Arduino FM Radio

James R. Di Re

Abdulrahman Kikia

Claudio Scione

Initial Objectives

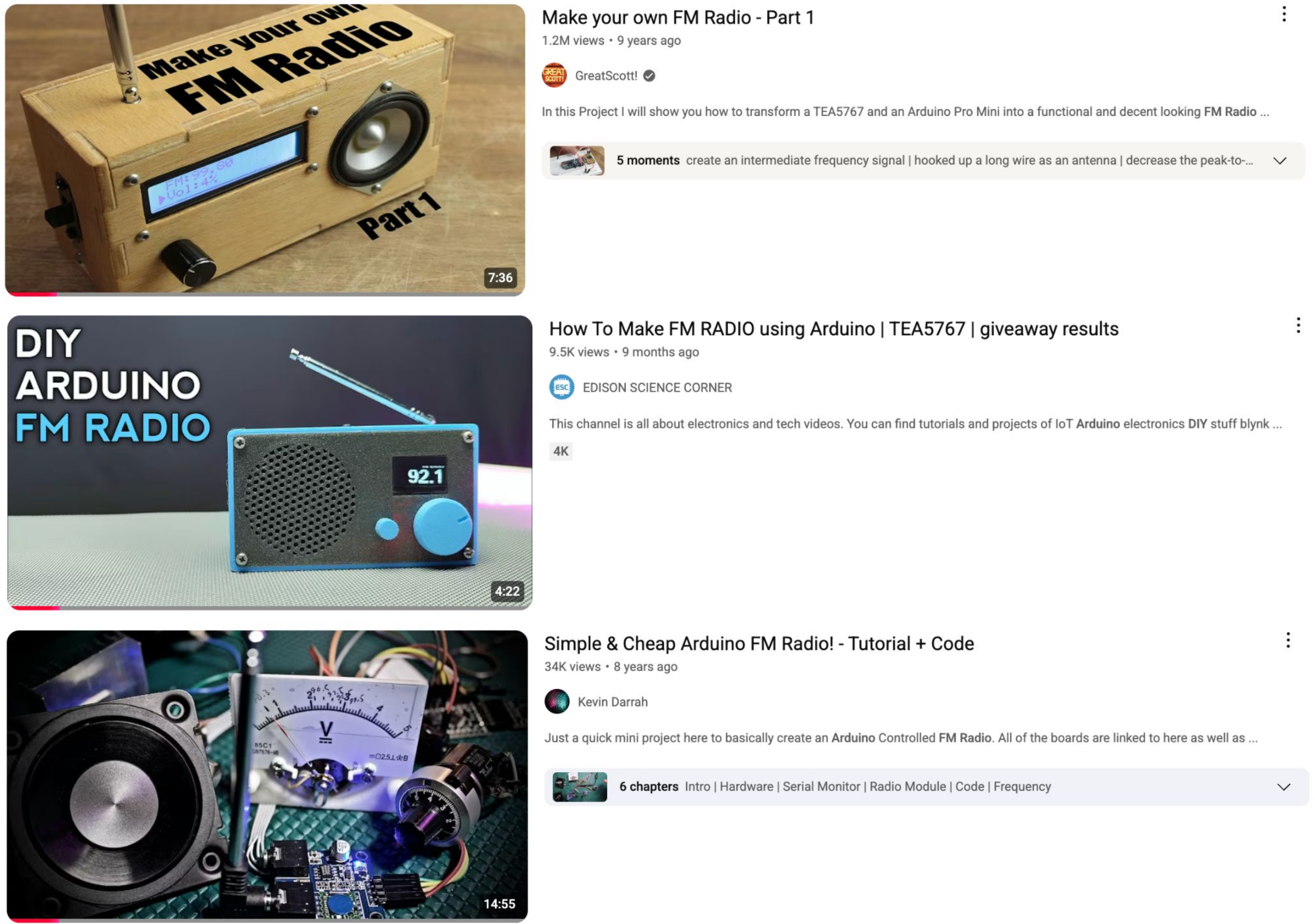
Abdulrahman

- Must be portable —------->
- Should support different bands of reception (FM, AM, Shortwave, etc.)
- Display information on an LCD
- Support presets (ex. 1, 2, 3, 4, 5)
- RDS (prints current song)
- Scanning
- Bluetooth compatibility (i.e. play audio from your phone)

Inspiration

James

- James’s idea. Wanted to do something involving audio (MIDI controller, synthesizer, etc.)
- Decided on radio after watching many YouTube videos and visiting Instructables links
- Intuitive - can pick up signal anywhere, for free
- Practical - can listen to music, news, talk shows, traffic report, etc. from the Montreal area



Make your own FM Radio - Part 1

1.2M views • 9 years ago

GreatScott!

In this Project I will show you how to transform a TEA5767 and an Arduino Pro Mini into a functional and decent looking **FM Radio** ...

5 moments create an intermediate frequency signal | hooked up a long wire as an antenna | decrease the peak-to-...

How To Make FM RADIO using Arduino | TEA5767 | giveaway results

9.5K views • 9 months ago

EDISON SCIENCE CORNER

This channel is all about electronics and tech videos. You can find tutorials and projects of IoT **Arduino** electronics **DIY** stuff blynk ...

4K

Simple & Cheap Arduino FM Radio! - Tutorial + Code

34K views • 8 years ago

Kevin Darrah

Just a quick mini project here to basically create an **Arduino** Controlled **FM Radio**. All of the boards are linked to here as well as ...

6 chapters Intro | Hardware | Serial Monitor | Radio Module | Code | Frequency

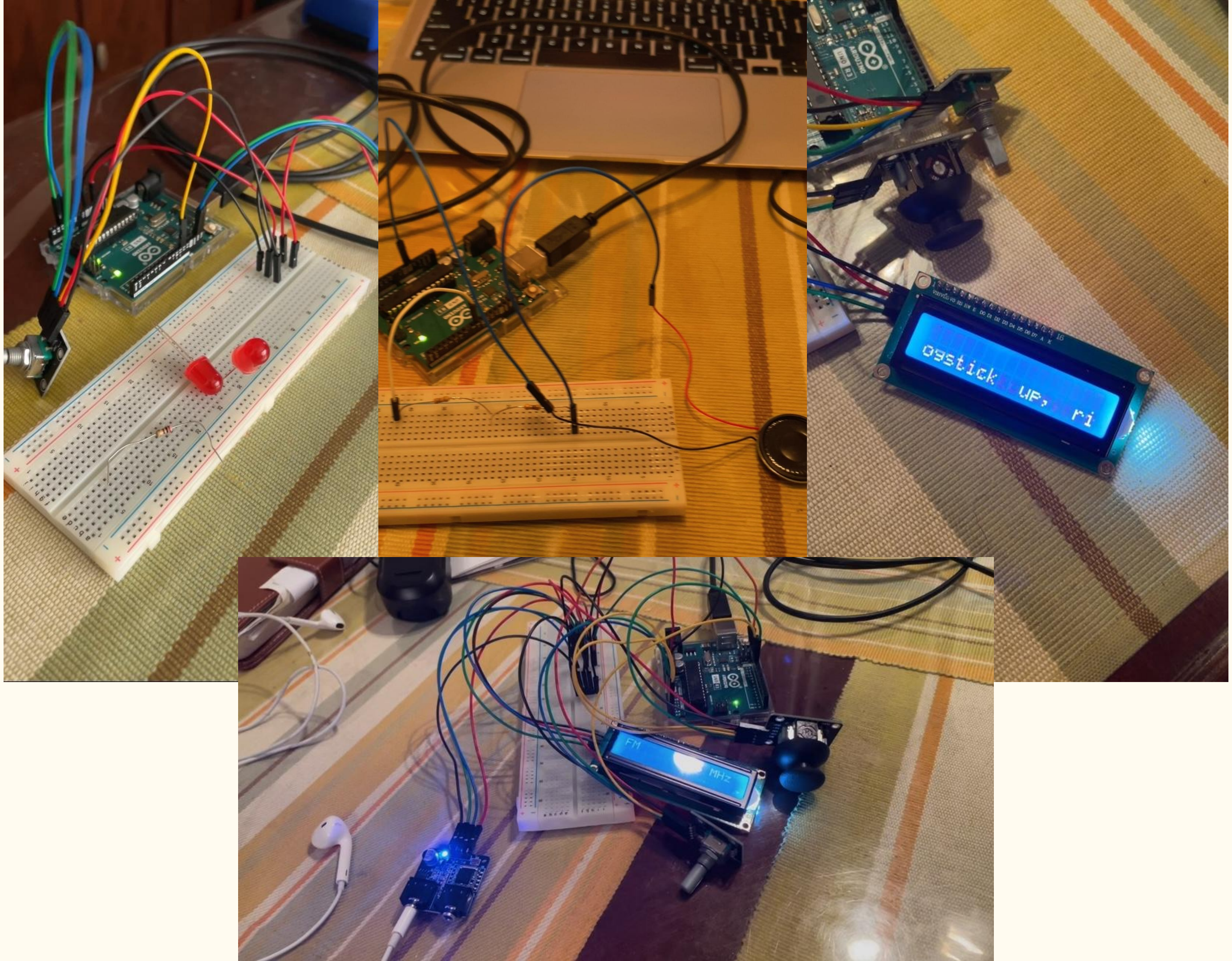
Design and Planning

- Initial radio sketches
- Problems
  - Speaker wires kept disconnecting
  - Speakers hard to mount on box (no screw holes)
- Adding features gradually (first rotary encoder, then LCD, then radio module, then amplifier and speaker)
- Cardboard prototype

Hardware Setup

- Arduino UNO Microcontroller
- Rotary Encoder (tuner knob)
- LCD Display (I^2C)
- TEA5767 FM Radio Module
- TDA7297 Amplifier
- Speakers
- Joystick
- Power Source
- Headphone Wire, Jumper Wires, Breadboard

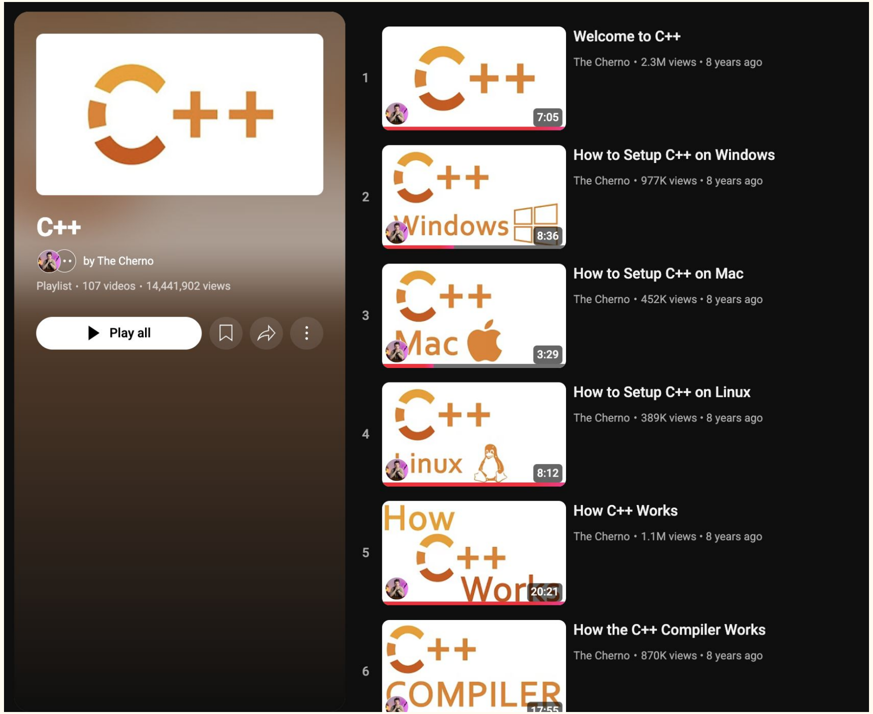
Abdulrahman


oystick UP, ri
FM
MHz

<!-- Code -->

- Already had some prior knowledge of CPP from this tutorial series —----------->
- Made writing the radio code simpler
- However, this was the first time putting that knowledge into practice

James



C++

by The Cherno

Playlist • 107 videos • 14,441,902 views

1 Welcome to C++ — The Cherno • 2.3M views • 8 years ago — 7:05

2 How to Setup C++ on Windows — The Cherno • 977K views • 8 years ago — 8:36

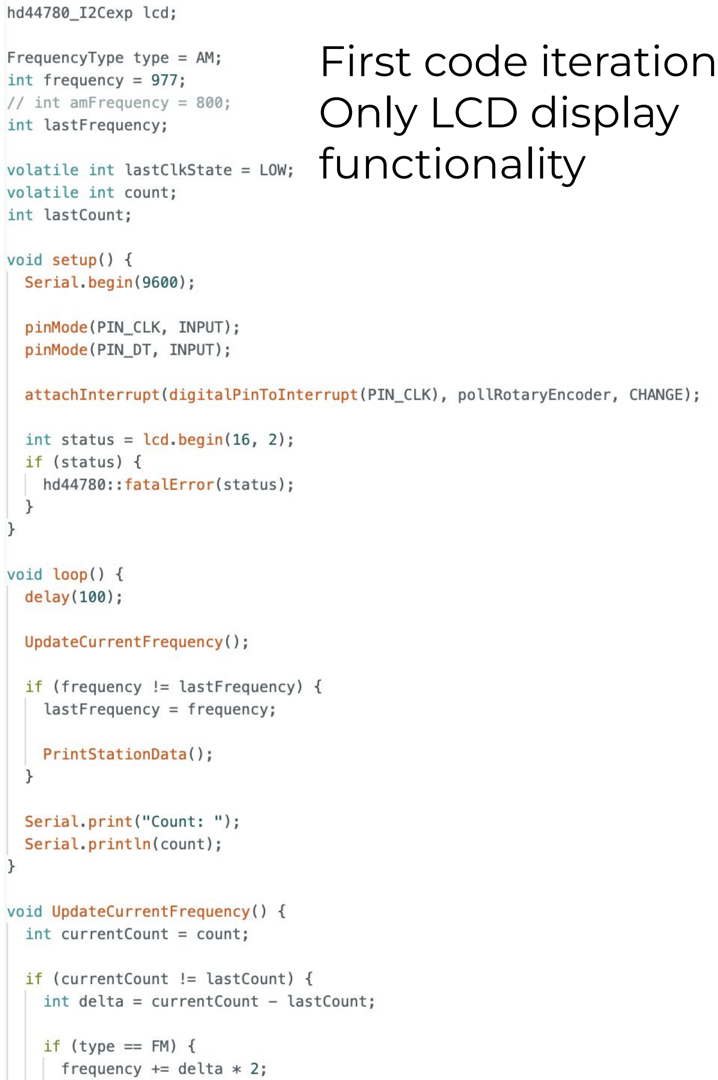
3 How to Setup C++ on Mac — The Cherno • 452K views • 8 years ago — 3:29

4 How to Setup C++ on Linux — The Cherno • 389K views • 8 years ago — 8:12

5 How C++ Works — The Cherno • 1.1M views • 8 years ago — 20:21

6 How the C++ Compiler Works — The Cherno • 870K views • 8 years ago

First code iteration: Only LCD display functionality



```
hd44780_I2Cexp lcd;

FrequencyType type = AM;
int frequency = 977;
// int amFrequency = 800;
int lastFrequency;

volatile int lastClkState = LOW;
volatile int count;
int lastCount;

void setup() {
  Serial.begin(9600);

  pinMode(PIN_CLK, INPUT);
  pinMode(PIN_DT, INPUT);

  attachInterrupt(digitalPinToInterrupt(PIN_CLK), pollRotaryEncoder, CHANGE);

  int status = lcd.begin(16, 2);
  if (status) {
    hd44780::fatalError(status);
  }
}

void loop() {
  delay(100);

  UpdateCurrentFrequency();

  if (frequency != lastFrequency) {
    lastFrequency = frequency;

    PrintStationData();
  }

  Serial.print("Count: ");
  Serial.println(count);
}

void UpdateCurrentFrequency() {
  int currentCount = count;

  if (currentCount != lastCount) {
    int delta = currentCount - lastCount;

    if (type == FM) {
      frequency += delta * 2;
```

Later code iteration: separated into six files!
Reasons: Scalability, more control, more organized

`main.cpp` -

Entry point of the program. Contains `setup()` and `loop()` functions

`RadioHeaders.h` -

Header file for all the following classes. Contains variables and function declarations. All other files `#include` this

`class RadioHandler { }` -

General class for managing many of the properties of the radio. Manages current station, states, etc

`class RadioInput { }` -

Handles functionality for the rotary encoder and the joystick

`class RadioDisplay { }` -

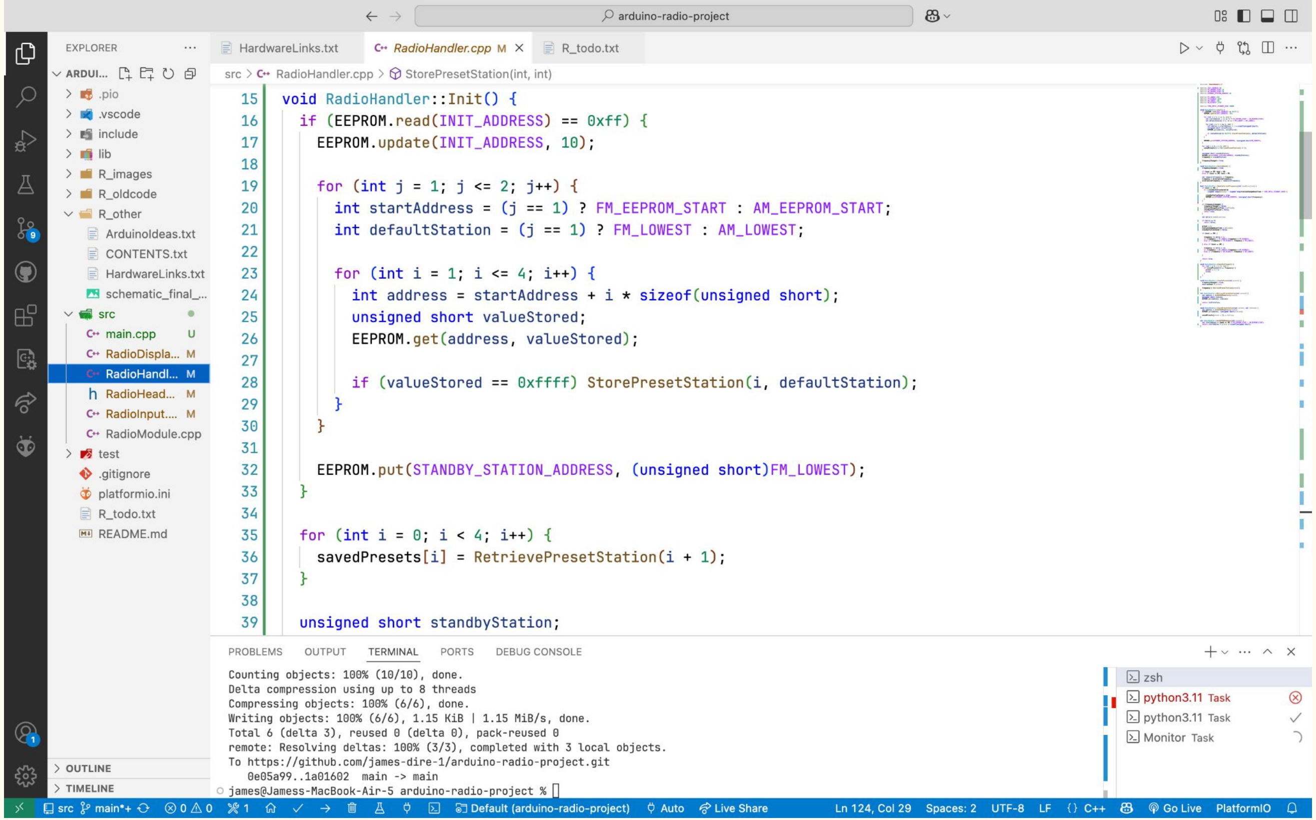
Handles everything to do with displaying information on the LCD screen, including methods for printing text

`class RadioModule { }` -

Handles communicating with the radio module and tuning to stations.

The switch to Visual Studio Code

- Allowed for easy version control
- Underlines errors and points out flaws in the code
- Easy refactoring tools
- Easy way to see project structure
- Extension used:

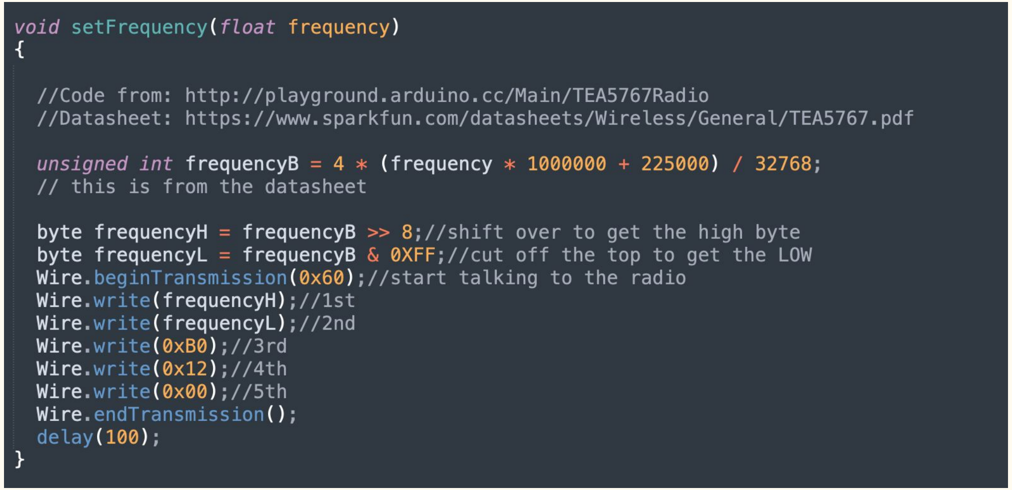


```
void RadioHandler::Init() {
  if (EEPROM.read(INIT_ADDRESS) == 0xff) {
    EEPROM.update(INIT_ADDRESS, 10);

    for (int j = 1; j <= 2; j++) {
      int startAddress = (j == 1) ? FM_EEPROM_START : AM_EEPROM_START;
      int defaultStation = (j == 1) ? FM_LOWEST : AM_LOWEST;

      for (int i = 1; i <= 4; i++) {
        int address = startAddress + i * sizeof(unsigned short);
        unsigned short valueStored;
        EEPROM.get(address, valueStored);

        if (valueStored == 0xffff) StorePresetStation(i, defaultStation);
      }
    }

    EEPROM.put(STANDBY_STATION_ADDRESS, (unsigned short)FM_LOWEST);
  }

  for (int i = 0; i < 4; i++) {
    savedPresets[i] = RetrievePresetStation(i + 1);
  }

  unsigned short standbyStation;
```

```
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.15 KiB | 1.15 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/james-dire-1/arduino-radio-project.git
   0e05a99..1a01602  main -> main
james@Jamess-MacBook-Air-5 arduino-radio-project %
```

Libraries Used

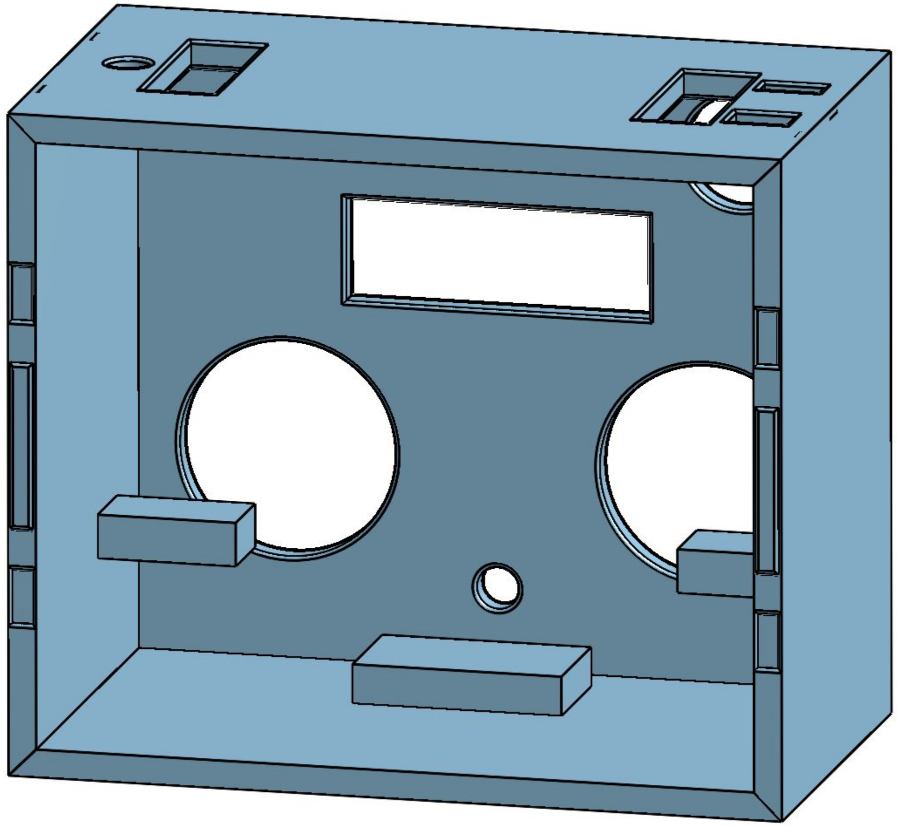
- hd44780 by Bill Perry. LCD library. Includes functions like `setCursor(0, 0)`, `print("")`, `clear()`
- RotaryEncoder by Matthias Hertel. Includes functions like `tick()` and `getDirection()`
- Also used code provided in the description of a YouTube video by Kevin Darrah, for tuning with the TEA5767 FM radio module



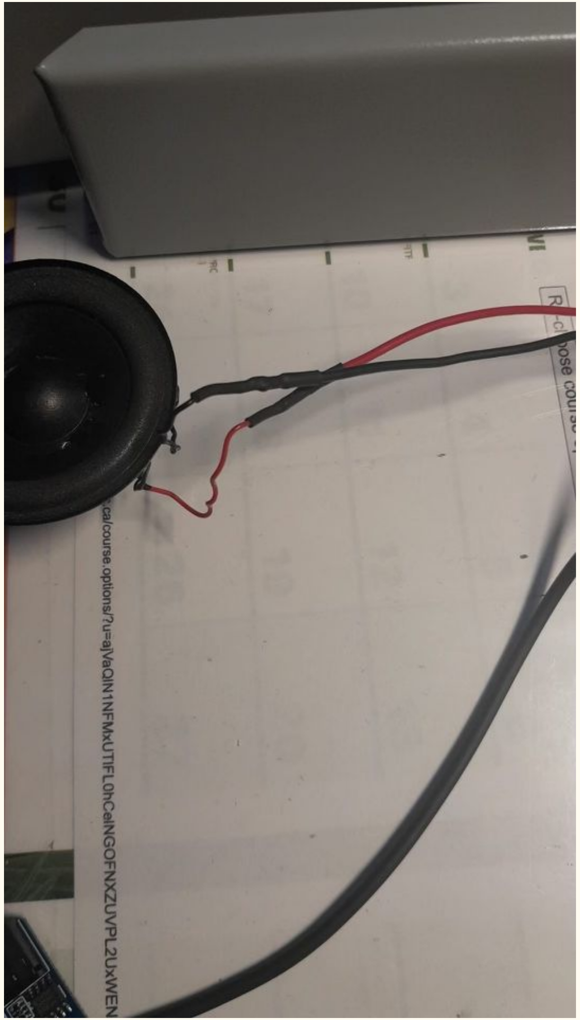
```
void setFrequency(float frequency)
{

  //Code from: http://playground.arduino.cc/Main/TEA5767Radio
  //Datasheet: https://www.sparkfun.com/datasheets/Wireless/General/TEA5767.pdf

  unsigned int frequencyB = 4 * (frequency * 1000000 + 225000) / 32768;
  // this is from the datasheet

  byte frequencyH = frequencyB >> 8;//shift over to get the high byte
  byte frequencyL = frequencyB & 0XFF;//cut off the top to get the LOW
  Wire.beginTransmission(0x60);//start talking to the radio
  Wire.write(frequencyH);//1st
  Wire.write(frequencyL);//2nd
  Wire.write(0xB0);//3rd
  Wire.write(0x12);//4th
  Wire.write(0x00);//5th
  Wire.endTransmission();
  delay(100);
}
```

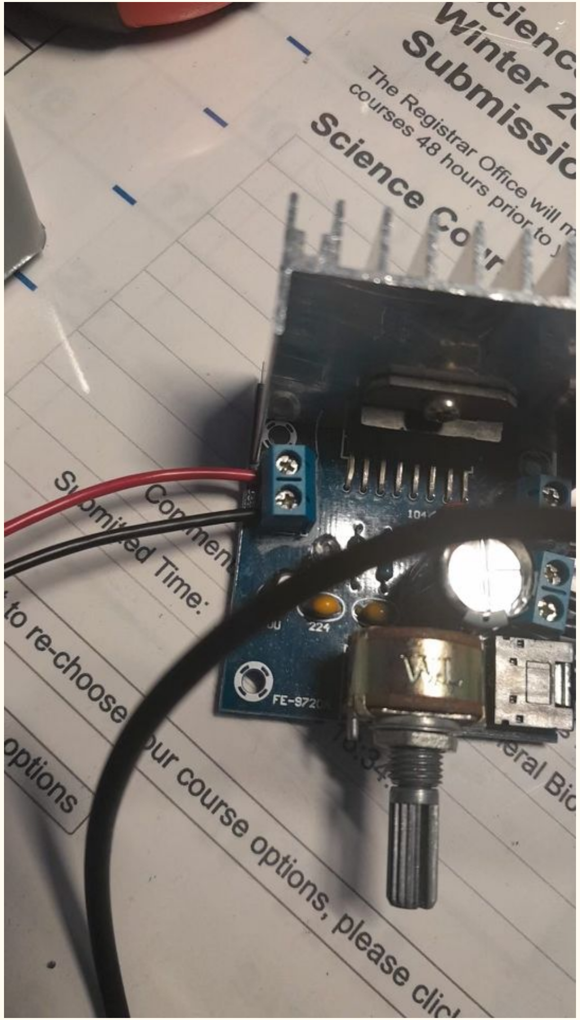
Initial Design for the box

Had to be changed to make the assembly possible

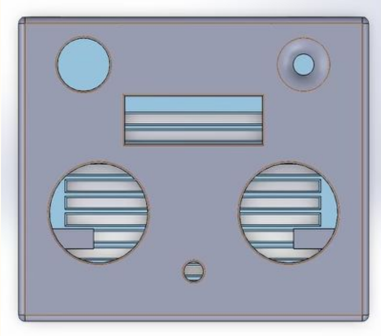


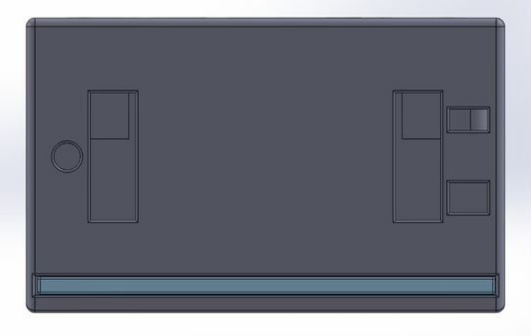
Abdulrahman

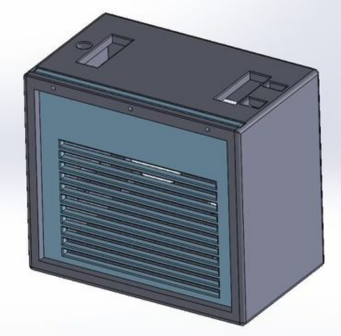




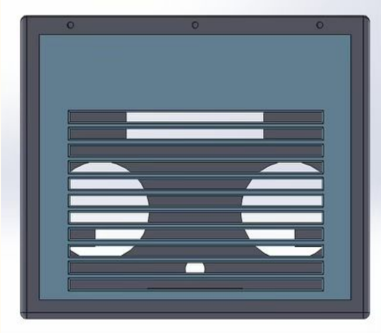
Claudio

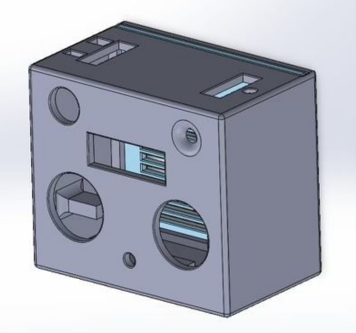


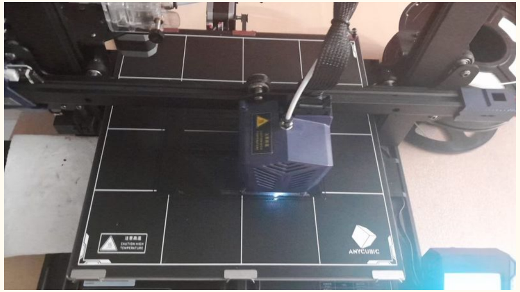




ANYCUBIC






ANYCUBIC

Results/Successes

- Functional FM tuning with knob
- Informative display that shows band, station, preset
- Tuning to presets (although some problems)
- Overwriting presets
- Decent audio quality
- Organized code that was easy to maintain

Claudio

James

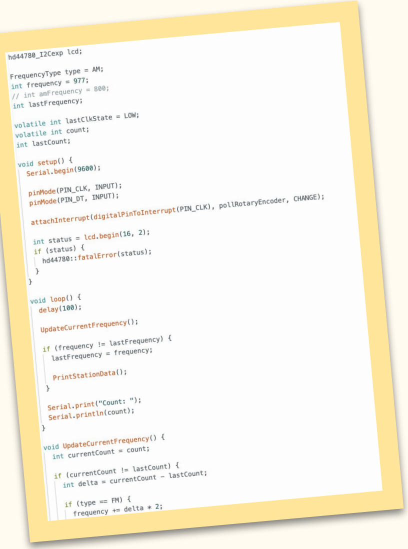
Persisting Issues:

- Interference noise caused by the LCD screen updating and joystick movement
- Joystick messing up the 1^{2}C communication lines. Causes code freeze ups. (Isolated problem by removing joystick code and changing station every 1 second)
- Volume knob can’t be turned all the way

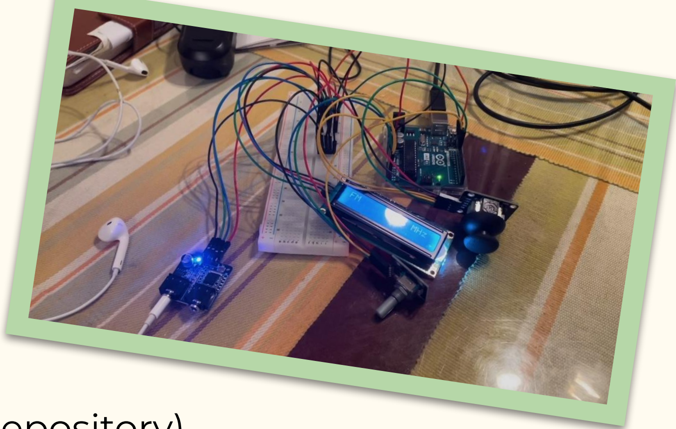
Lacking Features:

- Only FM support. Plans to try out different (more low-level, complicated) radio module, but no time
- No Bluetooth connectivity
- No RDS, scanning, bluetooth support

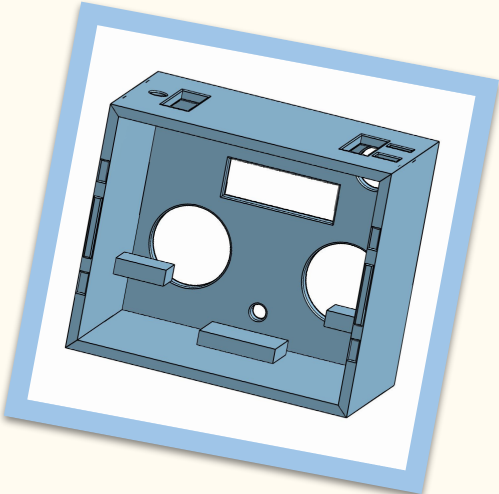
What We Learnt



```
hd44780_I2Cexp lcd;

FrequencyType type = AM;
int frequency = 977;
// int amFrequency = 800;
int lastFrequency;

volatile int lastClkState = LOW;
volatile int count;
int lastCount;

void setup() {
  Serial.begin(9600);

  pinMode(PIN_CLK, INPUT);
  pinMode(PIN_DT, INPUT);

  attachInterrupt(digitalPinToInterrupt(PIN_CLK), pollRotaryEncoder, CHANGE);

  int status = lcd.begin(16, 2);
  if (status) {
    hd44780::fatalError(status);
  }
}

void loop() {
  delay(100);

  UpdateCurrentFrequency();

  if (frequency != lastFrequency) {
    lastFrequency = frequency;

    PrintStationData();
  }

  Serial.print("Count: ");
  Serial.println(count);
}

void UpdateCurrentFrequency() {
  int currentCount = count;

  if (currentCount != lastCount) {
    int delta = currentCount - lastCount;

    if (type == FM) {
      frequency += delta * 2;
```



- Interference problems
- Git and GitHub (various commands, repository)
- How to integrate libraries into our project
- How to set up a VS Code environment
- Useful C functions like `sprintf()`
- How to package components into a box
- 3D printing



Abdulrahman

https://www.shutterstock.com/search/time-running-out

https://en.m.wikipedia.org/wiki/File:Visual_Studio_Code_1.35_icon.svg

https://commons.wikimedia.org/wiki/File:PlatformIO_logo.svg

https://www.amazon.ca/PRUNUS-J120-Shortwave-Rechargeable-Playing%E3%80%902023/dp/B0BMT86PBV