电子系统设计-2021秋

**会议信息自动提醒系统**

**戴增辉、尤恺聪、刘子驾、王力人**

目录

[一、 组员 2](#_Toc91272747)

[二、 项目背景 2](#_Toc91272748)

[三、 解决的方案 3](#_Toc91272749)

[3.1 整体框架 3](#_Toc91272750)

[3.2 软件端 4](#_Toc91272751)

[3.3 硬件端 6](#_Toc91272752)

[四、 分工 8](#_Toc91272753)

[五、 总结 9](#_Toc91272754)

[六、 成员感想 10](#_Toc91272755)

[七、 附录：源代码 11](#_Toc91272756)

[App.py 11](#_Toc91272757)

[WindowCapture.py 15](#_Toc91272758)

[Bluetooth.ino 16](#_Toc91272759)

# 一、 组员

1. 戴增辉 2018080094 无83
2. 尤恺聪 2018080076 无81
3. 刘子驾 2018080084 无88
4. 王力人 2018080086 无81

# 二、 项目背景

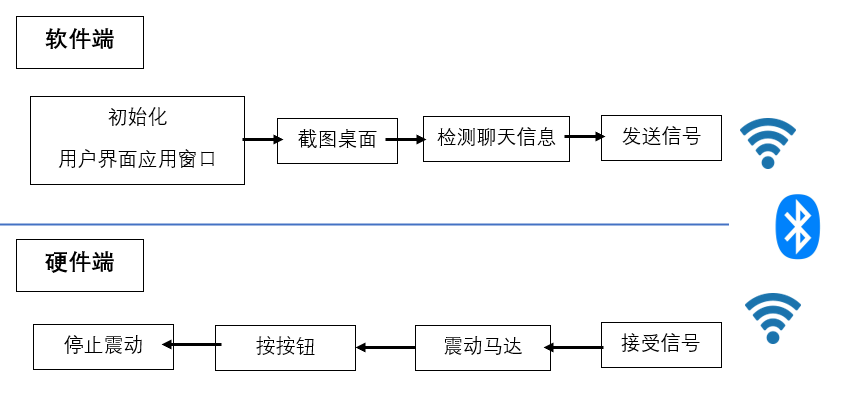
2020年开始，新冠疫情席卷全球，由于这一疫情持续时间长的特性，为了保证日常生活的持续，人们对会议软件，线上通讯系统和设备的需求不断增长。尽管各式各样的会议软件层出不穷，但他们的普遍功能集中于电脑或手机单一客户端层面，人机交互能力较为薄弱，并且时时必须受到手机或电脑设备的约束，导致使用场景毕竟有限。如：教学课堂中老师需要离开设备四处走动的场景，商业会议中需要离开设备进行演示的场景，产品发布会，公开演示等大型直播场景。以上场景中，如果单一依靠手机或电脑设备则会议系统的人机交互能力非常薄弱，观众和演讲者基本处于信息孤岛的状况，演讲者如果需要实时与观众进行交互则需要被手机或电脑设备的距离约束，并需要分散注意力关注观众的反馈。

我们提出一种解决方案，尝试抓住这个关键问题，通过增加便捷方便使用的外设，来增强会议系统，直播系统中的人机交互能力，力求打通以上场景中的会议多方存在的信息孤岛。

# 三、 解决的方案

## 3.1 整体框架

为了实现人机交互的功能以及软硬件的结合，我们提出了如图所示的解决方案架构。整体的解决方案可分为软件和硬件的部分。其中，软硬件通过蓝牙实现通信。

软件的主要功能是及时地检测到会议聊天窗口的信息，并在收到信息后将信号由蓝牙传送给硬件。在实现用户交互上，我们开发了应用界面，让用户可以手动选择开始/结束会议聊天窗口的信息，这样能增加用户的自主性和流程的灵活性。

至于检测信息的部分，由于会议软件的开发封闭性，因此无法透过利用连接API的方式来检测信息，最后我们选择了利用图像识别的方法来识别聊天窗口的信息。在用户点击“开始”按键之后，代码将会自动并且在相同的时间间隔内对桌面进行截图，然后再通过图像识别的方式检测新信息。当检测到新信息后，软件端会发送信号给硬件端。

在硬件方面，我们选择了Arduino Uno作为我们的单片机开发板。为了使Arduino能够与电脑进行通信，我们利用了蓝牙HC05作为通信媒介。在基础的单片机上，增设了按钮开关和马达/LED等。

当Arduino检测到从软件端传来的信息，将会触发马达的震动/LED的点亮，并且有不同程度的震动/闪烁方式，让用户端能够更好地分辨信息的数量。只有当用户按下开关按钮之后，才能够停止震动/闪烁。

以上为整体的方案框架，细节部分请参考接下来的介绍。

## 3.2 软件端

在会议软件层面的开发上面，我们选择了Python作为开发语言，并使用了PyQt5来进行桌面客户端界面的开发，这一套技术栈的好处是使用便捷，并且支持跨平台，在功能开发完善后，我们可以轻松地将软件移植到Win以外的其他设备和不同的系统中。如：Linux，MacOs，Android，IOS。

**聊天窗口信息检测：**

由于会议软件的封闭性，无法使用API的方式检测到信息，因此这里使用了图像识别的方式对聊天窗口的信息进行检测。代码主要可以分为截图模块和检测模块。

**截图模块：**

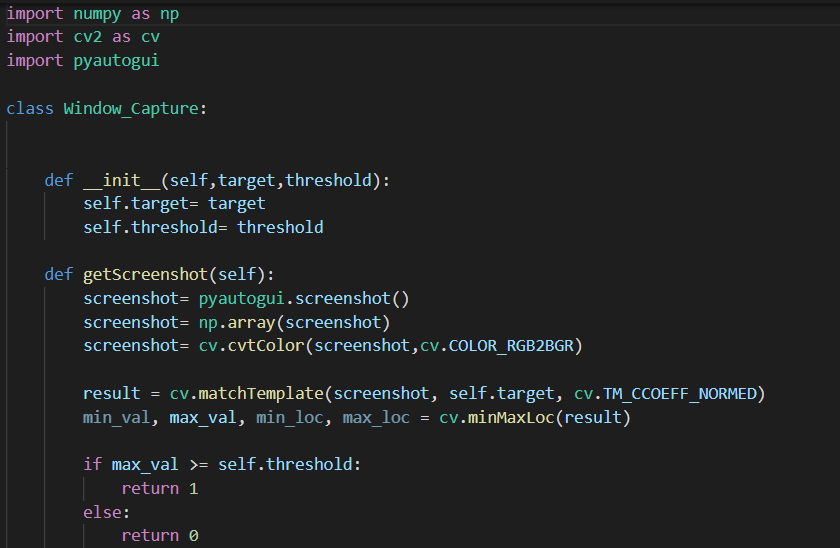
PyAutoGUI是一个纯Python的GUI自动化工具，它的功能包括用程序自动控制鼠标和键盘操作，利用它可以实现GUI的自动化任务。截图模块利用pyAutoGUI库中的screenshot函数，进行截图。

**检测模块：**

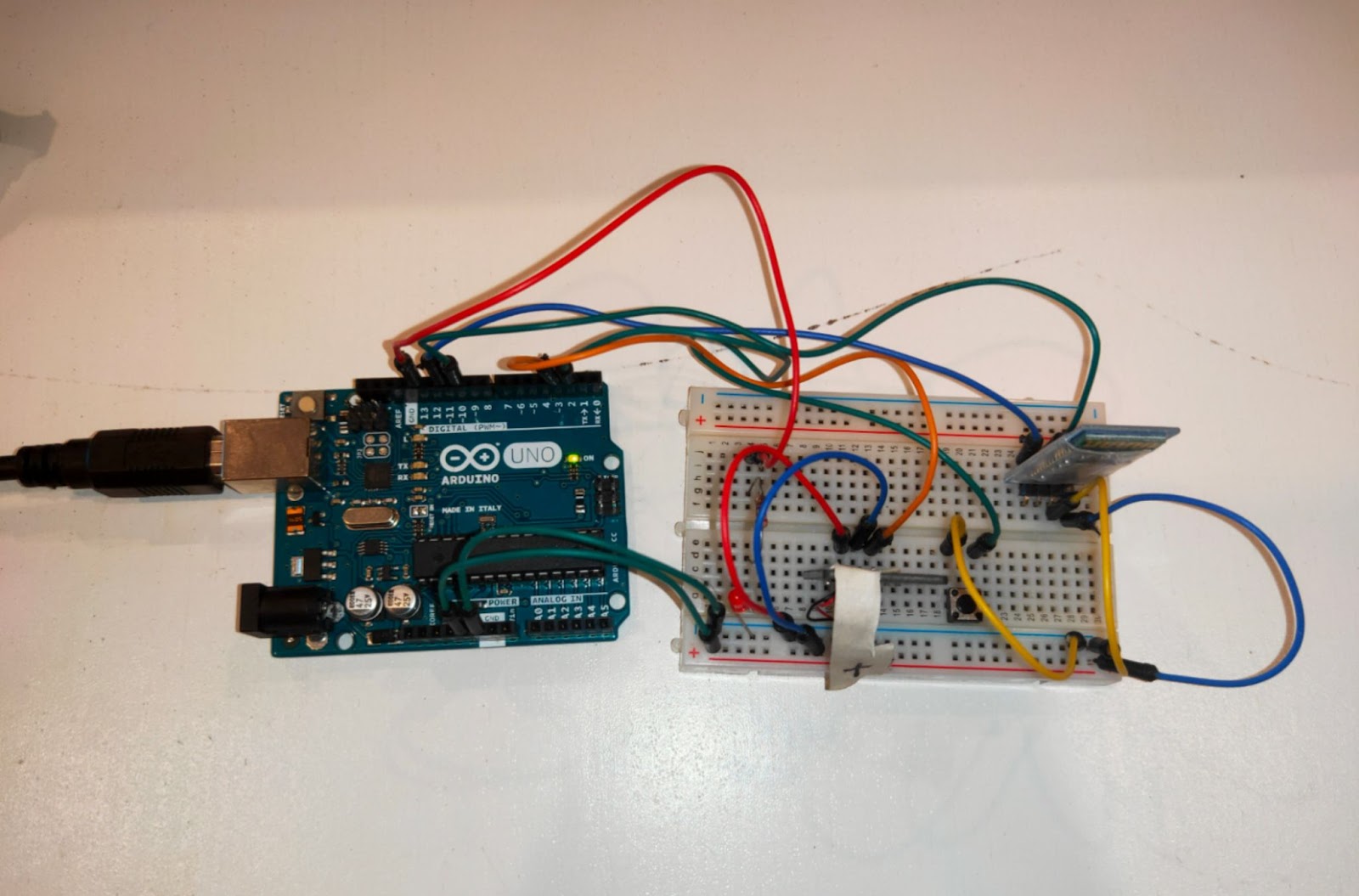
* Opencv的全称是Open Source Computer Vision Library，即计算机视觉开源库。它是基于C++语言的，但是也可以在Python使用。本次使用的函数包括：cvtColor,matchTemplate,minMaxLoc。
* cvtColor:对原图像实现灰度值、二值图、HSI等颜色形式转换。本次主要是将RGB转为BGR形式，以方便接下来的图像处理。
* matchTemplate:处理某一特定图像在模板中的位置，进行匹配。我们使用了TM\_CCOEFF\_NORMED(归一化相关系数匹配法），因为此方法较其他方法得出的结果更优。
* minMaxLoc:获取图像矩阵中的最大和最小值。用于处理匹配后的矩阵，数值越大，说明该位置的图像越接近目标图像。

检测的难点包括：不同会议软件的信息窗口图像不同，聊天窗口的位置可以改变，聊天信息的数量不同。这些都会降低信息检测的准确度，因此需要调整阈值来判断匹配的结果，以免发生误判的情况。在经过反复调试之后，我们发现将阈值设在0.8的效果最佳（1为完全匹配）。

截图和检测代码如下：

****

## 3.3 硬件端

****

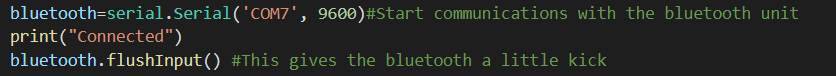
我们选用了Arduino UNO作为开发板。其他的硬件包括：蓝牙模块，震动模块，按钮。

蓝牙模块在初始化是就与电脑建立连接，之后便开始接收电脑发送的讯息。如果接收到新信息，震动模块将会开始震动，并根据等待时常增强震动。唯有用户在电脑端查看信息，或者按按钮，震动模块才会停止震动。

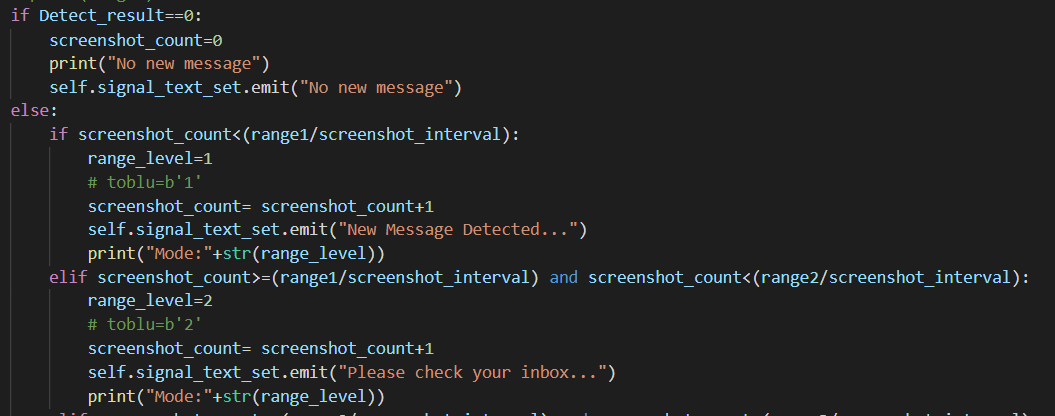
Arduino IDE上主要的需要include SoftwareSerial.h才能够将蓝牙进行连接。部分代码如下：

在电脑与硬件的连接方面，需要在Python代码中加入pySerial模块：

蓝牙连接的相应代码如下：



检测未查看信息的不同时长和对应的震动模式代码如下：



# 四、 分工

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **项目** | **备注** | **负责人** |
| 4/10-15/10 | 搜集资料，购买材料 | 购买材料需要等待2-4周才能收到 | 所有人 |
| 16/10-10/11 | 软件开发 | 编写图像识别程序 | 戴增辉 |
| 编写用户程序接口 | 刘子驾、王力人 |
| 整合软件各模块程序和调试 | 所有人 |
| 24/10-11/11 | 硬件开发 | 编写Arduino程序 | 尤恺聪 |
| 12/11-1/12 | 编写软硬件接口程序 | 尤恺聪 |
| 1/12-9/12 | 软硬件连接 | 实现软硬件连接的功能 | 尤恺聪、刘子驾、戴增辉 |
| 9/12-14/12 | 撰写总结报告 |  | 所有人 |
| 准备汇报内容 | 刘子驾、戴增辉 |

# 五、 总结

本次硬件设计作品完成了解决线上同学在会议软件上的消息无法及时得到线下的回应的问题。透过软硬件的结合，实现了人机交互的功能，也能够自动检测到会议的信息。线上的用户只要发送信息，就能够自动被远在另一端的电脑收到并通过蓝牙发送给马达，并且实时更新数据，也能够通过按钮将硬件的消息传送回软件部分，真正实现了软硬件的结合。

在实践的过程中，我们遇到了许多的困难：队友们无法同时在线下进行试验、对于软硬件的知识缺乏、会议软件的API不能够直接检测会议中的聊天信息、购买硬件的运输日期被延期、蓝牙的型号不能与电脑进行沟通、蓝牙与电脑的沟通延时长等等。但是，为了能够完成作品，我们还是尽全力把所有的困难都解决了，最后也如期完成作品。虽然作品还是有许多可以去优化的地方，但是这已经是我们尽了最大的努力去完成的作品。在解决问题的过程中，我们通过搜集网上的资料和询问有经验的人来帮助我们解决问题，比如原本我们打算利用API来对会议的聊天信息进行实时监控，但是通过搜集资料发现目前没有一个主流的会议软件提供聊天的API接口（Zoom仅仅提供用户之间的聊天信息API，而非实时会议聊天的API），因此最后决定利用图像识别的方式达到检测信息的目的。硬件的部分，由于有队友处于无法收到淘宝的国家或者因为疫情无法收到快递，因此不是所有同学都做了硬件的部分，最后也幸好队友能够及时收到硬件并完成了硬件的部分和整体的软硬件连接。

最后，十分感谢老师们和助教们的帮助，才能让我们顺利完成此课程。

# 六、 成员感想

**戴增辉：**

本次的实验可以说是困难重重，主要难点就是无法返校，并且其他队友都在不同的国家、地区，因此无法面对面进行试验。硬件的部分也就不能一起进行调试。此外，硬件的部分也需要自己购买，运输需要时间，导致最后有点拖慢了进度。但是，多亏队友们的配合以及老师和助教的帮助，才能让我们顺利完成本课程。在这次的实验中，我从想法的提出到实践的过程都对于软硬件设计有了更深的了解，我学会了如何利用openCV进行图像识别，和一些Arduino的硬件知识。除此之外，此方案的解决问题十分贴切我现在遇到的问题--线上同学的反馈无法及时得到线下老师的会议。所以希望这次的解决方案可以启发一些大佬们的帮忙，进而弥补我们现在由于技术和环境等的限制而无法实现理想结果的小遗憾。不过，我还是从这次的实验中学习到了很多知识，感谢队友，老师和助教。

**尤恺聪：**

我负责实现和调试Arduino硬件功能及其与python的交互。交互功能在网上可以找到很多的代码实现，我只需要加入Arduino端接收到信息后控制妈的和LED的功能。我遇到的最大困难是调试串口连接。Arduino IDE在上传时需要选择对外接口的选项，而先前提到的网上参考资料没有明确指出应如何选择。最终我们自己探索出，这个接口选项指的其实是代码上传的接口，应该选择主板ArduinoUno的接口，而Arduino端的蓝牙通信是靠serialconnect分配两个插针来实现。这次的任务让我更加了解了Arduino的调试方法以及蓝牙通信的实现。

**刘子驾：**

我们这次的实验工作在软件层面上尽量进行了一些扩展，硬件层面交给了一位同学全权负责，一方面是我们每个人都身处不同的国度，无法很好的进行硬件的协同开发。但也因此，我学到了很多软件系统和硬件结合需要掌握的知识。

**王力人：**

对我来说，参加这门课是我在清华的未来计划之一，当时我还在校园里。 但是由于 COVID-19，我不得不将它放到网上，这在尝试协调课程项目的每个方面都有其自身的挑战。 无论如何，我们都已尽最大努力按时完成我们的项目，并满足课程的总体要求。 由于我和我的队友承担了软件部分的责任，我可以说我已经学到了很多关于如何将我们迄今为止学到的编程语言组合在一起形成真实案例场景的知识。 最后我要感谢我们的队友们一起努力完成了赋予的任务。 我也想借此机会感谢我们的老师和我们的助教，让我们有机会参加在线课程，使我们能够满足我们系的学分要求。

# 七、 附录：源代码

## App.py

import PySide6.QtCore

import sys

import random

from PySide6.QtCore import Qt, QThread, Signal, Slot

from PySide6.QtGui import QAction, QImage, QKeySequence, QPixmap

from PySide6.QtWidgets import (QApplication, QComboBox, QGroupBox,

QHBoxLayout, QLabel, QLineEdit, QPushButton,

QSizePolicy, QVBoxLayout, QWidget,QGridLayout)

import cv2 as cv

import time,threading

import serial

from WindowCapture import Window\_Capture

chat1\_img= cv.imread('chat1.jpg',1)

chat2\_img = cv.imread('tencentchat.jpg',1)

target=chat1\_img #image we want to find from screenshot

screenshot\_interval= 5 #screenshot evey 5 seconds

range\_level=0 #range level of motor vibration

screenshot\_count=0 #index to count the screenshot that detected message

range1= 30 #range 1= 0-30s

range2= 60 #range 2= 30s-60s

range3= 90 #range 3= 60s-90s

range4= 120 #range 4= 90s-120s

threshold= 0.80 #threshold for detect image

toblu=b'0'

# bluetooth=serial.Serial('COM7', 9600)#Start communications with the bluetooth unit

# print("Connected")

# bluetooth.flushInput() #This gives the bluetooth a little kick

# flblu=1

# function and class from Qt

class MyWidget(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

State = 2

self.buttonState1 = QPushButton("Tencent")

self.buttonState2 = QPushButton("Zoom")

self.buttonStart = QPushButton("Start")

self.buttonEnd = QPushButton("End")

self.textBox = QLineEdit(self)

self.layout = QHBoxLayout()

self.layout.addWidget(self.buttonStart)

self.layout.addWidget(self.buttonEnd)

self.layout2 = QHBoxLayout()

self.layout2.addWidget(self.buttonState1)

self.layout2.addWidget(self.buttonState2)

self.layout.addWidget(self.textBox)

self.main\_layout = QVBoxLayout(self)

self.main\_layout.addLayout(self.layout2)

self.main\_layout.addLayout(self.layout)

self.setLayout(self.main\_layout)

self.buttonStart.clicked.connect(lambda: self.Detection(True))

self.buttonEnd.clicked.connect(lambda: self.Detection(False))

#self.signal\_text\_set.connect(self.SetText)

self.buttonState1.clicked.connect(lambda: self.selectState1(1))

self.buttonState2.clicked.connect(lambda: self.selectState2(2))

# run in thread

self.th = Thread(self)

self.th.signal\_text\_set.connect(self.SetText)

@Slot()

def Detection(self,startStop):

if(startStop):

# if ~flblu:

# bluetooth=serial.Serial('COM7', 9600)#Start communications with the bluetooth unit

# print("Connected")

# bluetooth.flushInput() #This gives the bluetooth a little kick

self.th.setState(startStop)

self.th.start()

else:

self.th.setState(startStop)

self.th.terminate()

time.sleep(1)

print("Stop")

# bluetooth.close() #Otherwise the connection will remain open until a timeout which ties up the /dev/thingamabob

print("Done")

# flblu=0

def SetText(self,str):

self.textBox.setText(str)

def selectState1(self,state):

self.th.setMode(state)

# print("State :" ,state)

self.buttonState1.setEnabled(False)

self.buttonState2.setEnabled(True)

def selectState2(self,state):

self.th.setMode(state)

# print("State :" ,state)

self.buttonState2.setEnabled(False)

self.buttonState1.setEnabled(True)

# new class & function add below

class Thread(QThread):

signal\_text\_set = Signal(str)

def \_\_init\_\_(self, parent=None):

QThread.\_\_init\_\_(self, parent)

self.startStop = None

self.state = 2

def setState(self,startStop):

self.startStop = startStop

def setMode(self,state):

self.state = state

def run(self):

# State = 1 zoom detect

# State = 2 Tencent detect

while self.startStop:

if(self.state == 2):

target = chat1\_img

else:

target = chat2\_img

Window= Window\_Capture(target,threshold)

Detect\_result= Window.getScreenshot()

# print(target)

if Detect\_result==0:

screenshot\_count=0

print("No new message")

self.signal\_text\_set.emit("No new message")

else:

if screenshot\_count<(range1/screenshot\_interval):

range\_level=1

# toblu=b'1'

screenshot\_count= screenshot\_count+1

self.signal\_text\_set.emit("New Message Detected...")

print("Mode:"+str(range\_level))

elif screenshot\_count>=(range1/screenshot\_interval) and screenshot\_count<(range2/screenshot\_interval):

range\_level=2

# toblu=b'2'

screenshot\_count= screenshot\_count+1

self.signal\_text\_set.emit("Please check your inbox...")

print("Mode:"+str(range\_level))

elif screenshot\_count>=(range2/screenshot\_interval) and screenshot\_count<(range3/screenshot\_interval):

range\_level=3

# toblu=b'3'

screenshot\_count= screenshot\_count+1

self.signal\_text\_set.emit("Please check your inbox...")

print("Mode:"+str(range\_level))

# bluetooth.write(tr.encode(str(3)))

elif screenshot\_count >= (range3 / screenshot\_interval) and screenshot\_count < (range4 / screenshot\_interval):

range\_level= 4

# toblu=b'4'

screenshot\_count = screenshot\_count + 1

self.signal\_text\_set.emit("Please check your inbox...")

print("Mode:"+str(range\_level))

else:

range\_level= 5

# toblu=b'5'

screenshot\_count= screenshot\_count+1

self.signal\_text\_set.emit("Please check your inbox...")

print("Mode:"+str(range\_level))

# self.signal\_text\_set.emit(range\_level)

# send range level to arduino

# bluetooth.write(range\_level)

# bluetooth.write(toblu)#bluetooth.write(str.encode(str(i)))

time.sleep(screenshot\_interval)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication([])

widget = MyWidget()

widget.resize(400,300)

widget.show()

sys.exit(app.exec())

## WindowCapture.py

import numpy as np

import cv2 as cv

import pyautogui

class Window\_Capture:

def \_\_init\_\_(self,target,threshold):

self.target= target

self.threshold= threshold

def getScreenshot(self):

screenshot= pyautogui.screenshot()

screenshot= np.array(screenshot)

screenshot= cv.cvtColor(screenshot,cv.COLOR\_RGB2BGR)

result = cv.matchTemplate(screenshot, self.target, cv.TM\_CCOEFF\_NORMED)

min\_val, max\_val, min\_loc, max\_loc = cv.minMaxLoc(result)

if max\_val >= self.threshold:

return 1

else:

return 0

## Bluetooth.ino

#include "SoftwareSerial.h"

SoftwareSerial serial\_connection(10, 11);//Create a serial connection with TX and RX on these pins

//#define BUFFER\_SIZE 64//This will prevent buffer overruns.

//char inData[BUFFER\_SIZE];//This is a character buffer where the data sent by the python script will go.

//int count=0;//This is the number of lines sent in from the python script

//int i=0;

int j=0;

char inChar=-1;

const int motorPin = 3; //motor transistor is connected to pin 3

const int buttonPin = 2; // the number of the pushbutton pin

const int ledPin = 13; // the number of the LED pin

int buttonState = 0; // variable for reading the pushbutton status

int holdstop = 0;

int prev=0;

void setup()

{

Serial.begin(9600);

serial\_connection.begin(9600);

pinMode(ledPin, OUTPUT); // initialize the LED pin as an output;

pinMode(motorPin, OUTPUT);// initialize the motor as an output;

pinMode(buttonPin, INPUT\_PULLUP);// initialize the pushbutton pin as an input:

}

void loop()

{

buttonState=digitalRead(buttonPin);

if(serial\_connection.available()>0 && buttonState==HIGH)//If there are any bytes then deal with them

{

char receive = serial\_connection.read(); //Read from Serial Communication

if(receive == '1') //If received data is 1, turn on the LED and send back the sensor data

{

prev=1;

}

if(receive == '2') //If received data is 1, turn on the LED and send back the sensor data

{

prev=2;

}

if(receive == '3') //If received data is 1, turn on the LED and send back the sensor data

{

prev=3;

}

if(receive == '4') //If received data is 1, turn on the LED and send back the sensor data

{

prev=4;

}

if(receive == '5') //If received data is 1, turn on the LED and send back the sensor data

{

prev=5;

}

Serial.println("current intensity:"+String(prev));

serial\_connection.println("Let's do this");//Then send an incrmented string back to the python script

holdstop=0;

}

if(prev!=0&&holdstop!=1){

for (j;j<prev;j++){

buttonState = digitalRead(buttonPin);

if (buttonState==HIGH){

digitalWrite(motorPin, HIGH); //vibrate

digitalWrite(ledPin, HIGH);

delay(100); // delay one milisecond

digitalWrite(motorPin, LOW); //stop vibrating

digitalWrite(ledPin, LOW);

delay((1000/prev)-100); //holtfup

}

else{

digitalWrite(motorPin, LOW);

digitalWrite(ledPin, LOW);

if(holdstop==0)holdstop=1;

break;

}

}

j=0;

}

}