Project Deliverable G: Prototype 2

GNG2101 [A03] – Professor Hanan Anis

Submitted by:

Team A2

Ayesha Khan, 300056179

Alessandro Furlano, 300116738

Aunonto Bhuiya, 300115942

Ethan Chan, 300006808

Dieudonne Lomamba, 300095212

Due Date: 09/11/20

University of Ottawa

Table of Contents

Introduction

Over the course of the past couple of weeks, Team 2 has been working on developing their second iteration of their device. This second prototype implements improvements discovered from an additional client meeting with Fran and her support staff (Justine). Similar to the previous prototype, the group was split into a software and a hardware team. Despite the problems with shipping due to COVID-19, the software team was able to develop some code based on the client meeting and needs.

Client Meeting

Client Meeting Preparation and Summary

In order to prepare for their third meeting with Fran and Justine (her support staff), Team 2 attempted to complete prototype 2 prior to the meeting. However, due to shipping delays brought on by COVID-19 they were unable to fully complete it. Instead, they presented their model to the client and asked about any potential concerns.

Client Meeting Feedback

From their meeting Team 2 discovered that due to Fran getting a new T.V, they may need to implement an external mic in their design to ensure that the audio received is as optimal as possible. The team were also informed of audio samples that had been sent to them for testing purposes. Although due to problems with shipping, they were unable to present the clients with a working device, Team 2 received integral information for their design.

Changes to Prototype

Based on their meeting, Team 2 will now be implementing an external mic to the main unit design. This microphone will allow the device to get clearer input from Fran, thus, making it more effective.

Prototype II

Hardware

Due to COVID-19, Team 2 was unable to complete their second physical prototype.

Software

Server and Audio Recognition Code

```
import socket
from pathlib import Path
from pydub import AudioSegment

def recognize(location = 'D:/Users/Ethan Chan/AppData/Local/Google/Cloud SDK/sf_test.flac'):
    r = sr.Recognizer()
    tempsplit = location.split('.')
    if len(tempsplit)!=2:
        raise Exception("filename doesnt not follow standard 1 location, 1 file")
    local = tempsplit[0]
    filetype = tempsplit[1]
    if filetype!= 'flac':
        location = translate(local,filetype)

    with sr.AudioFile(location) as source:
        audio = r.record(source)

    res = r.recognize_google(audio, language='en-US')
    print(res)
    return res
```

```
def translate(location,ftype):
    soundfile = location.split('/')
    sf = soundfile[-1] + '.flac'
    path = Path(location + '.flac')
    if path.is_file():
        ret = location + '.flac'
    else:
        s = AudioSegment.from_file(location+”.”+ftype,ftype)
        s.export(sf,format="flac")
        ret = location + '.flac'
    return ret


def send(MESSAGE = b"Hello, World!"):
    UDP_IP = "192.168.0.160" #127.0.0.1
    UDP_PORT = 5005
    print("UDP target IP: %s" % UDP_IP)
    print("UDP target port: %s" % UDP_PORT)
    print("message: %s" % MESSAGE)

    sock = socket.socket(socket.AF_INET, # Internet
                         socket.SOCK_DGRAM) # UDP
    sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))

def filteraudio(audio):
    alength = len(audio)
    ret = False
    for iter in range(alength-2):
        if audio[iter] == 'h':
            word = audio[iter:(iter+2)]
            if word == 'hey':
                ret = True
                break
            elif (word == 'hel')and(iter+2!=alength):
                if audio[iter+3]=='p':
                    ret = True
                    break
    return ret
```

Shown above is the code used for the voice recognition software, along with helper functions.

Recognize: Takes a file location and returns the recognized words in english. It uses the translation function if necessary

Translate: Takes the file location, and the files data type. If the file is not of the file type “flac” then the file is translated into that type. The location of the new file is returned.

Send: Takes a binary message and returns nothing (OK or true will be added later). Connection is established and the message is sent. The connection will later be split from this function for better functionality.

Filteraudio: Takes an audio string and returns a boolean. Used to check the words recognized with the designated words used by the client. If the word “hey” or “help” is within the string, then true is returned.

An example of using this code would be (without using the filteraudio):

```
message = recognize()
bmsg = str.encode(message)
send(bmsg)
```

Testing of the code:

4 audio files were provided by the client: “Bedside with noise machine.m4a”, “By bedside with TV on Vol level 15.m4a”, “By bedside.m4a”, and “By door.m4a”.

The filteraudio function was tested with these 4 files and no difference was found between the resulting flac files compared to flac files created through website usage.

The recognize function was then used with the following results:

Bedside with noise machine	go ahead and say hey

By bedside with TV on Vol level 15	go ahead and say hey
By bedside	say help
By door	okay it's a help

As shown above, the caretaker's voice was recognized while the client's was not. Testing with previously scrapped voice recognition software ended with the same result. However, listening to the audio itself, the client's voice was not clear enough to be heard by multiple members of the group that tried listening to them. A request has been sent for higher quality voice recordings and adjustments have been made to the project accordingly. At this point, it seems almost certain that the main device will require an extendable microphone as previously discussed.

Portable Unit Code

```
from gpiozero import Button
from gpiozero import RGBLED
from gpiozero import LED
from time import sleep
from gpiozero import Motor

# Call Connection Function
led = RGBLED(red=9, green=10, blue=11)
motor = Motor(forward=4, backward=14)
led.color = LED(18)
okBut = Button(4)
motor = Motor(17)
safecount = 0
audiocount = 0
situation1 = 0
situation2 = 0
situation3 = 0
```

```
def connection(self,  mainunitconc=None):
  if (mainunitconc == True):

  # server(True):
  else:
     connectioncount = 0
  led.red.on()
  sleep(1)
  led.color = (0, 0, 0)
  sleep(1)
  connectioncount += 1

  if (connectioncount > 180):
     motor.forward()
     sleep(5)
     motor.backward()
     sleep(5)


connection()


def server():
  situation1 = False
  while situation1 == False:
     signal = False
     if (signal == True):
        motor.forward()
        sleep(5)
        motor.backward()
        sleep(5)
        led.color = (0, 1, 0)
        sleep(1)
        led.color = (0, 0, 0)
        sleep(1)
     else:
        Situation1 = False
```

```
server()


def confirm():
  situation3 == False
  while situation3 == False:
    if okBut.is_pressed:
      oksignal = str.encode("ok signal")
      # send(oksignal)
  else:
    # neglect()
    Situation3 = False


confirm()


def neglect():
  while situation2 == False:

    if (recieve is "ok signal" and "Stop Signal" == False):
      safecount = safecount + 1
      audiocount = safecount / 3
      if (safecount > 3 and audiocount < 3):
        motor.forward()
        sleep(5)
        motor.backward()
        sleep(5)


      else:
      # Playsound insert audio queue code

    elif (recieve is "ok signal" == False and "Stop Signal" == True):
      safecount = safecount + 1
      audiocount = safecount / 3
      if (safecount > 3 and audiocount < 3):
        motor.forward()
        sleep(5)
        motor.backward()
```

```
            sleep(5)

        else:
        # Playsound insert audio queue code

    elif (recieve is "ok signal" == True and "Stop Signal" == False):
        safecount = safecount + 1
        audiocount = safecount / 3
        if (safecount > 3 and audiocount < 3):
            motor.forward()
            sleep(5)
            motor.backward()
            sleep(5)

        else:
                #Playsound insert audio queue code

    elif (recieve is "ok signal" and "Stop Signal" == True):

        motor.stop()
        # turn off sound queue sound
        led.off()
        Situation2 = True

        connection()


neglect()
```

The code for the portable unit can be seen above. The code is able to run the motor, LEDS, and buttons with different cases in the code. The leds and motor are able to be used in different functions . At the moment, no sound notification has been implemented into the device. Since in the current circumstances with regards to COVID-19, the components needed to effectively test the code above had not arrived in time. The code is able to run without any syntax errors in play but, without the proper components needed to test the code some errors might be seen later on.

Main Unit Code

```
from gpiozero import Button
from gpiozero import LED
from gpiozero import RGBLED

hereBut = Button(17)
powerBut = Button(22)
resetBut = Button(18)

largeLED = LED(18)
largeLED.red = 1
Flag4 = False
Flag3 = False


# insert code for server
# insert code for audio recognition


def Confirm():
  while Flag3 == False:
    if recieve is "ok signal":
      largeLED.red.on()
      Arrived()
    else:
      Flag3 = False


def Arrived():
  while Flag4 == False:
    if hereBut.is_pressed:
      largeLED.off()
      signal = str.encode("Stop Signal")
      send(signal)
      recognize()
    else:
      Flag4 = False
```

The code for the 2 computational functions in the main unit can be seen above. This code is used to assign functionality to the various buttons and LEDs in the main unit. Due to the current circumstances with regards to COVID-19, the components needed to effectively test the code above had not arrived in time. The code has been debugged for any syntax errors, however, at this time the team is unable to test the code. Currently, the code is designed to work for a button and an RGB LED that only sends a signal and indicates that a signal has been sent.

Project Plan

The project plan was updated in order to include a detailed schedule of the next two weeks. Deliverable I was added to the project plan as it was not there before. The Hardware tasks for prototype 2 were not completed as the components were not received in time.

Conclusion

Testing based on the audio provided by the client went poorly. However, steps have been taken to remedy this. Despite the setbacks, the software parts tested have provided promising results and will continue to be developed. As the team is still waiting on parts to be delivered, hardware objectives have been slowed but progress has been made on this as well. Thus, the project plan has been adjusted accordingly, and further testing has been planned upon other prototype objective completions.