# Final Project Presentation

Presented by: GNG5140 Night Call Bell Team
Zizheng Fan
Yacine Diagne

## Catalog

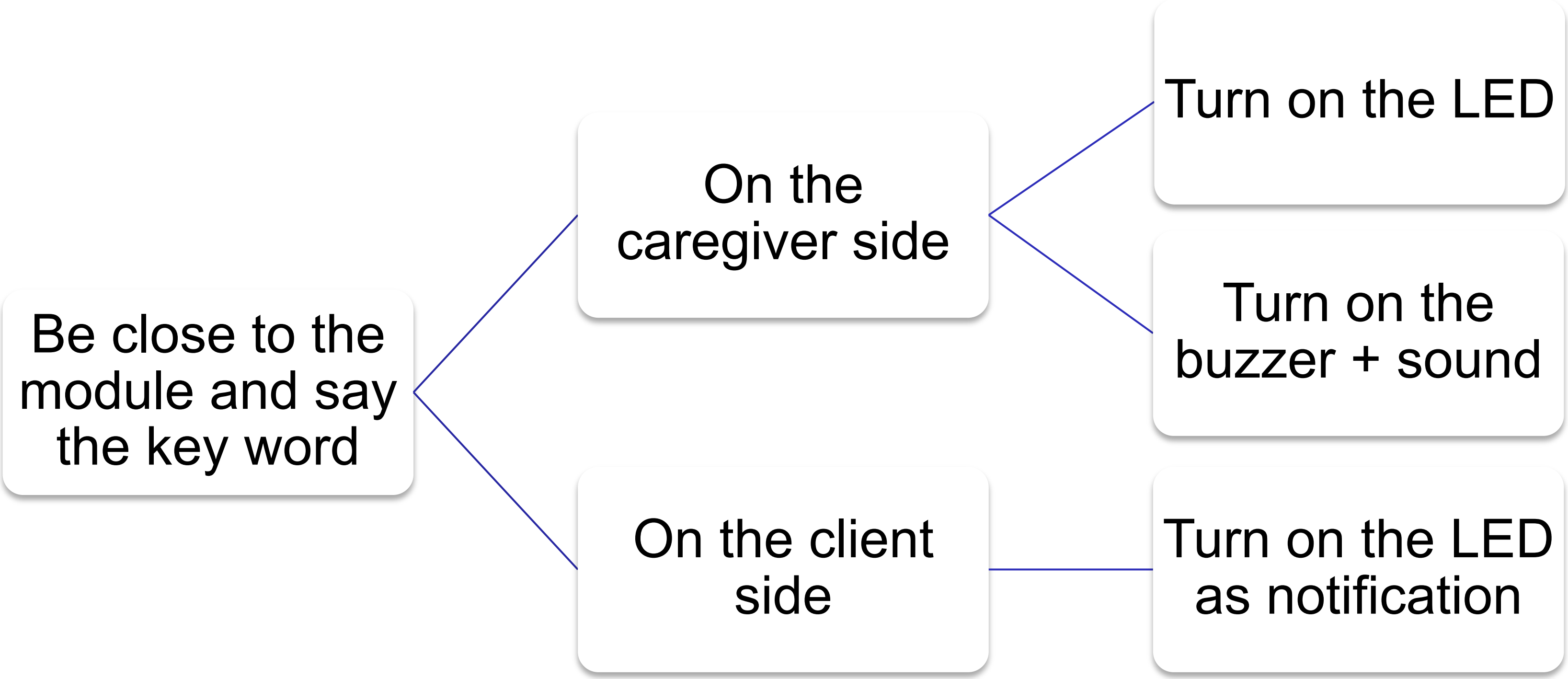
- Basic information
- Choice
- Program
- Revised prototype

# Project Overview

The Night Call Bell is a device that helps a person to communicate with another person. By using a sound, the device sends a notification to the other person. This device uses:

- a voice recognition module
- two rasberry pies
- LED + Buzzer

# Project Overview



Be close to the module and say the key word

On the caregiver side

Turn on the LED

Turn on the buzzer + sound

On the client side

Turn on the LED as notification

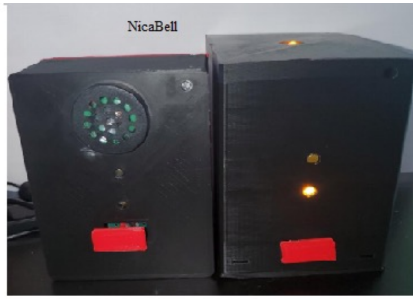
# Benchmarking

The Medical guardian

1

The mobile help

2



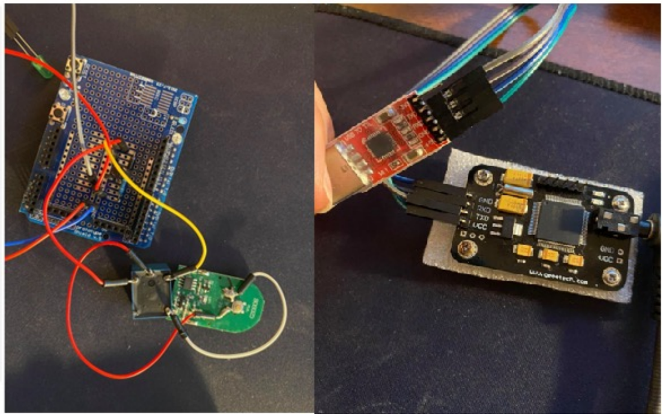
NicaBell

Call
For
Care

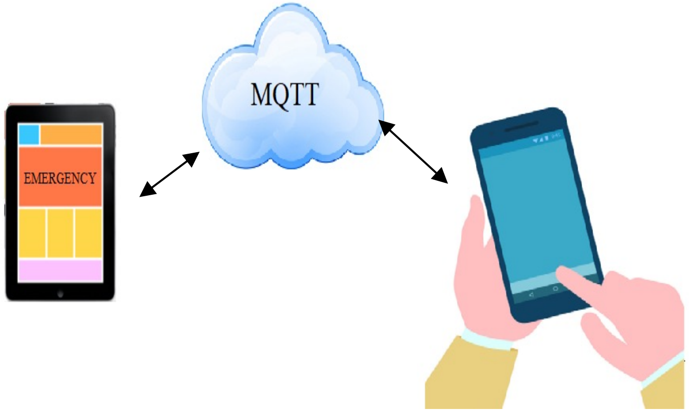
The life station

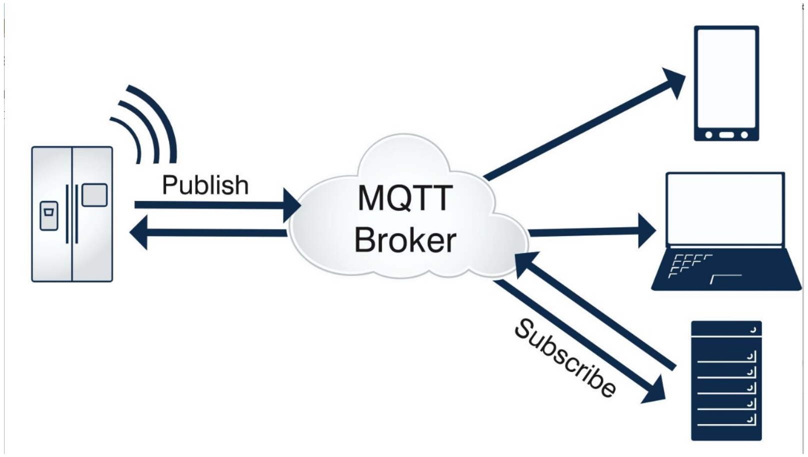
3

SmartTech



# Initial Idea





# First Client meeting and initial solution

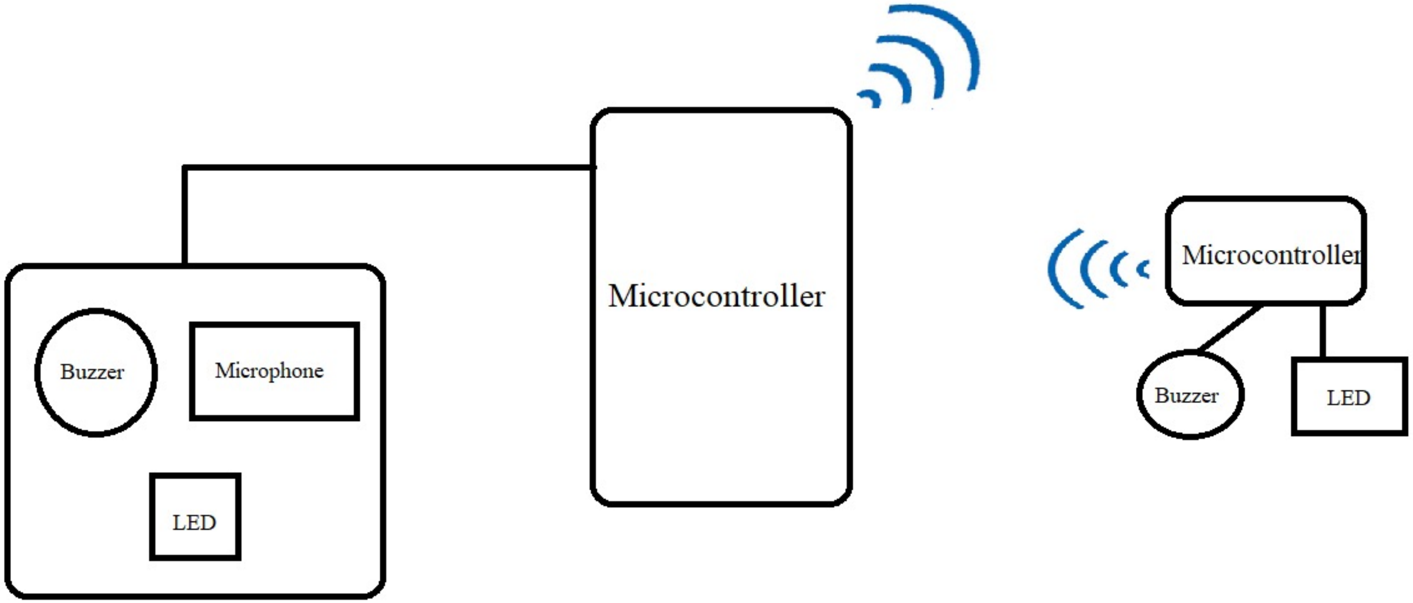
- She has difficulty moving her arm
- She is enable to project her voice

- Use Bluetooth instead of network
- Use local module or chip instead of API
- Use socket on main device and rechargeable Li-battery on receiver

# Requirement specification

| Needs |
|---|
| The device can quickly identify the voice of our client. |
| The device's ability to recognize sound will not be affected by background noise. |
| The device can quickly send alarm information to the staff. |
| It is best that the device can run without a network. |
| The operating device does not need to be borrowed through other devices such as mobile phones and computers. |
| The size of the transmitter is suitable for fixing on the table, and the size of the receiver is suitable for keeping in the pocket. |
| The device uses fixed power sources and sockets to provide power |
| The device has a good plastic package, preferably waterproof |

# Bloc Diagram



# Hardware choice

| | Raspberry pi | Arduino | OPI |
|---|---|---|---|
| Money Cost | 2 | 3 | 4 |
| Learning Cost | 4 | 2 | 3 |
| Functions | 5 | 5 | 3 |
| Community Support | 5 | 4 | 2 |
| **TOTAL** | **16** > | **14** > | **12** |

Criteria: From 1 to 5

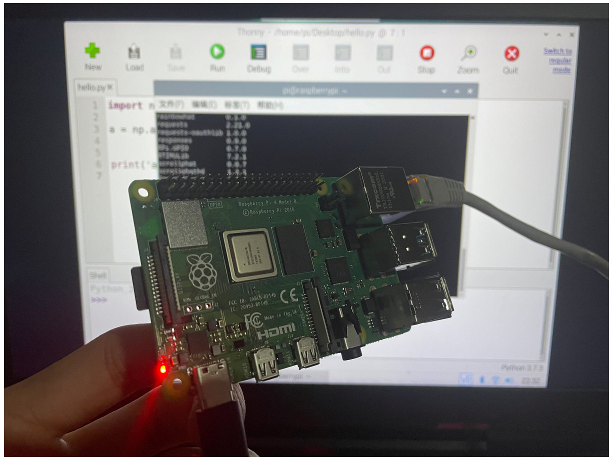
# Hardware Setting up

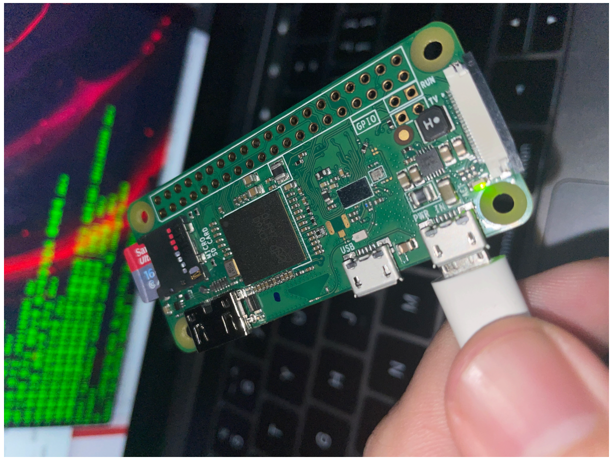
Bell Unit:

- Raspberry pi 4b 2G RAM
- 16G ROM
- Type-c Socket power

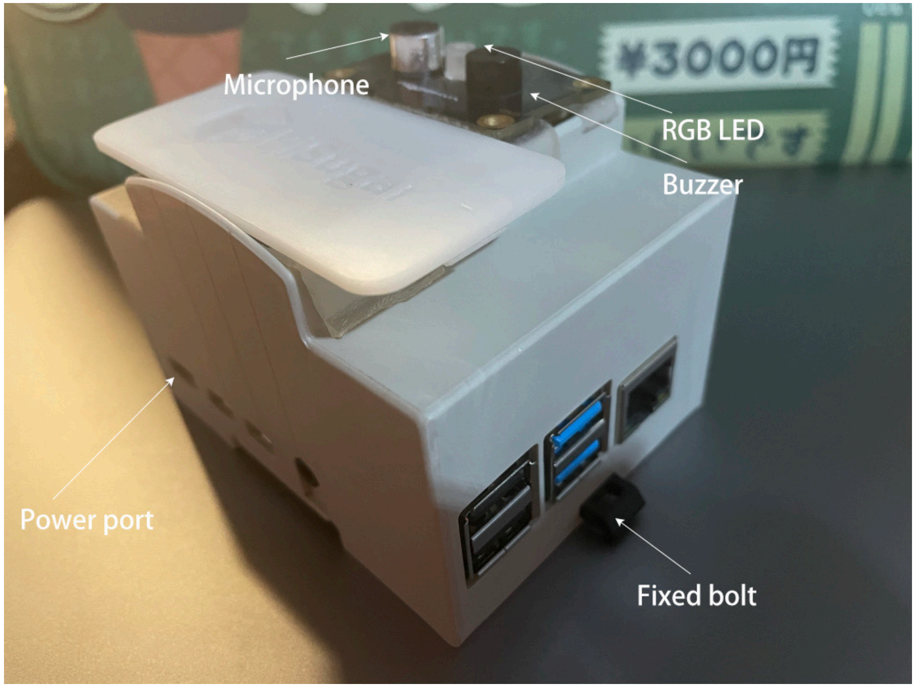
Portable Unit:

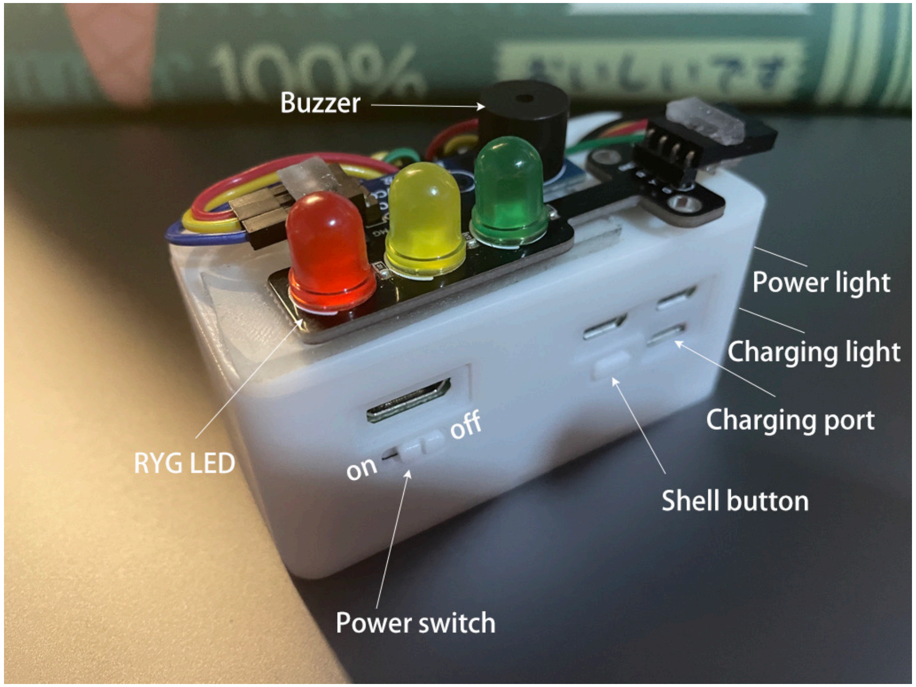
- Raspberry pi zerow 512M RAM
- 16G ROM
- I2C battery power





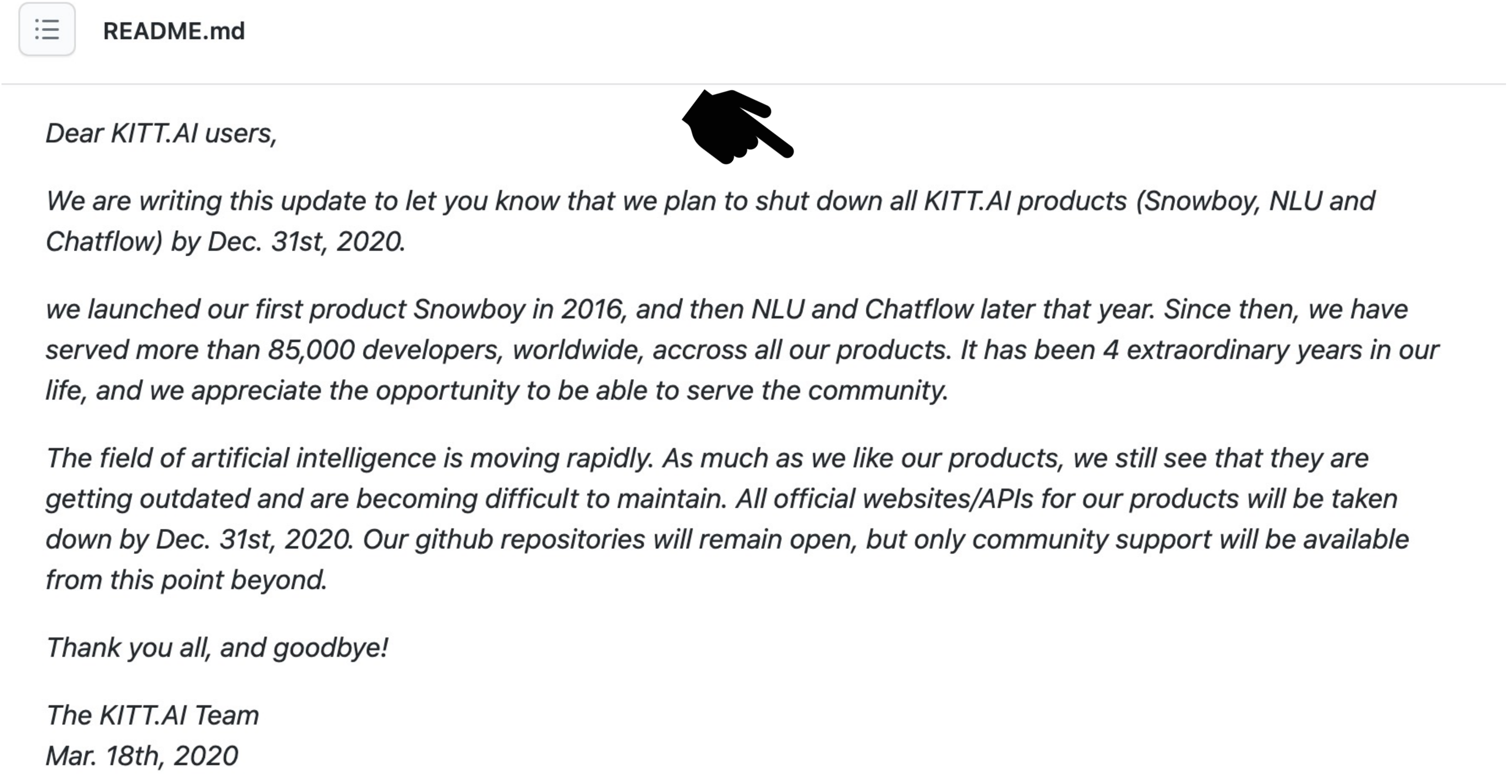
# Hardware Setting up





# Voice Recognition Choice

- Respeaker+snowboy



*Dear KITT.AI users,*

*We are writing this update to let you know that we plan to shut down all KITT.AI products (Snowboy, NLU and Chatflow) by Dec. 31st, 2020.*

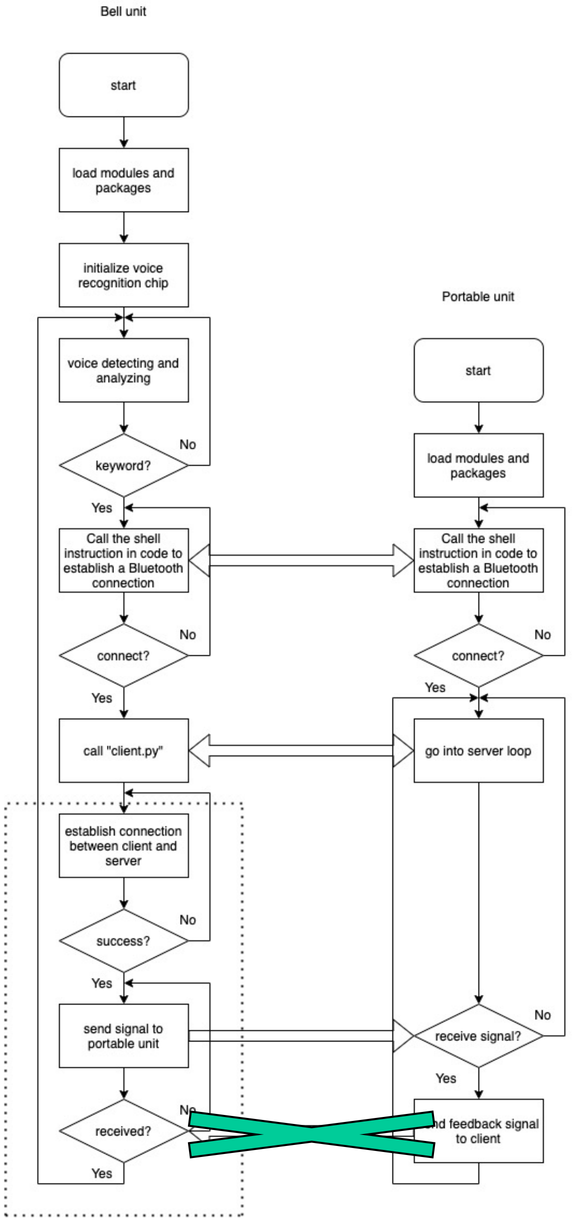
*we launched our first product Snowboy in 2016, and then NLU and Chatflow later that year. Since then, we have served more than 85,000 developers, worldwide, accross all our products. It has been 4 extraordinary years in our life, and we appreciate the opportunity to be able to serve the community.*

*The field of artificial intelligence is moving rapidly. As much as we like our products, we still see that they are getting outdated and are becoming difficult to maintain. All official websites/APIs for our products will be taken down by Dec. 31st, 2020. Our github repositories will remain open, but only community support will be available from this point beyond.*

*Thank you all, and goodbye!*

*The KITT.AI Team*
*Mar. 18th, 2020*

# Flowchart



Bell unit

start

load modules and packages

initialize voice recognition chip

voice detecting and analyzing

keyword?

No

Yes

Call the shell instruction in code to establish a Bluetooth connection

connect?

No

Yes

call "client.py"

establish connection between client and server

success?

No

Yes

send signal to portable unit

received?

No

Yes

Portable unit

start

load modules and packages

Call the shell instruction in code to establish a Bluetooth connection

connect?

No

Yes

go into server loop

receive signal?

No

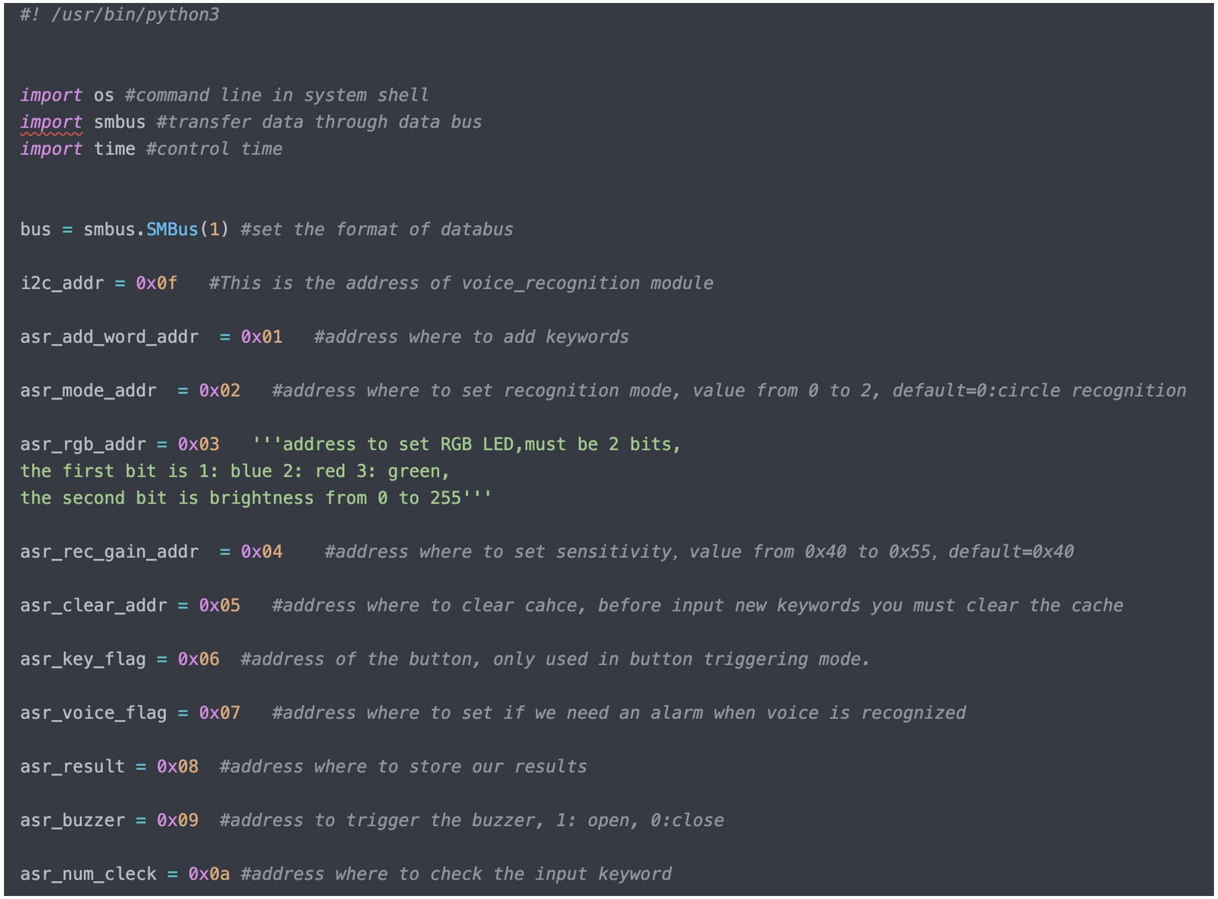
Yes

nd feedback signal to client

client.py file is included in the dotted frame

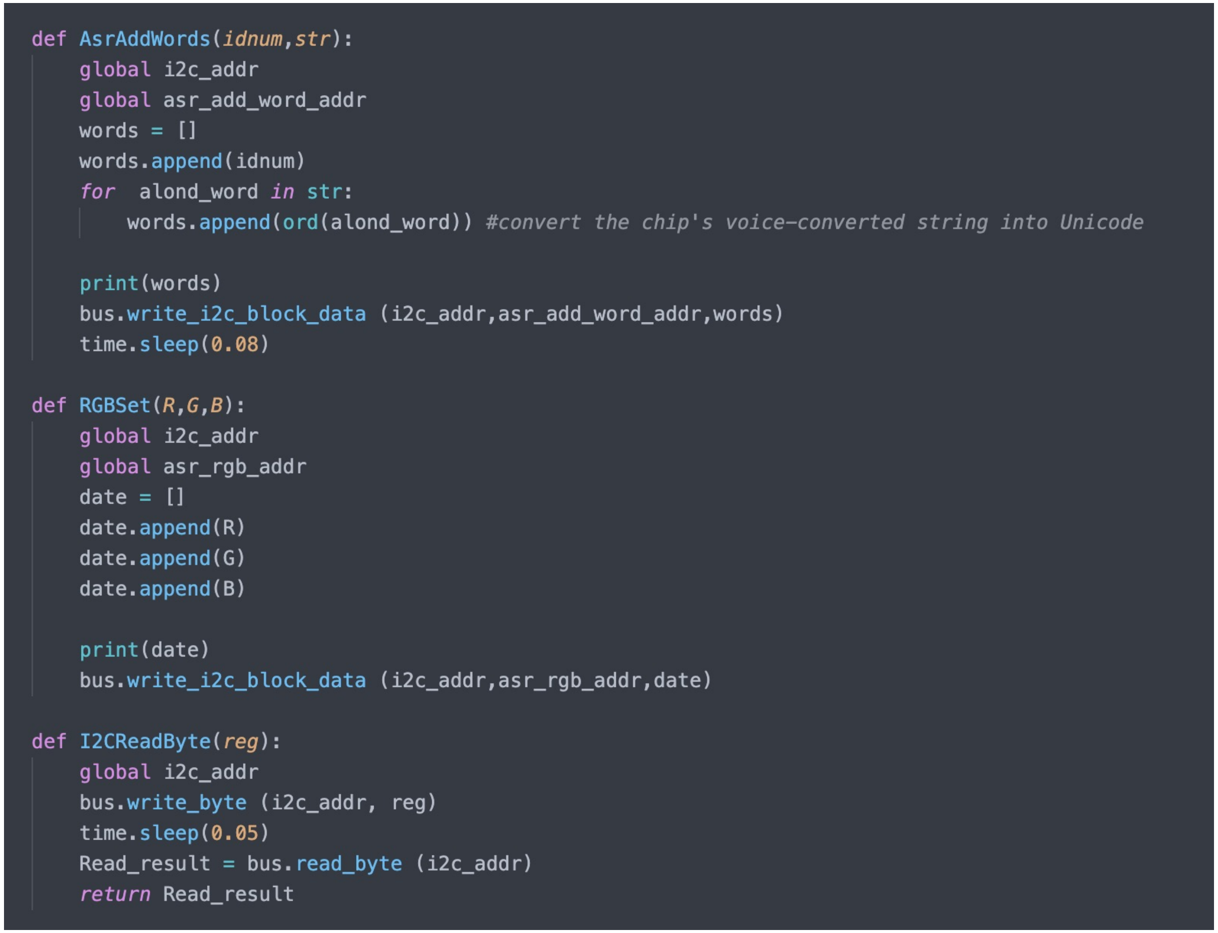
# Voice Recognition Code

| initialization | Function definition | If check | Main loop |
|---|---|---|---|
| Modules<br>Address | Keyword analysis<br>RGB control<br>Data control | Keywords | Continuously<br>Detecting |



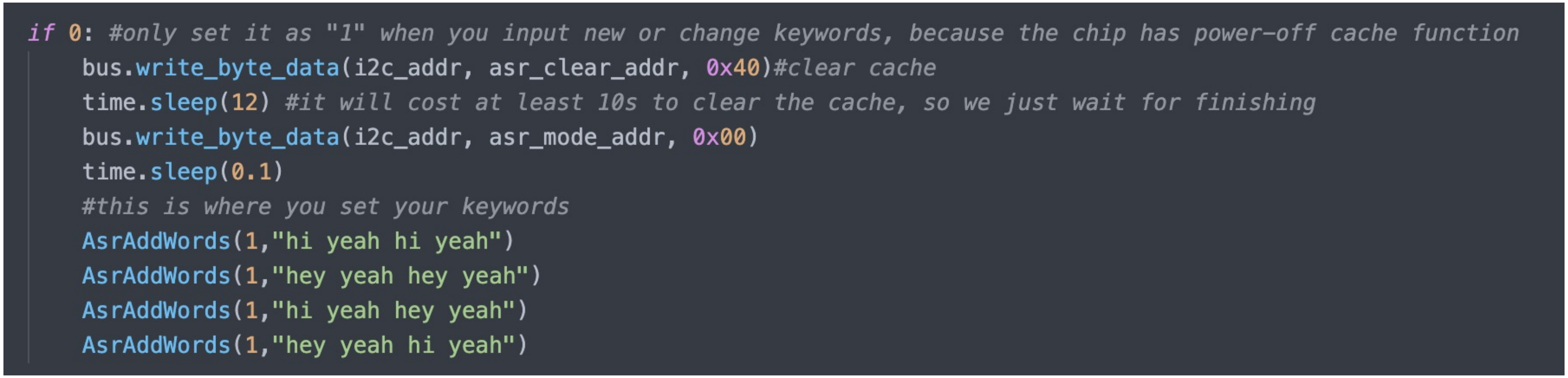
```
#! /usr/bin/python3


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
```

Define the device address and register address of the module for the following reference

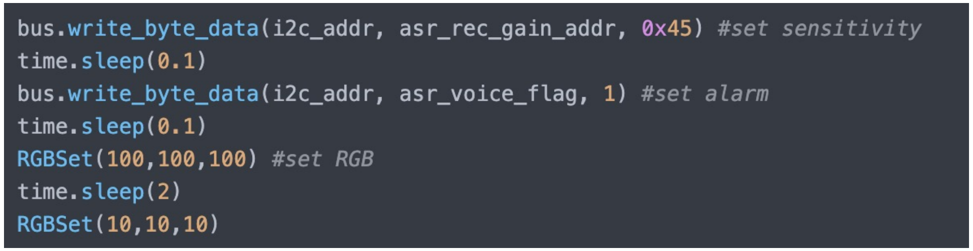


```
def AsrAddWords(idnum,str):
    global i2c_addr
    global asr_add_word_addr
    words = []
    words.append(idnum)
    for  alond_word in str:
        words.append(ord(alond_word)) #convert the chip's voice-converted string into Unicode

    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
```

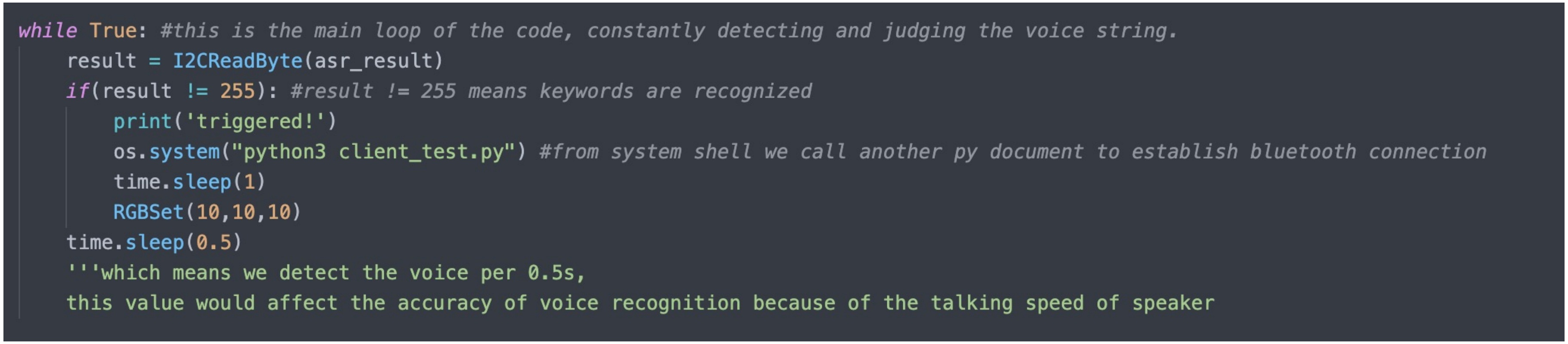
- Add and convert keyword
- Set RGB color
- Read data from chip



```
if 0: #only set it as "1" when you input new or change keywords, because the chip has power-off cache function
    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,"hi yeah hi yeah")
    AsrAddWords(1,"hey yeah hey yeah")
    AsrAddWords(1,"hi yeah hey yeah")
    AsrAddWords(1,"hey yeah hi yeah")
```

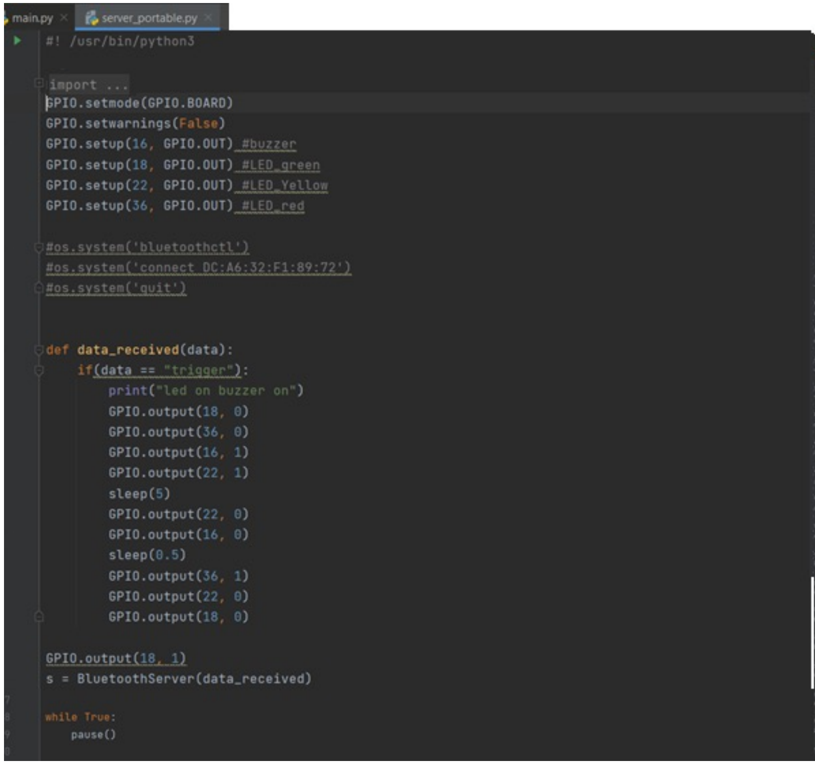


```python
bus.write_byte_data(i2c_addr, asr_rec_gain_addr, 0x45) #set sensitivity
time.sleep(0.1)
bus.write_byte_data(i2c_addr, asr_voice_flag, 1) #set alarm
time.sleep(0.1)
RGBSet(100,100,100) #set RGB
time.sleep(2)
RGBSet(10,10,10)
```

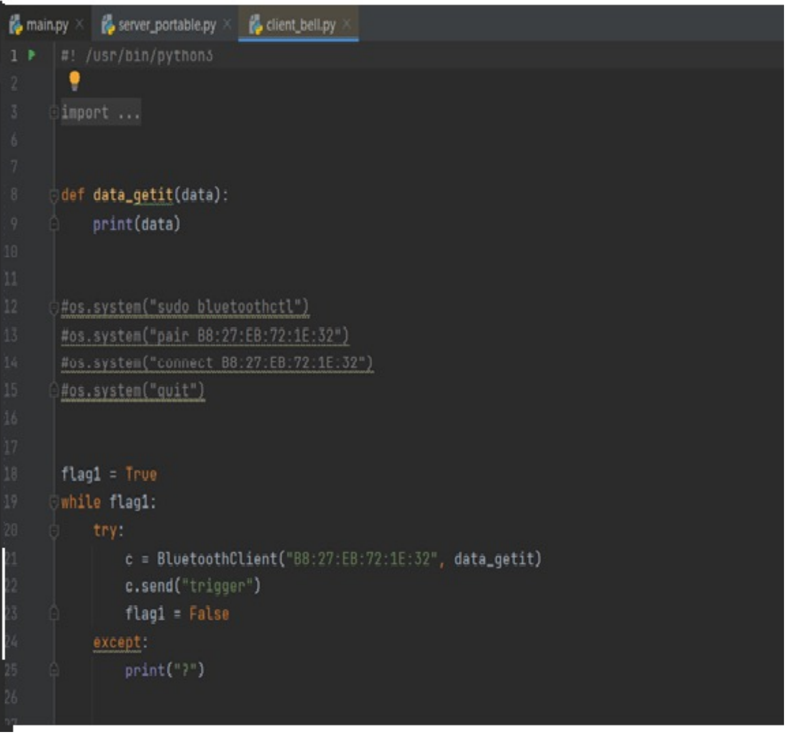


```python
while True: #this is the main loop of the code, constantly detecting and judging the voice string.
    result = I2CReadByte(asr_result)
    if(result != 255): #result != 255 means keywords are recognized
        print('triggered!')
        os.system("python3 client_test.py") #from system shell 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
```

# Bluetooth connections code



```
#! /usr/bin/python3

import ...
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

#os.system('bluetoothctl')
#os.system('connect DC:A6:32:F1:89:72')
#os.system('quit')


def data_received(data):
    if(data == "trigger"):
        print("led on buzzer on")
        GPIO.output(18, 0)
        GPIO.output(36, 0)
        GPIO.output(16, 1)
        GPIO.output(22, 1)
        sleep(5)
        GPIO.output(22, 0)
        GPIO.output(16, 0)
        sleep(0.5)
        GPIO.output(36, 1)
        GPIO.output(22, 0)
        GPIO.output(18, 0)

GPIO.output(18, 1)
s = BluetoothServer(data_received)

while True:
    pause()
```



```
#! /usr/bin/python3

import ...


def data_getit(data):
    print(data)


#os.system("sudo bluetoothctl")
#os.system("pair B8:27:EB:72:1E:32")
#os.system("connect B8:27:EB:72:1E:32")
#os.system("quit")


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

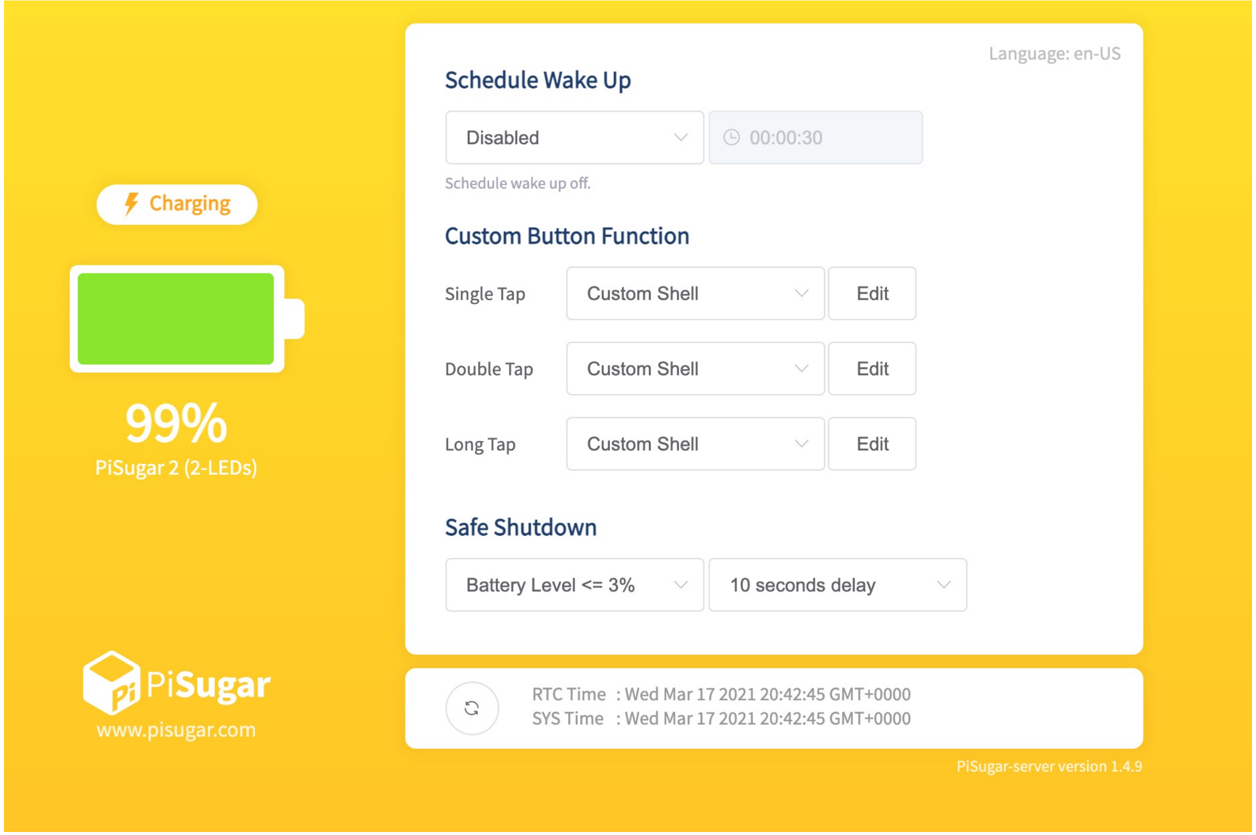
# Problem with the first prototype

- To the portable unit, it takes one minute from turning on the power switch to self-starting the program. And this time will be thirty seconds for the bell unit.
- If the two devices are not connected for a long time, they will automatically **cancel the pairing**, so when the device is started again, they cannot automatically connect via Bluetooth.

# The Second Client Meeting

- Full presentation
- Keyword discussion
- Alarming Mechanism
- Resetting Mechanism

# Button Functioning Code



Language: en-US

**Schedule Wake Up**

Disabled | 00:00:30

Schedule wake up off.

**Custom Button Function**

Single Tap: Custom Shell

Double Tap: Custom Shell

Long Tap: Custom Shell

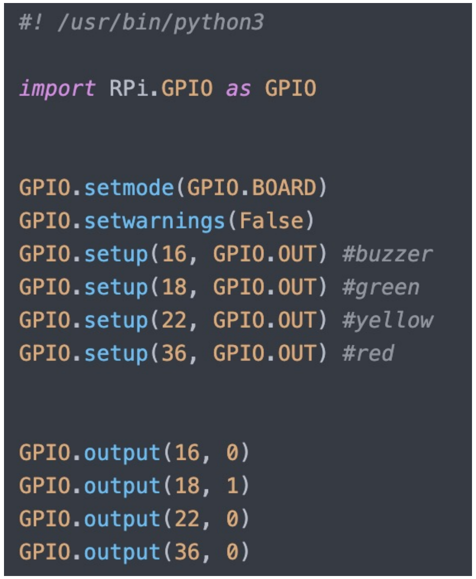
**Safe Shutdown**

Battery Level <= 3% | 10 seconds delay

RTC Time : Wed Mar 17 2021 20:42:45 GMT+0000
SYS Time : Wed Mar 17 2021 20:42:45 GMT+0000

PiSugar-server version 1.4.9

# Button Functioning Code



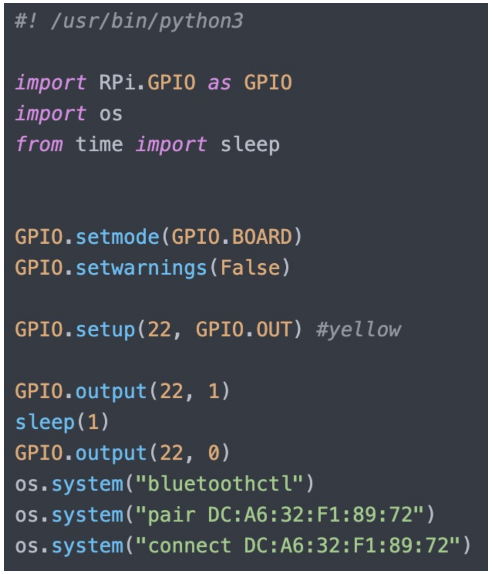
```python
#! /usr/bin/python3

import RPi.GPIO as GPIO


GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(16, GPIO.OUT) #buzzer
GPIO.setup(18, GPIO.OUT) #green
GPIO.setup(22, GPIO.OUT) #yellow
GPIO.setup(36, GPIO.OUT) #red


GPIO.output(16, 0)
GPIO.output(18, 1)
GPIO.output(22, 0)
GPIO.output(36, 0)
```

## One-tap

- Turn off the buzzer
- Turn the portable unit into initialized status

## Long-tap

- Manually connect two devices through Bluetooth



```python
#! /usr/bin/python3

import RPi.GPIO as GPIO
import os
from time import sleep


GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

GPIO.setup(22, GPIO.OUT) #yellow

GPIO.output(22, 1)
sleep(1)
GPIO.output(22, 0)
os.system("bluetoothctl")
os.system("pair DC:A6:32:F1:89:72")
os.system("connect DC:A6:32:F1:89:72")
```

# Demo Presentation

# Thank You