**Amplified FM Tuner Evaluation Board**

From: Aldo Ndreu, Ryan Antolin, & Erick Cantos  
Discipline: Computer Engineering Technology  
Date: 04/23/2018

# Declaration of Joint Authorship

We, Aldo Ndreu, Ryan Antolin, and Erick Cantos confirm that this work submitted for assessment is the joint work of ourselves and is expressed in our own words. Any uses made within of other works of any other author, in any form (ideas, equations, figures, previous technologies, tables, programs, texts) are properly acknowledged at the point of use. A list of the references used is included. Aldo Ndreu has handled the software and mobile application, while Ryan Antolin has handled the Database, and Erick Cantos has handled the hardware aspects of this project.

# Approved Proposal

## Executive Summary

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators such as the Speaker Bonnet & FM Tuner Evaluation Board - Si4703. The database used in this project will be Firebase and will store FM radio stations, and possibly favorited stations.

The mobile device functionality will include setting or selecting different FM radio stations, favoriting different FM radio stations, displaying song or station currently playing, and will be further detailed in the mobile application proposal. We will be collaborating with the following department which is the Prototype Lab (For Extra Help). In the winter semester we planned to form the group of Aldo Ndreu and following students, who are also building similar hardware this term Ryan Antolin and Erick Cantos. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project but will need some changes and adjustments in order to meet specific requirements. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 3-student group.

## Background

The problem solved by this project which we will be creating will be the capability of being able to connect to an amplified Speaker Bonnet via FM Radio. How this will work is by taking a mobile device and connecting to the database in order for the FM Radio stations to play through the Speaker Bonnet. A bit of background about these topics will include both devices being used. One of the device being used will be the FM evaluation board tuner chip. This device does more then tuning into FM stations, it can also detect both data service and radio broadcast data service. It can also be used to display station id and song to the user as well as have great filtering and carrying detection. This board will be able to pick up multiple radio stations and makes a great tool in order for it to be implemented with a Raspberry Pi. The other device used is the amplified speaker bonnet. By using the speaker bonnet, this will act as the output for FM Tuner sensor and will be the primary source in which the sound will be coming from. It will amplify the audio by boosting the signal in certain areas that you usually cannot hear with your current mobile speakers.

We have searched for prior art via Humber’s IEEE subscription selecting “My Subscribed Content” and have found and read which provides insight into similar efforts.

## Concluding remarks

This proposal presents a plan for providing a solution for FM Tuner sensor to be connected with the amplified speaker bonnet for high quality sound. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative capstone project demonstrating my ability to learn how to support projects. We request approval of this project.

# Abstract

Amplified sound is important when wanting to hear certain audio at a greater and increased state. By connecting an FM tuner sensor to a speaker bonnet, the outputted sound should be at a level where it is enjoyable and meets user satisfaction when tuning to different frequencies. This system will allow users to tune into certain frequencies while also producing amplified sound. The FM tuner sensor will obtain data through the mobile application from the database and allow different frequencies to be sent to the Raspberry Pi in order to allow users to listen to certain stations. The database will grab most of the sensors data, and make it available to an Android application. The application will display different user selections as well as a favourites tab to satisfy user preference. This system has the potential to be unique given all the necessary implementation and should provide users great satisfaction when used assuming they have internet connectivity.

Table of Contents

[Declaration of Joint Authorship iii](#_Toc512207200)

[Approved Proposal v](#_Toc512207201)

[Executive Summary v](#_Toc512207202)

[Background vi](#_Toc512207203)

[Concluding remarks vi](#_Toc512207204)

[Abstract vii](#_Toc512207205)

[Illustration List xii](#_Toc512207206)

[1. Introduction 1](#_Toc512207207)

[2. Project Description 2](#_Toc512207208)

[2.1 Problem 2](#_Toc512207209)

[2.2 Rationale Behind Project 2](#_Toc512207210)

[2.3 Project Scope 2](#_Toc512207211)

[2.4 Requirement Specifications 4](#_Toc512207212)

[2.4.1 Database 4](#_Toc512207213)

[2.4.2 Mobile Application 4](#_Toc512207214)

[2.4.3 Software 5](#_Toc512207215)

[2.4.4 Hardware 5](#_Toc512207216)

[2.5 Project Overview 6](#_Toc512207217)

[2.5.1 Bill of Materials 6](#_Toc512207218)

[2.5.2 Time Commitment 6](#_Toc512207219)

[2.5.3 Mechanical Assembly 7](#_Toc512207220)

[2.5.4 PCB and Soldering 9](#_Toc512207221)

[2.5.5 Power Up 10](#_Toc512207222)

[2.5.6 Unit Testing 10](#_Toc512207223)

[2.5.7 Production Testing 11](#_Toc512207224)

[2.6 Problems Encountered 12](#_Toc512207225)

[2.6.1 Analog to Digital 12](#_Toc512207226)

[2.6.2 Discarding the Speaker Bonnet 12](#_Toc512207227)

[2.7 Approaches 13](#_Toc512207228)

[2.7.1 Analog to Digital 13](#_Toc512207229)

[2.7.2 Discarding the Speaker Bonnet 13](#_Toc512207230)

[2.8 Walkthrough of System 14](#_Toc512207231)

[2.8.1 Raspberry Pi 14](#_Toc512207232)

[2.8.2 Phone Application 14](#_Toc512207233)

[2.8.3 Server (Firebase) 14](#_Toc512207234)

[3. Progress Reports 15](#_Toc512207235)

[3.1 Report 1 15](#_Toc512207236)

[3.2 Report 2 18](#_Toc512207237)

[3.3 Report 3 19](#_Toc512207238)

[3.4 Report 4 22](#_Toc512207239)

[4. Conclusions 24](#_Toc512207240)

[5. Recommendations 25](#_Toc512207241)

[6. Technical References 26](#_Toc512207242)

[7. Appendices 26](#_Toc512207243)

[7.1 Microcontroller Firmware 26](#_Toc512207244)

[7.1.1 FM Tuner Firmware 26](#_Toc512207245)

[7.1.2 Speaker Bonnet Firmware 32](#_Toc512207246)

[7.2 Android Phone Application 38](#_Toc512207247)

[7.2.1 Station List Adapter 38](#_Toc512207248)

[7.2.2 Station 42](#_Toc512207249)

[7.2.3 FavouriteListFragment 47](#_Toc512207250)

[7.2.4 StationListFragment 53](#_Toc512207251)

[7.2.5 SharedPreference 62](#_Toc512207252)

[7.2.6 MainActivity 65](#_Toc512207253)

[7.2.7 RadioActivity 71](#_Toc512207254)

[7.2.8 activity1.xml 78](#_Toc512207255)

[7.2.9 activity\_main.xml 80](#_Toc512207256)

[7.2.10 content\_main.xml 81](#_Toc512207257)

[7.2.11 fragment\_station\_list.xml 82](#_Toc512207258)

[7.2.12 listfragment.xml 82](#_Toc512207259)

[7.2.13 radio\_fragment.xml 83](#_Toc512207260)

[7.2.14 station\_list\_item.xml 83](#_Toc512207261)

[7.2.15 menu\_option.xml 85](#_Toc512207262)

[7.2.16 colors.xml 85](#_Toc512207263)

[7.2.17 dimens.xml 86](#_Toc512207264)

[7.2.18 strings.xml 86](#_Toc512207265)

[7.2.19 strings\_fav\_msg.xml 88](#_Toc512207266)

[7.2.20 styles.xml 89](#_Toc512207267)

[7.2.21 styles.xml (fr) 89](#_Toc512207268)

[7.2.22 build.gradle 91](#_Toc512207269)

[7.2.23 AndroidManifest.xml 94](#_Toc512207270)

# Illustration List

2.3a UML Diagram. p.2

2.5.3a PCB Schematic p.7

2.5.3b PCB Layout p.8

# 1. Introduction

The task of being able to deliver amplified quality sound while being able to connect a device or sensor to this type of device requires specific hardware setup. To this end, we have developed an integrated solution by creating a speaker bonnet which acts as the output to the FM tuner sensor by picking up the local frequencies and producing them in an amplified state.

Our system will have a database that will have local radio station frequencies which will be accessible via an android application and Raspberry Pi.

Despite possible redundancy of our system due to other products capable of playing FM radio stations, and other amplified speaker created and used, we nonetheless feel that our system will be unique in the sense that we are combining both sensors in a way that has never been done before.

Due to both sensors needing to be connected to specific pins, our system needed both sensors to be able to interact together without any conflicting issues, which we solved by creating a custom PCB board as well as acquiring a header in order to have the appropriate connection.

We will make all source code for our Android Application using Android Studio and in addition we will be using Python in order to communicate to the Raspberry Pi available.

# 2. Project Description

## 2.1 Problem

Creating an amplified FM Tuner which is functionable through a cellular device is not an easy task to accomplish. Most currently existing systems are either different in the way they are build, or don’t offer the same functionality in order to comply with user satisfaction.

## 2.2 Rationale Behind Project

In order to solve this problem, a small-scale, inexpensive, amplified FM Tuner was created in order to produce radio signals through a Speaker Bonnet. With the use of a cellular device, this system will be capable of grabbing data from the firebase database which will then trigger the Raspberry Pi to run the FM Tuner through the Speaker Bonnet making it unique to systems that have been deployed in the past.

## 2.3 Project Scope

To meet these requirements, this system has been developed with a certain set of criteria that had to be met in order to solve the problem. The system must be able to communicate with different aspects of integration such as the mobile application, database, and the hardware. In order for the system to perform it’s required tasks, the system must be able to read data from a database, which contain specific fields for each radio stations. To facilitate the smart phone application interface, the hardware must be able to retrieve information from the database. A follow up diagram is shown for further elaboration based on our system scheme.

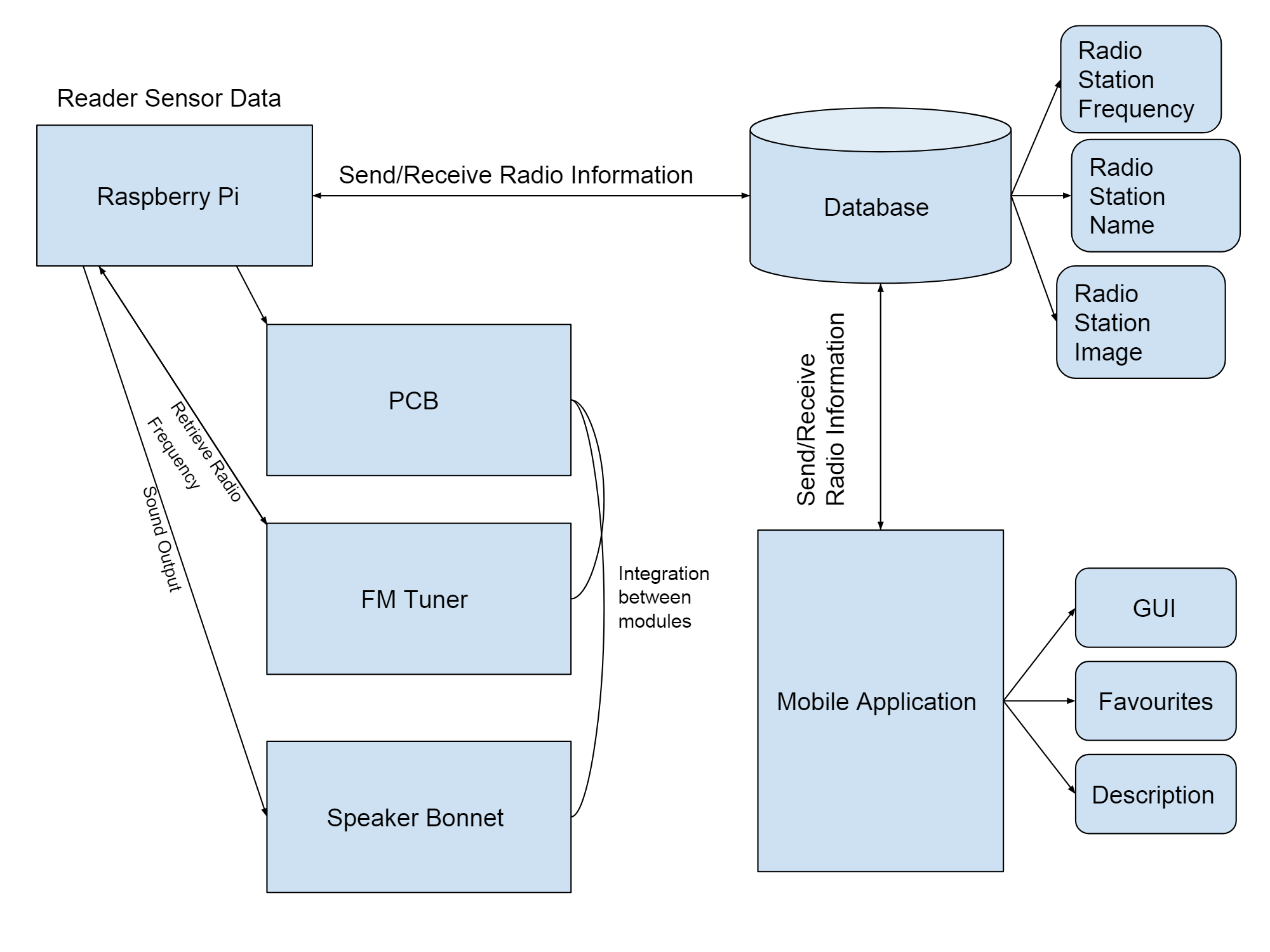


Image 2.3a UML Diagram

## 2.4 Requirement Specifications

### 2.4.1 Database

There has been a Firebase database created in order for this project to store different values for multiple radio frequencies. This database holds a variety of frequency information regarding each station in relation to the FM Tuner evaluation board. The database contains a single table for each radio frequency (or station), and the fields in the table include details such as the name, frequency number, and a description or logo of each FM radio station. Dummy data has been stored so far as the connection between the database, app, and Raspberry Pi has yet to be established. (Developed by Ryan Antolin)

### 2.4.2 Mobile Application

There has been a mobile application (currently only available on Android platforms) created which will take the data from the database and correspond it to the appropriate FM radio station to display all necessary information when selected. This will trigger the FM tuner to find the station that is selected to play and a corresponding image will be taken from an online source (with copyrights) according to each station. The application created is setup with a selection of different radio stations that currently play through hard coded radio streams. The favorites portion of the app has been created that allow users to favourite certain radio stations by using shared preference. The main focus now is retrieving frequencies from the database in order for them to be played through the Raspberry Pi. This will remove the hard-coded links for each radio station and will be using the FM Tuner board as it’s source for playing radio stations. (Developed by Aldo Ndreu)

### 2.4.3 Software

The software aspect of this project will combine the code used on the FM Tuner with the drivers installed specifically for the speaker bonnet. This part of the project requires the python code to be taken from the FM tuner sensor and be combined with the speaker bonnet. The speaker bonnet uses an installation process along with a specific script in order to run and play sound which is then integrated with the python code used for the FM tuner sensor. Once the hardware has been correctly setup and the correct code has been applied, they should perform the appropriate task which is getting the frequencies from the FM tuner and audio being outputted through the speaker bonnet. (Developed by Aldo Ndreu)

### 2.4.4 Hardware

The hardware implemented for this project has been specifically designed and created to connect both sensors being used onto a single platform. Both sensors require specific pin connection on the Raspberry Pi which is why a fritzing diagram and other layouts have been created in order to brainstorm the connection procedure. A custom PCB board has also been created in order for the connections to align according to requirements and two header pins are used in order to connect both sensors into the appropriate slots. Once all the appropriate procedures have been followed, there should be a successful connection by both sensors. (Developed by Erick Cantos)

# 2.5 Project Overview

### 2.5.1 Bill of Materials

1x Spark Fun FM Tuner Evaluation Board Si4703 – $28.99

1x Straight Break Away Headers - $2.99

1x Adafruit Speaker Bonnet for Raspberry Pi – $12.95

1x Raspberry Pi starter kit - $99.99

1x 8 Ohm 3-Watt speaker - $1.95

2x 22 AWG gauge electrical wire (3 cm) – $14.50 (10m / 32ft)

2x Alligator clips - $7.73 (Pack of 10)

1x Female connection header 8 positions .1” TIN - $0.87

1x Female connection header 40 positions .1” TIN - $3.49

1x Stacking Raspberry Pi header 40 positions - $3.94

Product prices are subjected to change and will vary over time due to currency changes and/or

supplier price change.

### 2.5.2 Time Commitment

After the acquisition of the required materials, it is expected that at least five hours of time

must be allocated in order to reproduce the project if the code included is used. This estimate

on completion takes into consideration that the person building the project has adept

knowledge on soldering and programming. However, if the code is developed yourself more

time may be expended into completing the project. The importance of correctly ordering the

parts and thoroughly following the build instructions minimizes the opportunity to create

any project errors or possible delays.

### 2.5.3 Mechanical Assembly

FM Radio Tuner:

1. Prepare the Spark Fun FM Tuner Evaluation Board Si4703 by soldering the straight

break away headers onto it. These pins will be connected to the Si4703 GPIO terminals

1. Once the pins have been soldered onto the Si4703 board, plug in a set of

headphones or Speakers with a 3.5mm jack into the Si4703 board.

Speaker:

1. First, start with the preparation of the Adafruit Speaker Bonnet by soldering the twenty

position female header onto the GPIO terminals located on the border. It is considered

easier to solder the rest of the pins if you begin by soldering the first two pinouts and

the last two to keep the header in place while you solder the rest of the pins.

1. When the soldering is complete, take the two pieces of electrical wire and screw one

wire into the positive terminal on the right speaker terminal and the other wire into the

negative terminal on the right speaker connector. If you wanted to use an additional

speaker, it could be wired to the left speaker terminal using the same method.

1. Lastly, connect one end of one alligator clip onto the electrical wire on the positive

terminal and the other end will connect to the positive terminal on the speaker. The

other alligator clip will hook onto the electrical wire on the negative terminal which will

then connect to the negative terminal on the speaker.

Printed Circuit Board:

1. The schematic for the PCB is provided within the Project Files Folder. When

designing/replicating the diagram consider that the PCB must fit the raspberry pi. The

recommended size for the PCB should reach 55mm in width and 57mm in height to sit all

components comfortably together. This is first adjustment that should be completed prior

making the design.

1. After the PCB size orientation has been completed, connections from raspberry pi to

header connections must be implemented. The schematic below should be a reference

on how the circuit is connected.

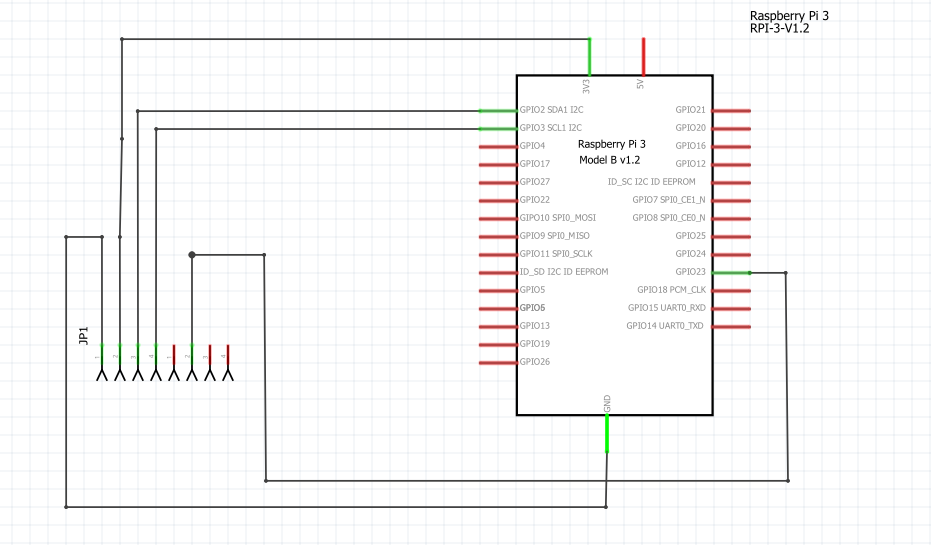


Image 2.5.3a PCB schematic

Five out of the eight pins are used on the JP1 header which are SCL, SDA, GND, 3v3, and

GPIO23. These pins will enable sensor reading and power up to the FM Tuner.

1. Following the schematic should come the PCB layout. Ensure there is enough space

between the raspberry pi connections and the header for the FM Tuner prior to submitting

Gerber files for manufacturing.

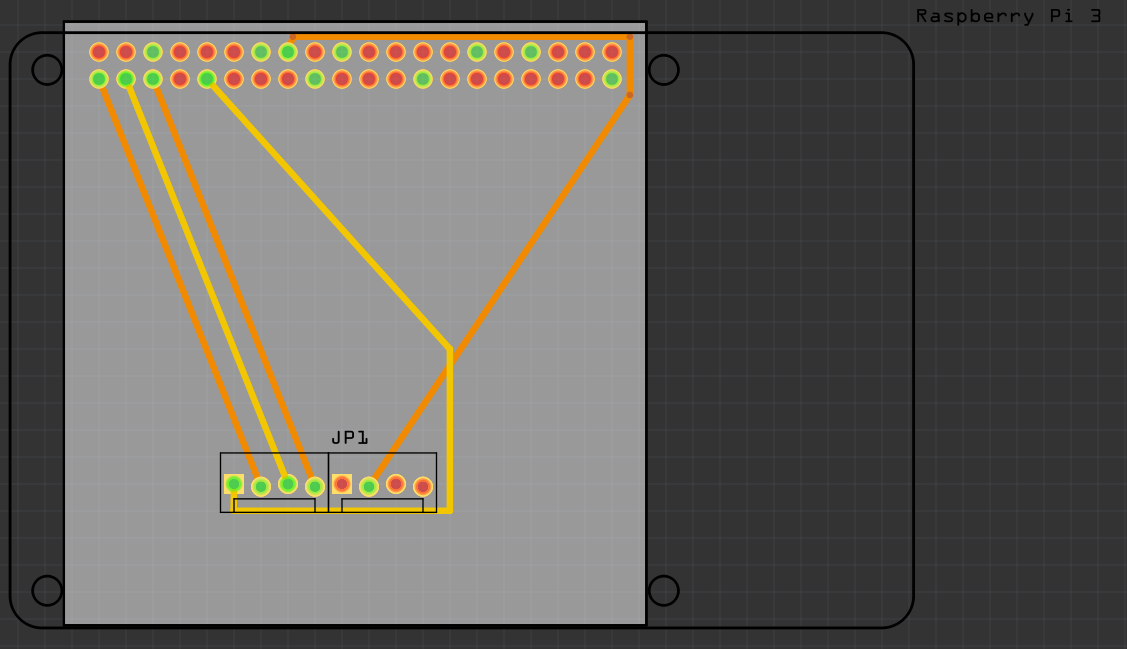


Image 2.5.3b PCB layout

1. The printed board will now be prepared to have the header connections soldered on.

Solder on the eight-pin header on top of the PCB following up with soldering the Raspberry

Pi stackable header underneath the Raspberry Pi GPIO.

1. Finally, complete the assembly by aligning the stackable header portion of the PCB to

sit comfortably on the Raspberry Pi. Seat the speaker bonnet directly on top of the

stackable header. Lastly, take the FM Tuner and seat it into the eight-pin header across

from the speaker bonnet. It is now ready for software testing.

### 2.5.4 PCB and Soldering

The PCB will provide a platform that will enable the integration between the speaker

bonnet and the FM tuner. Both these devices will be mounted onto a single board so that

communications can be established between them and progress their functionality together.

Depending on the type of software development tool that you use, some traces that are

connected on the top side of the board may have to connected manually to the to bottom

of the board and header pins. If this is the case then it is suggested that you acquire electrical

wire with about 18 mm in diameter and solder the wire onto a trace with enough wire to pass

through a pinout hole. Once the wire is through the pinout hole solder the header pin and wire

onto the board at the same time.

### 2.5.5 Power Up

First, if you are booting for the first time you will need to insert a micro SD card into the micro

SD reader on the Raspberry pi. The micro SD card must contain an image of the Raspberry pi

NOOBS software so we can install an operating system and can be downloaded from the link:

https://www.raspberrypi.org/downloads/noobs. Plug in the power, boot to the Raspberry

Pi, and select one of the Raspbian preinstalled operating systems. After the installation of the

operating system you must configure basic settings before anything else, these settings include

configuration of the internet, and VNC viewer. Lastly, users should also have I2C enabled in

order to complete this project. Follow these steps to enable I2C:

**Start> Raspberry Pi configuration > Interfaces and ensure I2C is enabled.**

### 2.5.6 Unit Testing

A reminder that the FM Tuner must have a 3.5 mm headphone connected to it for proper

testing. After the verification of headphones, we will need to ensure that the FM Tuner sensor

is operating correctly by opening a terminal on the raspberry pi and entering the command

“sudo i2cdectec -y 1”. The use of this command will display an output of many addresses and

you must ensure that the sensor is connected at address 10. Afterwards the correct python

code must be compiled to test functionality of the sensor which can be found in the Firmware

portion of the technical document. Save the python file as sudo yourfilename.py and compile

the program within the terminal using the command “python filename.py”. This command

will compile and run the program if there are no errors and will be executed.

Moving onto the Speaker bonnet, you are required to install the drivers and acquire the scripts

through the Adafruit website. There is an alternative which is using the advanced version but it

is not recommended for a novice user. To run the script, use the command “curl –

sS https://raw.githubusercontent.com/adafruit/Raspberry-Pi-Installer-

Scripts/master/i2amp.sh | bash”. You will be prompted to reboot your Raspberry Pi device

after the installation. You will be able to use the speakers once the Raspberry Pi is rebooted.

Finally, you must open a terminal and download mpg123 which will be the media player for the

device. You can do so by running the command “sudo get-apt install -y mpg123”. Personal

music can now be playable via mpg123. On a side note ensure files do not have spaces and

they are replaced with dashes. Spaces on UNIX based operating systems are not comparable to

spaces on the Windows Operating Systems.

# 2.5.7 Production Testing

If the program for the FM Tuner runs successfully without any errors then the output will

display multiple registry values and prompt the use to enter a radio station. As an example,

if the user would like to listen to the radio station 99.9 Virgin radio, simply enter 999. This is

programmed to run in a loop so that every time the user is prompted to enter a radio station

without having to re-run the program. If the FM Tuner is not providing any registry values,

ensure that the sensor is seated correctly onto the PCB. Ensure I2C is enabled at all times.

# 2.6 Problems Encountered

### 2.6.1 Analog to Digital

The FM Tuner had been designed to input and output analog signals whereas the Speaker

Bonnet was designed to input digital signals and output analog signals. With the initial goal of

having a singular piece of hardware in mind, we needed to combine these two modules without

anything else being intermediary. Without additional hardware this proved to be difficult.

### 2.6.2 Discarding the Speaker Bonnet

During one of the phases of the project we had speculated using only the speaker attached to

the Speaker Bonnet. The sound output was not a huge success since the speaker was not able

to amplify the sound loud enough on its own. In order to amplify the sound

additional hardware would have to be developed. This of course would change our project

direction drastically.

# 2.7 Approaches

### 2.7.1 Analog to Digital

Several approaches were taken when considering that we needed to take analog signals

and convert them to digital. The first approach was with the use of a USB sound card adapter

that had a line in for the mic. We were sure we were able feed the analog signals coming from

FM Tuner into the raspberry pi and have it redirect the output into the speaker bonnet. As

it still remains as a plausible solution we ultimately did not go through with it since it required

an abundance of much more research when we discovered that we possibly needed to work

with codecs. The second approach was the solution that allowed the modules to be versatile.

Two python scripts were created with the purpose of controlling each module. The scripts will

connect to a shared data base which will allow both devices to see what they both should be

doing. Instead of a hardware integration, the integration had now become a virtual

integration which allowed both modules to act in unison.

### 2.7.2 Discarding the Speaker Bonnet

To address this issue, we had the FM Tuner and the Speaker Bonnet connect to a shared

database through python scripts. We had developed the scripts to get the modules to interpret

the database and perform their own tasks accordingly. This set us back on track and we did not

need to eliminate the Speaker Bonnet anymore.

# 2.8 Walkthrough of System

### 2.8.1 Raspberry Pi

The FM Tuner sensor and Speaker Bonnet were connected through a PCB which was mounted on the Raspberry Pi’s 20pin GPIO pin-out. The Speaker Bonnet sensor and the FM tuner sensor will then wait for both python scripts that will sync at the same time when audio is playing back requiring connection to the database.

### 2.8.2 Phone Application

The current phone application has predefined radio stations implemented on it’s main screen. Each radio station will sync the data from the phone application onto the database which will send through each time a user clicks on a station and hits the play button. This will change the frequency values to the station that is played from the phone application.

### 2.8.3 Server (Firebase)

Firebase is the database used where we defined each stations frequency, name, and description. These fields will show up on the phone application once we have a child made on firebase and called on the java application. From there, each time a user switches stations, it will change the frequency value table that is connected with the python script which will read the user station that was selected on the application.

# 3. Progress Reports

### 3.1 Report 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Erick Cantos <erickcantos@live.ca>** | | Mar 5  https://mail.google.com/mail/u/0/images/cleardot.gif |  | https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
| |  | | --- | | to Kristian, Aldo, me | | | |

Dear Kristian,

        This is will be an update on our individual progress in regard to the Radio project that we are developing. In conjunction with our progress, we also would like to present our financial status and project problems with proposed solutions.

        Throughout the past weeks, my partners Ryan and Aldo have worked persistently into completing different aspects of the technical report for our Radio project. They have contributed by subdividing key topics on the technical report that will pave our foundations for completing our Radio project such as methodologies, implementations, general requirements, and work breakdown. The key to our success will lie behind our technical documentation. We can ensure that each group member is up to date with the latest version of this documentation. As a group, we strive to ensure that all our information stays accurate and remains inclusive where all things are written are agreed on by all members of the group.

        As a group member that joined during the beginning stages of development, I took on the responsibility to ensure we can incorporate our own PCB so we could integrate Ryan’s Adafruit Stereo Speaker Bonnet (<https://www.adafruit.com/product/3346>) with Aldo’s Sparkfun FM Tuner Evaluation Board(<https://www.sparkfun.com/products/12938>) onto a single raspberry pi device. In contrast to how they were running on individual raspberry pi’s. The PCB mock designs went through two drafts before my partners and I compromised on the second design. Following the design completion, I made sure the PCB was submitted and printed at the Humber College prototype lab.

        Upon design submission, our team members agreed that this was the best time to order the parts so we could start the soldering process as soon as possible. We simply needed two types of connections headers which we ordered of the website [https://www.digikey.ca](https://www.digikey.ca/).

        The first part is an 8-position female connection header in which we needed to situate Aldo’s FM tuner which cost $0.87 (<https://www.digikey.ca/products/en?keywords=%20S7006-ND> ). And the second was a stacking header for the Raspberry PI which made it easy to accommodate Ryan’s speaker which cost $3.94 (<https://www.digikey.ca/products/en?keywords=%09%201528-1783-ND>).

        We’ve spent the total amount of $13.44 which includes the shipping and handling fees and the governmental tax. Last week all the parts arrived on Tuesday in which we organized a group meeting to solder the parts correctly onto the printed PCB with the assistance of Vlad and Kelly over in the prototype room.

        Followed by completing the hardware, we began to face problems during our functionality stages. The PCB that I worked extremely hard on creating was fully capable of supporting and operating the Speaker Bonnet and FM Tuner, but individually only. The FM tuner uses a 3.5-millimeter audio jack that acts as both the speaker and the antenna for the radio. And Ryan’s speaker uses the GPIO pinouts on the raspberry pi which uses digital signals. This is where the problem arises where Ryan’s speaker is not receiving any sound.

        The group has been working on possible solutions conscientiously. We’ve reached out to Kelly in the prototype lab, he suggested that we purchase a USB audio adapter with an auxiliary cable that will feed the sound from the FM Tuner back into the Raspberry Pi. This will include some sort of manipulation to get the sound output onto the Speaker Bonnet. We are faced with a difficult task and do not know if this is a guaranteed solution. We are on restricted time and we have worked extremely hard to achieve what we have now.

        We are very attentive to any suggestions, advice, or ideas that you may have to offer on the integration of our project. Furthermore, as the worst situation would it be possible if dropping Ryan’s bonnet speaker was the last option so we could focus on the FM Tuner and the software aspect of this project.

Sincerest regards,

Erick Cantos

### 3.2 Report 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Aldo Ndreu <n01108481@humbermail.ca>** | | Mar 9  https://mail.google.com/mail/u/0/images/cleardot.gif |  | https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
| |  | | --- | | to erickcantos, me  https://mail.google.com/mail/u/0/images/cleardot.gif | | | |

Good afternoon sir,

This is Aldo from your CENG355 course and I am emailing you today regarding the issue our group is having when combining my FM Tuner and Ryan's speaker bonnet. We went ahead with trying the solution of using a USB audio adapter with an auxiliary cable in order to get the sound from my sensor back into the Raspberry Pi and to output through Ryan's speaker but are unable to. We are unable to get my radio to output through Ryan's speaker and we tried multiple methods such as editing configuration files by changing the default output of the Raspberry Pi and looking at sound files with no solution. The USB audio adapter has input jack where we plugged an aux cable into it and back to my sensor and also an output jack where the output is coming from whatever is plugged in even though we want the sound to come from Ryan's speaker. When we run our projects, both of them happen to run on their own separately such as Ryan's speaker is playing audio from an online source and my FM Tuner is playing the stations through whatever is plugged into the USB output which happened to be headphones when we did the testing. It is like the two projects are not even connected to each other. We talked to Vlad and Kelly and even with the input they provided, we were still unable to find a clear solution. We are still working extremely hard to try and get this issue solved but are wondering worse case scenario if we can drop Ryan's speaker and just focus on creating the app and database given the amount of time we have left.

Thank-you,

Aldo Ndreu

### 3.3 Report 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Aldo Ndreu <n01108481@humbermail.ca>** | | Mar 19  https://mail.google.com/mail/u/0/images/cleardot.gif |  | https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
| |  | | --- | | to erickcantos, me, kristian.medri  https://mail.google.com/mail/u/0/images/cleardot.gif | | | |

Dear Kristian,

     This is will be a status update on our integration progress in regard to the radio project that we have been developing throughout the semester. In conjunction with our current progress, we also would like to present our financial status and project problems with proposed solutions.

     Throughout the past weeks, myself and my partners Erick and Ryan have worked persistently into completing different aspects of the technical report for our radio project.

     As a group, we have divided specific tasks we need to accomplish in certain stages of development to ensure the integration progress is successful. Erick took on the responsibility to ensure we can incorporate our own PCB so we could integrate Ryan’s Adafruit Stereo Speaker Bonnet (<https://www.adafruit.com/product/3346>) with my Sparkfun FM Tuner Evaluation Board (<https://www.sparkfun.com/products/12938>) onto a single Raspberry Pi device by creating PCB designs. The PCB mock designs went through two drafts before Erick submitted them to be printed at the Humber College prototype lab.

     Upon the design submission, our team members agreed that this was the best time to order the parts so we could start the soldering process as soon as possible. We simply needed two types of connections headers which we ordered of the website ([https://www.digikey.ca](https://www.digikey.ca/)).

     The first part is an 8-position female connection header in which we needed to situate my FM Tuner which cost $0.87 (<https://www.digikey.ca/products/en?keywords=%20S7006-ND> ). And the second part was a stacking header for the Raspberry PI which made it easy to accommodate Ryan’s speaker which cost $3.94 (<https://www.digikey.ca/products/en?keywords=%09%201528-1783-ND>).

     We’ve spent a total amount of $13.44 which includes the shipping and handling fees as well as governmental tax. We also inquired about a USB audio adaptor which we were able to get a hold of free of charge in order to continue with the integration of our hardware as well as an auxiliary cable.

     Followed by completing the hardware, we began to face problems during our functionality stages. The PCB that Erick worked extremely hard on creating was fully capable of supporting and operating the Speaker Bonnet and FM Tuner, but individually only. The FM Tuner uses a 3.5-millimetre audio jack that acts as both the output speaker and the antenna for the radio. Meanwhile, Ryan’s speaker uses the GPIO pinouts on the Raspberry Pi which uses digital signals. This is where the problem arises where Ryan’s speaker is not receiving any sound from the FM Tuner rather it's own audio from an online source.

     The group has been working on possible solutions conscientiously. We’ve previously reached out to Kelly in the prototype lab, in which he suggested that we purchase a USB audio adapter with an auxiliary cable that will feed the sound from the FM Tuner back into the Raspberry Pi. We tried implementing the USB audio adaptor as a solution but still had no luck in integrating my FM Tuner to associate with Ryan's Speaker Bonnet. The auxiliary cable from my FM Tuner that connects to the USB audio adaptor is practically feeding the audio back to the USB where it can only be heard through an output device. Through more research and findings, we have attempted numerous possible methods such as editing configuration files by changing the default output of the Raspberry Pi as well as looking at and modifying sound files with no solution. The group is still hard at work in attempts to find a permanent solution to this issue. With the issue at hand, the group has still progressed in creating our database as well as the application aspect of our project. We have yet to figure out how we can integrate these three aspects of our project in order for them to work in cohesion.

     We are very attentive to any suggestions, advice, or ideas that you may have to offer on the integration of our project. Furthermore, as the worst situation would it be possible if dropping Ryan’s Speaker Bonnet was the last option so we could focus on the FM Tuner and the software aspect of this project.

Sincerest regards,

Aldo Ndreu

### 3.4 Report 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **Ryan Antolin <ryan.antolin@gmail.com>** | | Apr 1  https://mail.google.com/mail/u/0/images/cleardot.gif |  | https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
| |  | | --- | | to kristian.medri, n01108481, erickcantos | | | |

Dear Kristian,

This will be a status update regarding the integration progress of our radio project that has been in under development throughout this semester. With the current progress we initially have at the moment, we want to present our troubleshooting status.

Throughout the weeks of trying our best to implement other ways to get our project working, myself and my partners Erick and Aldo have worked to attempt all the troubleshooting aspects to continue forward into integration.

As a group, we first made a flowchart to figure out what needs to be targeted and see it in a bigger picture than our schematics. We all contributed to our respective parts on the flowchart. As a result, we figured that we made a list of all the troubleshooting ideas we need until our demonstration day.

Our methods into tackling our problems are as  followed: configuration of the Raspberry Pi files, forcing audio through HDMI, new code/hybrid code (start from scratch or revamping python code), play audio separately, use an analog signal to play to speaker, or ADC converter.

As we attempted to implement some of these methods, there were problems we came across that still led to us having trouble with the integration. We tried to configure the Raspberry Pi files but every time we would try to play a song through the FM tuner, it will still only play on the FM tuner. If we play audio separately, there would be synchronization problems with audio coming through with the speaker bonnet. If we use an ADC converter, it will require to buy it online which we cannot obtain before demonstration day as well as it may not be a guaranteed solution to our problem.

We, as a group, are still trying our best to troubleshoot our project. Furthermore, if we cannot get anything to work from our troubleshooting techniques, we will most likely play the audio separately as it is the closest method to our integrated modules actually working.

Sincerest regards,   
Ryan Antolin

# 4. Conclusions

This system has been developed to utilize radio signals that will meet our goals into integrating the FM Tuner sensor with a speaker bonnet in order to play from analog to digital. These modules are connected to each other through a PCB which allows the process of having a built-in speaker and FM Tuner running simultaneously. This system processes data from a mobile device onto the database which will then send frequencies to the FM Tuner sensor to allow communication between the mobile device and sensor. Once the project reaches completion, it should be able to meet all requirements we need in order to finish off the integration goals.

# 5. Recommendations

During the weeks of working on our project prototype, we were thinking of being able to make a case design for the Raspberry Pi and the speaker so that the speaker will be on top of the Pi in an enclosure protecting the sensors, PCB, and the Pi. Although the price will go up, the aesthetics of the prototype would look way better than how it is setup right now. Hardware modifications we could have done to this was adding two 4W 3ohm speaker than one 1W 8ohm so that it could be much louder for others to hear and possibly get stereo to work on both speakers in order to use two different signals instead of one signal (mono). Application modifications we could have done was add a volume slider to control the audio being played back onto the Pi since volume is controlled from the script. We could have added many more stations on the application and on the firebase to show that all the frequencies are able to be captured on the FM Tuner sensor.

# 6. Technical References

[1] Bluetooth in wireless communication. (n.d.). Retrieved September 18, 2017, from http://ieeexplore.ieee.org/document/1007414

[2] SparkFun FM Tuner Evaluation Board - Si4703. (n.d.). Retrieved from https://www.sparkfun.com/products/12938

[3] Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

[4] Lumpkins, W. (n.d.). The MobiAria Wireless Bluetooth Speaker. Retrieved September 18, 2017, from http://ieeexplore.ieee.org/document/6685931

[5] Bodson, D. (n.d.). Digital Audio Around the World. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5641649

[6] Pauli, M. (2017, May 5). Miniaturized Millimeter-Wave Radar Sensor for High-Accuracy Applications. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7885501

# 7. Appendices

### 7.1 Microcontroller Firmware

### 7.1.1 FM Tuner Firmware

#Import Statements

import RPi.GPIO as GPIO

import smbus

import time

from pyrebase import pyrebase

#Connecting To Database

from pyrebase import pyrebase

config = {

"apiKey": "AIzaSyD1C1oFmuUKBPeRyCjj5Z4G7m443A5H4lw",

"authDomain": "radiopi-6998d.firebaseapp.com",

"databaseURL": "https://radiopi-6998d.firebaseio.com",

"projectId": "radiopi-6998d",

"storageBucket": "radiopi-6998d.appspot.com",

"serviceAccount": "/home/pi/Documents/RadioPi-c20135ff3ec2.json"

}

firebase = pyrebase.initialize\_app(config)

#Authentication

auth = firebase.auth()

#authenticate a user

user = auth.sign\_in\_with\_email\_and\_password("aldondreu9@gmail.com", "bluejays123")

db = firebase.database()

all\_values = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

print all\_values

#Initialization of FM Tuner

i2c = smbus.SMBus(1) #use 0 for older RasPi

GPIO.setmode(GPIO.BCM) #board numbering

GPIO.setup(23, GPIO.OUT)

GPIO.setup(0, GPIO.OUT) #SDA or SDIO

#put SI4703 into 2 wire mode (I2C)

GPIO.output(0,GPIO.LOW)

time.sleep(.1)

GPIO.output(23, GPIO.LOW)

time.sleep(.1)

GPIO.output(23, GPIO.HIGH)

time.sleep(.1)

address = 0x10 #address of SI4703 from I2CDetect utility

print "Initial Register Readings"

reg = i2c.read\_i2c\_block\_data(address, 0, 32)

print reg

#write x8100 to reg 7 to activate oscellitor

list1 = [0,0,0,0,0,0,0,0,0,129,0]

w6 = i2c.write\_i2c\_block\_data(address, 0, list1)

time.sleep(1)

#write x4001 to reg 2 to turn off mute and activate IC

list1 = [1]

#print list1

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

time.sleep(.1)

#write volume

print "Doing VOlume lowest setting"

list1 = [15,0,0,0,0,0,15]

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

#write channel

print "Setting Channel, pick a strong one"

nc = 1011 #this is 101.1 The Fox In Kansas City Classic Rock!!

nc \*= 10 #this math is for USA FM only

nc -= 8750

nc /= 20

list1 = [1,128, nc]

#set tune bit and set channel

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

time.sleep(1) #allow tuner to tune

# clear channel tune bit

list1 = [1,0,nc]

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

reg2 = i2c.read\_i2c\_block\_data(address,64, 32)

print reg2 #just to show final register settings

#You should be hearing music now!

#Headphone Cord acts as antenna

var = 1

while var == 1 :

print "Setting FM Frequency:"

single\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

print "Tuning", single\_value

nc = int(single\_value)

nc \*= 10 #this math is for USA FM only, Greetings from San Jose California

nc -= 8750

nc /= 20

print "Param: ", nc

list1 = [1,128, nc]

#set tune bit and set channel

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

time.sleep(.1) #allow tuner to tune

# clear channel tune bit

list1 = [1,0,nc]

w6 = i2c.write\_i2c\_block\_data(address, 64, list1)

reg2 = i2c.read\_i2c\_block\_data(address,64, 32)

print reg2 #Your newly entered Freq Param value will show up happily sitting at the 4th position of the registers array

var2 = single\_value

while var == 1 :

single\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

if (var2 != single\_value):

print "Changing the Tuner", single\_value

break

### 7.1.2 Speaker Bonnet Firmware

import vlc

import time

import urllib

from pyrebase import pyrebase

#Connect to Database

config = {

"apiKey": "AIzaSyD1C1oFmuUKBPeRyCjj5Z4G7m443A5H4lw",

"authDomain": "radiopi-6998d.firebaseapp.com",

"databaseURL": "https://radiopi-6998d.firebaseio.com",

"projectId": "radiopi-6888d",

"storageBucket": "radiopi-6998d.appspot.com",

"serviceAccount": "/home/pi/Documents/RadioPi-c20135ff3ec2.json"

}

firebase = pyrebase.initialize\_app(config)

auth = firebase.auth()

user = auth.sign\_in\_with\_email\_and\_password("ErickCantos2018@gmail.com", "HumberCollege")

db = firebase.database()

var2 = None

var = 1

while var == 1 :

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

print single\_value

var2 = single\_value

# This block can turn into switch case statement

if(single\_value == 973):

p = vlc.MediaPlayer('http://newcap.leanstream.co/CHBMFM?args=3rdparty\_02&uid=1a03ae7d-8b81-4a05-83c5-ecdaad822d6d')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

elif(single\_value == 941):

p = vlc.MediaPlayer('http://cbc\_r2\_tor.akacast.akamaistream.net/7/364/451661/v1/rc.akacast.akamaistream.net/cbc\_r2\_tor')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

elif(single\_value == 969):

p = vlc.MediaPlayer('http://radio\_cklg-lh.akamaihd.net/i/VAN969\_1@183241/index\_48\_a-b.m3u8?sd=10&rebase=on')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

elif(single\_value == 925):

p = vlc.MediaPlayer('http://radio\_ckis-lh.akamaihd.net/i/TOR925\_1@176956/index\_48\_a-p.m3u8?sd=10&rebase=on')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

elif(single\_value == 935):

p = vlc.MediaPlayer('http://newcap.leanstream.co/CFXJFM?args=3rdparty\_02&uid=fca8e4a9-dcf4-4b05-9b6a-85af80a32d74')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

elif(single\_value == 1035):

p = vlc.MediaPlayer('http://ice23.securenetsystems.net/CIDC?playSessionID=556E08CF-155D-C0F3-03C6BFB41A25C35B')

p.play()

time.sleep(2)

var2 = single\_value

while var == 1:

oldsingle\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

single\_value = int(oldsingle\_value)

if(var2 != single\_value):

print "Changed Tuned", single\_value

p.stop()

break

else:

print "Error: No radio station found on data base"

break

#Instance = vlc.Instance()

#player = Instance.media\_player\_new()

#media = Instance.media\_new(url)

#player.set\_media(media)

#player.play()

#time.sleep(2)

#p = vlc.MediaPlayer(url)

#p.play()

#time.sleep(2)

while var == 1:

pass

single\_value = db.child("Stations").child("Frequency").child("Name").get(user['idToken']).val()

if(var2 != single\_value):

print "Changed Tune", single\_value

break

# 7.2 Android Phone Application

### 7.2.1 Station List Adapter

package ca.humber.radiopi.radiopi.adapter;

import java.util.List;

import android.app.Activity;

import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ArrayAdapter;

import android.widget.ImageView;

import android.widget.TextView;

import ca.humber.radiopi.radiopi.R;

import ca.humber.radiopi.radiopi.beans.Station;

import ca.humber.radiopi.radiopi.utils.SharedPreference;

public class StationListAdapter extends ArrayAdapter<Station> {

private Context context;

List<Station> stations;

SharedPreference sharedPreference;

public StationListAdapter(Context context, List<Station> stations) {

super(context, R.layout.station\_list\_item, stations);

this.context = context;

this.stations = stations;

sharedPreference = new SharedPreference();

}

private class ViewHolder {

TextView stationNameTxt;

ImageView favoriteImg;

}

@Override

public int getCount() {

return stations.size();

}

@Override

public Station getItem(int position) {

return stations.get(position);

}

@Override

public long getItemId(int position) {

return 0;

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

ViewHolder holder = null;

if (convertView == null) {

LayoutInflater inflater = (LayoutInflater) context

.getSystemService(Activity.LAYOUT\_INFLATER\_SERVICE);

convertView = inflater.inflate(R.layout.station\_list\_item, null);

holder = new ViewHolder();

holder.stationNameTxt = (TextView) convertView.findViewById(R.id.txt\_sta\_name);

holder.favoriteImg = (ImageView) convertView.findViewById(R.id.imgbtn\_favorite);

convertView.setTag(holder);

} else {

holder = (ViewHolder) convertView.getTag();

}

Station station = (Station) getItem(position);

holder.stationNameTxt.setText(station.getName());

/\*If a station exists in shared preferences then set heart\_red drawable

\* and set a tag\*/

if (checkFavoriteItem(station)) {

holder.favoriteImg.setImageResource(R.drawable.heart\_red);

holder.favoriteImg.setTag(getContext().getString(R.string.red));

} else {

holder.favoriteImg.setImageResource(R.drawable.heart\_grey);

holder.favoriteImg.setTag(getContext().getString(R.string.grey));

}

return convertView;

}

/\*Checks whether a particular station exists in SharedPreferences\*/

public boolean checkFavoriteItem(Station checkStation) {

boolean check = false;

List<Station> favorites = sharedPreference.getFavorites(context);

if (favorites != null) {

for (Station station : favorites) {

if (station.equals(checkStation)) {

check = true;

break;

}

}

}

return check;

}

@Override

public void add(Station station) {

super.add(station);

stations.add(station);

notifyDataSetChanged();

}

@Override

public void remove(Station station) {

super.remove(station);

stations.remove(station);

notifyDataSetChanged();

}

}

### 7.2.2 Station

package ca.humber.radiopi.radiopi.beans;

public class Station {

private int id;

private String name;

private String link;

private String description;

private String imageURL;

private String freq;

public Station() {

super();

}

public Station(int id, String name, String freq, String description) {

super();

this.id = id;

this.link = link;

this.name = name;

this.description = description;

this.imageURL = imageURL;

this.freq = freq;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLink() {

return link;

}

public void setLink(String link) {

this.link = link;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public String getImageURL() {

return imageURL;

}

public void setImageURL(String imageURL) {

this.imageURL = imageURL;

}

public String getFreq(){

return freq;

}

public void setFreq(String freq){

this.freq = freq;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + id;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Station other = (Station) obj;

if (id != other.id)

return false;

return true;

}

@Override

public String toString() {

return "Station[id=" + id + ", name=" + name + ", link= " + link + ", description="

+ description + ", imageURL=" + imageURL + "]";

}

}

### 7.2.3 FavouriteListFragment

package ca.humber.radiopi.radiopi.mFragments;

import java.util.List;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.Intent;

import android.support.v4.app.Fragment;

import android.content.DialogInterface;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.AdapterView.OnItemLongClickListener;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.Toast;

import ca.humber.radiopi.radiopi.R;

import ca.humber.radiopi.radiopi.RadioActivity;

import ca.humber.radiopi.radiopi.adapter.StationListAdapter;

import ca.humber.radiopi.radiopi.beans.Station;

import ca.humber.radiopi.radiopi.utils.SharedPreference;

public class FavouriteListFragment extends Fragment{

public static final String ARG\_ITEM\_ID = "favorite\_list";

Intent i;

ListView favoriteList;

SharedPreference sharedPreference;

List<Station> favorites;

Activity activity;

StationListAdapter stationListAdapter;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

activity = getActivity();

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_station\_list, container, false);

// Get favorite items from SharedPreferences.

sharedPreference = new SharedPreference();

favorites = sharedPreference.getFavorites(activity);

//Checking if anything is available

if (favorites == null) {

showAlert(

getResources().getString(R.string.no\_favorites\_items),

getResources().getString(R.string.no\_favorites\_msg));

} else {

if (favorites.size() == 0) {

showAlert(

getResources().getString(R.string.no\_favorites\_items),

getResources().getString(R.string.no\_favorites\_msg));

}

favoriteList = (ListView) view.findViewById(R.id.list\_station);

//Whether or not favourites is empty we still need the listview

if (favorites != null) {

stationListAdapter = new StationListAdapter(activity, favorites);

favoriteList.setAdapter(stationListAdapter);

favoriteList.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View arg1, int position, long arg3) {

//Copied code from StationListFragment onItemClick

Station station = (Station) parent.getItemAtPosition(position);

String description = station.getDescription();

String imageURL = station.getImageURL();

String freq = station.getFreq();

i = new Intent(getActivity(), RadioActivity.class);

i.putExtra(getActivity().getString(R.string.description), description);

i.putExtra(getActivity().getString(R.string.imageurl), imageURL);

i.putExtra(getActivity().getString(R.string.freq), freq);

startActivity(i);

}

});

//set our heart image to red or grey

//however if its on favouritelistfragment it will be red by default

//and grey only for a moment then it will be removed from the list

favoriteList.setOnItemLongClickListener(new OnItemLongClickListener() {

@Override

public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {

ImageView button = (ImageView) view.findViewById(R.id.imgbtn\_favorite);

String tag = button.getTag().toString();

if (tag.equalsIgnoreCase(getString(R.string.grey))) {

sharedPreference.addFavorite(activity, favorites.get(position));

Toast.makeText(

activity, activity.getResources().getString(R.string.add\_favr),

Toast.LENGTH\_SHORT).show();

button.setTag(getContext().getString(R.string.red));

button.setImageResource(R.drawable.heart\_red);

} else {

sharedPreference.removeFavorite(activity, favorites.get(position));

button.setTag(getContext().getString(R.string.grey));

button.setImageResource(R.drawable.heart\_grey);

stationListAdapter.remove(favorites.get(position));

Toast.makeText(

activity, activity.getResources().getString(R.string.remove\_favr),

Toast.LENGTH\_SHORT).show();

}

return true;

}

});

}

}

return view;

}

public void showAlert(String title, String message) {

//if favourites is empty display a dialog box that instructs the user how to add a favourite

if (activity != null && !activity.isFinishing()) {

AlertDialog alertDialog = new AlertDialog.Builder(activity).create();

alertDialog.setTitle(title);

alertDialog.setMessage(message);

alertDialog.setCancelable(false);

// setting OK Button

alertDialog.setButton(AlertDialog.BUTTON\_POSITIVE, getString(R.string.ok),

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

dialog.dismiss();

// activity.finish();

getFragmentManager().popBackStackImmediate();

}

});

alertDialog.show();

}

}

@Override

public void onResume() {

super.onResume();

}

}

### 7.2.4 StationListFragment

package ca.humber.radiopi.radiopi.mFragments;

import java.util.ArrayList;

import java.util.List;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.support.v4.app.Fragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.AdapterView.OnItemLongClickListener;

import android.widget.ImageView;

import android.widget.ListView;

import android.widget.Toast;

import ca.humber.radiopi.radiopi.MainActivity;

import ca.humber.radiopi.radiopi.R;

import ca.humber.radiopi.radiopi.RadioActivity;

import ca.humber.radiopi.radiopi.adapter.StationListAdapter;

import ca.humber.radiopi.radiopi.beans.Station;

import ca.humber.radiopi.radiopi.utils.SharedPreference;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

public class StationListFragment extends Fragment implements OnItemClickListener, OnItemLongClickListener {

public static final String ARG\_ITEM\_ID = "station\_list";

/\*

Initialized Strings for the stations frequencies.

\*/

public static String boom\_Name, boom\_Freq, boom\_Description;

public static String cbc\_Name, cbc\_Freq, cbc\_Description;

public static String humbercollege\_Name, humbercollege\_Freq, humbercollege\_Description;

public static String kiss\_Name, kiss\_Freq, kiss\_Description;

public static String theMove\_Name, theMove\_Freq, theMove\_Description;

public static String z1035\_Name, z1035\_Freq, z1035\_Description;

Intent i;

Activity activity;

//This is where our data is going to end up

ListView stationListView;

List<Station> stations;

//Which then gets passed to this adapter

StationListAdapter stationListAdapter;

SharedPreference sharedPreference;

//DATABASE REFERENCES

private FirebaseDatabase mDatabase;

private DatabaseReference reference;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

activity = getActivity();

sharedPreference = new SharedPreference();

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_station\_list, container, false);

findViewsById(view);

//Initializing those above references

mDatabase = FirebaseDatabase.getInstance();

reference = mDatabase.getReference(getString(R.string.stations));

reference.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

//A lot of repeated code to read in all our strings from our Firebase Database

//This should be done in a for loop...

if (dataSnapshot.exists()){

boom\_Name = dataSnapshot.child("Boom973").child("Name").getValue().toString();

boom\_Freq = dataSnapshot.child("Boom973").child("Frequency").getValue().toString();

boom\_Description = dataSnapshot.child("Boom973").child("Description").getValue().toString();

cbc\_Name = dataSnapshot.child("CBCMusic941").child("Name").getValue().toString();

cbc\_Freq = dataSnapshot.child("CBCMusic941").child("Frequency").getValue().toString();

cbc\_Description = dataSnapshot.child("CBCMusic941").child("Description").getValue().toString();

humbercollege\_Name = dataSnapshot.child("HumberCollege969").child("Name").getValue().toString();

humbercollege\_Freq = dataSnapshot.child("HumberCollege969").child("Frequency").getValue().toString();

humbercollege\_Description = dataSnapshot.child("HumberCollege969").child("Description").getValue().toString();

kiss\_Name = dataSnapshot.child("Kiss925").child("Name").getValue().toString();

kiss\_Freq = dataSnapshot.child("Kiss925").child("Frequency").getValue().toString();

kiss\_Description = dataSnapshot.child("Kiss925").child("Description").getValue().toString();

theMove\_Name = dataSnapshot.child("TheMove935").child("Name").getValue().toString();

theMove\_Freq = dataSnapshot.child("TheMove935").child("Frequency").getValue().toString();

theMove\_Description = dataSnapshot.child("TheMove935").child("Description").getValue().toString();

z1035\_Name = dataSnapshot.child("z1035").child("Name").getValue().toString();

z1035\_Freq = dataSnapshot.child("z1035").child("Frequency").getValue().toString();

z1035\_Description = dataSnapshot.child("z1035").child("Description").getValue().toString();

//This is usually outside the ValueEventListener

setStations();

stationListAdapter = new StationListAdapter(activity, stations);

stationListView.setAdapter(stationListAdapter);

stationListView.setOnItemClickListener(StationListFragment.this);

stationListView.setOnItemLongClickListener(StationListFragment.this);

}

}

@Override

public void onCancelled(DatabaseError databaseError) {}

});

return view;

}

private void setStations() {

//Read in all our strings into an array list of type station

Station boom973 = new Station(1, boom\_Name, boom\_Freq, boom\_Description);

Station cbc941 = new Station(2, cbc\_Name, cbc\_Freq, cbc\_Description);

Station humbercollege969 = new Station(3, humbercollege\_Name, humbercollege\_Freq, humbercollege\_Description);

Station kiss925 = new Station(4, kiss\_Name, kiss\_Freq, kiss\_Description);

Station theMove935 = new Station(5, theMove\_Name, theMove\_Freq, theMove\_Description);

Station z1035 = new Station(6, z1035\_Name, z1035\_Freq, z1035\_Description);

stations = new ArrayList<>();

stations.add(boom973);

stations.add(cbc941);

stations.add(humbercollege969);

stations.add(kiss925);

stations.add(theMove935);

stations.add(z1035);

}

private void findViewsById(View view) { stationListView = (ListView) view.findViewById(R.id.list\_station);}

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

Station station = (Station) parent.getItemAtPosition(position);

String description = station.getDescription();

String freq = station.getFreq();

// String imageURL = station.getImageURL();

i = new Intent(getActivity(), RadioActivity.class);

i.putExtra(getActivity().getString(R.string.description), description);

i.putExtra(getActivity().getString(R.string.freq), freq);

//i.putExtra(getActivity().getString(R.string.imageurl), imageURL);

//reference.child("Frequency").child("Name").setValue(cbc\_Freq);

startActivity(i);

/\*Use station.postion to decide which stream is being selected then run Radio activity\*/

}

@Override

public boolean onItemLongClick(AdapterView<?> arg0, View view, int position, long arg3) {

//Handles all the favouriting tasks . This function is repated in FavouriteListFragment

ImageView button = (ImageView) view.findViewById(R.id.imgbtn\_favorite);

String tag = button.getTag().toString();

if (tag.equalsIgnoreCase(getString(R.string.grey))) {

sharedPreference.addFavorite(activity, stations.get(position)); // Figure the wrong 2nd argument...

// Found out I had a package problem for both of the sharedPreference.

Toast.makeText(activity, activity.getResources().getString(R.string.add\_favr), Toast.LENGTH\_SHORT).show();

button.setTag(getString(R.string.red));

button.setImageResource(R.drawable.heart\_red);

} else {

sharedPreference.removeFavorite(activity, stations.get(position)); // Figure the wrong 2nd argument...

// Found out I had a package problem for both of the sharedPreference.

button.setTag(getString(R.string.grey));

button.setImageResource(R.drawable.heart\_grey);

Toast.makeText(activity, activity.getResources().getString(R.string.remove\_favr), Toast.LENGTH\_SHORT).show();

}

return true;

}

@Override

public void onResume() {

super.onResume();

}

}

### 7.2.5 SharedPreference

package ca.humber.radiopi.radiopi.utils;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import android.content.Context;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import ca.humber.radiopi.radiopi.beans.Station;

import com.google.gson.Gson;

public class SharedPreference {

public static final String PREFS\_NAME = "STATION\_APP";

public static final String FAVORITES = "Station\_Favorite";

public SharedPreference() {

super();

}

// This four methods are used for maintaining favorites.

public void saveFavorites(Context context, List<Station> favorites) {

SharedPreferences settings;

Editor editor;

settings = context.getSharedPreferences(PREFS\_NAME,

Context.MODE\_PRIVATE);

editor = settings.edit();

Gson gson = new Gson();

String jsonFavorites = gson.toJson(favorites);

editor.putString(FAVORITES, jsonFavorites);

editor.commit();

}

public void addFavorite(Context context, Station station) {

List<Station> favorites = getFavorites(context);

if (favorites == null)

favorites = new ArrayList<Station>();

favorites.add(station);

saveFavorites(context, favorites);

}

public void removeFavorite(Context context, Station station) {

ArrayList<Station> favorites = getFavorites(context);

if (favorites != null) {

favorites.remove(station);

saveFavorites(context, favorites);

}

}

//get favourites in JSON format and convert to gson and and to an arraylist

public ArrayList<Station> getFavorites(Context context) {

SharedPreferences settings;

List<Station> favorites;

settings = context.getSharedPreferences(PREFS\_NAME,

Context.MODE\_PRIVATE);

if (settings.contains(FAVORITES)) {

String jsonFavorites = settings.getString(FAVORITES, null);

Gson gson = new Gson();

Station[] favoriteItems = gson.fromJson(jsonFavorites,

Station[].class);

favorites = Arrays.asList(favoriteItems);

favorites = new ArrayList<Station>(favorites);

} else

return null;

return (ArrayList<Station>) favorites;

}

}

### 7.2.6 MainActivity

package ca.humber.radiopi.radiopi;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.res.Configuration;

import android.graphics.Color;

import android.os.Bundle;

import android.support.v4.content.ContextCompat;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.Toast;

import com.aurelhubert.ahbottomnavigation.AHBottomNavigation;

import com.aurelhubert.ahbottomnavigation.AHBottomNavigationItem;

import ca.humber.radiopi.radiopi.mFragments.FavouriteListFragment;

import ca.humber.radiopi.radiopi.mFragments.StationListFragment;

import java.util.Locale;

public class MainActivity extends AppCompatActivity implements AHBottomNavigation.OnTabSelectedListener{

AHBottomNavigation bottomNavigation;

//CREATED THE BOTTOM NAVIGATION TOOLBAR USING THE ACTIVITY\_MAIN LAYOUT FILE

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

bottomNavigation= (AHBottomNavigation) findViewById(R.id.myBottomNavigation\_ID);

bottomNavigation.setOnTabSelectedListener(this);

this.createNavItems();

}

private void createNavItems()

{

//CREATE ITEMS TO DISPLAY ON BOTTOM NAVIGATION BAR

AHBottomNavigationItem radioItem=new AHBottomNavigationItem(R.string.Radio,R.drawable.radio,R.color.colorBottomNavigationAccent);

AHBottomNavigationItem favoritesItem=new AHBottomNavigationItem(R.string.Favorites,R.drawable.favorites,R.color.colorBottomNavigationAccent);

AHBottomNavigationItem musicplayerItem=new AHBottomNavigationItem(R.string.MusicPlayer,R.drawable.mp3player,R.color.colorBottomNavigationAccent);

//ADD ITEMS TO BOTTOM NAVIGATION BAR

bottomNavigation.addItem(radioItem);

bottomNavigation.addItem(favoritesItem);

bottomNavigation.addItem(musicplayerItem);

//PROPERTIES GIVEN TO THE BOTTOM NAVIGATION BAR ICONS

bottomNavigation.setAccentColor(ContextCompat.getColor(MainActivity.this, R.color.LightGolden));

bottomNavigation.setDefaultBackgroundColor(Color.parseColor(getString(R.string.backgroundColorString)));

bottomNavigation.setCurrentItem(0);

}

//CREATED onTabSelected WHICH ALLOWS MOVEMENT BETWEEN FRAGMENTS. IF ON SELECTED FRAGMENT, THE OTHER FRAGMENTS

//WILL TURN GREY. I WAS ALSO ABLE TO MOVE THE MP3 ACTIVITY AS A FRAGMENT IN OUR BOTTOM NAVIGATION BAR.

@Override

public boolean onTabSelected(int position, boolean wasSelected) {

if(position==0)

{

StationListFragment stationFragment = new StationListFragment();

getSupportFragmentManager().beginTransaction().replace(R.id.content\_id,stationFragment).commit();

}else if(position==1)

{

//FavoritesFragment favoritesFragment=new FavoritesFragment();

FavouriteListFragment favouritesFragment=new FavouriteListFragment();

getSupportFragmentManager().beginTransaction().replace(R.id.content\_id,favouritesFragment).commit();

}else if(position==2) {

//Switched from MusicPlayer (removed in this version) to the RadioActivity.

Intent intent = new Intent(MainActivity.this, RadioActivity.class);

startActivity(intent);

}

return true;

}

//CREATED MENU FOR OUR TOOLBAR IN ORDER TO HAVE THE ABOUT PAGE

@Override

public boolean onCreateOptionsMenu(Menu menu) {

//Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_option, menu);

return true;

}

//CREATED ON BACK PRESSED WHICH GIVES THE USER A DIALOG BOX CONFIRMING IF THEY WANT TO EXIT THE APP OR NOT

@Override

public void onBackPressed() {

new AlertDialog.Builder(this)

.setIcon(android.R.drawable.ic\_dialog\_alert)

.setTitle(getString(R.string.closing\_title))

.setMessage(getString(R.string.closing\_message))

.setPositiveButton(getString(R.string.yes), new DialogInterface.OnClickListener()

{

@Override

public void onClick(DialogInterface dialog, int which) {

finish();

}

})

.setNegativeButton(getString(R.string.no), null)

.show();

}

//CREATED LOGIC BEHIND ABOUT PAGE. WHEN SELECTED, IT WILL GO TO THE APPROPRIATE CLASS AND DISPLAY THE

//APPROPRIATE FILE

public boolean onOptionsItemSelected(MenuItem item) {

//Handle action bar item clicks here. The action bar will

//automatically handle clicks on the Home/Up button, so long

//as you specify a parent activity in AndroidManifest.xml.

Intent bar\_intent = null;

switch (item.getItemId()) {

/\* case R.id.About:

Intent intent = new Intent(MainActivity.this, About.class);

startActivity(intent);

break;

We will not have an about.class for the sake of demonstrating.

\*/

case R.id.en:

Locale locale = new Locale(getString(R.string.en));

Locale.setDefault(locale);

Configuration config = new Configuration();

config.locale = locale;

getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());

Toast.makeText(this, getString(R.string.translate\_english), Toast.LENGTH\_LONG).show();

recreate();

break;

case R.id.fr:

Locale locale2 = new Locale(getString(R.string.fr));

Locale.setDefault(locale2);

Configuration config2 = new Configuration();

config2.locale = locale2;

getBaseContext().getResources().updateConfiguration(config2, getBaseContext().getResources().getDisplayMetrics());

Toast.makeText(this, getString(R.string.translate\_french), Toast.LENGTH\_LONG).show();

recreate();

break;

}

return super.onOptionsItemSelected(item);

}

}

### 7.2.7 RadioActivity

package ca.humber.radiopi.radiopi;

import android.app.Activity;

import android.content.Intent;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.media.AudioManager;

import android.media.MediaPlayer;

import android.os.AsyncTask;

import android.os.Bundle;

import android.provider.MediaStore;

import android.view.View;

import android.view.WindowManager;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

import ca.humber.radiopi.radiopi.MainActivity;

import ca.humber.radiopi.radiopi.R;

import ca.humber.radiopi.radiopi.RadioActivity;

import ca.humber.radiopi.radiopi.adapter.StationListAdapter;

import ca.humber.radiopi.radiopi.beans.Station;

import ca.humber.radiopi.radiopi.utils.SharedPreference;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

public class RadioActivity extends Activity {

Button b\_play;

MediaPlayer mediaPlayer;

boolean prepared = false; //These are just for checking if were ready to play the stream

boolean started = false; //and if the stream has started

String name, freq, description, imageURL;

URL url1;

private FirebaseDatabase mDatabase;

private DatabaseReference reference;

@Override

public void onCreate(Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.activity1);

mDatabase = FirebaseDatabase.getInstance();

reference = mDatabase.getReference(getString(R.string.stations));

//Leave Radio Page open until user locks the phone or exits the page

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

Toast.makeText(this, getResources().getString(R.string.radioToast), Toast.LENGTH\_LONG).show();

//Passing string values from onItemClick() in StationListFragment

Bundle bundle = getIntent().getExtras();

freq = bundle.getString(getString(R.string.freq));

imageURL = bundle.getString(getString(R.string.imageurl));

description = bundle.getString(getString(R.string.description));

TextView descripView = (TextView) findViewById(R.id.descripView);

descripView.setText(description);

b\_play = (Button) findViewById(R.id.b\_play);

//b\_play.setEnabled(false);

//b\_play.setText(R.string.LOADING);

//Setting up madia player here

mediaPlayer = new MediaPlayer();

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

//new PlayerTask().execute(streamLink);

//checking for a real URL that gets passed in from bundle

try {

url1 = new URL(imageURL);

} catch (MalformedURLException e) {

e.printStackTrace();

}

//Setting pause/play button, after the loading is complete

b\_play.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view){

if(started){

started = false;

mediaPlayer.pause();

//b\_play.setText(R.string.PLAY);

activityChange();

} else {

started = true;

mediaPlayer.start();

b\_play.setText(R.string.MAIN\_MENU);

reference.child("Frequency").child("Name").setValue(freq);

}

}

});

}

//This task is for setting our radio stream with the first string passed in doInBackground

/\* class PlayerTask extends AsyncTask<String, Void, Boolean> {

@Override

protected Boolean doInBackground(String...strings){

try {

mediaPlayer.setDataSource(strings[0]);

mediaPlayer.prepare();

prepared = true;

} catch (IOException e) {

e.printStackTrace();

}

return prepared;

}

@Override

protected void onPostExecute(Boolean aBoolean){

super.onPostExecute(aBoolean);

b\_play.setEnabled(true);

b\_play.setText(R.string.PLAY);

}

} \*/

public void activityChange(){

Intent intent = new Intent(RadioActivity.this, MainActivity.class);

startActivity(intent);

}

@Override

protected void onPause() {

super.onPause();

if(started){

mediaPlayer.pause();

}

}

@Override

protected void onResume() {

super.onResume();

if(started){

mediaPlayer.start();

}

}

@Override

protected void onDestroy(){

super.onDestroy();

if(prepared){

mediaPlayer.release();

}

}

}

### 7.2.8 activity1.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:background="@color/LightYellow">

<Button

android:id="@+id/b\_play"

android:layout\_width="160dp"

android:layout\_height="wrap\_content"

android:text="@string/PLAY"

tools:layout\_editor\_absoluteX="147dp"

tools:layout\_editor\_absoluteY="230dp"

android:layout\_centerVertical="true"

android:layout\_centerHorizontal="true"/>

<ImageView

android:id="@+id/im1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="109dp"

app:srcCompat="@color/LightYellow"/>

<TextView

android:id="@+id/descripView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="95dp"/>

</RelativeLayout>

### 7.2.9 activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:fitsSystemWindows="true"

tools:context="ca.humber.radiopi.radiopi.MainActivity">

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:theme="@style/AppTheme.AppBarOverlay">

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

android:background="?attr/colorPrimary"

app:popupTheme="@style/AppTheme.PopupOverlay"/>

</android.support.design.widget.AppBarLayout>

<include layout="@layout/content\_main"/>

<com.aurelhubert.ahbottomnavigation.AHBottomNavigation

android:id="@+id/myBottomNavigation\_ID"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom"/>

</android.support.design.widget.CoordinatorLayout>

### 7.2.10 content\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="16dp"

android:paddingTop="1dp"

android:paddingLeft="1dp"

android:paddingRight="1dp"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:context="ca.humber.radiopi.radiopi.MainActivity"

android:id="@+id/content\_id"

tools:showIn="@layout/activity\_main">

</RelativeLayout>

### 7.2.11 fragment\_station\_list.xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/LightYellow">

<ListView

android:id="@+id/list\_station"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:divider="#030303"

android:dividerHeight="1dp"

android:drawSelectorOnTop="true"

android:footerDividersEnabled="false"

android:scrollbarStyle="outsideOverlay"

android:layout\_marginBottom="?attr/actionBarSize">

</ListView>

</RelativeLayout>

### 7.2.12 listfragment.xml

<?xml version="1.0" encoding="utf-8"?>

<ListView xmlns:android="http://schemas.android.com/apk/res/android"

android:background="@color/LightYellow"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/list"/>

### 7.2.13 radio\_fragment.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical" android:layout\_width="match\_parent"

android:background="@color/LightYellow"

android:layout\_height="match\_parent">

</LinearLayout>

### 7.2.14 station\_list\_item.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:background="@color/LightYellow"

android:descendantFocusability="blocksDescendants">

<RelativeLayout

android:id="@+id/pdt\_layout\_item"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_alignParentTop="true">

<TextView

android:id="@+id/txt\_sta\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_alignParentTop="true"

android:textColor="@color/Black"

android:textSize="20sp"

android:textStyle="bold"

android:padding="12dp"/>

<ImageView

android:id="@+id/imgbtn\_favorite"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentEnd="true"

android:background="@null"

android:contentDescription="@string/favorites"/>

</RelativeLayout>

<View

android:layout\_width="match\_parent"

android:layout\_height="1dp"

android:layout\_below="@+id/pdt\_layout\_item"

android:background="@color/LightYellow"/>

</RelativeLayout>

### 7.2.15 menu\_option.xml

<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:title="@string/french\_translate\_tab"

android:id="@+id/fr"

app:showAsAction="never"/>

<item

android:title="@string/english\_translate\_tab"

android:id="@+id/en"

app:showAsAction="never"/>

<item

android:title="@string/about\_us"

android:id="@+id/About"

app:showAsAction="never"/>

</menu>

### 7.2.16 colors.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="colorPrimary">#800000</color>

<color name="colorPrimaryDark">#800000</color>

<color name="colorAccent">#FF4081</color>

<color name="LightYellow">#f0ead6</color>

<color name="Black">#000000</color>

<color name="White">#FFFFFF</color>

<color name="LightGolden">#FAFAD2</color>

</resources>

### 7.2.17 dimens.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

<!-- Default screen margins, per the Android Design guidelines. -->

<dimen name="activity\_horizontal\_margin">16dp</dimen>

<dimen name="activity\_vertical\_margin">16dp</dimen>

<dimen name="fab\_margin">16dp</dimen>

</resources>

### 7.2.18 strings.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">RadioPi</string>

<string name="GroupName">RadioPi</string>

<string name="LOADING">LOADING</string>

<string name="PLAY">PLAY</string>

<string name="MAIN\_MENU">MAIN MENU</string>

<string name="Radio">Radio</string>

<string name="Favorites">Favorites</string>

<string name="favorites">favorites</string>

<string name="stations">Stations</string>

<string name="MusicPlayer">MP3</string>

<string name="about\_us">About Us</string>

<string name="closing\_message">Are you sure you want to close this application?</string>

<string name="closing\_title">Closing Application</string>

<string name="yes">Yes</string>

<string name="no">No</string>

<string name="link">streamLink</string>

<string name="grey">grey</string>

<string name="red">red</string>

<string name="no\_permission">Please Allow Permission To Access MP3</string>

<string name="start\_duration">00:00:00</string>

<string name="created\_by">RadioPi Created By Erick, Aldo, Ryan</string>

<string name="version\_num">Version 2.0</string>

<string name="website">https://github.com/AldoNdreu/SensorEffectorProject/tree/master/documentation/CENG355</string>

<string name="stationlist">station\_list</string>

<string name="favouriteslist">favourites\_list</string>

<string name="stationapp">STATION\_APP</string>

<string name="stationfavourite">Station\_Favourite</string>

<string name="ok">OK</string>

<string name="backgroundColorString">#800000</string>

<string name="imageurl">imageURL</string>

<string name="description">description</string>

<string name="translate\_english">Translated to English!</string>

<string name="translate\_french">Translated to French!</string>

<string name="english\_translate\_tab">English Translation</string>

<string name="french\_translate\_tab">French Translation</string>

<string name="fr">fr</string>

<string name="en">en</string>

<string name="radioToast">Please Allow Up To 30 Seconds For Radio Stream To Load</string>

<string name="freq"></string>

</resources>

### 7.2.19 strings\_fav\_msg.xml

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="add\_favr">Added to Favorites</string>

<string name="remove\_favr">Removed from Favorites</string>

<string name="no\_favorites\_items">No Favorites</string>

<string name="no\_favorites\_msg">Long press on item to add to favorites</string>

<string name="favorites\_remove\_msg">Long press on item to remove from favorites</string>

</resources>

### 7.2.20 styles.xml

<resources>

<!-- Base application theme. -->

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary">@color/colorPrimary</item>

<item name="colorPrimaryDark">@color/colorPrimaryDark</item>

<item name="colorAccent">@color/colorAccent</item>

</style>

<style name="AppTheme.NoActionBar">

<item name="windowActionBar">false</item>

<item name="windowNoTitle">true</item>

</style>

<style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

<style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />

</resources>

### 7.2.21 styles.xml (fr)

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">RadioPi</string>

<string name="GroupName">RadioPi</string>

<string name="LOADING">CHARGEMENT....</string>

<string name="PLAY">JOUER</string>

<string name="PAUSE">ARRÊTEZ</string>

<string name="Radio">Radio</string>

<string name="Favorites">Favoris</string>

<string name="favorites">favoris</string>

<string name="stations">Stations</string>

<string name="MusicPlayer">MP3</string>

<string name="about\_us">À propos de nous</string>

<string name="closing\_message">Êtes-vous sûr de vouloir fermer cette application?</string>

<string name="closing\_title">Demande de clôture</string>

<string name="yes">Oui</string>

<string name="no">Non</string>

<string name="grey">gris</string>

<string name="red">rouge</string>

<string name="no\_permission">Veuillez autoriser la permission d`accéder à MP3</string>

<string name="start\_duration">00:00:00</string>

<string name="created\_by">XM Lecteur de musique Créé par RadioPi</string>

<string name="version\_num">Version 1.0</string>

<string name="website">https://github.com/AldoNdreu/SensorEffectorProject</string>

<string name="add\_favr">Ajouté aux Favoris</string>

<string name="remove\_favr">Retiré des Favoris</string>

<string name="no\_favorites\_items">Pas de Favoris</string>

<string name="no\_favorites\_msg">Appuyez longuement sur l`élément à ajouter aux favoris</string>

<string name="favorites\_remove\_msg">Appuyez longuement sur l`élément à supprimer des favoris</string>

<string name="translate\_english">Traduit en anglais!</string>

<string name="translate\_french">Traduit en français!</string>

<string name="english\_translate\_tab">Traduction anglaise</string>

<string name="french\_translate\_tab">Traduction française</string>

<string name="radioToast">S`il vous plaît permettre jusqu`à 30 secondes pour le flux radio à charger</string>

</resources>

### 7.2.22 build.gradle

apply plugin: 'com.android.application'

android {

compileSdkVersion 27

buildToolsVersion '27.0.3'

defaultConfig {

applicationId "ca.humber.radiopi.radiopi"

minSdkVersion 21

targetSdkVersion 26

versionCode 1

versionName "1.0"

testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

buildscript {

repositories {

jcenter()

google()

}

dependencies {

classpath 'com.android.tools.build:gradle:3.1.1'

classpath 'com.google.gms:google-services:3.2.0'

}

}

allprojects {

repositories {

jcenter()

google()

}

}

dependencies {

implementation 'com.google.firebase:firebase-database:11.8.0'

compile fileTree(include: ['\*.jar'], dir: 'libs')

androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {

exclude group: 'com.android.support', module: 'support-annotations'

})

compile 'com.android.support.constraint:constraint-layout:1.0.2'

testCompile 'junit:junit:4.12'

compile 'com.android.support:appcompat-v7:26.1.0'

compile 'com.android.support:support-v4:26.1.0'

compile 'com.android.support.constraint:constraint-layout:1.0.2'

compile 'com.android.support:recyclerview-v7:26.1.0'

compile 'com.android.support:design:26.1.0'

compile 'com.aurelhubert:ahbottomnavigation:1.3.3'

compile 'com.github.medyo:android-about-page:1.1.1'

implementation 'com.google.code.gson:gson:2.8.2'

}

apply plugin: 'com.google.gms.google-services'

### 7.2.23 AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="ca.humber.radiopi.radiopi">

<!-- normal permissions -->

<uses-permission android:name="android.permission.INTERNET" />

<permission android:name="android.permission.MEDIA\_CONTENT\_CONTROL" />

<!-- danger permissions -->

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity

android:name=".MainActivity"

android:label="@string/app\_name"

android:configChanges="locale|orientation"

android:theme="@style/AppTheme.NoActionBar">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".RadioActivity" />

</application>

</manifest>