

# Arduino FM Radio

James R. Di Re

Abdulrahman Kikia

Claudio Scione

## Initial Objectives

- 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'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.





# 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

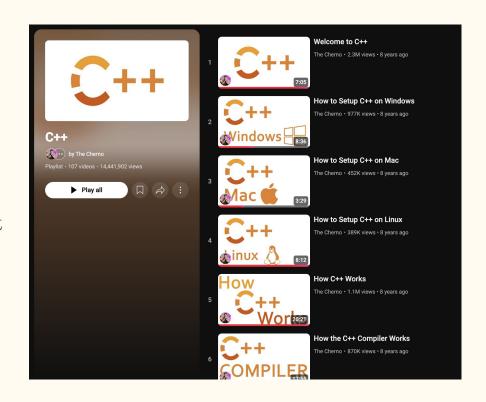
List hardware





### <!-- 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



```
Later code iteration: separated into six files!
                      First code iteration:
FrequencyType type = AM;
int frequency = 977;
                                                          Reasons: Scalability, more control, more organized
// int amFrequency = 800;
                      Only LCD display
int lastFrequency;
volatile int lastClkState = LOW; functionality
                                                           main.cpp -
volatile int count:
int lastCount:
                                                           Entry point of the program. Contains setup () and loop () functions
void setup() {
 Serial.begin(9600);
                                                           RadioHeaders.h -
 pinMode(PIN_CLK, INPUT);
                                                           Header file for all the following classes. Contains variables and function
 pinMode(PIN_DT, INPUT);
                                                           declarations. All other files #include this
 attachInterrupt(digitalPinToInterrupt(PIN_CLK), pollRotaryEncoder, CHANGE);
 int status = lcd.begin(16, 2);
 if (status) {
                                                           class RadioHandler { } -
  hd44780::fatalError(status):
                                                           General class for managing many of the properties of the radio. Manages
                                                           current station, states, etc
void loop() {
 delay(100);
                                                           class RadioInput { } -
 UpdateCurrentFrequency();
                                                           Handles functionality for the rotary encoder and the joystick
 if (frequency != lastFrequency) {
  lastFrequency = frequency;
  PrintStationData();
                                                           class RadioDisplay { } -
                                                           Handles everything to do with displaying information on the LCD screen,
 Serial.print("Count: ");
                                                           including methods for printing text
 Serial.println(count);
void UpdateCurrentFrequency() {
                                                           class RadioModule { } -
 int currentCount = count;
 if (currentCount != lastCount) {
                                                           Handles communicating with the radio module and tuning to stations.
  int delta = currentCount - lastCount;
  if (type == FM) {
   frequency += delta * 2;
```

hd44780 I2Cexp lcd:



## The switch to Visual Studio Code

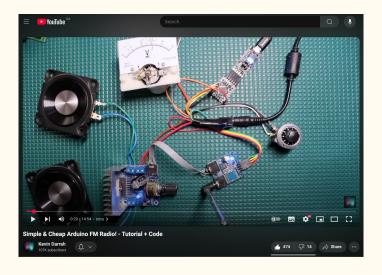
- Allowed for easy version control
- Underlines errors and points out flaws in the code
- Easy refactoring tools
- Easy way to see project structure
- Extension used:



```
83~
                                                                                                                                                         EXPLORER
                     HardwareLinks.txt
                                        C++ RadioHandler.cpp M X R_todo.txt
                                                                                                                                                    D ~ 0 th II ...
ARDUI... [3 [7] (1) [9]
                     src > C++ RadioHandler.cpp > 分 StorePresetStation(int, int)
oiq. 👼 <
                             void RadioHandler::Init() {
> xscode
                               if (EEPROM.read(INIT_ADDRESS) == 0xff) {
> m include
                       17
                                 EEPROM.update(INIT_ADDRESS, 10);
                       18
> R_images
                       19
                                  for (int j = 1; j <= 2; j++) {
                       20
                                    int startAddress = (i == 1) ? FM EEPROM START : AM EEPROM START:
∨ ■ R_other
                       21
                                    int defaultStation = (j == 1) ? FM_LOWEST : AM_LOWEST;
   Arduinoldeas.txt
                       22
   CONTENTS.txt
                       23
                                    for (int i = 1; i <= 4; i++) {
   HardwareLinks.txt
                                      int address = startAddress + i * sizeof(unsigned short);
                       25
                                      unsigned short valueStored;
   C++ main.cpp
                                      EEPROM.get(address, valueStored);
   C++ RadioDispla... M
                       27
     RadioHandl... M
                       28
                                      if (valueStored == 0xffff) StorePresetStation(i, defaultStation):
   h RadioHead... M
                       29
   C++ RadioInput.... M
                       30
   C++ RadioModule.cpp
                       31
                       32
                                 EEPROM.put(STANDBY_STATION_ADDRESS, (unsigned short)FM_LOWEST);
                       33
  platformio.ini
                       34
  R_todo.txt
  M README.md
                       35
                               for (int i = 0; i < 4; i++) {
                       36
                                 savedPresets[i] = RetrievePresetStation(i + 1);
                       37
                       38
                       39
                               unsigned short standbyStation;
                      Counting objects: 100% (10/10), done.
                      Delta compression using up to 8 threads
                                                                                                                                              > python3.11 Task
                      Compressing objects: 100% (6/6), done.
                                                                                                                                             ≥ python3.11 Task
                      Writing objects: 100% (6/6), 1.15 KiB | 1.15 MiB/s, done.
                      Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
                                                                                                                                              > Monitor Task
                      remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
                      To https://github.com/james-dire-1/arduino-radio-project.git
OUTLINE
                        0e05a99..1a01602 main -> main
                      iames@Jamess-MacBook-Air-5 ard∪ino-radio-project % [
                          ெ ✓ → 🗎 🗸 🛡 🖸 🗑 Default (arduino-radio-project) 🛡 Auto 🔗 Live Share
                                                                                                      Ln 124, Col 29 Spaces: 2 UTF-8 LF () C++ 😝 🖗 Go Live PlatformIO 🚨
```

## 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



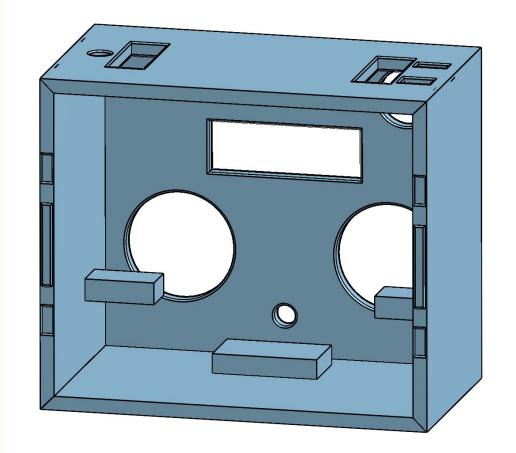
```
void setFrequency(float frequency)
{
    //Code from: http://playground.arduino.cc/Main/TEA5767Radio
    //Datasheet: https://www.sparkfun.com/datasheets/Wireless/General/TEA5767.pdf

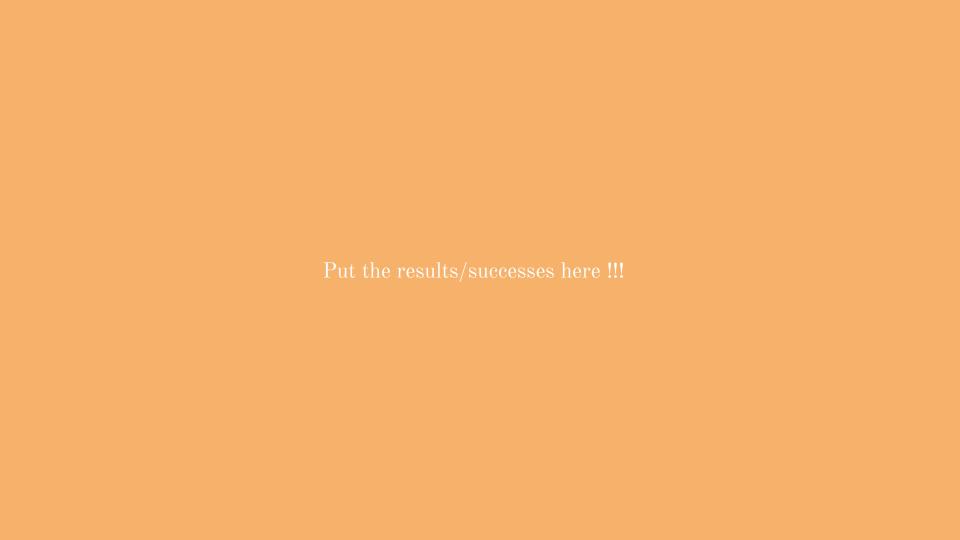
unsigned int frequencyB = 4 * (frequency * 1000000 + 225000) / 32768;
    // this is from the datasheet

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(frequencyL);//2nd
    Wire.write(frequencyL);//2nd
    Wire.write(0xB0);//3rd
    Wire.write(0x12);//4th
    Wire.endTransmission();
    delay(100);
}
```

# Initial Design for the box

Had to be changed to make the assembly possible







#### Persisting Issues:

- Interference noise caused by the LCD screen updating and joystick movement
- Joystick messing up the 1<sup>2</sup>C communication lines. Causes code freeze ups. (Isolated problem by removing joystick code and changing station every 1 second)

#### 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



| 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           |

 $\underline{https://www.amazon.ca/PRUNUS-J120-Shortwave-Rechargeable-Playing\%E3\%80\%902023/dp/B0BMT86PBV}$