LAMBTON COLLEGE

Embedded Systems and Engineering Design

PROJECT REPORT

SMART GLASSES PROJECT

A 4th Semester Project for post-graduate Diploma for Embedded Systems and Engineering Design

By:

GURSEWAK SINGH C0725167
JAIDEEP SINGH C0727763
SOURAV HARISH C0732272

Submitted To:

TAKIS ZOURNTOS

ABSTRACT

The maturing field of wearable computing aims to inter-weave computing devices into everyday life. This report focuses on smart glasses, one of the categories of wearable computing devices which is very present in the media and expected to be a big market in the next years. It analyses the differences from smart glasses to other smart devices, introduces many possible applications for different target audiences and gives an overview of the different smart glasses which are available now or should be available in the next few years. Interesting technological features of the smart glasses are highlighted and explained. The smart glasses assignment is a raspberry pi zero w project that has some advanced features. Smart glasses are going to be a big market in upcoming years.

ACKNOWLEDGMENT

We want to express our thanks to, Prof Takis Zourntos, for going through every detail, giving expert advice and encouragement throughout this difficult project to make this project a success.
Last but not least, we would like to thank our friends and family for their help in every way for the success of this project.

INTRODUCTION

Smart glasses are wearable minicomputer glasses that makes a person to see whatever is going on in his phone, other than the view he can see through the clear glasses. Moreover, the augmented reality can be displayed into the glass view. In today's era, people are too busy even to take out their phone out of the pocket while doing some work, so here this Smart glass is very helpful to sort out this kind of problems. For example, this smart glass can be used to get to know who is calling or texting on our phone while the phone is still in our pocket.

Other than this it can be used by a person who is riding a bicycle and does not have an empty hand to carry the phone out with google maps on. it could be a very useful device for people daily use and can help people stay connected with their love ones in real time. Which could be really use full in emergency settings.

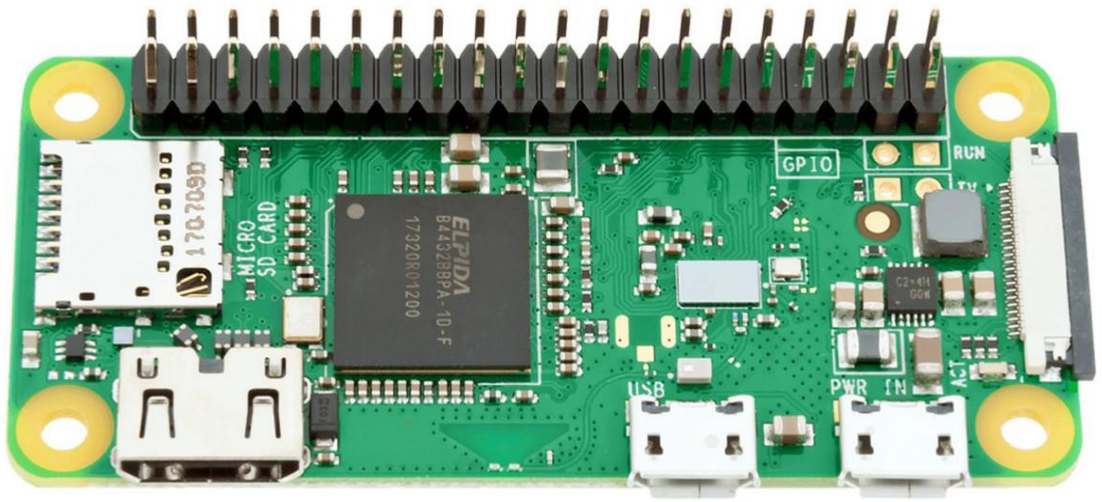
With advancement in technology, new features and specs are introduced with every new Model launched. But in this project, we have tried to make glasses that are affordable and eases our daily routine life.

HARDWARE DESCRIPTION

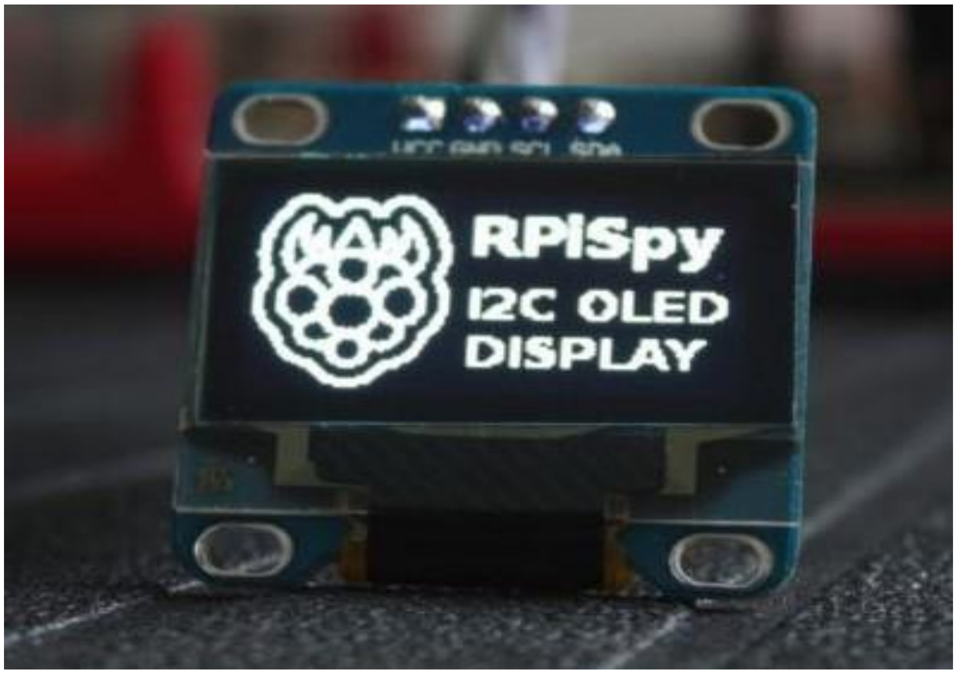
Following hardware components are required for this project

- Raspberry pi zero w
- OLED Display
- Power bank module
- 5v battery
- Microphone
- Speaker
- Connecting wires
- WM8960 audio module
- Switch

Raspberry pi zero w: can be described as the upgraded version of raspberry pi zero as it comes with certain additional inbuilt features. It is very ideal for making embedded internet of things (IOT) projects as it comes with inbuilt Bluetooth and WIFI. It has been designed to be as compact and as flexible as possible with mini connectors and 40 pin GPIO.

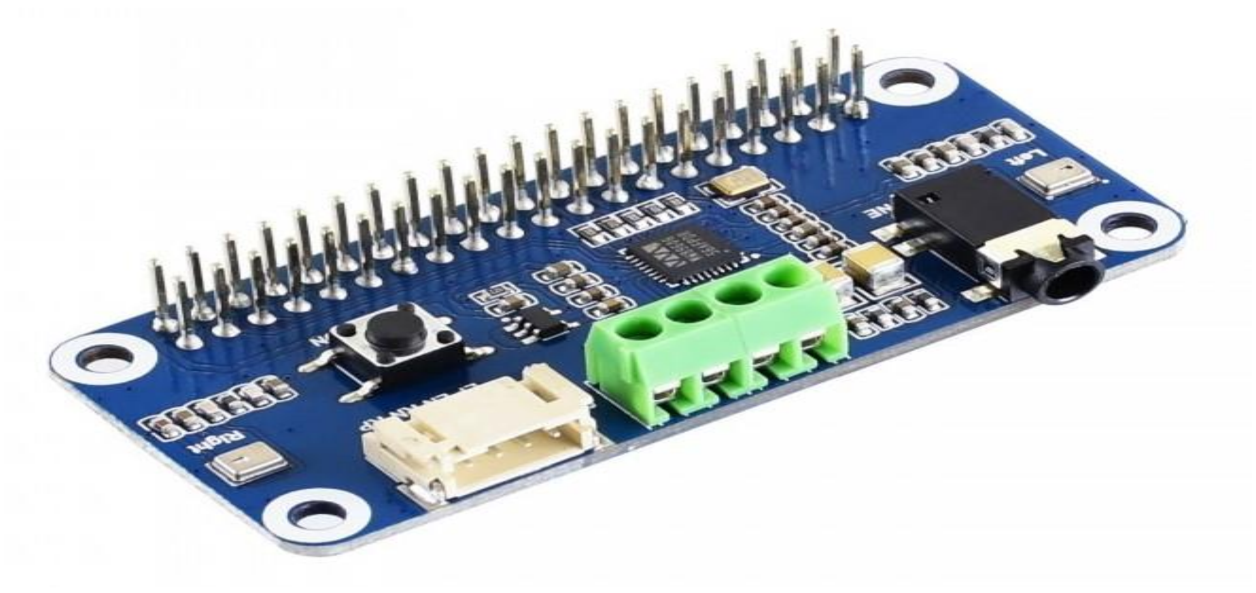


OLED Screen: A 4 pin monocolor OLED (organic light emitting diode) of 0.96-inch display with 128x64 pixels. This display consumes less power in comparison to other displays. The four pins are GND, VCC, SLC and SDA.

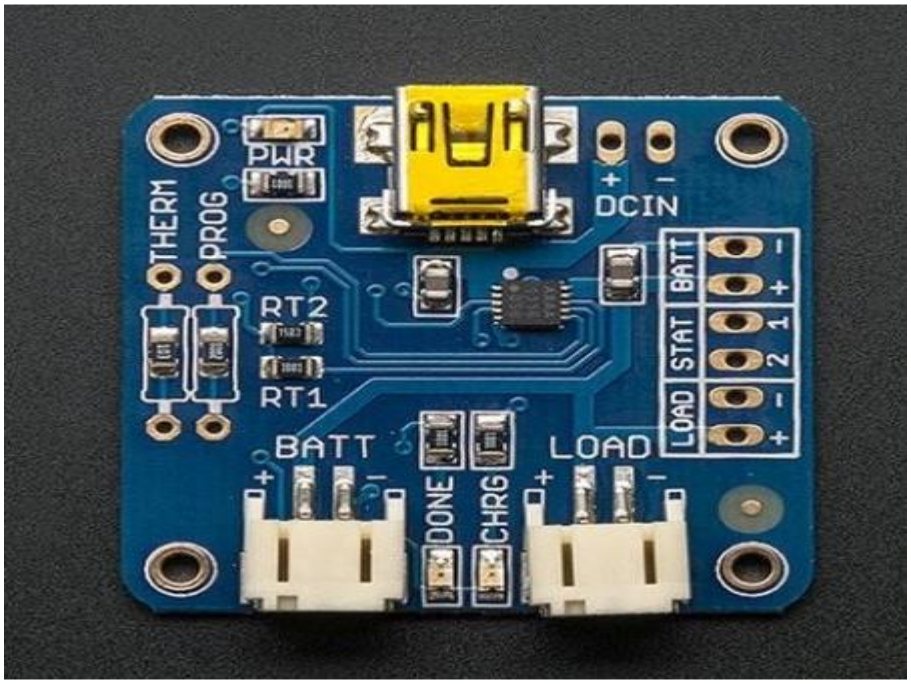


WM8960 Audio HAT Module: is particularly designed for raspberry pi. It is very useful module because of its compatibility, low power consumption and supports stereo encoding and decoding. This module uses I2C interface for communication and even supports earphone drivers and speaker drivers. This device has onboard dual-channel speaker interface that can be used to play music directly in speakers. This module has an inbuilt 3.5 mm earphone jack.

Power supply: 5v
Audio Interface: I2S
Control interface: I2C
Logic voltage: 3.3v



Power Bank Module: A lithium ion and lithium polymer charger MCP73833 is used. Function of this module is power management in project.

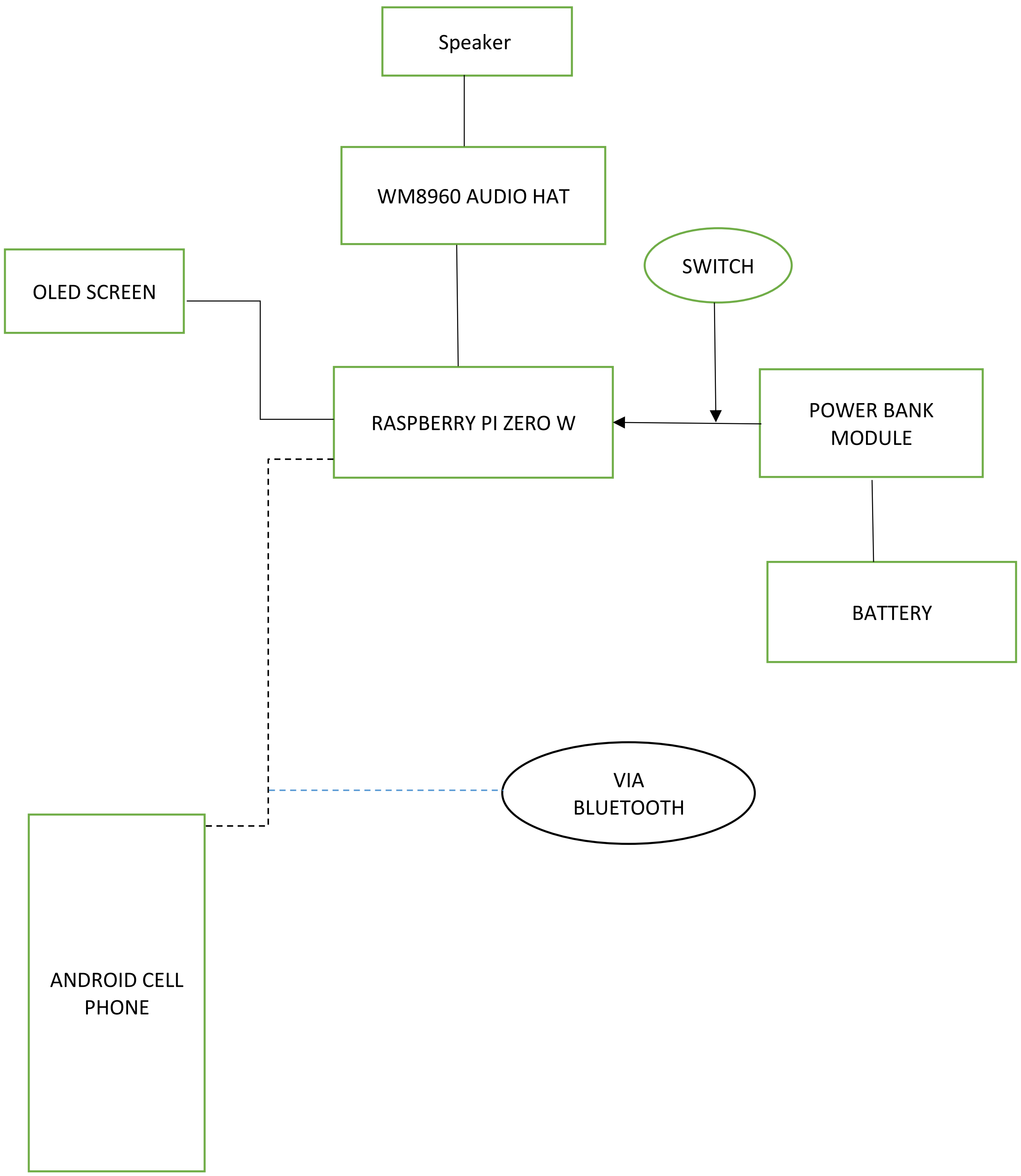


Battery: A powerful LP803860 lithium polymer 2000mAh battery is used.

Cell Phone: Another important requirement for this project is an android cell phone.

Miscellaneous: For this project we need a piece of magnifying glass (prism), a small piece of a plane mirror, a pair of glasses, case, connecting wires and switch. An Android application SPP client is also used for this project.

BASIC BLOCK DIAGRAM



SET UP RASPBERRY PI ZERO W:

- Download Raspbian, SSH and config file in your pc.
- Put all of these in microSD card.
- Enable SSH
- Enable WIFI

```
 country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
ssid="WIFI_SSID"
scan_ssid=1
psk="WIFI_PASSWORD"
key_mgmt=WPA-PSK
}
```

Add username and password of your Wi-Fi.

- SSH into raspberry pi zero w by inserting the microSD card and power with micro USB Cable.
- Open Linux terminal and check for connection.

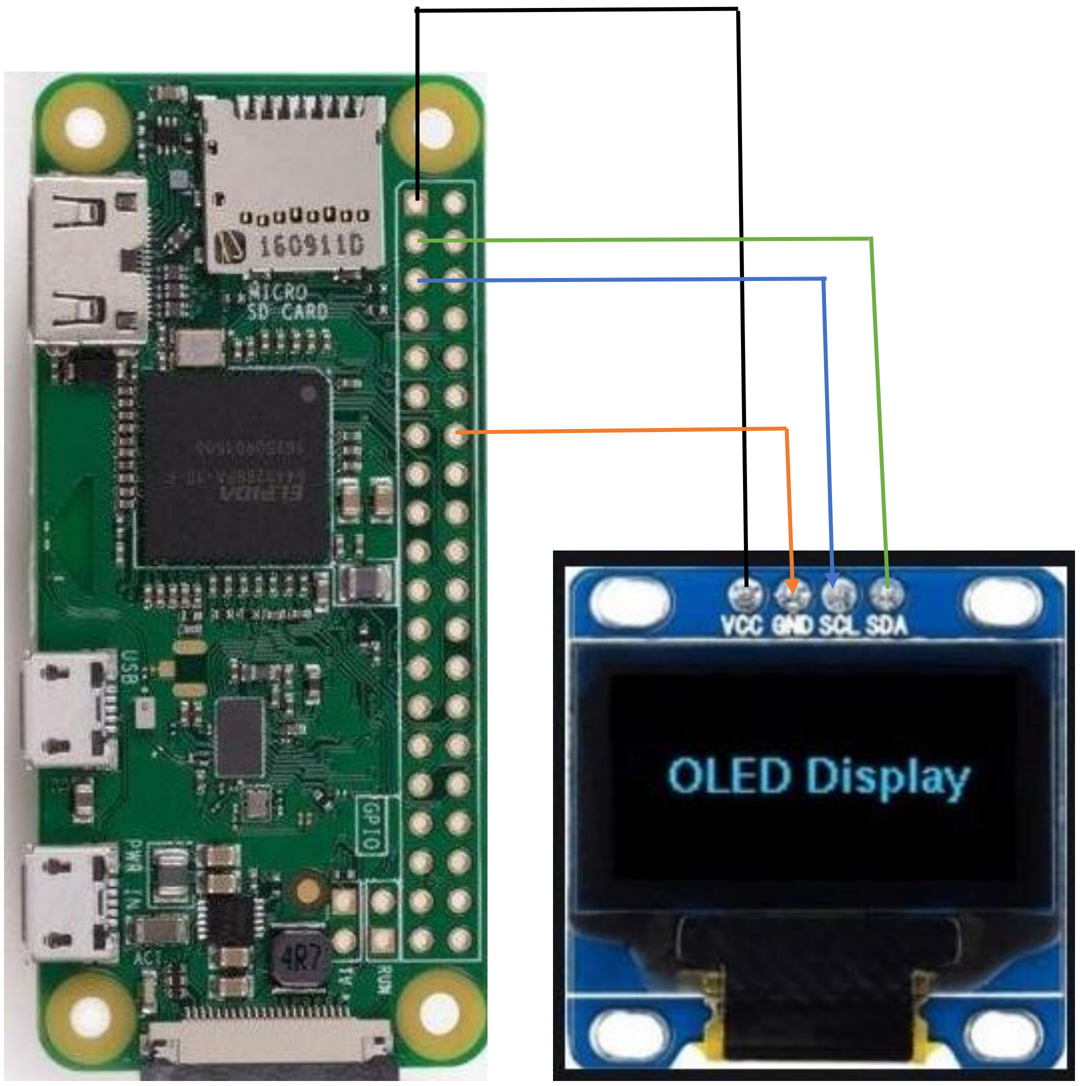
Enter- ssh pi@raspberrypi.local
password:

Your raspberry pi zero W is ready for use.

CONNECTING RASPBERRY PI ZERO W AND OLED SCREEN

Raspberry pi zero w has 40 GPIO pins and OLED screen has 4 pins.

OLED PIN	GPIO PIN	NOTES
VCC	1	3.3 VOLT
GND	14	GROUND
SCL	5	I2C
SDA	3	I2C



You can connect VCC pin to either pin 1 or 17 as they both provide 3.3v
You can connect ground pin to either pin 6, 9, 14, 20, 25, 30, 34 or 39 as they all provide ground.

<u>Enable I2C Interface:</u>

1. Firstly, we need to update the Pi software to the latest, issue

sudo apt-get update
sudo apt-get upgrade.

2. Enable I2C and SPI modules directly with raspi-config tool, issue a

sudo raspi-config

Then go to menu Advanced Option, select i2c and under question" Would you like the i2c kernel module to be loaded by default ?", select Yes

3. To be able to compile you will need the compiler and some other tools, issue

sudo apt-get install build-essential git-core libi2c-dev i2c-tools lm-sensors

4. Get all the file from GitHub dedicated repo :

git clone https://github.com/hallard/ArduiPi_OLED

5. detect if the OLED is connected by issue a

   sudo i2cdetect -y 1

6. run a program and check if everything is working.

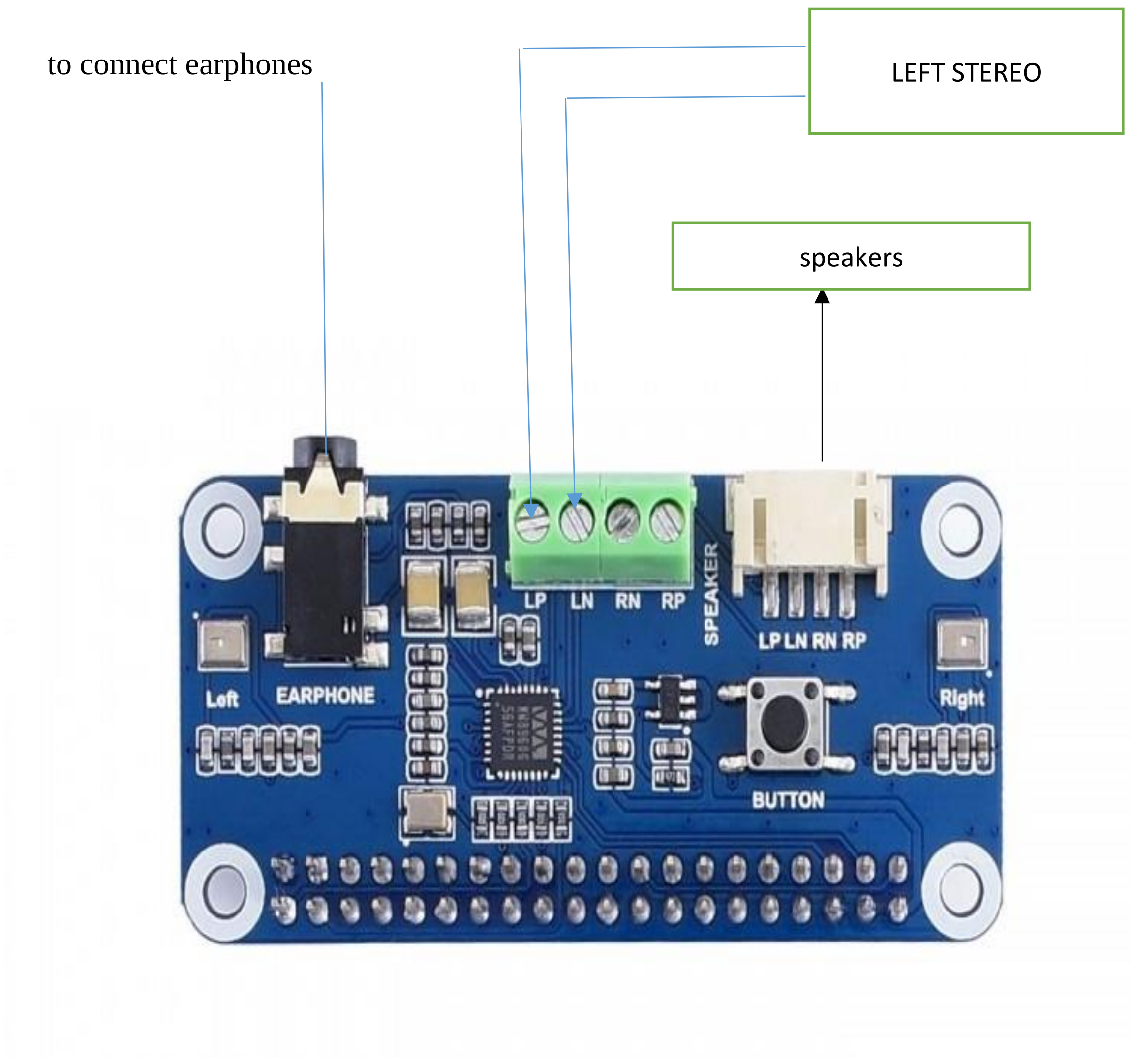
CONNECT SPEAKER TO WM8960 MODULE

A WM8960 Audio HAT module has 4 sockets to connect two stereos. It also has a port to connect headphones with the system and an additional socket to connect speakers to play music.

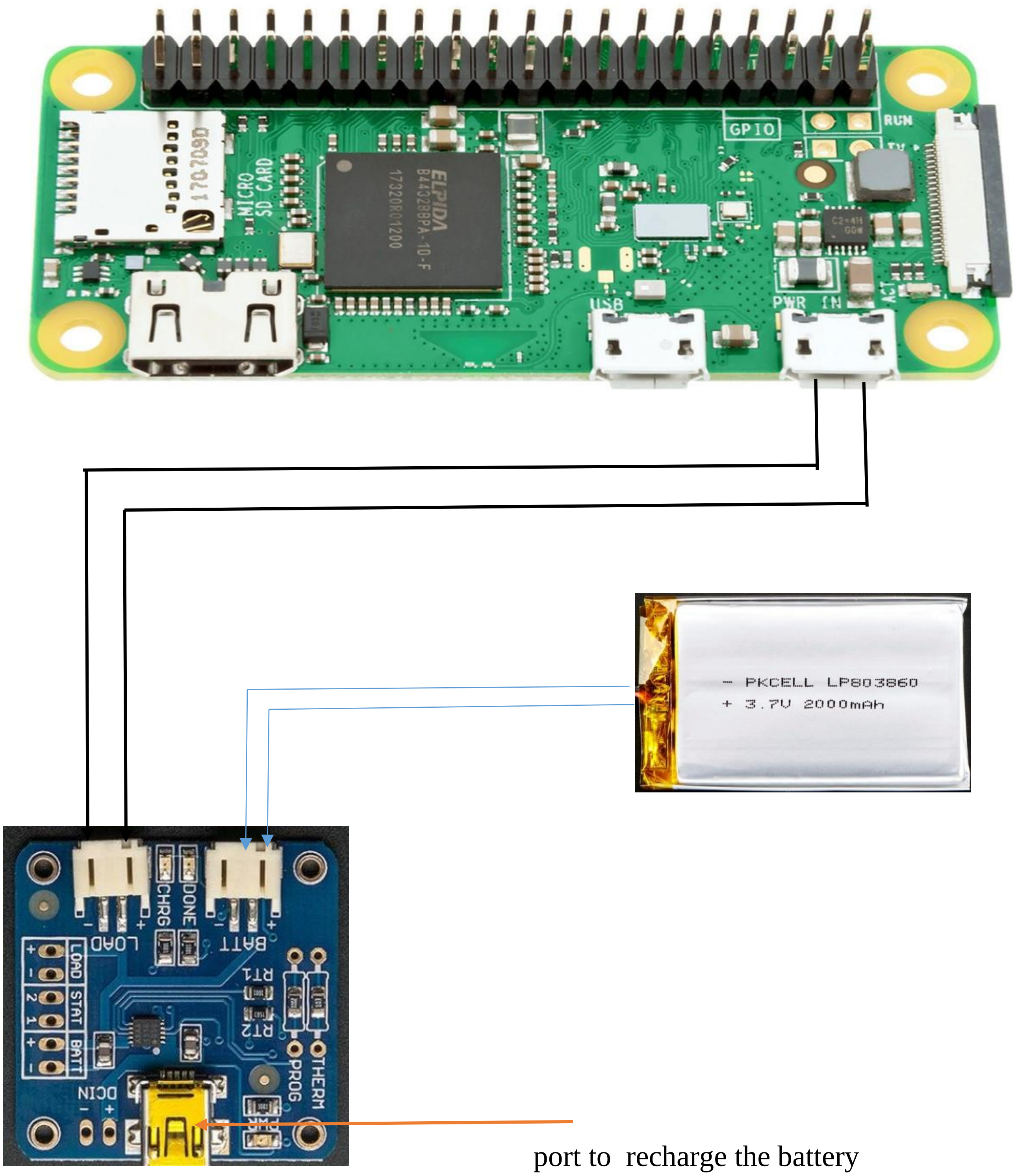
Install drivers for WM8960

1. Clone the driver from https://github.com/waveshare/WM8960-Audio-HAT
2. cd WM8960-Audio-HAT
3. sudo ./install.sh
4. sudo reboot
5. for checking the drivers sudo dkms status
6. it should show the kernel version else you need to install it again
7. Check sound card status of Raspberry Pi with command aplay -l and arecord -l
8. Use arecord to record audio and play
9. sudo arecord -f cd -Dhw:1 | aplay -Dhw:1

10. To setting sound and adjust volume, you can use alsamixer tool
11. sudo alsamixer
12. Set Default Soundcard
13. sudo vi /usr/share/alsa/alsa.conf
14. defaults.ctl.card 0
15. defaults.pcm.card 0
16.Modify these statements from 0 to 1
17. to play mp3
18. sudo apt-get install mpg123
19. sudo mpg123 test.mp3



CONNECT PI ZERO W , BATTERY AND POWER MODULE



The power module has two ports. One port is connected to the rechargeable battery and other port with the raspberry pi zero W's power socket. There are two ports on Pi 0 W one is for power and another one is USB.

The function of this module is power management. The battery is used to power this project and this module manages the power to the project.

There is third port on power module and it is used to recharge the battery in case fully drained.

A small switch is joined on the line connecting power module and raspberry just to switch off the power in case we are not using device.

WM8960 Audio HAT Drivers

https://github.com/waveshare/WM8960-Audio-HAT

Adafruit OLED Display Drivers

https://github.com/hallard/ArduiPi_OLED

you can check this GitHub account for any assistance

https://github.com/sourav5360/Final-year-project-

WM8960 Audio HAT Module

https://www.waveshare.com/w/upload/5/54/WM8960_Audio_HAT_User_Manual_EN.pdf

To solve the Bluetooth issues related to the raspberry pi Zero W

https://learn.pi-supply.com/make/fix-raspberry-pi-3-bluetooth-issues/

SOFTWARE DESCRIPTION

The most complicated thing about this project is coding and compatibility issues amongst the various modules that we are using in this project. The Bluetooth is not compatible with Raspbian lite so we need a Raspbian stretch version to sought out the issues with WM8960 module.

The WM8960 is a low power stereo codec featuring Class D speaker drivers to provide 1W per channel into 8Ω loads. Guaranteed low leakage, excellent PSRR and pop/click suppression mechanisms enable direct battery connection for the speaker supply. This IC integrates a complete microphone interface and a stereo headphone driver. External component requirements are drastically reduced as no separate microphone, speaker or headphone amplifiers are required. Advanced on-chip digital signal processing performs automatic level control for the microphone or line input. Stereo 24-bit Delta Sigma converters are used with low power over-sampling digital interpolation and decimation filters and a flexible digital audio interface. The master clock can be input directly or generated internally by an onboard PLL, supporting most commonly used clocking schemes.

We have developed the entire code in visual studio and the cross-tool chain is working perfectly. The compatibility issue is certain with the Bluetooth while using it with raspberry pi zero w.

Advanced TCP options that are often set with setsockopt, such as receive windows and the Nagle algorithm, don't make sense in Bluetooth, and can't be used with RFCOMM sockets. Aside from this, the byte ordering, and socket addressing structure differences, programming RFCOMM sockets is virtually identical to programming TCP sockets. To accept incoming connections with a socket, use bind to reserve operating system resource, listen to put it in listening mode, and accept to block and accept an incoming connection. Creating an outgoing connection is also simple and merely involves a call to connect. Once a connection has been established, the standard calls to read, write, send, and recieve can be used for data transfer.

```
#include "ssd1306_i2c.h"
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>

#include <pthread.h>
struct ps
{
        int st;
        pthread_t *thr;
};

//A mutex can be statically initialized by assigning
PTHREAD_MUTEX_INITIALIZER in its definition, as follows:
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
//

int status = 0;
void *recvsocket(void *arg)//
{

struct ps *p = (struct ps *)arg;
int st = p->st;
char s[1024];

while(1)
{
memset(s, 0, sizeof(s));

//read data from bluetooth.

int rc = read(st, s, sizeof(s));
if (rc <= 0)//
break;
char *word = "clear";
printf("phoneï¼š%s", s);
                //check clear and display string
if
(strstr(s,word)==NULL)
{
printf("Display OLED\n");
ssd1306_clearDisplay();
ssd1306_drawString(s);
ssd1306_display();
}

else{
printf("clear display\n");
ssd1306_clearDisplay();
}

//delay(100);

}

pthread_mutex_lock(&mutex);            //Locks a mutex object
status = 0;
pthread_mutex_unlock(&mutex);          //release a mutex object
pthread_cancel(*(p->thr));             //cancel thread
return NULL;
}

void *sendsocket(void *arg)
{
int st = *(int *)arg;
char s[1024];
while(1)
{
        memset(s, 0, sizeof(s));
        read(STDIN_FILENO, s, sizeof(s));
        //write data to phone
        write(st, s, strlen(s));
}
        return NULL;
}

int main(int arg, char *args[])
```

```
{
   struct sockaddr_rc loc_addr = { 0 },
   rem_addr = { 0 };
   char buf[1024] = { 0 };
   int s, client, bytes_read;
   socklen_t opt = sizeof(rem_addr);        //ssd1306 begin

ssd1306_begin(SSD1306_SWITCHCAPVCC, SSD1306_I2C_ADDRESS);
   // allocate socket
   s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

   // bind socket to port 1 of the first available
   // local bluetooth adapter

   loc_addr.rc_family = AF_BLUETOOTH;
   loc_addr.rc_bdaddr = *BDADDR_ANY;//any possible bluetooth adaptor
   loc_addr.rc_channel = (uint8_t) 1;
   bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));          //bind socket

   // put socket into listening

listen(s, 1);
pthread_t thrd1, thrd2;
   while(1){
// accept one connection
printf("start pthread\n");
//wait for connection
client = accept(s, (struct sockaddr *)&rem_addr, &opt);
pthread_mutex_lock(&mutex);


/*Locks a mutex object, which identifies a mutex. If the mutex is already locked
by another thread, the thread waits for the mutex to become available. The
thread that has locked a mutex becomes its current owner and remains the
owner until the same thread has unlocked it. When the mutex has the attribute of
recursive, the use of the lock may be different. When this kind of mutex is locked
multiple times by the same thread, then a count is incremented and no waiting
thread is posted. The owning thread must call pthread_mutex_unlock() the same
number of times to decrement the count to zero.  */
status++;
pthread_mutex_unlock(&mutex);            //Releases a mutex object.
if (status>1){
close(client);
continue;
}

if(client==-1){
printf("failure\n");
}

ba2str( &rem_addr.rc_bdaddr, buf );

fprintf(stderr, "accepted connection from %s\n", buf);

//memset(buf, 0, sizeof(buf));

struct ps ps1;
ps1.st=client;
ps1.thr=&thrd2;

pthread_create(&thrd1,NULL,recvsocket,&ps1);   //create receive thread;The
pthread_create() function starts a new thread in the calling  process.
                    pthread_detach(thrd1);
//The pthread_detach() function shall indicate to the implementation that storage
for the thread thread can be reclaimed when that thread terminates.

pthread_create(&thrd2,NULL,sendsocket,&client);          //create send thread;
pthread_detach(thrd2);
}

close(client);                      //close client
 close(s);                          //close socket

}
```

Streaming Bluetooth Audio from Phone to Raspberry Pi Using ALSA

1. Enabling Audio Profile Sink Role

1.1. Open the configuration file for blue alsa service

```
  sudo nano /lib/systemd/system/bluealsa.service
```

1.2. Search the line starts with "ExecStart" and add a profile option with a2dp-sink as below.

```
ExecStart=/usr/bin/bluealsa -p a2dp-sink
```

1.3 Reboot.

```
     Sudo reboot
```

2. Bluetooth Pairing and Connection

2.1 Launch BlueZ command line interface.

```
Bluetoothctl
```

2.2 Setup a pairing agent.

```
default-agent
```

2.3 Make the Raspberry Pi discoverable.

```
discoverable on
```

2.4 On your phone, search and select your Raspberry Pi from Bluetooth menu

2.5 Confirm the pairing on both your phone and Raspberry Pi.

```
Request confirmation
[agent] Confirm passkey 847261 (yes/no): yes
```

2.6 Authorize A2DP service (first 32 bits: 0000110d).

```
Authorize service
[agent] Authorize service 0000110d-0000-1000-8000-00805f9b34fb (yes/no): yes
```

2.7 Trust the phone so that Raspberry Pi will automatically accept connections from the phone from the next time.

```
trust XX:XX:XX:XX:XX:XX
```

2.8 Exit from BlueZ command line interface.

```
exit
```

3. Audio Routing

Forward audio from the phone to Raspberry Pi’s output.

```
bluealsa-aplay 00:00:00:00:00:00
```

4. Verify

Launch a media player and play some music on your phone. You should be able to hear the music from Raspberry Pi’s speaker

Troubleshoot : Sound Cutting Out

Raspberry Pi’s (Zero W) on-board Bluetooth+WiFi combo chip has an issue and the Bluetooth sound cuts out when WiFi is enabled. If you don’t need WiFi, you can just disable the WiFi interface to avoid the sound cutting issue

```
sudo ip link set wlan0 down
```

If you need WiFi, one of the workarounds is to disable the on-board Bluetooth and use a Bluetooth dongle instead. Below are the steps to do that

1. Open “/etc/modprobe.d/raspi-blacklist.conf”.

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

2. Add lines below and save.

```
blacklist btbcm
blacklist hci_uart
```

3. Connect Bluetooth dongle on Raspberry Pi’s USB port.

4. Restart the Raspberry Pi

DEMO DAY EXPERIENCE

Demo day was held on April 27, 2020. It was very exciting for us after so much hard work; we finally completed our project, which was fully functional with different behavior. Hardware and software were working perfectly without any issues. The display is also working perfectly. Everybody had their own project in class with different concept and features. We were second last group called for demonstration by our instructor and he asked us few questions about the hardware designing, about the problem faced during the project, code and division of labor amongst our group. He gave his expert advice by studying our code which helped us to improve the code and overall performance of our project. We really enjoyed the day and getting suggestions from respected instructor and friends to improve our project. This experience was generally new for us, as it was first time experience that we presented our project online. Finally, our professor gave us advice on what things to focus on in future so that we could have good opportunities in jobs. Instructor has very happy and satisfied by our work.

FUTURE PLAN

Building this project was a great experience and wonderful team effort. We started this project with a proposal and division of labor among ourselves. The process was quite clear we started from planning the components, designing Hardware, implementing hardware in best way possible, then working on our code. It is interesting when you know your code is going to affect the hardware, so we had to be very careful writing it. This technology has advanced over the years and we were thrilled to read about its development and future concepts in pipeline. There are still certain possibilities of development in this project. We have learned a lot in this project and apply our skills to make the world a better place.

APPENDIX

Website Links:

Sourav Harish : Sourav5560.github.io

Jaideep Singh : jaideeprajput.github.io

Gursewak Singh : gursewak95.github.io

GitHub Links :

https://github.com/sourav5360/Final-year-project-

https://github.com/jaideeprajput

WM8960 Audio-Hat-Drivers-

https://github.com/waveshare/WM8960-Audio-HAT

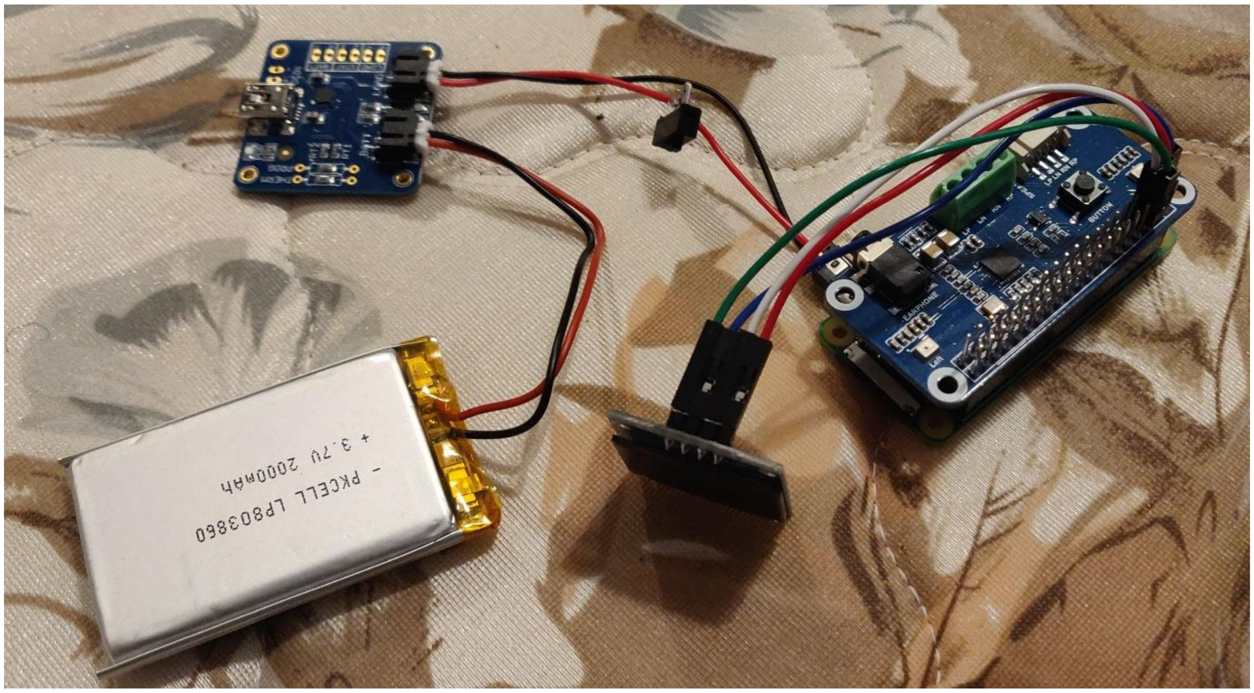
https://github.com/jaideeprajput/WM8960-Audio-HAT

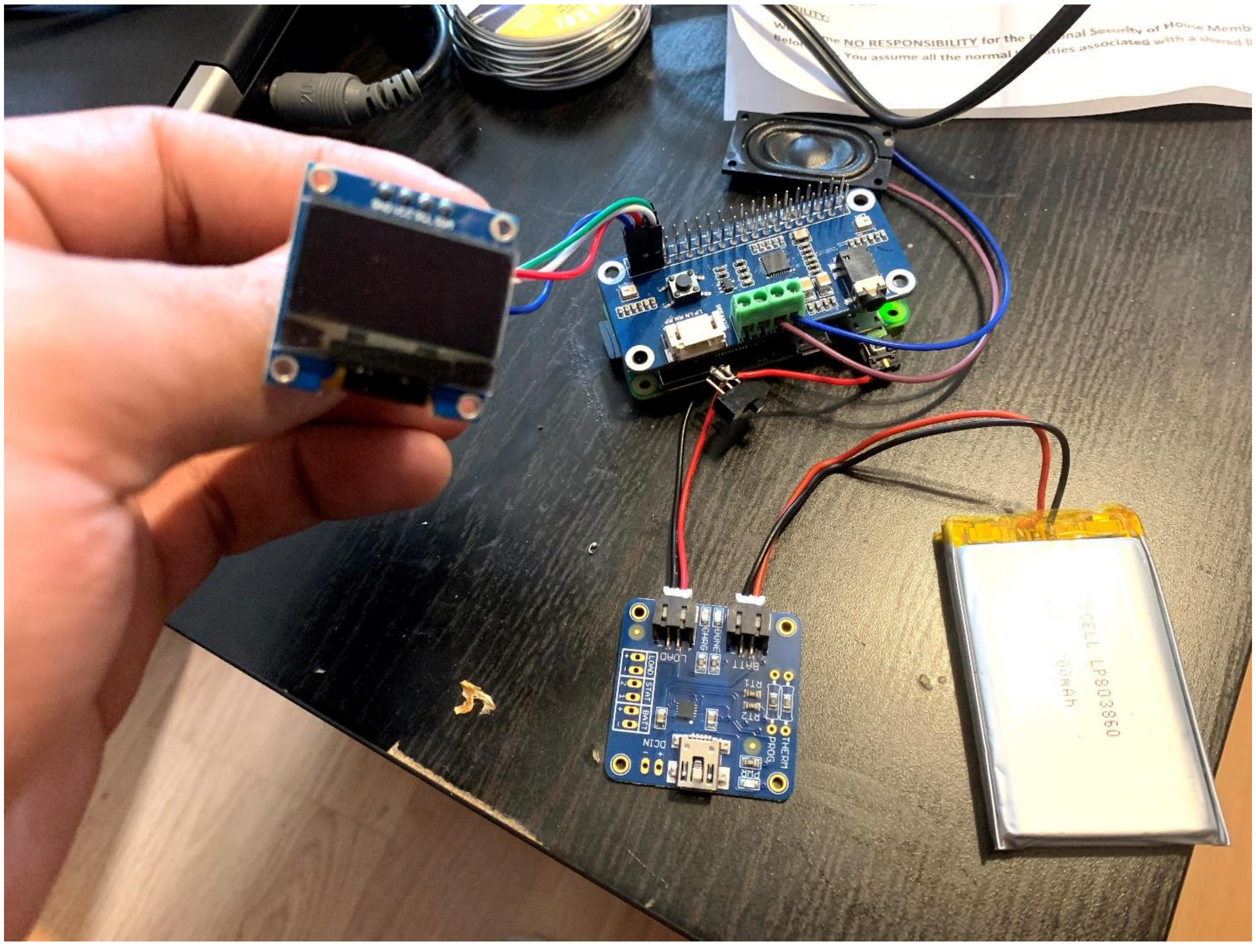
Visit GitHub links for the CPP file of code.

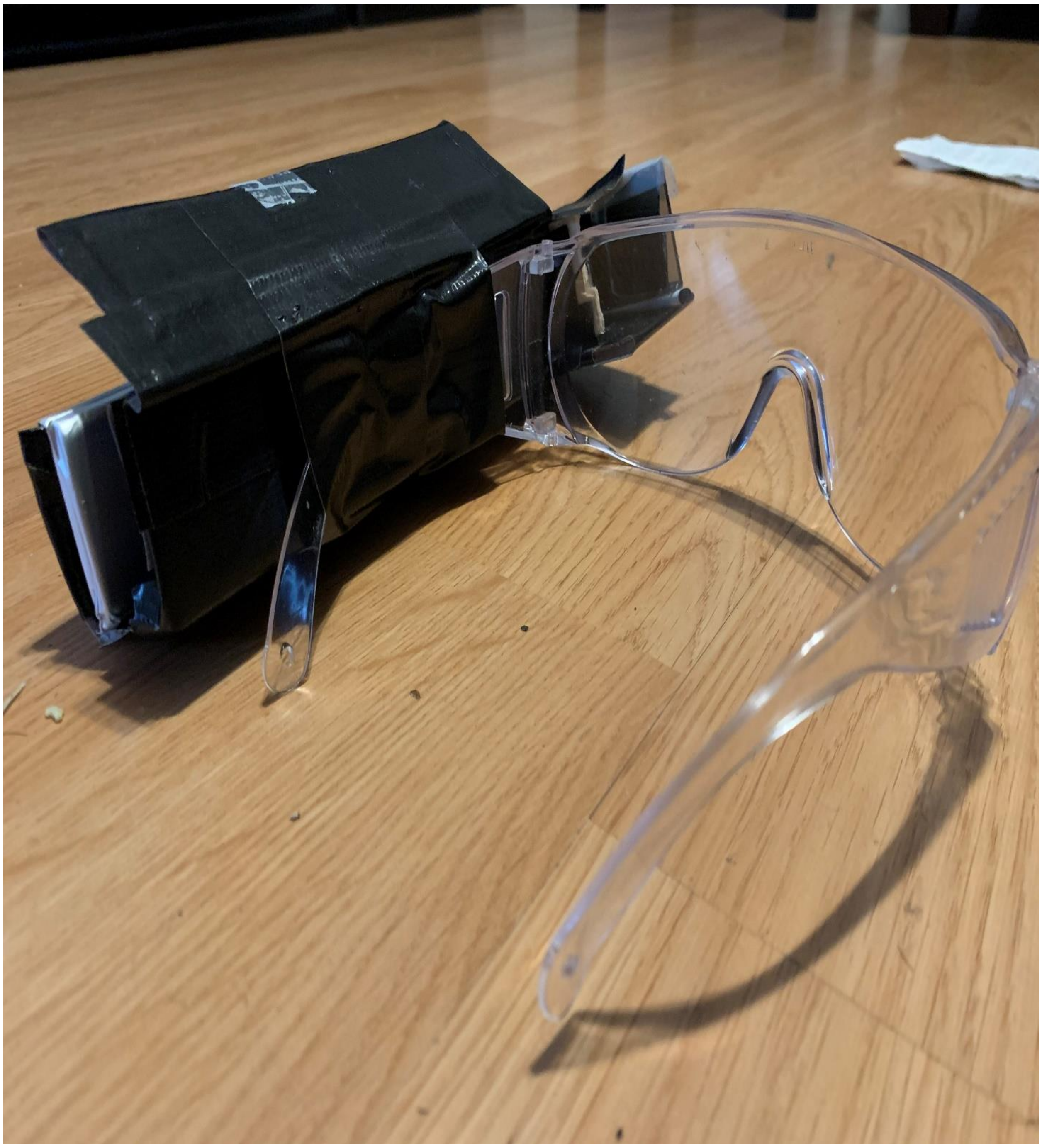
PDF files are also available on GitHub.

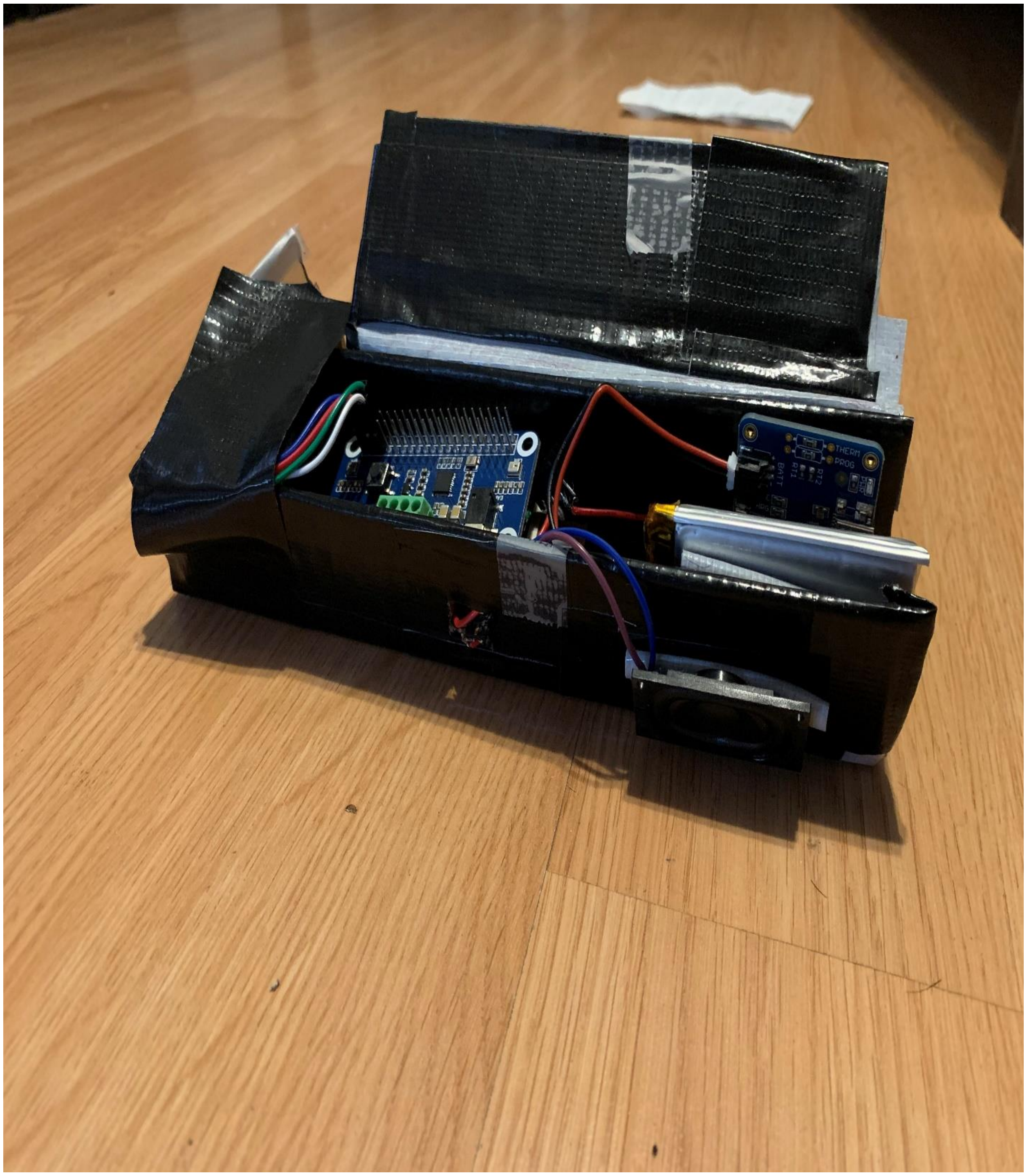
There is scratch code in this report you can modify according to your needs and specification.

Below are some Pictures we clicked during our project development


- PKCELL LP803860
+ 3.7V 2000mAh






THERM
PROG
BATT

REFERENCES

Bluetooth in C: https://people.csail.mit.edu/albert/bluez-intro/x502.html

OLED screen: https://www.winstar.com.tw/products/oled-module/graphic-oled-display/4-pin-oled.html

http://www.glassappsource.com/castar

http://www.brueckner.com/en/brueckner-servtec/services/remote-services/remote-service-tools/