Project Deliverable G: Final Prototype Report

Submitted by

[Night Call Bell Team]

[Zizheng Fan, 300161358]

[Yacine Diagne, 7902246]

Date: 31/03/21

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
RGB	Red, green and blue
LED	Light-emitting diode

1 Introduction

In this week's project work, we completed the final testing and improvement work. We used the voice file sent by the customer to adjust the voice keywords, which greatly improves the accuracy of recognition. At the same time, the design is further tested according to the professor's suggestions. Finally, we also held a group meeting and simulation for the upcoming design day pitch.

2 Final Prototype

2.1 Software Design

2.1.1 Voice Recognition Module

In order to realize the function of voice recognition, we decide to use Speech Recognition Module of YAHBOOM which is a Chinese company but you actually could buy its products in North America.

It has already integrated LD3320 chip which provides a speech recognition module based on its internal MCU. We can directly enter the recognition term and the corresponding serial number of the entry through I2C, set the mode of the module (loop detection, password trigger, key trigger), and obtain the identified result, which can be used directly without knowing the internal processing. And the module is also integrated with a buzzer and a RGB lamp which we can control its color and set the switch to identify the prompt sound through I2C.

Here, we will show you how our codes function in detail.

```
#! /usr/bin/python3


import os #command line in system shell
import smbus #transfer data through data bus
import time #control time


bus = smbus.SMBus(1) #set the format of databus

i2c_addr = 0x0f   #This is the address of voice_recognition module

asr_add_word_addr  = 0x01   #address where to add keywords

asr_mode_addr  = 0x02   #address where to set recognition mode, value from 0 to 2, default=0:circle recognition

asr_rgb_addr = 0x03   #address to set RGB LED,must be 2 bits, the first bit is 1: blue 2: red 3: green, the second bit is brightness from 0 to 255

asr_rec_gain_addr  = 0x04    #address where to set sensitivity，value from 0x40 to 0x55，default=0x40

asr_clear_addr = 0x05   #address where to clear cahce, before input new keywords you must clear the cache

asr_key_flag = 0x06  #address of the button, only used in button triggering mode.

asr_voice_flag = 0x07   #address where to set if we need an alarm when voice is recognized
```

```
asr_result = 0x08  #address where to store our results

asr_buzzer = 0x09  #address to trigger the buzzer, 1: open, 0:close

asr_num_cleck = 0x0a #address where to check the input keyword
```

we import three modules. “os” is for write command in shell. “smbus” is for controlling data bus. “time” is for counting time in this code. After finishing this, we can step into the address definition part. We refer to the official user manual and define the address of each register on the chip hardware to facilitate the next use. There are a lot of addresses that we didn't use later, but we wrote them down anyway.

```
def AsrAddWords(idnum,str):
    global i2c_addr
    global asr_add_word_addr
    words = []
    words.append(idnum)
    for  alond_word in str:
        words.append(ord(alond_word)) #convert the chip's voice-converted string into Unicode

    print(words)
    bus.write_i2c_block_data (i2c_addr,asr_add_word_addr,words)
    time.sleep(0.08)

def RGBSet(R,G,B):
    global i2c_addr
    global asr_rgb_addr
    date = []
    date.append(R)
    date.append(G)
    date.append(B)

    print(date)
    bus.write_i2c_block_data (i2c_addr,asr_rgb_addr,date)

def I2CReadByte(reg):
    global i2c_addr
    bus.write_byte (i2c_addr, reg)
    time.sleep(0.05)
    Read_result = bus.read_byte (i2c_addr)
    return Read_result
```

The first one is keyword_adding_function. It adds the entry sequence number and the keyword of the entry, this function writes the entry register address to be operated, and then write the phrase sequence number and the keyword string that identifies the phrase one byte after another where the append functions. And the ord function is to convert string format to Unicode to facilitate our comparison

The second one is to control the RGB color of LED

The last function is data_reading_function. This function firstly writes the register value to be read to the module, that is, what is read here is the detection result, so what is written is the address value of the result storage register, and then it reads the module to obtain the identified value. So that we could get a result to check if we detected the keyword. If the keyword has not been detected, the returned result will be default 255, otherwise, it will be the value you set.

```
if 0: #only set it as "1" when you input new or change keywords
    bus.write_byte_data(i2c_addr, asr_clear_addr, 0x40)#clear cache
    time.sleep(12) #it will cost at least 10s to clear the cache, so we just wait for finishing
    bus.write_byte_data(i2c_addr, asr_mode_addr, 0x00)
    time.sleep(0.1)
    #this is where you set your keywords
    AsrAddWords(1, "hey yeah hey yeah")
```

Now, it’s time to go into if_check. This part is to check if there are new keywords needed to be input in registers or old keywords changed. If there are, we should set the condition to 1, otherwise, set it to 0, so that we actually could skip this part in daily use.

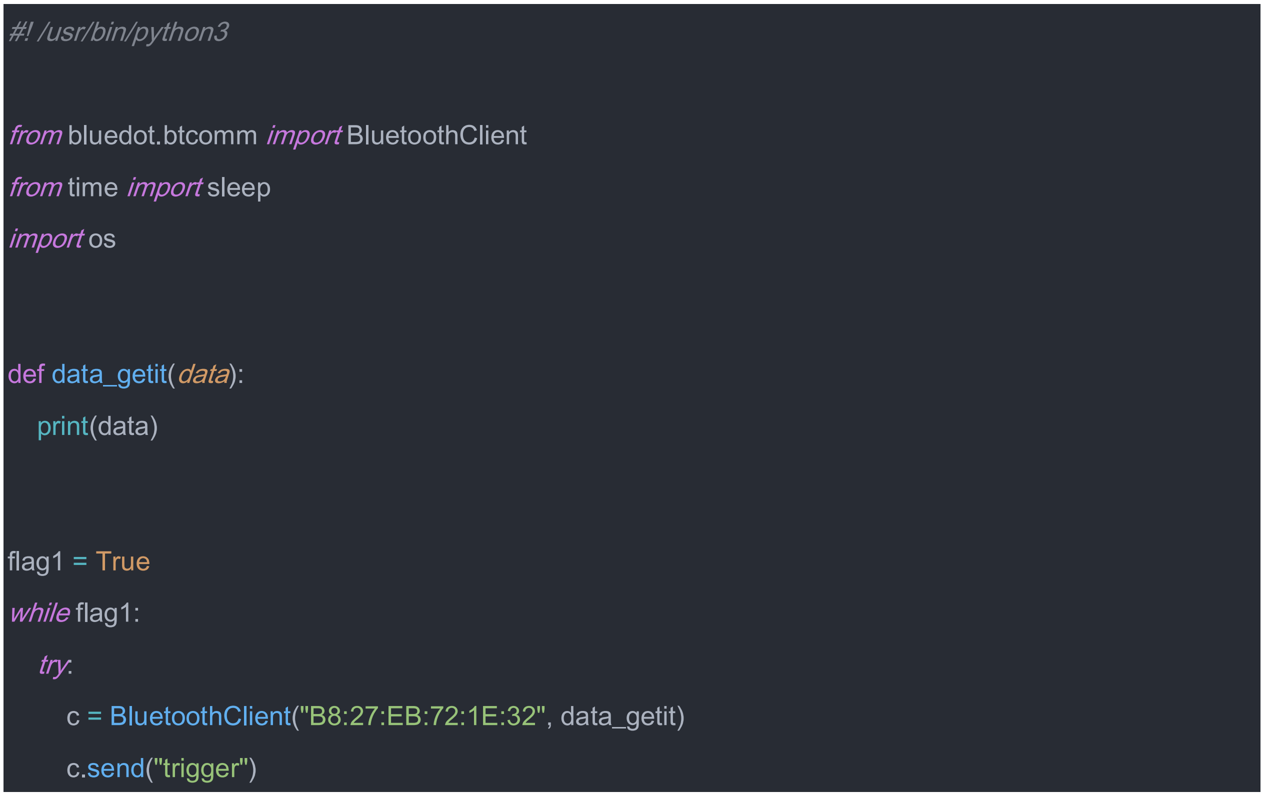
```
bus.write_byte_data(i2c_addr, asr_rec_gain_addr, 0x45) #set sensitivity
time.sleep(0.1)
bus.write_byte_data(i2c_addr, asr_voice_flag, 1) #set alarm
time.sleep(0.1)
RGBSet(100,100,100) #set RGB
time.sleep(2)
RGBSet(10,10,10)


while True: #this is the main loop of the code, constantly detecting and judging the voice string.
    result = I2CReadByte(asr_result)
    if(result != 255): #result != 255 means keywords are recognized
        print('triggered!')
        os.system("python3 client_test.py") #from system shell we call another py document to establish bluetooth connection
        time.sleep(1)
        RGBSet(10,10,10)
    time.sleep(0.5) #which means we detect the voice per 0.5s, this value would affect the accuracy of voice recognition because of the talking speed of speaker
```

Finally, this is the main loop which means it is the actually continuously running part of our code. While True means it will forever be running itself till the end of this world.

When the if notice the gotten result is not default 255 which means the chip just detected our keyword, it will call another py document which is used to establish Bluetooth connection and send alarm to portable unit.

2.1.2 Bell Unit Bluetooth Client Module

After multi-comparison, we finally choose bluedot as the module for Bluetooth data transmission.



```
#! /usr/bin/python3

from bluedot.btcomm import BluetoothClient
from time import sleep
import os


def data_getit(data):
    print(data)


flag1 = True
while flag1:
    try:
        c = BluetoothClient("B8:27:EB:72:1E:32", data_getit)
        c.send("trigger")
        flag1 = False
    except:
        print("miss!")
```

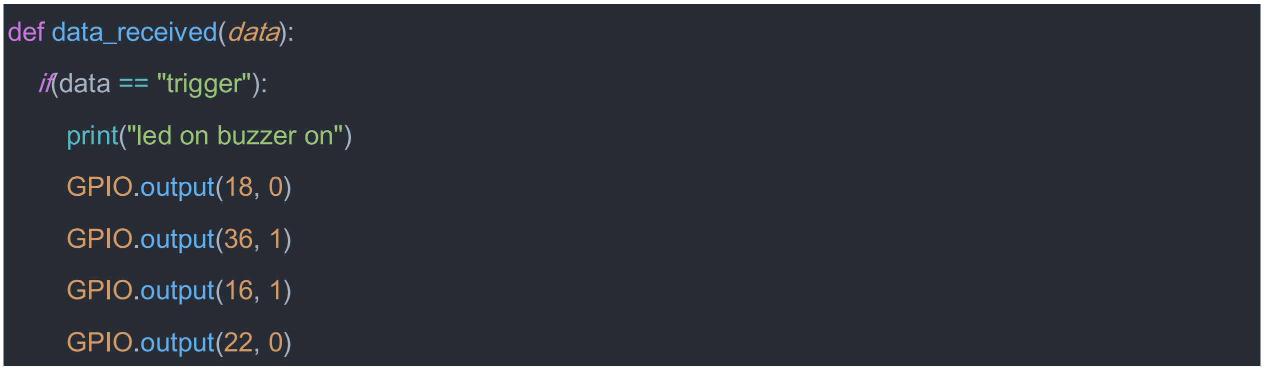
As you can see, when establishing Bluetooth communication, we use a while-try loop with flag to ensure the success of data transmission.

2.1.3 Portable Unit Bluetooth Server Module

Here, we determine to use RPi.GPIO module to control the GPIO of raspberry because they the best compatibility. Also, Bluetooth communication is realized by bluedot.

```
#! /usr/bin/python3

import RPi.GPIO as GPIO
from bluedot.btcomm import BluetoothServer
from signal import pause
from time import sleep


GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(16, GPIO.OUT) #buzzer
GPIO.setup(18, GPIO.OUT) #LED_green
GPIO.setup(22, GPIO.OUT) #LED_Yellow
GPIO.setup(36, GPIO.OUT) #LED_red
```

This part is to initialize GPIO pins, and tell the raspberry pi which pin is to be used.



```
def data_received(data):
  if(data == "trigger"):
    print("led on buzzer on")
    GPIO.output(18, 0)
    GPIO.output(36, 1)
    GPIO.output(16, 1)
    GPIO.output(22, 0)
```

Here, we defined a function which would be run when Bluetooth server has gotten the already set trigger word “trigger”. In this function, when portable has received the trigger signal from bell unit, it will turn off green led, and have the buzzer keep beeping, the red led keep shining until the button on shell is pressed.

2.2 Hardware Design

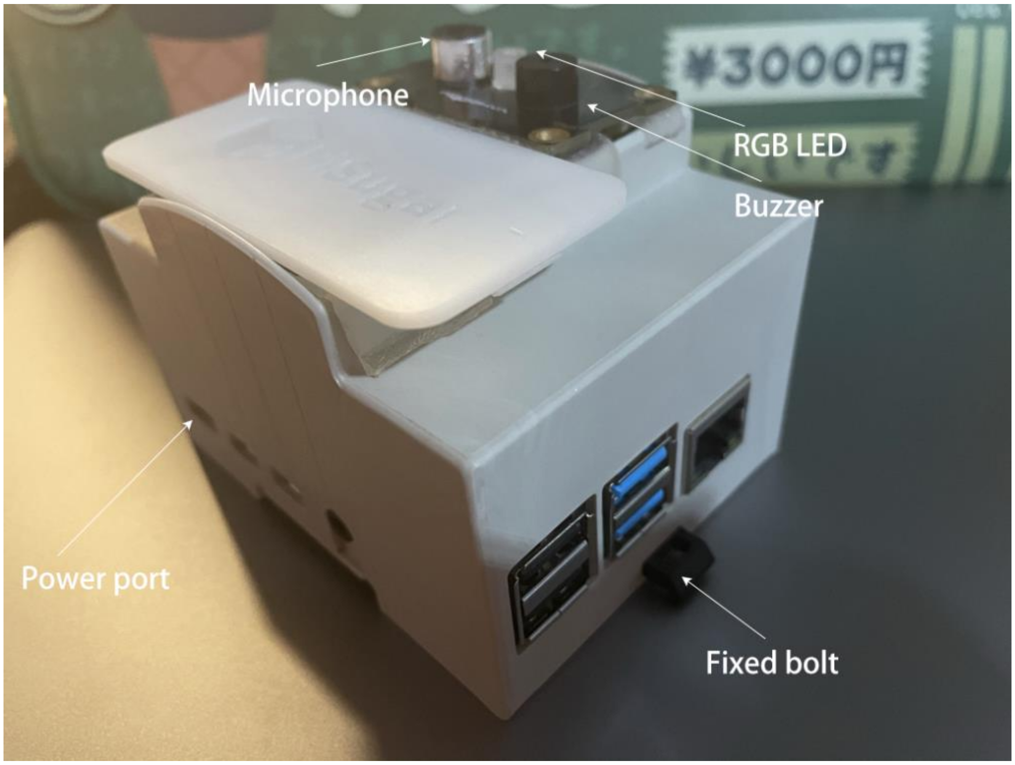


Figure1: Bell Unit

The ports or modules marked with white arrows here are the parts that users will use in our product.

It can be seen that the nano-magic tape has excellent performance in pasting. At the same time, we also use the accessories of the shell to seal the product, so as to ensure that the internal circuit of the product will not accumulate dust, but also can play some role in waterproof protection.

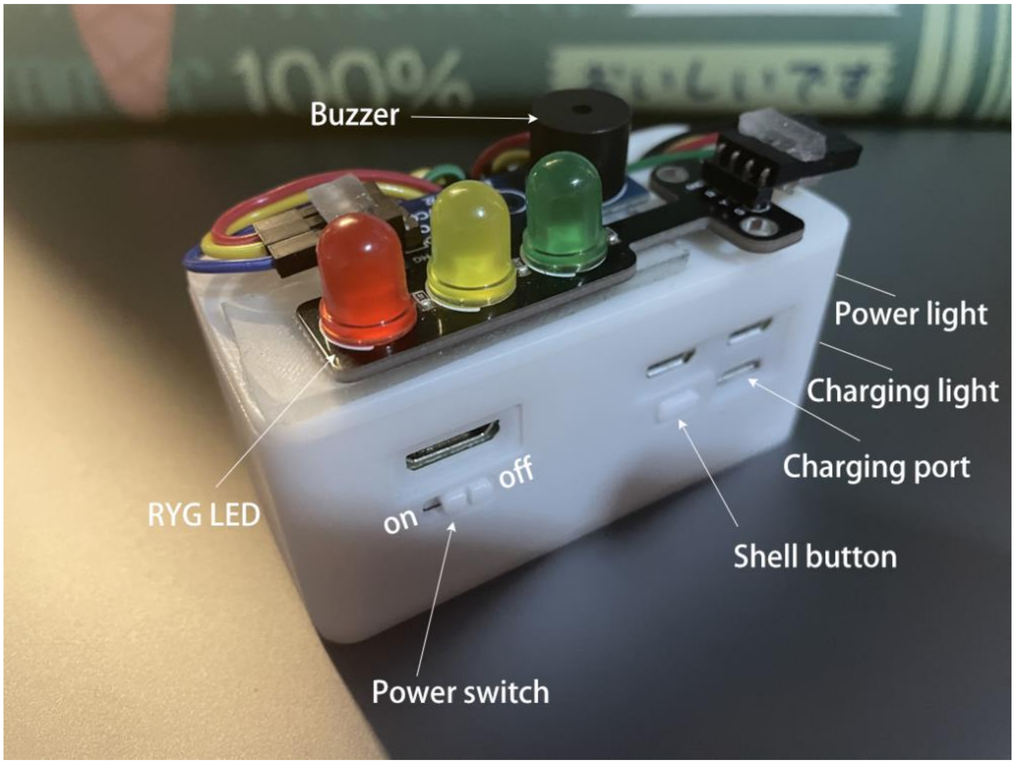


Figure2: Portable Unit

We have indeed tried various ways to put the LED and buzzer in the shell, but this seems impossible. Even if the wire is not taken into account, the LED is so big that we could not close the lid of its shell . And after the buzzer is placed in the shell, the volume of the beep is greatly reduced, and we are worried that it will not serve as a reminder and warning. Therefore, we decided to fix it outside the shell with nano-magic tape. Also, we fixed the exposed electronic devices and made it waterproof.

2.3 Scalability and Usability

Our final product still has strong functional scalability. We can connect the mechanical button at the GPIO interface and secure it to the top cover. In this way, it can be easily used by the elderly who are not physically disabled. Because the reliability of mechanical buttons is much higher than that of speech recognition, this will greatly improve the reliability of our products.

2.4 Sustainability

Our design adopts the shell scheme provided by PiSugar. The electronic devices are almost seamlessly connected to the shell and are fixed with screws and nuts to ensure that there will not be any loosening. After the final decision on the product design, we will use hot melt glue to completely seal the exposed parts of the device so as to achieve the function of waterproof and durable. At the same time, we will also glue and fix the wire with 502 glue to ensure its durability.

3 Design Day Pitch

Explain why the problem is relevant ("So What?")

There are many elder people with disability and they need help.

Explain the basic user requirements ("Who Cares?")

For those who could not move, voice activation is a better choice.

To facilitate the nurse, we need a light, small, wireless and portable device.

Explain the differentiation in your design or the key aspects that make your product better ("Why you?")

Voice triggered, robust wireless connection, faster response, portable and nice-looking hardware

Introduce the topic of the report

4 Further testing

	Testing	**Results**
1	Invite different people (another group idiom, my parents) to say key words 5 times each person and observe the success rate of speech recognition	The success rate is still maintained in a high number which is about 93.3%
2	Test of signal penetrating ability against the wall	When separated by a wall, the signal strength is still excellent. When two walls are separated, the signal strength is seriously affected. We don't expect the signal to pass through three walls, but this kind of use is also rare.

Table1: Further Testing

5 Conclusions and Recommendations for Future Work

After the design presentation, we tested and improved our products according to the professor's questions and suggestions. In the next few weeks, we will study how to expand the scalability of our products. At the same time, after the final version of the product design is decided, we will use hot melt glue and 502 glue to completely seal the exposed area of the product, greatly enhancing its durability, sustainability and water resistance.