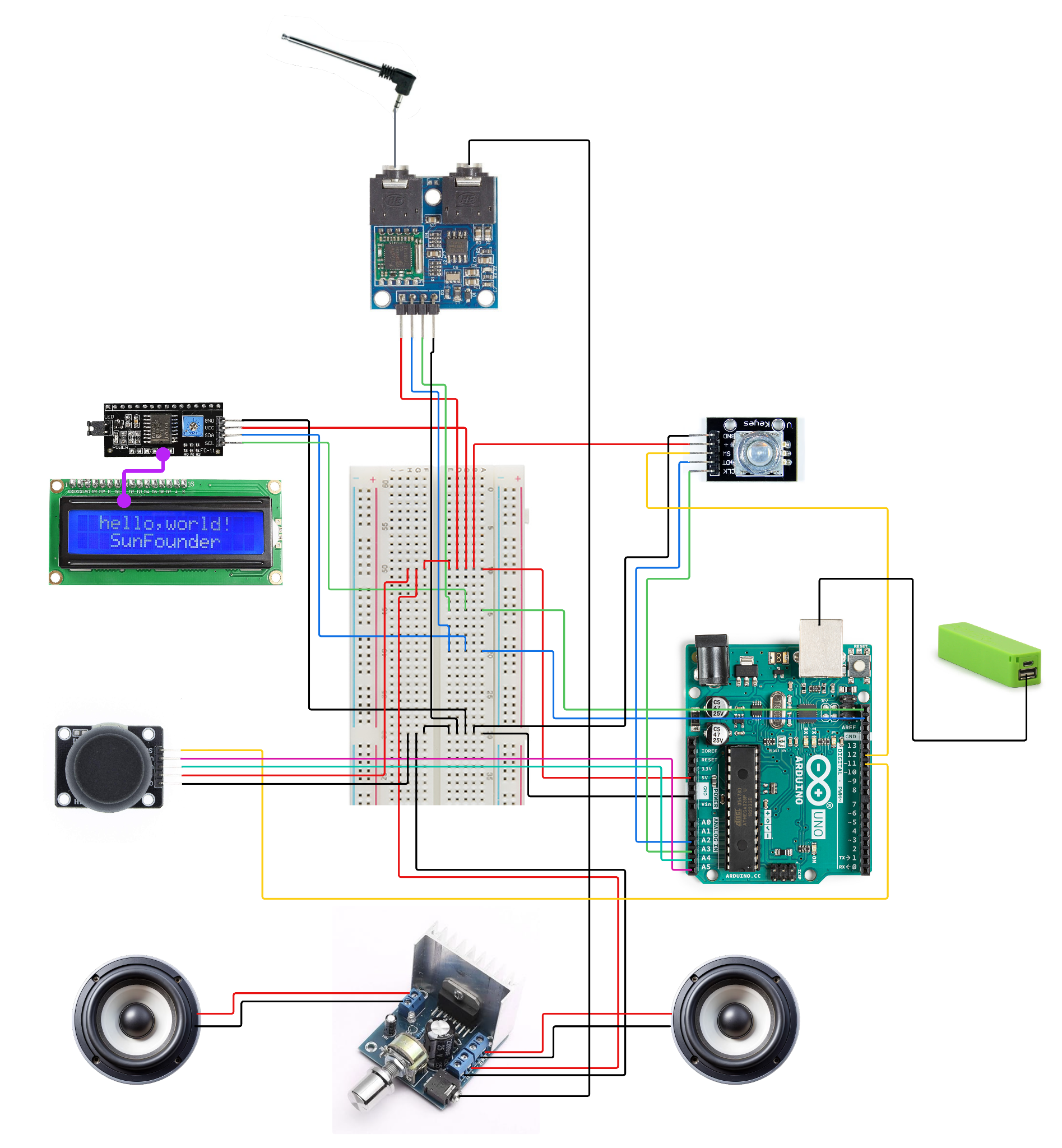
The screen, antenna and rotary encoder have screw holes so mounting them wasn’t an issue. However, using the battery was our real issue. The connector that the battery uses is very stiff and doesn’t come easily out of its terminals. So removing the battery wasn’t a feasible option to cut the circuit. After trying different options, we ended up using a power bank as our power source, as it can be easily removed and charged when needed. For the arduino board, we added holes for accessing its USB and power ports when mounted.

Initially the idea was to have the six faces of the box snap into each other through a connector that was meant to hold them by pressure. However, as the pieces are small and would break with high pressure, we had to change the design into a box, of which the rear pieces slide out for quick access. The breadboard sits in the bottom and connects to the other pieces. The speaker drivers and the amplifier sit on extruded platforms made for them, while other pieces are mounted to the walls of the chassis. The result is a radio that is fairly portable, and is comparable to other radios of its kind in size.

## **Implementation**

### **Hardware Setup**

* Arduino Uno microcontroller
* Rotary encoder with button - for frequency tuning
* LCD display that is I2C compatible - for displaying radio data
* TEA5767 FM radio module with antenna - for receiving radio signals
* TDA7297 amplifier - for amplifying the audio signal coming from the radio module and for allowing volume control
* Male-to-male headphone wire - for sending the audio signal from the radio module to the amplifier
* Speakers
* Joystick - for choosing between different preset stations (up, down, left, right = 4 presets)
* Power source: power bank OR 9V battery OR power from a computer
* Male-to-male jumper wires
* Male-to-female jumper wires
* Breadboard



### **Software Development**

Firstly, James already had C++ knowledge from a tutorial series he had watched on YouTube a few years ago, so because of that, it was a lot easier for him to write code for the project compared to someone just starting out. However, this was still his first time putting that knowledge into real practice. James recommended to the other group members that they watch some videos from the tutorial series too.

The features in the code were at first programmed in the Arduino IDE, and were added gradually (for example, first the code for the rotary encoder and LCD display were added, then the code for the radio module, then the code for the joystick, then the code for preset saving, etc.). The code was always kept in a maintainable condition that was simple to expand upon. This was a top priority, as James knew from past projects that untidy code gets exponentially hard to maintain. Eventually, even though the code was kept organized, it was still overwhelming to keep all the functionality in one file. Thus, it was separated into six separate files: the main cpp file, a main header file, a class for managing radio state, a class for handling input, a class for handling the LCD display, and a class for handling the radio module. Later on in development, this proved to be a great choice, as each of these files grew in size. Looking back, it would not have been feasible to keep everything in a single file.

However, separating the code into different files didn’t come without its own problems. For one thing, there were many bugs that got introduced as a result, and all of them were isolated to being issues regarding C++’s copying of objects in unexpected areas. As a result, in fixing these bugs, the team grew more familiar with C++’s copying conventions, and became more aware of when it was appropriate to use pointers (\*) or references (&).

Eventually, the entirety of the code was moved to Visual Studio Code. This provided many advantages over the Arduino IDE. For one thing, VS Code allowed for easy integration with version control software such as Git and GitHub. Also, it underlines compilation errors in real-time, a feature which the Arduino IDE lacks entirely.

### **Sensor Documentation**

* Antenna

The antenna picks FM signals and transmits them into the amplifier.

* Amplifier

The amplifier picks the sounds signal (from the antenna) and transmits them into the speakers. It also has a potentiometer that controls the power given to the speaker (the volume coming out of them).

* Rotary Encoder

The rotary encoder helps with navigating radio stations. Each step in the rotation gives the arduino board a signal to either increase or decrease the frequency picked from the antenna by 0.2, which is by how much FM signal changes.

* Joystick

The joystick has two potentiometers for dual axis rotation. It can detect the rotation of the stick in eight directions and in our case is the optimal way to select and tune-in on presets.

## **Results**

## **Analysis**

## **Discussion and Conclusion**

## **References**

## **Acknowledgements**