IT Carlow
4rd Year Project, 2020

Technical Specification Brief

Evan Whelan - C00230300
Written in LaTeX

15/10/2020

Contents

1 Introduction

In this brief, I will outline the code and hardware setup of my project. The project will be available for download through GitHub and a user can download it and set up the necessary hardware to utilise my project for themselves.

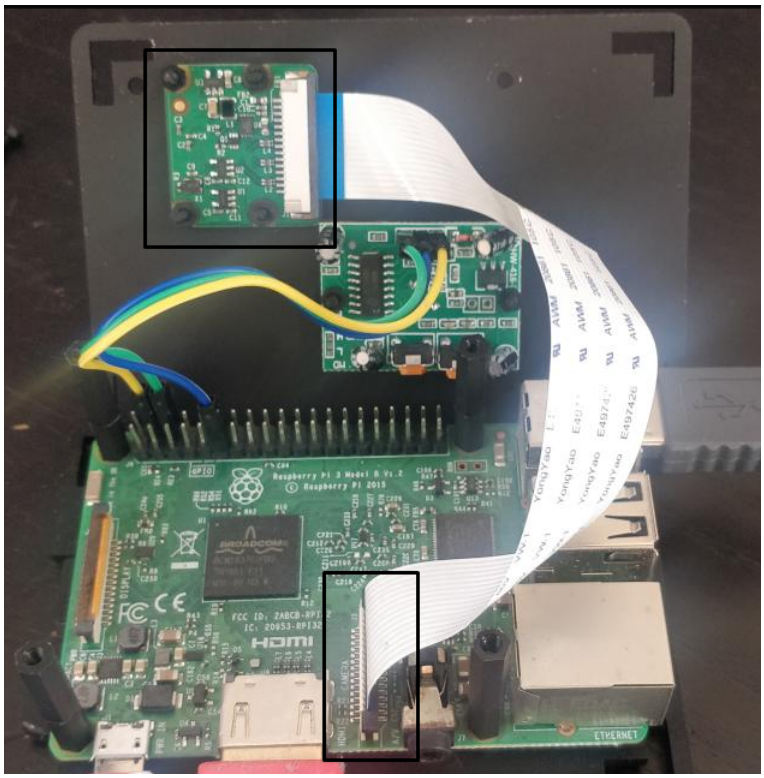
2 Hardware setup

2.1 Raspberry PI

This project will work on any raspberry PI later than the standard Raspberry PI 3. It is required that the PI model has a camera port along with GPIO pins, access to the internet and at least 8GB of storage. The raspberry PI OS must be updated once installed with the Raspbian OS by issuing the command "sudo apt update". Once updated the user must then download and install Python onto their machine by issuing the command "sudo apt install python3 idle3".

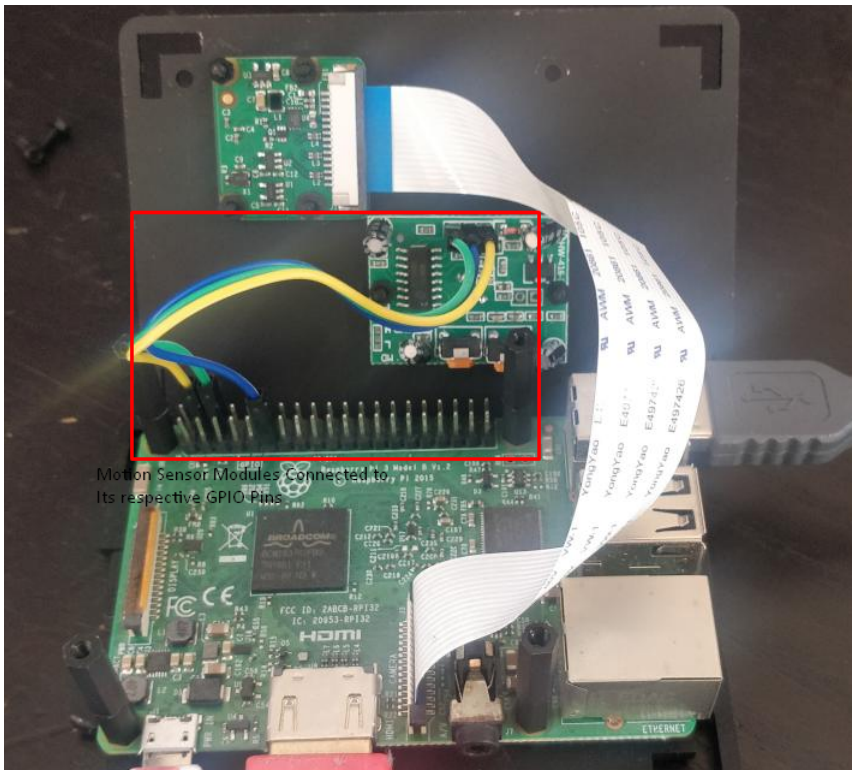
2.2 Camera Install

The camera must be installed through the dedicated camera port on the raspberry PI, once installed the camera must then be activated within the raspberry Pi's operating system by typing the following command "sudo raspi-config". You must then select setting 7 and click to enable the camera utility. You should then test that the camera is in fact working by taking a test photograph by typing the following command into the CLI "raspistill -v -o test.jpg" I have below a photograph of the camera module setup.



2.3 Motion Sensor Install

The motion sensor should come with the motion sensor module, connector wires and an optional mounting bracket to mount it to a compatible Raspberry PI case. Plug each one of the connecting wires into the connector pins on the motion sensor, taking note of which wire is connected to which pin. All 3 of the pins will be labelled with their respective function (3-5V power IN, digital OUT and ground). The connector wires should then be connected with their respective pins on the raspberry PI GPIO pins. I would recommend using pin 6 for ground, pin 2 for the 3-5V power and pin 26 (GPIO 7) for the digital input pin. See my setup below.



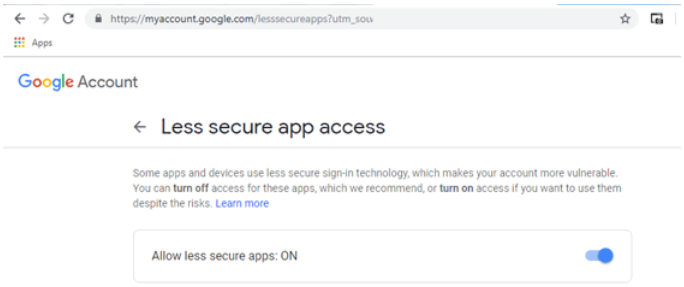
3 The code

3.1 Dependencies

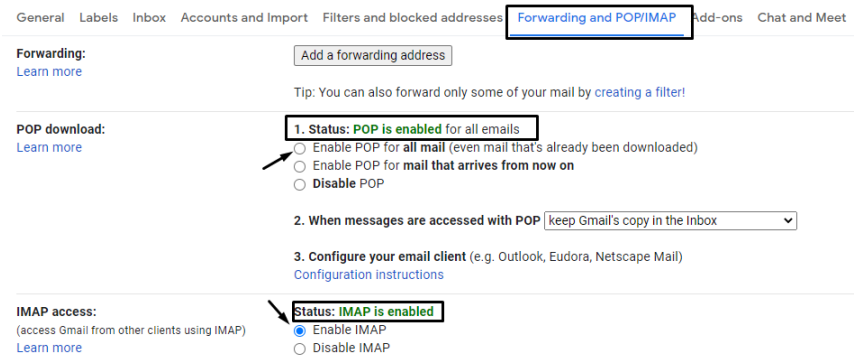
You will need to download and install SMTP email server for Python before doing any of the code. To do so, open the CLI and type the following command "sudo apt-get install ssmtp".

3.1.1 Email setup

It is highly recommended that you set up an alternative email that you just use for sending the email via the script. This is because there is various settings that need to be enabled that can lower your security on your email. Once you have created the new email, click on the cog in the top right to access the settings, click show more, then select security on the right hand side of the screen. Under "sign in and security" select "Less secure app access" and then ensure that it is enabled.



Now you need to enable IMAP and POP on your email to allow it to automatically send emails. To do this, once again select the cog symbol on your email then select "view all settings". At the top bar, select "Forwarding and POP/IMAP". Here, click to enable POP and then click to enable the IMAP function. Make sure to refresh the page to ensure that both have been enabled.



3.2 Sensor Code

3.2.1 Imports

Various libraries have to be imported for the code to function. These imports include the GPIO and camera port libraries while the emailing function requires the importation of SMTPLib as well as imghdr to allow for the sending of attachments in image form. MIME will also need to be imported to allow for the encoding of the email as well as the attachment and the SSL library will need to be imported to allow for the encryption aspect.

```
import RPi.GPIO as GPIO
from picamera import PiCamera
import time
from datetime import datetime

import smtplib
import imghdr
import ssl
from email.message import EmailMessage
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
from email.mime.base import MIMEBase
from email import encoders
import os.path

import imghdr
from email.message import EmailMessage
```

3.2.2 Loops

Next we have the Main loops of the code. Here we use the Try Except loop to keep running the code further within the loop until a specific action is made. In this case, once the user hits a key on the keyboard after the program is running the loop will stop and the system will turn off. The next loop is a While True loop which will keep running while the motion sensor is sending a True Boolean through the GPIO ports, showing that there's movement within its scope.

```
try:
    time.sleep(1)
    print("Ready")

    while True:
        if GPIO.input(PIR_PIN):
```

3.2.3 Saving the footage

Once the While loop is initialised due to the motion sensor detecting motion, the code will then print "motion detected" to the terminal while taking note of the time and date. The code will then capture and save an image after a 2 second delay and name the photo the time and date at which the motion was detected.

```
print("Motion Detected!")

timestamp = time.strftime("%Y%m%d-%H%M%S")
image_name = 'IMG_' + timestamp + '.jpeg'

camera.start_preview()
time.sleep(2)
camera.capture(image_name)
camera.stop_preview()
```

3.3 Email code:

3.3.1 Login and forming the email layout

SMTP requires the sender's email and password as well as the email of the recipient/homeowner. This is why it is good practice to create an email just for this program. MIME is used in conjunction with STMP to properly encode the message as well as the attachments sent. Due to the password being stored in plain text within the code, I need to stress the importance of making sure your Admin account on the raspberry PI uses a strong access password. Here, I have listed the code where it signs in and forms the subject and body of the email that will be sent.

```
email_user = 'itcc00230300@gmail.com'
email_password = 'TestPassword123'
email_send = 'evanwhelan7@gmail.com'

subject = 'Security Camera'

msg = MIMEMultipart()
msg['From'] = email_user
msg['To'] = email_send
msg['Subject'] = subject

body = 'Movement was detected on your security camera!'
msg.attach(MIMEText(body,'plain'))
```

3.3.2 Sending the image attachment

To send an image attachment, we must use the MIME encoding functionality to properly format the .jpg file. Here I have specified the attachment's name and not the directory, as it will automatically search the directory the code is stored in for the named file. Once python opens the file, it must be set as a part of the MIME message. After all information is entered into the MIME string, it

is then encoded using base64 encoding and the image is then assigned as an attachment payload. The payload is then attached to the MIME email contents (named msg in my code). The email is now successfully created.

```
filename = image_name
attachment  = open(filename,'rb')

part = MIMEBase('application','octet-stream')
part.set_payload((attachment).read())
encoders.encode_base64(part)
part.add_header('Content-Disposition',"attachment; filename= "+filename)

msg.attach(part)
```

3.3.3 Securing the Email

The email is secured by the use of SSL which allows for the encryption of the email to the sender's email as well as providing authenticity assurance. SSL is added at the end when the email is about to be sent and is initialised by the following code.

```
ssl_context = ssl.create_default_context()
server.starttls(context=ssl_context)
```

3.3.4 Sending the Email

The email is sent by by first calling the login function and specifying the account details as the functions parameters. Then the sendmail function is called with the email's content as the parameters. The email has now been securely sent and now the While loop will start again after a 2 second delay.

```
server.starttls(context=ssl_context)
server.login(email_user,email_password)

server.sendmail(email_user,email_send,text)
server.quit()
```