Front Door Security

CS-300 Final Project

Brad Ritzema (bdr22), Zachary Chin (zc26)

Professor Schuurman

CS-300-A

5/6/2020

Project Design Document

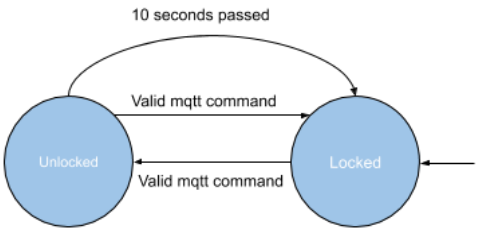
Project Summary:

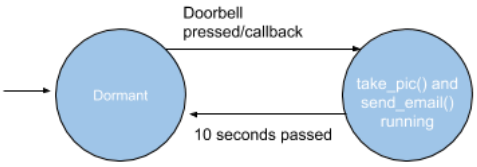
For our final project, we decided to make a door security system. To implement it, we had a variety of parts that had to work together. First, we had a button that simulated a doorbell, and a camera that took a picture. When the doorbell is rung, the camera captures a picture of who is at the door and sends it to the specified user's email. Second, we had a motion sensor (due to damages, it is simulated by a button) that once it detects motion, an LED turns on. Third, we have a servo motor that serves as a lock; the user can run a python program and lock/unlock the door from anywhere.

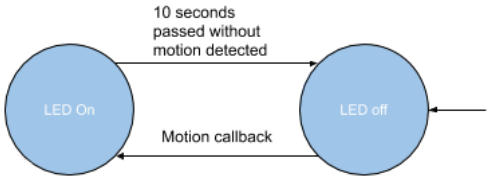
Social, Security, or Privacy Issues:

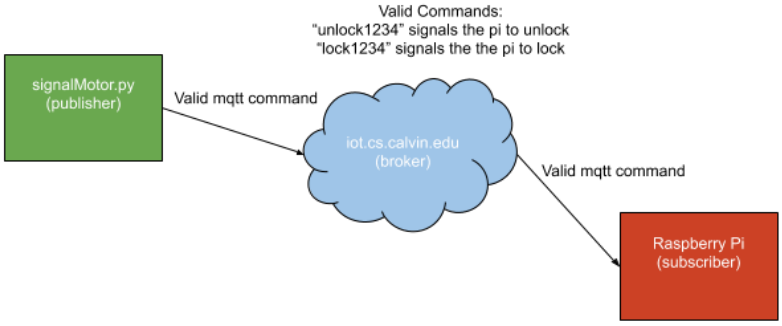
We built this project with a few issues at the forefront of our minds. One of the biggest issues we foresaw was security. We did not want just anyone to be able to access the Raspberry Pi and unlock/lock the door. To combat this problem, we used TLS encryption with MQTT along with valid passwords for the lock/unlock. So, when you run the SignalMotor.py program you need to know the correct password to either lock or unlock the door. This password is securely transferred (no one can view the data) and is verified by the Raspberry Pi. Also, the door automatically unlocks after 10 seconds. The door can be both manually and automatically locked. Along with the lock, the emailed pictures are secure. We considered implementing facial recognition to signal an unlock of the door but this conflicted with the economic design norm. In order to implement a facial recognition system that is secure enough to not accept images of people, the price of the entire system significantly goes up. Finally, with usability in mind, we added the motion sensor and light. In the night, the LED will illuminate those at the door, and increase the quality of the nighttime pictures.

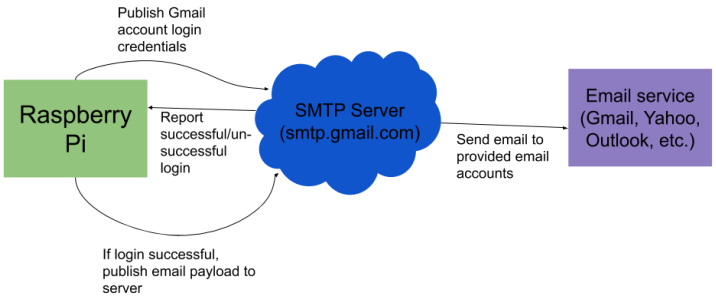
State Diagram/Data Flow Charts:











Fritzing Diagram:

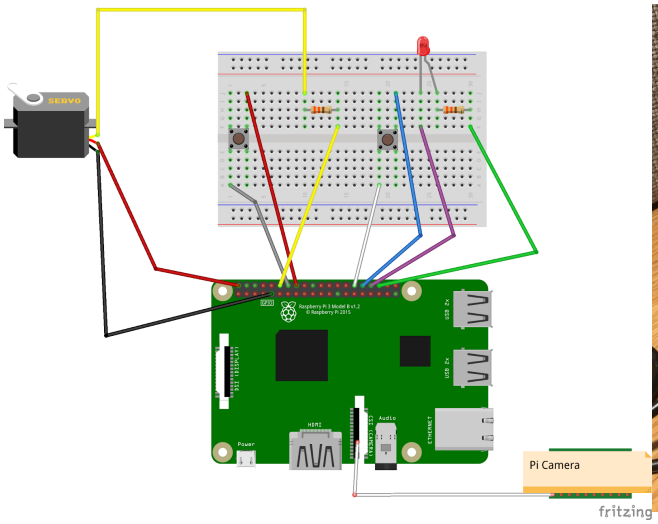
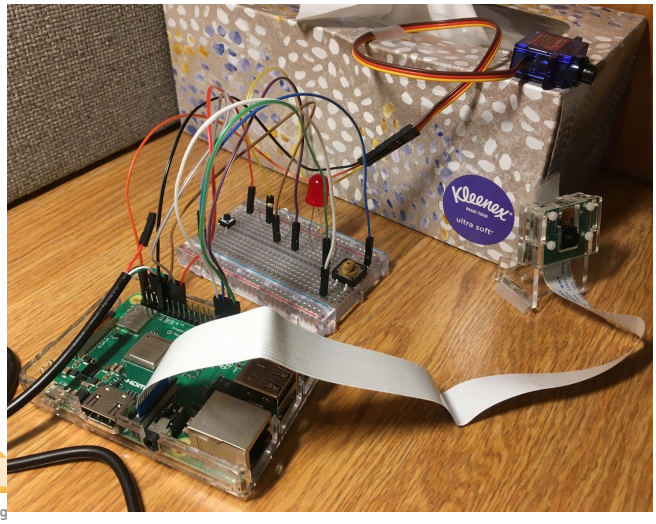
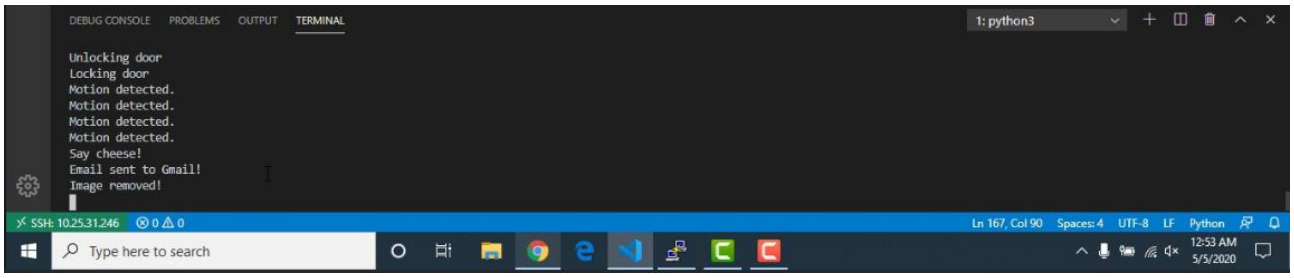


Photo of Completed Prototype:

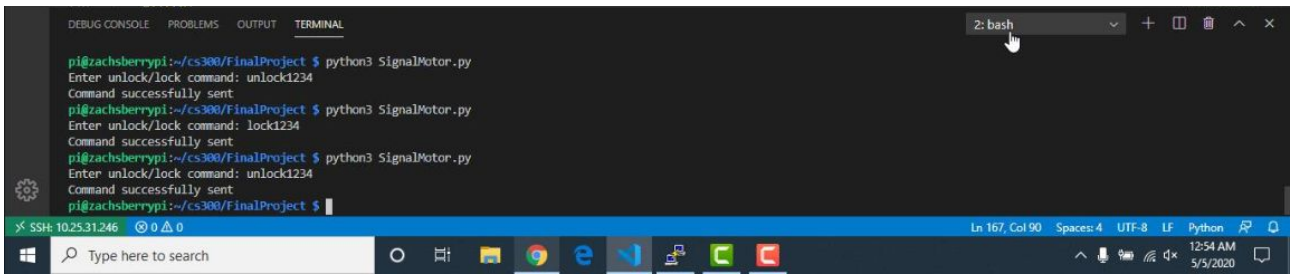


Sample output:

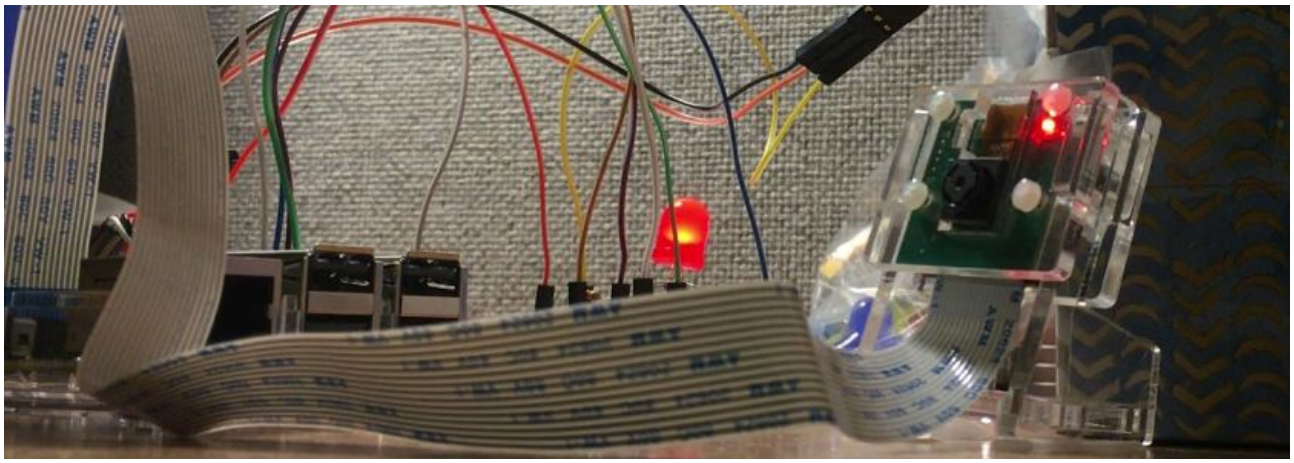
FrontDoorSecurity.py



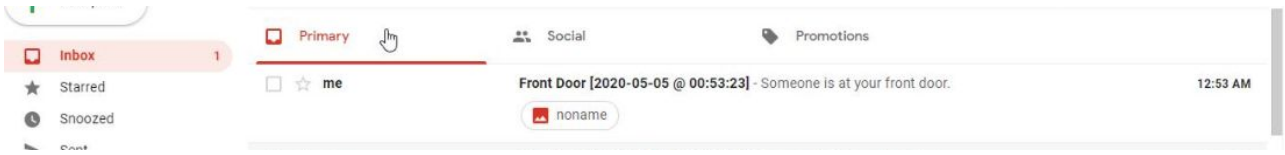
SignalMotor.py

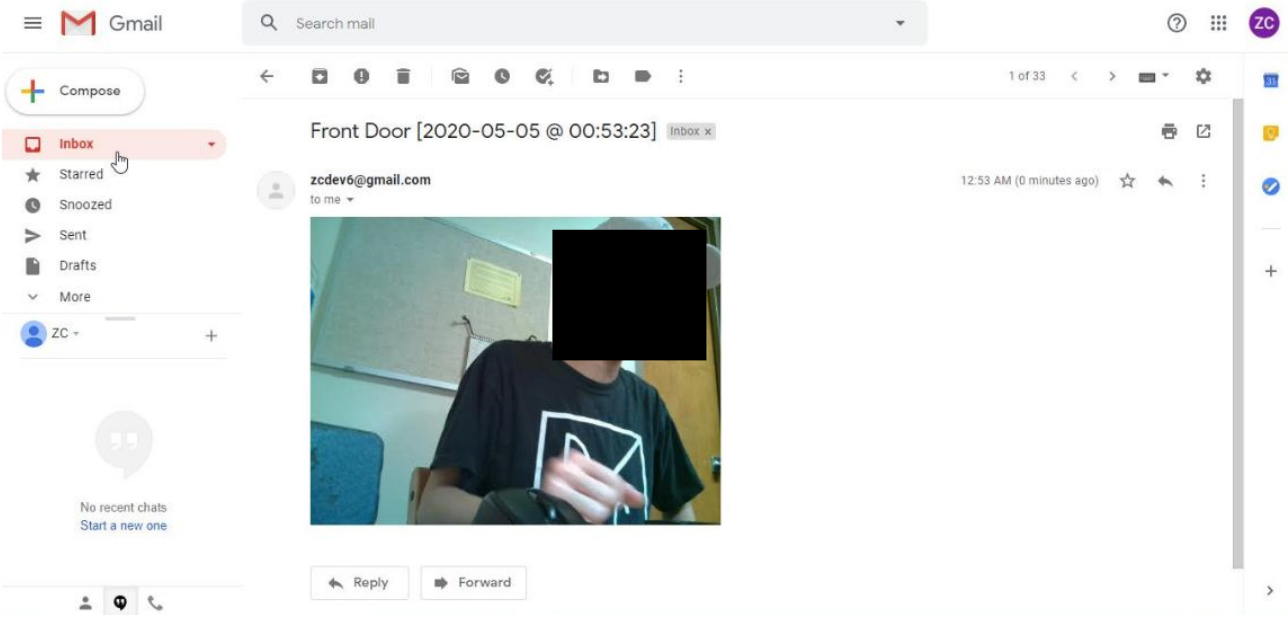


Actuators



Email inbox:




Gmail
Search mail
ZC
Compose
Inbox
Starred
Snoozed
Sent
Drafts
More
ZC
No recent chats
Start a new one
1 of 33
Front Door [2020-05-05 @ 00:53:23]
Inbox
zcdev6@gmail.com
to me
12:53 AM (0 minutes ago)
Reply
Forward

Appendix:

SignalMotor.py

```
#-----------------------------------------------------------------------
# SignalMotor.py
#
# Names: Brad Ritzema, Zachary Chin
# Course: CS-300-A
# Last Modified: 5/6/2020
#
# Publishes messages to the Raspberry Pi to control the locked/unlocked
#   state; linked with FrontDoorSecurity.py through MQTT.
#-----------------------------------------------------------------------
import paho.mqtt.client as mqtt
import time
import os

# Constants
BROKER = [redacted]
USERNAME = '[redacted]' # broker username
PASSWORD = [redacted]" # broker password
PORT = 8883
QOS = 0
TOPIC = 'bdr22/door'
CERTS = '/etc/ssl/certs/ca-certificates.crt'
VALIDUNLOCK = "unlock1234"
VALIDLOCK = "lock1234"

# Callback when a connection has been established with the MQTT broker
def on_connect(client, userdata, flags, rc):
    if rc==0:
        print('Connected to',BROKER)
    else:
        print('Connection to',BROKER,'failed. Return code=',rc)
        os._exit(1)

# While command incorrect:
while True:
    command = input ("Enter unlock/lock command: ")
    if (command != VALIDLOCK and command != VALIDUNLOCK):
        print("Invalid unlock/lock command, try again")
    else:
        break

# Setup MQTT client and callbacks
client = mqtt.Client()
client.on_connect = on_connect
# Securely connect to MQTT broker
client.username_pw_set(USERNAME, password=PASSWORD)
client.tls_set(CERTS)
client.connect(BROKER, PORT, 60)
try:
    client.publish(TOPIC, command)
    print("Command successfully sent")
```

```
except KeyboardInterrupt:
    print("Done")
    client.disconnect()
```

FrontDoorSecurity.py

```
#-----------------------------------------------------------------------
# FrontDoorSecurity.py
#
# Names: Brad Ritzema, Zachary Chin
# Course: CS-300-A
# Last Modified: 5/6/2020
#
# Note: enter "sudo pigpiod" in the terminal prior to starting the program
# Simulates a front door security system, including an automated porch
#   light, door lock, and doorbell. Porch light is triggered after sensing
#   motion; the doorbell causes the camera to take a picture and send the
#   resulting image to provided email address(es).
#
# Door lock status can be controlled via SignalMotor.py (ideally can be
#   run from anywhere); the two programs are linked through MQTT. Once
#   unlocked, door automatically locks after 10 seconds.
#-----------------------------------------------------------------------
import RPi.GPIO as GPIO
import time
import pigpio
import paho.mqtt.client as mqtt
import os

# For RPi Camera
from datetime import datetime
from picamera import PiCamera

# For creating/sending email
import html
import mimetypes
from email.headerregistry import Address
from email.message import EmailMessage
from email.utils import make_msgid
from pathlib import Path
import smtplib
import ssl

# Create camera object
camera = PiCamera()

# Constants for doorbell
DOORBELL = 12
DOORBELL_BOUNCETIME = 10000

# Constants for porch light and motion sensor
MOTION_SENSOR = 23
LED = 16
MOTION_BOUNCETIME = 300

# Constants for servo motor
MOTOR = 18              # Connect servomotor to BCM 18
DELAY = 2               # Delay to avoid bouncing
LOCKPOSITION = 1500
UNLOCKPOSITION = 2300
```

```
ITERATIONTIME = 1

# Constants for mqtt
BROKER = '[redacted]'
MQTT_USERNAME = "[redacted]" # broker username
MQTT_PASSWORD = "[redacted]" # broker password
MQTT_PORT = 8883
QOS = 0
TOPIC = 'bdr22/door'
CERTS = '/etc/ssl/certs/ca-certificates.crt'
VALIDUNLOCK = "b'unlock1234'"
VALIDLOCK = "b'lock1234'"

# Constants for Gmail
GMAIL_SERVER = 'smtp.gmail.com'
GMAIL_PORT = 465
GMAIL_USER = "[redacted]"
GMAIL_PASS = "[redacted]"

# Mailing list (can add other emails to list if needed)
sent_to = ["[redacted]"]

# setup doorbell
GPIO.setmode(GPIO.BCM)
GPIO.setup(DOORBELL, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# setup porch light and sensor
GPIO.setup(MOTION_SENSOR, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(LED, GPIO.OUT)
motionStartTime = 0               # Initialize motionStartTime


#-----------------------------------------------------------------------
# send_email()
#
# Creates an EmailMessage to store the email's data and sends the payload
#   string (which includes the picture data) from the provided Gmail
#   account to the specified email address.
#-----------------------------------------------------------------------
def send_email(dateAndTime, imgName):
    title = dateAndTime.strftime("%Y%m%d-%H%M")
    path = Path(imgName)

    msg = EmailMessage()
    msg["Subject"] = "Front Door [" + dateAndTime.strftime("%Y-%m-%d @
      %H:%M:%S") + "]"
    msg["From"] = GMAIL_USER
    msg["To"] = sent_to

    # Specify content of email shown in inbox preview
    msg.set_content("Someone is at your front door.")

    # Code below referenced from
    # https://stackoverflow.com/questions/19171742/send-e-mail-to-gmail-with-
    # inline-image-using-python
```

```
    #-------------------------------------------------------------------------

    # Create a unique string suitable for an RFC 2822-compliant Message-IDheader
    #   => defaults to domain name (@AV70211.ad.calvin.edu)
    cid = make_msgid()[1:-1]          # removes the <> around the string

    # Attaches HTML alternative to multipart message
    msg.add_alternative(
        '<img src="cid:{cid}" alt="{alt}"/>'
        .format(cid=cid, alt=html.escape(title, quote=True)),
        subtype='html')

    # Guesses the type of the image to send (.png, .jpg, et cetera)
    maintype, subtype = mimetypes.guess_type(str(path))[0].split('/', 1)
    msg.get_payload()[1].add_related(  # image/png
        path.read_bytes(), maintype, subtype, cid="<{cid}>".format(cid=cid))

    #-------------------------------------------------------------------------

    try:
        server = smtplib.SMTP_SSL(GMAIL_SERVER, GMAIL_PORT)
        server.ehlo()                           # Open connection to server
        server.login(GMAIL_USER, GMAIL_PASS)     # Login to Gmail account
        server.sendmail(GMAIL_USER, sent_to, msg.as_string())   # Send the email
        server.close()                           # Close connection to server
        print('Email sent to Gmail!')
    except:
        print("Email failed to send.")


#-------------------------------------------------------------------------
# take_pic()
#
# Takes and stores a picture on the local drive, then calls send_email()
#   to send the email with the picture attached. Deletes the picture from
#   the local drive once this process is complete.
#-------------------------------------------------------------------------
def take_pic():
    global camera
    time.sleep(1)                   # Allows the camera to adjust brightness
    now = datetime.now()
    nowFormatted = now.strftime("%Y%m%d-%H%M")
    imgName = nowFormatted + ".jpg" # Name of image is the date followed by time
    print("Say cheese!")            # **** FOR DEBUGGING ****
    camera.capture(imgName, resize = (432, 324)# Reduces size of image by factor
                                                # of 6
    send_email(now, imgName)
    os.remove(imgName)              # Removes image from local drive
    print("Image removed!")         # **** FOR DEBUGGING ****


#-------------------------------------------------------------------------
# doorbell_callback()
#
# Callback function for when the doorbell button is pressed.
```

```
#-----------------------------------------------------------------------
def doorbell_callback(channel):
    take_pic()                  # Take picture


#-----------------------------------------------------------------------
# motion_callback()
#
# Callback function for when the motion sensor detects motion
#   (Button is used to simulate motion sensor functionality)
#-----------------------------------------------------------------------
def motion_callback(channel):
    print("Motion detected.")
    global motionStartTime
    motionStartTime = time.time()
    GPIO.output(LED, True)

# if doorbell button pressed
GPIO.add_event_detect(DOORBELL, GPIO.FALLING, callback=doorbell_callback,
bouncetime=DOORBELL_BOUNCETIME)

# if motion sensor activated (motion sensor button is pressed)
GPIO.add_event_detect(MOTION_SENSOR, GPIO.FALLING, callback=motion_callback,
bouncetime=MOTION_BOUNCETIME)


#-----------------------------------------------------------------------
# unlock_door()
#
# Sets the servo motor to the unlock position to unlock the door.
#-----------------------------------------------------------------------
def unlock_door():
    print('Unlocking door')
    pi.set_servo_pulsewidth(MOTOR, UNLOCKPOSITION)
    global unlockStartTime
    unlockStartTime = time.time()
    global lockedState
    lockedState = False

#-----------------------------------------------------------------------
# lock_door()
#
# Sets the servo motor to the lock position to lock the door.
#-----------------------------------------------------------------------
def lock_door():
    print('Locking door')
    pi.set_servo_pulsewidth(MOTOR, LOCKPOSITION)
    global lockedState
    lockedState = True


# Setup servo motor (lock/unlock)
pi = pigpio.pi()                # connect to pigpiod
unlockStartTime = 0              # Initialize unlockStartTime
lockedState = True         # Default to lock state
```

```
if not pi.connected:            # test connection
    exit(0)
lock_door()                      # Lock door automatically on program start


# Client-Broker Setup

#-------------------------------------------------------------------------
# on_connect()
#
# Confirms connection with desired MQTT broker.
#-------------------------------------------------------------------------
def on_connect(client, userdata, rc, *extra_params):
    print('Connected with result code='+str(rc))


#-------------------------------------------------------------------------
# on_message()
#
# Handles input taken from the server topic.
#   If input is VALIDUNLOCK:
#       Unlock door
#   If input is VALIDLOCK:
#       Lock door
#-------------------------------------------------------------------------
def on_message(client, data, msg):
    if msg.topic == TOPIC:
        if str(msg.payload) == VALIDUNLOCK:
            unlock_door()
        if str(msg.payload) == VALIDLOCK:
            lock_door()

# Setup MQTT client and callbacks
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.username_pw_set(username=MQTT_USERNAME, password=MQTT_PASSWORD)
client.tls_set(CERTS)

# Connect to MQTT broker and subscribe to the button topic
client.connect(BROKER, MQTT_PORT, 60)
client.subscribe(TOPIC)
client.loop_start()

try:
    while True: # continuously run
        time.sleep(ITERATIONTIME)

        # If the porch light has been on for longer than 10 seconds, turn porch
        #   light off
        if (time.time() - motionStartTime) > 10 and GPIO.input(LED) == True:
            GPIO.output(LED, False)
        # If the door has been unlocked for longer than 10 seconds, lock door
        if (time.time() - unlockStartTime) > 10 and lockedState == False:
            lock_door()
```

```
# If you do want to turn it off...
except KeyboardInterrupt:
    print("\nDone.")
    client.disconnect()
    camera.close()
    GPIO.cleanup()              # clean up GPIO
```