***FourOneSix Sound***

Software and Hardware Specification Sheet

By: Romario Tulloch and Andrew Le

Study: Computer Engineering Technology

April 26th,2019

# Declaration of Joint Authorship

We, Romario Tulloch and Andrew Le, confirm that the work we are submitting for assessment is the work created by ourselves, and is from our own words. Any work taken from any other author (programs, figures, techniques, illustrations, or any other material) are properly cited at the point of it’s use. A bibliography is provided at the end of this report, which would contain all sources used in this report.

# Proposal

1. The name of our project will be the FourOneSix Sound.
2. The summary of the project that we will be putting together is to implement the use of our firebase database to be operating on our hardware. Our hardware is an audio amplifier that will be operating from the raspberry pi 3. The courses that are related to out project are Software project (CENG 319), hardware production tech (CENG 317) and Embedded Systems (CENG 252).
3. Our plan to implement the project is to ensure that the mobile application is working as it should then to ensure that the hardware (audio amplifier) is also operating as it should. The next step will be to figure out how to use the raspberry pi to read from the firebase database automatically. The project will me managed in a systematically way as the critical path will be used as a guide to implement all the steps in putting the project together.

1. In regards to the timeframe of completing the project, it seems very likely as out application needs to undergo a few alterations and the hardware aspect is basically finished. If there are any problems that arises, we will be dedicated more time and effort to fix the problem and implement measures to reduce problems in the future.
2. There are few similar products in the market right now such as Spotify and Sound cloud but the difference with our app is that is tailored for the specified hardware.
3. For test cases so far, we can test that App streams the media from the database and use a hard connection from a mobile device to the amplifier to test that the amplifier is still working as it should.
4. In conclusion this project will be a 14-week long project that we will try to incorporate our previous software with the hardware to make a fully functional system that satisfies the need of real-world problems.

# Abstract

The purpose of this project was to create a device(hardware) that would work with a raspberry pi and an app on our phone(software) as an assignment for the computer engineering technology program at Humber College. The hardware device we are creating is a speaker using an audio amplifier that was chosen as our hardware device. The phone app allows for users to send music files to an offset database. This database will store all the account information and also the music files that each account uploads into the database. When command by the phone app, the music files in the database will be downloaded and played through the audio amplifiers/speakers. We feel that this is a great product as many people listen to music regularly and speakers are in very high demand and our product is an easy and user-friendly product. The rest of the report will go through a more thorough analysis of each separate part of the project and will provide more general details about the product.

# Table of Contents

Table of Contents

[Declaration of Joint Authorship 1](#_Toc7107166)

[Proposal 2](#_Toc7107167)

[Abstract 4](#_Toc7107168)

[Table of Contents 5](#_Toc7107169)

[Illustrations 7](#_Toc7107170)

[Introduction 8](#_Toc7107171)

[Project Description 9](#_Toc7107172)

[Requirement Specifications 9](#_Toc7107173)

[Hardware 9](#_Toc7107174)

[Software 9](#_Toc7107175)

[Database 10](#_Toc7107176)

[Build Instructions 11](#_Toc7107177)

[Budget for Materials Required 11](#_Toc7107178)

[Time Schedule 12](#_Toc7107179)

[Assembly of Pi 13](#_Toc7107180)

[PCB Design Files 15](#_Toc7107181)

[PCB Soldering 16](#_Toc7107182)

[Wiring 17](#_Toc7107183)

[Power Up 19](#_Toc7107184)

[Case Design 21](#_Toc7107185)

[Assembly for Hardware 21](#_Toc7107186)

[Installing Python 23](#_Toc7107187)

[Code for Sensor 25](#_Toc7107188)

[Database Design 27](#_Toc7107189)

[Mobile Application 27](#_Toc7107190)

[Status Report 33](#_Toc7107191)

[Status Update 1 33](#_Toc7107192)

[Status Update 2 34](#_Toc7107193)

[Status Update 3 34](#_Toc7107194)

[Status Update 4 35](#_Toc7107195)

[Problems Encountered 37](#_Toc7107196)

[Solutions 38](#_Toc7107197)

[Future Plans 39](#_Toc7107198)

[Conclusion 40](#_Toc7107199)

[Bibliography 41](#_Toc7107200)

# Illustrations

Figure 1 – Sensor

Figure 2 – Schedule

Figure 3 – PCB Gerber Files

Figure 4 –PCB Top View

Figure 5 –PCB Bottom View

Figure 6 –Sensor Wired

Figure 7 – Interfacing Menu

Figure 8 –I2C Option

Figure 9 –Top of Enclosure

Figure 10 –Inside of Enclosure

Figure 11 –Full Enclosure

Figure 12 –Verification

Figure 13 –Blinkatest Output

# Introduction

The task of testing auto sound systems can be a very tedious task. Plenty of steps that are needed to be take can be monitored and logged to ensure the proper testing of a system especially in the essence to optimize the amplifier levels. This process can even be very expensive in terms of using special hardware to test decibel levels.

We have developed a system that will play specific audio files via the amplifier. The specialized software that we have created will log volume levels manually to a database to keep track of how high the amplifier can go before distortion or clipping occurs.

There are other options that are available to the public that can test amplifier output but does not include the option of logging to a database. We feel that this can be a valuable resource in the audio industry.

# Project Description

## Requirement Specifications

### Hardware

The hardware portion of this project will be a joint effort between each member of the group as there are many responsibilities in order for everything to function as intended. The enclosure and hardware design will be built by Romario, Romario will also be handling testing the sensor. Both members, Andrew and Romario will be working on the connection between the Raspberry Pi and the sensor. The Integration of components may require additional help from every member due to problems that may occur during development.

To create this project, we used a Stereo 2.8W Class D Audio Amplifier - I2C Control AGC - TPA2016, as our sensor of choice. This sensor is able to run from 2.7 volts – 5.5 volts from a DC power source. As this sensor is of class “D”, this sensor has very high efficiency (89% efficient when using an 8-ohm speaker at 1.5 Watts). This sensor also comes with a built in thermal and over current protection. It has an Output Power: 2.8W at 4Ω, 10% THD, 1.7W at 8Ω, 10% THD, with 5V Supply.

### Software

The project utilizes a smartphone capable of running Android API 21 or higher. An up to date version of Android Studio was used to build the mobile application. A Raspberry Pi 3 was implemented with connection between the hardware and application. Updating the Raspbian OS to its newest version was used throughout the project. The mobile application will be used to work alongside the hardware components.

The software being used is an app created on android studio by us. The creation of the app was fairly simple and is very user friendly as it is easy to use and easy to navigate through the application. The main purpose of this android application is to be able to upload and download the music that you want through your phone into the firebase database. This app will allow for those songs to be played once uploaded and the app will also communicate to the hardware through the songs as they are inserted into the database. The app will consist of five major screens and also allows for users to log in and out.

### Database

The database was designed by Andrew. The database connection is established and connected to the created Android application. This connection allows for songs to be stored in the firebase storage and have many different readings stored in firebase database, these readings consist of the “Play” option, the selected URL option, all the song names and their download URLs beside them, and the “Volume” option. These values are able to be read and retrieved by the Raspberry Pi and our sensor. Account details are also saved onto the database as users can log in and out of their accounts when they want to.

Not only are we using firebase database but we are also using firebase storage. Using firebase storage allows us to save files that the user would like to save and also provides download URLs and other information on the file. This is very important as those download URLs are the way for the songs to be retrieved through both the phone (if the user wants to play the music through the phone) and through the hardware (if the user wants to play the music through the hardware).

# Stereo 2.8W Class D Audio Amplifier - I2C Control AGC - TPA2016Build Instructions

To create this project, we used a Stereo 2.8W Class D Audio Amplifier - I2C Control AGC - TPA2016, as our sensor of choice. This sensor is able to run from 2.7 volts – 5.5 volts from a DC power source. As this sensor is of class “D”, this sensor has very high efficiency (89% efficient when using an 8-ohm speaker at 1.5 Watts). Figure 1 - Sensor This sensor also comes with a built in thermal and over current protection. It has an Output Power: 2.8W at 4Ω, 10% THD, 1.7W at 8Ω, 10% THD, with 5V Supply.

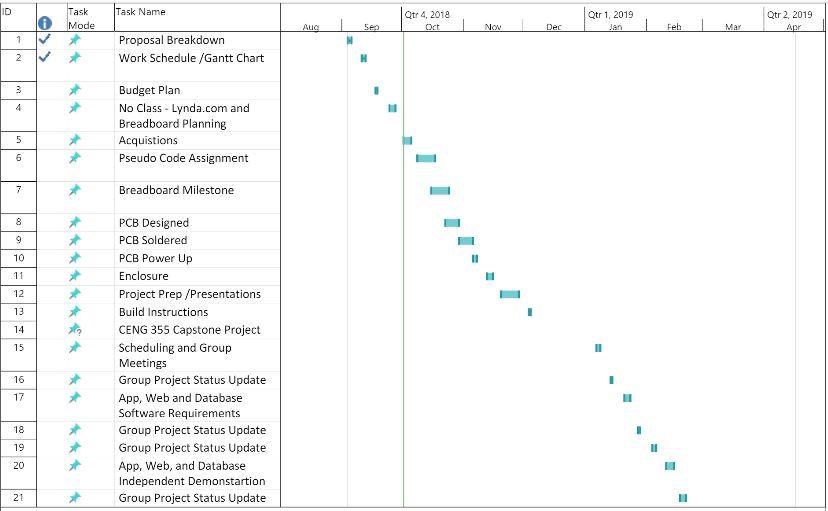
### Budget for Materials Required

* **Raspberry Pi 3B+ & Casing - $100** (https://www.amazon.ca/CanaKit-Raspberry-Starter-Premium-Black/dp/B07BCC8PK7/ref=sr\_1\_2?crid=3I43CI9QJZ2KK&keywords=rpi3+b%2B&qid=1556049628&s=gateway&sprefix=rpi3%2Caps%2C144&sr=8-2)
* **TPA2016D2 - $22** (https://www.adafruit.com/product/1712)
* **Speakers - $20** (https://www.amazon.ca/Gikfun-Speaker-Stereo-Loudspeaker-Arduino/dp/B01N74TGFM/ref=sr\_1\_3?keywords=3w+speakers&qid=1556049704&s=gateway&sr=8-3)
* **3.5mm Headphone Jack Adapter - $11** (https://www.amazon.ca/Cerrxian-Terminal-Headphone-Converter-Adapter/dp/B06WRRGYMM/ref=sr\_1\_5?keywords=3.5mm+to+speaker+terminal&qid=1556049771&s=gateway&sr=8-5)
* **Speaker Wires - $5** (Already Attained)
* **USB Extensions - $20** (Already Attained)
* **Wood for Enclosure - $15** (Home Depot)
* **TOTAL: $194**

***\*NB – All prices are quoted as after shipping\****

### Time Schedule

Realistically, this project should not take too long if all materials and facilities are available to you. The materials themselves might take a week to arrive due to shipping, but the actual process of assembling and programming should not take longer than a week if proper time is given. A couple of hours each day can be dedicated towards the different aspects of the project to make time usage more efficient and effective. For us, this project took around 2 whole semesters (8 months) to finish along with an average work time of around 2.5 hours a week. Here is the time schedule we followed:



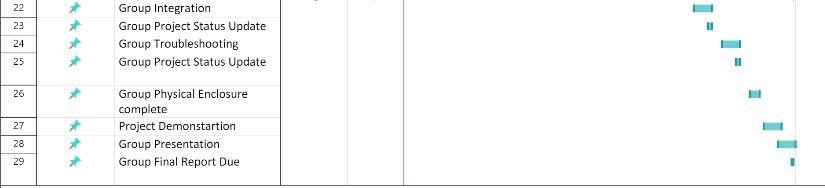


Figure 2- Schedule

### Assembly of Pi

These steps will cover how to set up the Raspberry Pi 3 B+ properly so that you have the ability to log in and test your sensors capabilities.

1. Format an SD card with a minimum of 8GB to be used for the OS of the Pi. You can use the following link to download a SD card formatting software: https://www.sdcard.org/downloads/formatter\_4/index.html
2. Download and unzip the latest version of the OS for the Raspberry Pi to your SD card. Download NOOBS in the link as that will ensure that you will have almost everything required when starting: https://www.raspberrypi.org/downloads/noobs/
3. Once the image is on the SD card, remove it from the pc and insert it in the Pi. Now plug in a separate monitor, mouse, keyboard, HDMI, Ethernet cable, and power supply to the Pi in their corresponding ports. The Pi turns on automatically when the power is plugged in.
4. Upon the boot up session, select Raspbian as the operating system for the Pi and follow the instructions as they appear. You may also change the keyboard layout on the bottom during initial boot. The US layout is highly recommended.
5. Once installation is completed, you should be brought to the desktop. Connect yourself to either Wifi or wired connection in order to perform the next few steps.
6. Open the terminal in the top left corner of the screen and input the following lines (this takes quite a long time): Shell wget https://raw.githubusercontent.com/six0four/StudentSenseHat/master/firmware/hshcribv01.sh \ -O /home/pi/hshcribv01.sh chmod u+x /home/pi/hshcribv01.sh /home/pi/hshcribv01.sh
7. Now it is time to set up a VNC connection so that you can access your Pi on any computer screen. From the Start Menu, go -> Preferences->Raspberry Pi Configuration->Interfaces, then set VNC to Enabled. Now on the desktop in the top right corner, you should see a VNC logo. When you click it, you should see an IP address for your Pi which will be used to connect it via the VNC software. Download the software on any computer you wish to communicate with the Pi: https://www.realvnc.com/en/connect/download/vnc/
8. Once the software is installed, connect the Ethernet cable from the Pi to your computer of choice to have a direct connection. Now you can simply input the same address you found in the Pi in the VNC software and it should connect.
9. To turn off the Pi, type sudo power down in the terminal.

If you are still unsure or struggling with a part in particular, this video provides a step by step explanation for everything required: <https://www.youtube.com/watch?v=xBlxuf_LSCM>

### PCB Design Files

1. The PCB was designed using a software called Fritzing, It can be downloaded from this link. <http://fritzing.org/home/>
2. Install the program from running the .exe file.
3. Download the fritzing repository to import the I2C sensor templates into the Fritzing program from this repository. [**https://github.com/adafruit/Fritzing-Library**](https://github.com/adafruit/Fritzing-Library)
4. Click on the PCB design tab.
5. Import the I2C sensor by typing TPA2016A2 in the search bar.
6. Import the Raspberry Pi 3B+.
7. Draw the PCB board over the GPIO pins of the RPI.
8. Set up the final view as shown below.

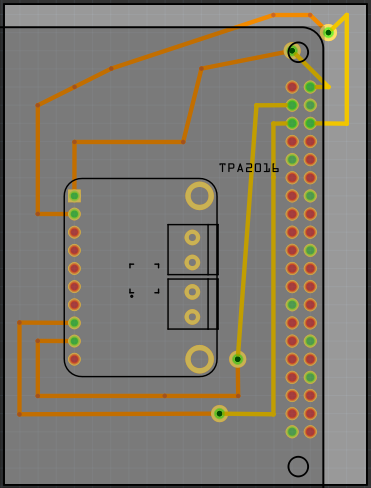


Figure 3- PCB Gerber Files

### PCB Soldering

1. Using the same rules as when soldering the Sensors, solder pieces of wire in between the vias on the PCB board.
2. Use a 10-pin header to place on top part of the PCB board. Solder the pins from the back. Ensure they have enough solder as this connects the pins of the amplifier to the pins of to GPIO.
3. Use a 6-pin connector from underneath the PCB so that that the input pins will fit into the GPIO pins of the RPI. Solder from the top side of the PCB and again ensure solid connections are made.

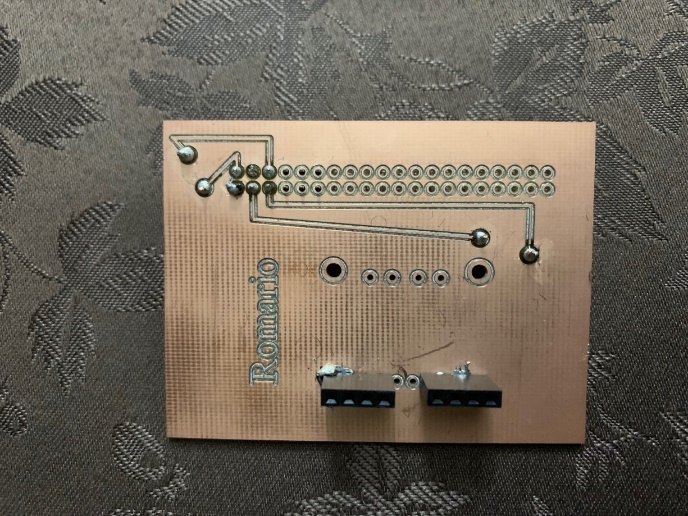
Top view:

Figure 4- PCB Top View

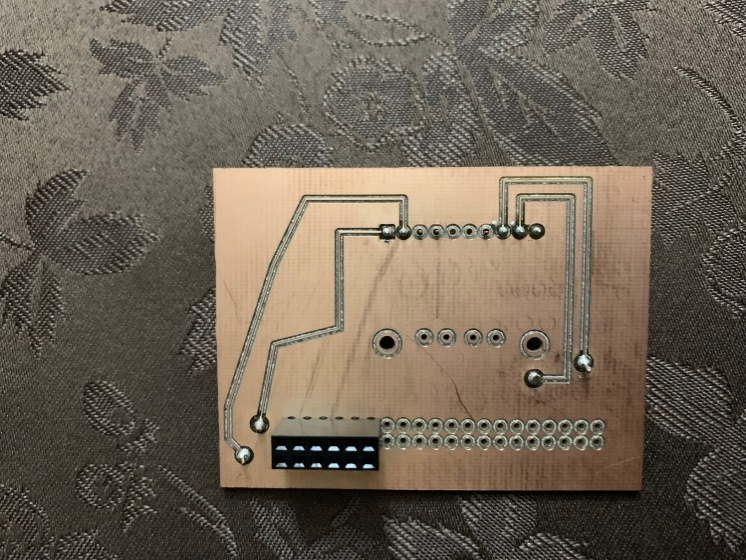
Bottom view:

Figure 5- PCB Bottom View

### Wiring

1. First the pins for the input of the amplifier will have to be soldered. Using 2 red wires of equal length, solder them onto the top of the amplifier pins that says R+ and L+.
2. Secondly, using 2 black wires of equal length, solder them onto the top of the amplifier pins that says R- and L-.

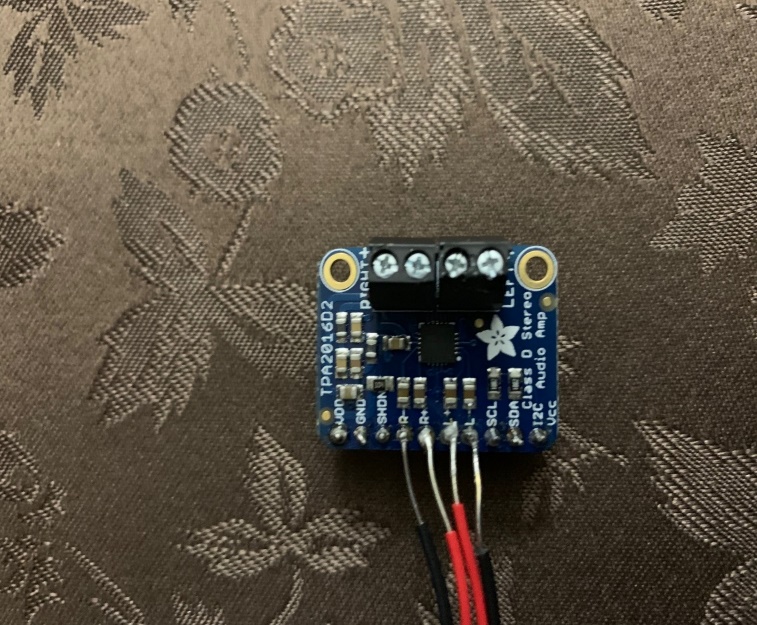


Figure 6- Sensor Wired

1. On the positive terminals of the speakers, solder the red wire to each speaker terminal. Solder a black wire onto the negative terminal of each speaker.
2. The two black terminals on the amplifier are the speaker output terminals. Use a small Philips screw driver and loosen the screws.
3. Place the positive lead of the R speaker into the positive terminal of the R output on the amplifier.
4. Place the negative lead of the R speaker into the negative terminal of the R output on the amplifier.
5. Place the negative lead of the L speaker into the negative terminal of the L output on the amplifier.
6. Place the positive lead of the L speaker into the positive terminal of the L output on the amplifier.
7. Tighten all the screes to secure each wire as you go along.
8. Place the amplifier pins in the 10-pin connector soldered onto the PCB. Ensure that the pins are corresponding to each other i.e. VDD TO VDD & VCC to VCC.
9. Insert the 3.5mm to terminal adapter into the 3.5mm jack on the raspberry pi.
10. Use a small Philips screw driver to loosen the screw terminals.
11. Place the black wires from the input side of the amplifier (L- & R-) into the ground (GND) terminal on the adapter and tighten the screw.
12. Place the red wires from the input side of the amplifier (L+ & R+) into the corresponding terminals on the adapter and tighten the screws.

### Power Up

In this section, we will now see if everything works, this works on whether you have soldered your PCB or you normally wired it onto your circuit board. Once connected boot up the Raspberry Pi, open the terminal window and follow these steps:

1. This will bring you into the configuration tool

sudo rasp-config

1. Use your arrows keys to go down and select "Interfacing Options"

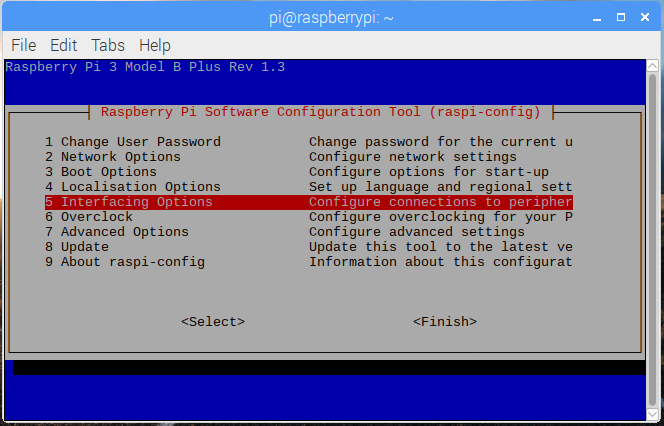
* 

Figure 7 - Interfacing Menu

1. Select I2C and submit yes. It should display ARM I2C is enabled.

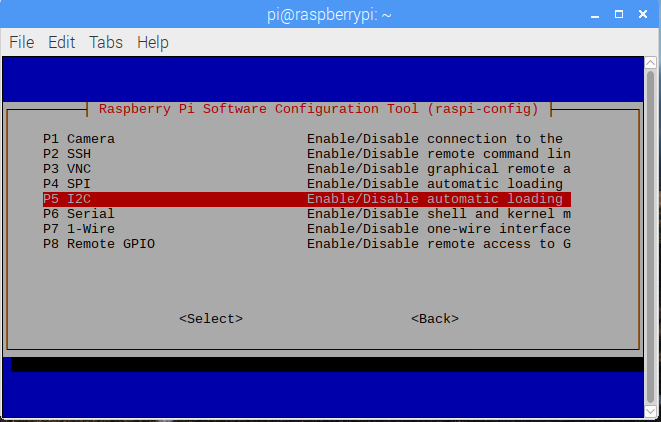


Figure 8- I2C Option

1. Exit by selecting the finish option. By using the sudo i2cdetect -y 1 command, it should your address for the sensors being used.

* Address is 58.

### Case Design

1. Cut 2pieces of (6x6x0.2) inches MDF wood. This will be used as the base of the case and the cover of the case.
2. Cut 4 pieces of (6x3x0.2) inches MDF wood to be used as the sides of the enclosure.
3. Using wood glue, place the 4 pieces of (6x3x0.2) pieces of wood on an upright angle perpendicular to the base of the enclosure, glue the contacts together on all edges.
4. Using a circular saw cut a 1-inch diameter whole in the side of one of the sides of the enclosure.
5. Proceed to make a 2nd hole in the adjacent side of the enclosure.
6. Make two 1-inch holes each in the cover of the enclosure as well.
7. Leave the components unassembled as the cover will be glued on the side frames after assembly (explained below).

### Assembly for Hardware

1. Place the assembled hardware in the case.
2. The two holes on each frame side will be used as the exit holes for the peripherals.
3. Place the speakers through the holes through the rear of the top cover of the enclosure.
4. From the outside of the cover, use small self-tapping screws to drill through the cover and place it self though the holes in the speakers designated for mouthing.
5. Take small nuts to thread through the screws to hold to speakers in place.
6. Finally, glue the cover onto the side frame.

The finished product should look like this:

Figure 9- Top of Enclosure

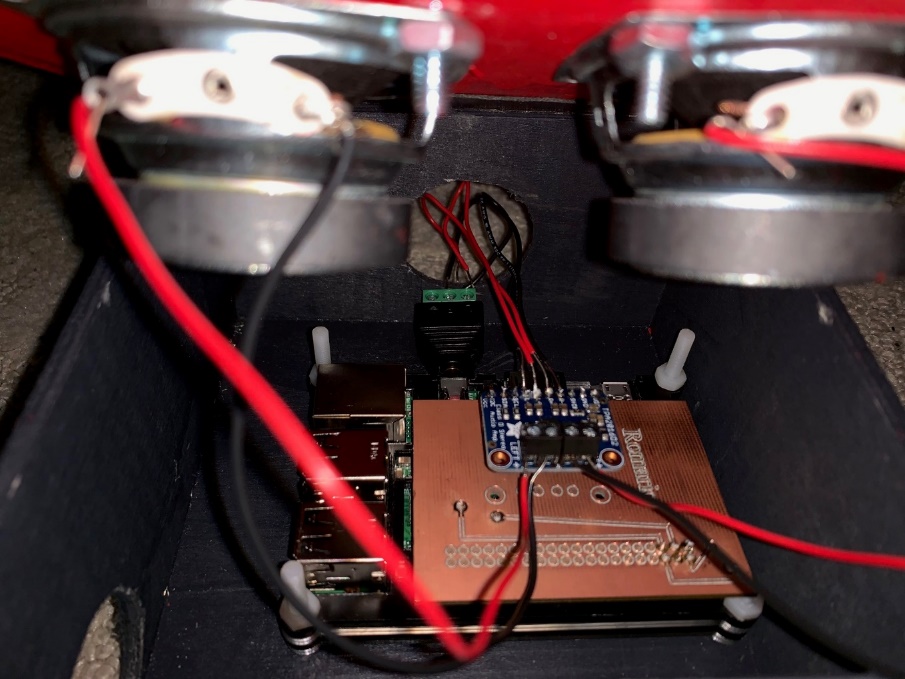


Figure 10 – Inside of Enclosure



Figure 11 – Full Enclosure

### Installing Python

For this project we will be coding in Python. For us to use Python, we need to install circuit Python will need to be installed, the instructions are as followed:

The following tests were experimented using Adafruit:

1. Run the update commands for the Raspberry Pi.  
 sudo apt-get upgrade

2. When done installing, run the command line for the python tools

sudo pip3 install --upgrade setuptools

3. Verify you have I2C Enabled

ls /dev/i2c\*



Figure 12 – Verification

4. Begin to install the Python Libraries

pip3 install RPI.GPIO

5. Use the following command to install adafruit-blinka:

pip3 install adafruit-blinka

To test if Python works, open python in the Raspberry Pi (it should be installed at this point), and write an example file to sample output.

import board  
import digitalio  
import busio  
   
print("Hello blinka!")  
   
# Try to great a Digital input  
pin = digitalio.DigitalInOut(board.D4)  
print("Digital IO ok!")  
   
# Try to create an I2C device  
i2c = busio.I2C(board.SCL, board.SDA)  
print("I2C ok!")  
   
# Try to create an SPI device  
spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)  
print("SPI ok!")  
   
print("done!")

Save it, then run it on the command line by typing

python3 blinkatest.py

The following should be seen

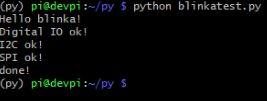


Figure 13 - Blinkatest Output

### Code for Sensor

from firebase import firebase

firebase = firebase.FirebaseApplication('https://fouronesixsound-51999.firebaseio.com/', None)

import urllib.request

import pygame

import urllib.request

import decimal

while 1==1:

Play = firebase.get('/Play', None)

url = firebase.get('/URL',None)

volume = firebase.get('/Volume', None)

if Play==1:

urllib.request.urlretrieve(url,'/home/pi/Desktop/mysong.mp3')

pygame.mixer.init()

pygame.mixer.music.load("mysong.mp3")

vol = decimal.Decimal(volume)

Volume = vol \* decimal.Decimal('.01')

pygame.mixer.music.set\_volume(Volume)

pygame.mixer.music.play()

while pygame.mixer.music.get\_busy() == True:

pass

### Database Design

The database was mostly designed by Andrew. The database connection is established and connected to the created Android application. This connection allows for songs to be stored in the firebase storage and have many different readings stored in firebase database, these readings consist of the “Play” option, the selected URL option, all the song names and their download URLs beside them, and the “Volume” option. These values are able to be read and retrieved by the Raspberry Pi and our sensor. Account details are also saved onto the database as users can log in and out of their accounts when they want to.

### Mobile Application

Upon opening of the app, the splash screen will pop up and have a button that takes users onto the login screen where users can register an account or if they are already registered, they can sign into the app. If users try to log in with incorrect passwords or emails, notifications will pop up corresponding to the error notifying the user that logging in has failed. If invalid emails or passwords are inserted while trying to register as a new user, a notification will pop up indicating that registering has failed. Once everything is created and the user is successfully logged on the user can press the Start Application button and they will be taken to the home screen. This button allows access for the application even for users that do not want to create an account. The app was created to be user friendly and clean, so that users would feel encouraged to use it rather than being intimidated and worried that they might press the wrong button and mess up. With this in mind the home screen is very simple as it contains a few different features. On the home page users can adjust the volume level they would like to play the songs using the progress bar on the bottom of the page, they can then upload that volume to the database using the Upload Volume button. If users have a song selected, they can use the play button to start playing their songs on the hardware, this is indicated by a prompt below the play button. In the top right corner of the home page, the menu bar is located. This menu bar allows for navigation throughout the application, and allows for users to go to the different pages. The different pages consist of the Upload page, where users can upload their songs to the database, the Song List page, where users can see the songs stored in the database and select the songs to be played either through their phones or the hardware, and the About Us page, where users can learn a bit more information about us. The menu bar also consists of the Sign Out option, where users can sign out of their accounts, and the Quit option, where users can close the application completely.

**Test Cases**

Authentication:

The first thing they will see is if they can login, for new users they would have to register. If the user attempts to login without registering, a message will appear noting that the account is not registered. When the user inputs an incorrect email address such as @email.com, it would not work due to the authentication recognizing that this is not a proper email address. Therefore, the user would have to use either Gmail, Hotmail, Yahoo, etc. Once everything is met, the account is now registered and saved onto the firebase authentication. They can now log in using the newly registered account information and if there are any mistakes in the email or password, the appropriate warning will appear prompting the user that there is a mistake. When you are logged into an account, there is a message at the bottom of the screen indicating which account is logged in by displaying the email address signed in.

Volume Uploading:

On the home page you will be able to adjust the volume of the music being played through the hardware. When testing the volume adjustment, we had to make sure that all the different volumes when pushed to the database was properly being set and that said volume was also being the volume set in the python code and the hardware volume will be adjusted accordingly.

Play Button:

To test the play button, we checked the database to see if the child called “Play” was created. Once that child was created the value set of Play should have been either zero or one. When testing, when the play button is pressed the value of Play should be set to one and when the pause button is pressed the value of Play should have been zero. When testing with the Python code, when Play was equal to one, when the play button was pushed, the download and playing of the song should have started, this was constantly tested throughout the creation of the device.

Uploading Songs:

To test the uploading of songs when the Select Songs button was clicked, a prompt for permission to access files from the phone would show up and if permission was not granted, the ability to upload songs from the phone’s files will not be able to work. If permission was granted, users will be able to see all of their files but would only be able to click on and insert mp3 files into the database. When a file is selected and ready to be uploaded, the upload songs button should be pressed. When this button is pressed the file should be stored into the firebase storage and the download URL that is given in firebase storage, should be stored in the firebase database along with the given name from the file under a child called Uploads.

Retrieving Songs:

When retrieving the songs, using the card view, the file name of the files should show up into the middle of the card. If multiple songs are in the database, multiple card views will be displayed, all of them with their respective file names in the middle. When anyone of these cards are clicked on, the URL of that song will be sent into the firebase database under the URL child. This child contains the next song that will be played through the hardware device. If you want to play the songs just through your phone and not through the hardware, when the card is clicked on, not only will it send the download URL to the database but a music player will show up allowing for the songs to be played through the phone instead of the hardware.

**Android Components**

Many android components and libraries are used during development of the app, each with their own specific purpose.

Major methods used for general overall use:

* **Intent (this, MainActivity.class)** - this method is often called when switching between screens (Often used with buttons or TextView links)
* **findViewById (int id)** - this method is used frequently to locate and interact with views found from layout resource files that are attached to the current activity
* **toastmakeTest(applicationContext, text, duration) -** this method is used often to send feedback to the user when they do a certain task like logging in.
* **onCreateView() -** similar to onCreate, but allows the view of other associated fragments.

Main methods for user authentication:

* **signInWithEmailAndPassword -** this method is called when the user gains access to their respective account once the proper email and password are entered
* **getCurrentUser() -** this method retrieves the data from the current user logged into the server, thereby accumulating further input they enter and save to the database

Main Methods for all Firebase Activities:

* **ChildEventListener () -** Listens for any activities to the children.
* **ValueEventListener () -** Looks for data changes in specific location of the database.
* **onChildAdded () -** This method allows the android device to pull data from the data structure and read from the database. In turn when a new child is added to the location to which it was added. For example, the temperature readings from Firebase.
* **onChildedChanged () -** This method will update the data when a child location has changed. Whenever there is a new temperature reading**.**
* **onChildRemoved () -** An event when the child is removed. Whenever content is being removing from the database.
* **onChildMoved () -** This method is used when the location of the child changes. Used to sort data on the database.
* **onDataChange () -** Similar to onChildedChanged, but will reads the static snapshot of the information at the given path at the timed event.
* **writeData()** - Will write the database with whatever the user is saving. This also includes the current user ID.
* **FirebaseDatabase.getInstance().getReference() –** this method refers to the firebase database and gets the information of the given firebase database.

Main methods for home:

* **setOnClickListener(new View.OnClickListener) -** When a button is placed this allows the button to be set. In doing so it becomes it creates a function called onClick(View v). As a result, the system allows all executeable code that is under onClick(View) once the button is pressed.
* **setValue() –** this method sets the value of a certain process

Main methods for upload:

* **setOnClickListener(new View.OnClickListener) -** When a button is placed this allows the button to be set. In doing so it becomes it creates a function called onClick(View v). As a result, the system allows all executeable code that is under onClick(View) once the button is pressed.
* **getLastPathSegment() –** this method gets the decoded last segment in the path.
* **getDownloadURL() -** this method asynchronously retrieves a long lived download URL with a revokable token. This can be used to share the file with others.
* **addOnCompleteListener –** this method works like onClickListener but instead of working on click, it works on completion of a certain process.
* **setValue() –** this method sets the value of a certain process.

Main methods for download:

This method uses a recycler view. This is a view group that has the main function of rendering any adapter-based view.

In the downloads we also use a card view. A card view is a frame layout with a rounded corner and background shadow.

We also use another process called an adapter. An adapter object is supposed to be the bridge between an AdapterView and all of the data for that view. It fills in all the AdapterViews with the appropriate data and content.

* **RecyclerView.Adapter -** To handle the data collection and bind it to the view
* **setOnClickListener(new View.OnClickListener) -** When a button is placed this allows the button to be set. In doing so it becomes it creates a function called onClick(View v). As a result, the system allows all executeable code that is under onClick(View) once the button is pressed.
* **getAdapter() –** This method gets the adapter for this view.

# Status Report

### Status Update 1

**2. Honestly report recent and current progress of each team member.**

- Thus far, the project has been progressing very well. The speaker terminals have been re soldered to enable proper conducting of power between the amplifier and the input terminals of the speakers. The case for the raspberry pi has begun to be rebuilt as needed. In relation to the software aspect of the project, the database is being restructured to facilitate new features such as uploading song from the physical device.

**3. Problems and hyperlinks.**

-The only issue for the project that is being faced is trouble shooting the raspberry pi to allow fire base to operate on it. This is a mandatory feature that needs to be fixed as soon as possible. The Github account has been attached as a whole to show the hardware section and the software section. Moving forward a third repository will be created to store any current update that is made. <https://github.com/RomarioT>

**4. Financial update.**

**-**In regards to the financial aspect of the project, the raspberry pi kit was purchased for $90, a set of speaker drivers were purchased for $30, speaker wires were already attained prior to the project, the amplifier sensor was purchased for $40 after shipping costs and the materials for the case sum up to be $20. This gives a cumulative amount of $180.

Best Regards,

Romario Kevin T. & Andrew Le

### Status Update 2

**2. Progress Update**

Work towards finishing the software is almost complete as test code to upload files to a database has been created and needs to be tested. Progress on connecting the Pi and the firebase is still being worked on and should be working soon.

**3. Problems and Hyperlinks**

Connecting the raspberry Pi to the firebase is still causing some trouble as a solution to that problem hasn't been found yet. Another issue that occurred was that the gradle in Android studio could not compile thus testing the code could not be done.

Here is another link to the GitHub: <https://github.com/RomarioT>

**4. Financial Update**

As of right now there have been no financial changes since our last e-mail.

### Status Update 3

Good Day,

**2. Honestly report recent and current progress of each team member.**

- Thus far, the project has been progressing better now. The hardware aspect of the project should be completed in roughly 1.5 weeks. The MDF base board for the speakers and the pi was ordered and has to be put together. The 3.5mm jack terminal to speaker input was ordered as well but is still awaiting arrival. In regards to the software aspect of the project, The application has been completed for all the requirements.

**3. Problems and hyperlinks.**

-The only issue for the project that is being faced is trouble shooting the raspberry pi to allow fire base to operate on it. This is a mandatory feature that needs to be fixed as soon as possible. The GitHub account has been attached as a whole to show the hardware section and the software section. Moving forward a third repository will be created to store any current update that is made. https://github.com/RomarioT

**4. Financial update.**

- In regards to the financial aspect of the project, the raspberry pi kit was purchased for $90, a set of speaker drivers were purchased for $30, speaker wires were already attained prior to the project, the amplifier sensor was purchased for $40 after shipping costs and the materials for the case sum up to be $20. The 3.5mm jack to speaker level input was purchased for $10. This gives a cumulative amount of $190.

Best Regards,

Romario Kevin T. & Andrew Le

### Status Update 4

Good Day Mr. Tian,

I would like to inform you about the progress in terms of troubleshooting of the project. Thus far the android application that Andrew is working on is in the final stages. So far, the initial screens operate as they should - the login screen, the main media player, sending volume information to the firebase database, uploading audio files to the database and actively adjusting volume information. There is an issue that is being faced attempting to retrieve a list of all the songs that are uploaded to the database as well as downloading them. The action plan to rectify this is to do research the use of array list and attempt to re implement that function in the application.

In terms of progress for the hardware aspect of the project handled by Romario, as you know we have recently successfully connected the raspberry pi to the fire base database. With that being said, we can actively view information stored on the firebase. The speakers have been fully tested and are in working order along with the amplifier. This was done by connecting the speakers to a 3.5mm adapter then connected to a device. The sound output was minimal. The adapter was then connected to the amplifier via the RPI and the speaker output was far louder. This shows the amplifier is in working order. An image of the address of the I2C will also be included in the final report to show that all the hardware is working with each other. One issue that was faced was that the PCB was not working properly. the fritzing files have been resent to the prototype lab to be remade and re soldered. The base and casing for the project was also not fully completed as the PCB needs to re soldered first.

On a side note we are missing the checklist and the build instructions. Andrew has completed his part of the build instructions and I am working on my part (the hardware). I needed to include some pictures in the file to complete it however the PCB needs to be remade. All the theoretical part of the build instructions has been completed. The files will be published to the GitHub today.

Best Regards,

Romario Kevin Tulloch

# Problems Encountered

During this project many problems occurred to us.

One problem that occurred during the creation of this product was when sending the song URL to the database, the wrong download URL was always wrong. This was a major issue as using the download URL was the only way that we would be able to access the song files. Without this our project would not be able to work as we would have no songs to download or even play. This issue took a lot of time to fix and also had a huge importance/priority for us as it is a key concept in our project.

Another problem that we encountered was the PCB. This was a huge problem as without the PCB we would not be able to complete this project. The PCB allows for a stable connection between the Raspberry Pi and our sensor. Without this the Raspberry Pi would not be able to connect to the sensor thus our speaker would not be able to play music through the amplifier. The problem that occurred was, we were missing the inclusion of both VCC and VDD as we thought that we would be able to create the PCB using only one of the two.

Not only did we have major issues but there were some minor issues as well. One minor issue we found was that older versions of the implementation were inserted. This caused problems with compiling and also caused the app to crash.

# Solutions

With problems there also must be a solution or the project would not have worked. Here are our solutions to all of the problems we had:

With the wrong URL being sent into the database, a lot of research was conducted into solving this problem and many different attempts were made from the research. The main issue was resolved by putting the contents in which the URL is getting retrieved into an onSuccessListener. Putting the contents in which the URL is getting retrieved into an onSuccessListener solved this problem as after this was completed the right download URL was being stored in the database and the ability to download the songs was complete.

With the PCB issue, we first checked to see if the PCB was the issue or if the sensor was the issue. This led to rewiring the sensor straight to the Raspberry Pi rather than using the PCB. This test would allow us to confirm our suspension of the PCB being the issue. Once wired, we tested to see if the audio amplifier would work, and it did, this led to us knowing it was the PCB that was wrong. To fix this issue, we created a new PCB using connections to both VCC and VDD and sent the files to the Humber Prototype Lab to be created. Once the PCB was created, we connected the PCB to the Raspberry Pi once again and tested to see if the address of the sensor was being registered and read through i2c detect.

To solve the minor issues, in the build gradle, we reviewed all the implementations and made sure that they were the most up to date versions and this allowed for the app to compile and would also stop the app from crashing when trying to access the different screens.

# Future Plans

As our project is in the early stages of development, the plan for the product now can greatly improve. Whether it be with hardware, software, or anything else. One thing that can be upgraded is the actual speaker we are using. When developing the hardware, we can have different versions of our product with different speaker sizes and power. This will allow for users to have more options when trying to decide which they want; they can have a stronger speaker or a weaker speaker. Secondly, we would like to implement a way for the users to be able to skip the song and move onto the next song. As of right now, our app allows for users to select the song that they would like to play next, but the newly selected song will only begin to play once the current song finishes playing, we would like to implement a fast forward button or a stop button that would all the users to either stop the current song playing or fast forward to the end, so that the next song can be played. Another feature we would like to adjust is to the database. We would like to use a larger database as users might have a lot of songs that they would like to be played and saved. This might cause an issue as the space in the database could be limited, thus we would like to expand it and allow for more storage of the music. In the future, we would like to create a new enclosure. A new enclosure for our product that would be smaller would be the ideal new enclosure. Allowing for a smaller less bulky enclosure would allow users to take their speaker with them, making the speaker more portable and more user-friendly. Along with making the enclosure more portable and smaller, different colors could be incorporated as it would allow users to pick the colors that they would like and would allow for a little bit more personality to the speaker.

# Conclusion

To conclude, FourOneSix Sound is an android based project that was created to allow users to play their favorite songs either through their phones or through the hardware device that we created. Our android app consists of three main screens, the Home page, the Upload page, and the Song List page. These pages all work together to create a great app that allows for our hardware device, made from a Stereo 2.8WClass D Audio Amplifier - I2C Control AGC - TPA2016, to play songs that the user wants.

# Bibliography

Firebase for Android: File Storage. (n.d.). Retrieved from <https://code.tutsplus.com/tutorials/firebase-for-android-file-storage--cms-27376>

Android Developers. (n.d.). Retrieved from <https://developer.android.com/>

Angelova, G. A. (n.d.). Uploading and downloading files to/from Google Cloud Firestore. Retrieved from <https://stackoverflow.com/questions/49529957/uploading-and-downloading-files-to-from-google-cloud-firestore>

Using the RecyclerViewEdit PagePage History. (n.d.). Retrieved from <https://guides.codepath.com/android/using-the-recyclerview>

Adafruit Industries. (n.d.). Stereo 2.8W Class D Audio Amplifier - I2C Control AGC - TPA2016. Retrieved from <https://www.adafruit.com/product/1712>

(n.d.). Retrieved from <https://www.pygame.org/docs/ref/music.html>