

Table of Contents

- Section 1 Overview of the application 1
 - A. What is the application about?..... 1
 - B. Summary of the steps that will be described 1
 - C. How does the final RPI set-up looks like? 2
 - D. How does the web application look like? 3
- Section 2 Hardware requirements 6
 - Hardware checklist..... 6
- Section 3 Doorbell for the smart door system 6
 - A. Completed Fritzing Diagram 6
 - B. Connect the Buzzer 7
 - C. Connect the Button..... 7
- Section 4 LCD Display for the smart door system 8
 - A. Completed Fritzing Diagram 8
 - B. Connect the LCD Display 9
 - C. Connect the LED Indicator 9
- Section 5 NFC/ RFID Reader to read access card 10
 - A. Completed Fritzing Diagram 10
- Section 6 Picam Livestream..... 11
 - A. Installing uv4l 11
 - B. Installing ffmpeg 11
 - C. Testing the livestream 12
- Section 7 Smart Home System 13
 - A. Completed Fritzing Diagram 13
 - B. LEDs for the room and the air-conditioning 14
 - C. Temperature Sensor 15
- Section 8 Coding the programs 16
 - A. Code the Main Program 16
 - B. Coding the website 24
 - C. Adding styles to the website..... 28
 - D. Preparing the database 29
- Section 9 Running the programs 31

A. Run the program.....31
B. Output of program.....32

Section 1

Overview of the application

A. What is the application about?

This application is a smart home system which is divided into 2 main components. The first component is the smart door system which consists of a doorbell, access card system, LCD display to show the time, and whether the access card was allowed or denied, a LED indicator light to show that the door is unlocked, a motion sensor to capture motion outside of the house, with a camera to capture a picture when motion is detected.

The second component is the smart home system which consists of 2 LEDs, 1 representing the lights in the house and the other to represent the air-conditioning of the house. 2 buttons are also included to turn the LEDs off and on to represent that the air-conditioning and lights can be turned on or off using buttons. However, using the webpage of the application, the user can turn the air-conditioning or the lights on and off remotely. A temperature and humidity sensor is also used to capture the temperature and humidity in the house and a graph of the day’s temperature can be seen on the website as well.

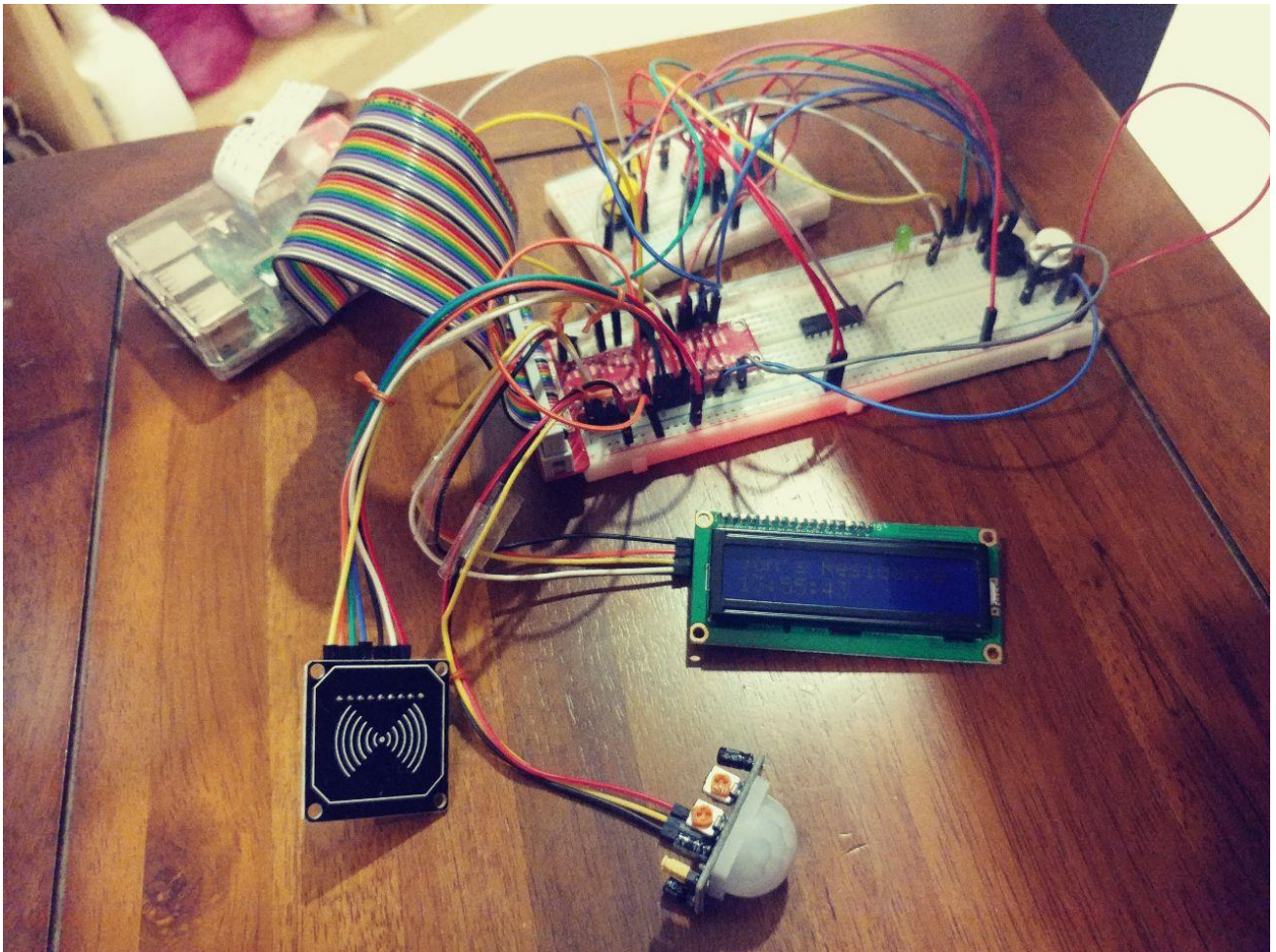
This application will help to ease the lives of home owners as they can ensure their home security using the door access system and motion sensor to detect movements and capture pictures of suspicious activities outside their house as well as a camera livestream, that allows home owners to see what is going on outside their house when they are away. The smart home system also provides home owners with convenience as they can see whether their lights or their air-conditioning is turned on or off, so that they can turn it off while they are out in the case that they forgotten to off it before leaving the house. The temperature chart also allows home owners to monitor their home temperature and they can then choose to turn on the air-conditioning before they reach home if they see that the temperature at home is high, allowing them to come back to a cool home and relax.

B. Summary of the steps that will be described

Provide a bullet list of the steps that will be covered in the other parts of this tutorial

	Section	Description
1)	Overview	
2)	Hardware requirements	Provides overview of hardware required
3)	Doorbell for smart door system	Provides a step by step guide on how to wire the doorbell system of the smart door system
4)	LCD Display for the smart door system	Provides a step by step guide on how to wire the LCD display of the smart door system
5)	NFC/ RFID Reader to read access card	Provides a step by step guide on how to wire the NFC/ RFID card reader to read the access card and grant access of the user to the home.
6)	Motion Sensor to capture images	Provides a step by step guide on how to wire the motion sensor so that it can detect motion outside the house
7)	Smart Home System	Provides a step by step guide on how to wire the LEDs and the temperature sensor of the components in the house
8)	Coding the programs	Provides a step by step guide on how to create and program the application correctly
9)	Running the programs	Provides expected outputs if the program is runned properly

C. How does the final RPI set-up looks like?



D. How does the web application look like?



JON'S SMART HOME WEB INTERFACE

View Temperature Values

LIGHT SWITCH

Turn on lights

Turn off lights

OFF

AIR-CON SWITCH

Turn on air-con

Turn off air-con

OFF



JON'S SMART HOME WEB INTERFACE

View Temperature Values

LIGHT SWITCH

Turn on lights

Turn off lights

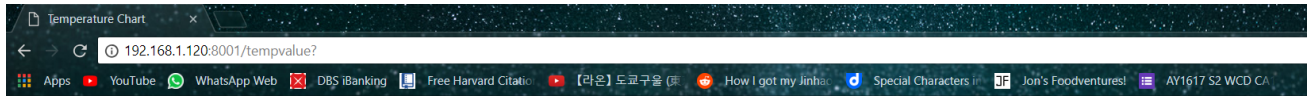
ROOM LIGHT SWITCHED ON

AIR-CON SWITCH

Turn on air-con

Turn off air-con

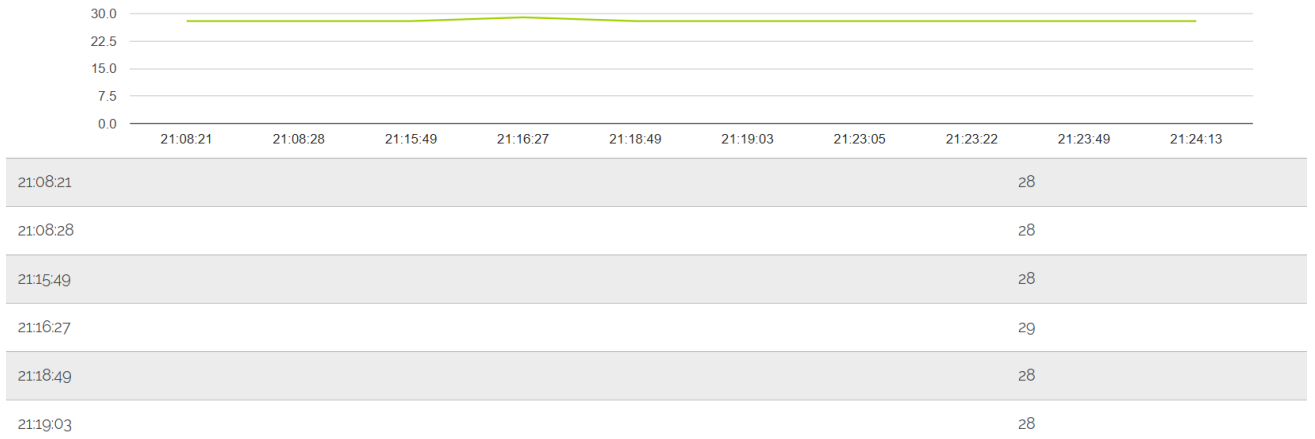
AIR-CONDITIONING SWITCHED OFF



JON'S SMART HOME WEB INTERFACE

Home

TEMPERATURE VALUES CAPTURED



Section 2

Hardware requirements

Hardware checklist

To complete this application, you will need:

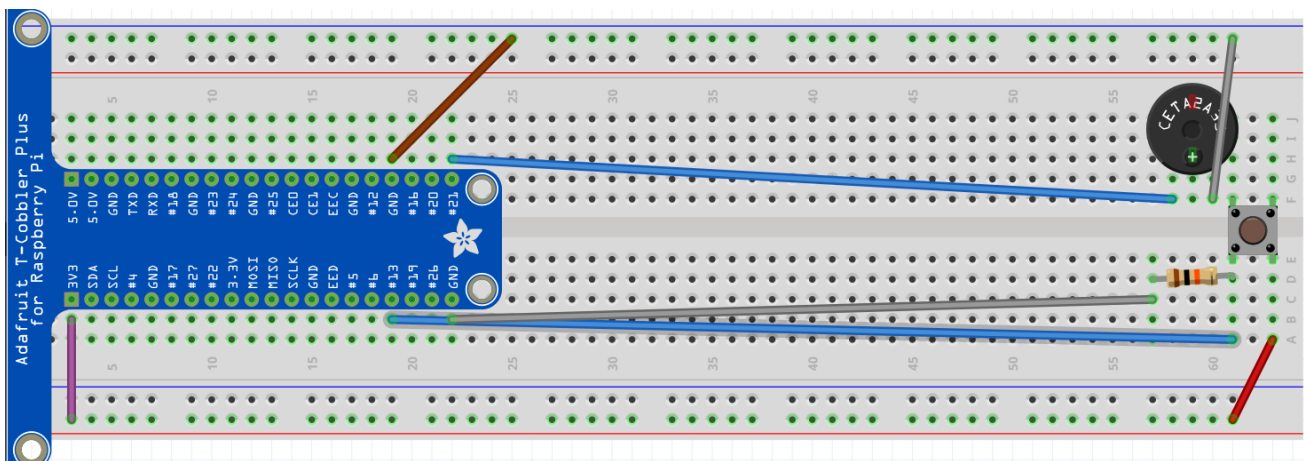
- 1 I2C LCD Display
- 1 RFID / NFC MFRC522 Card Reader Module
- 1 DHT11 Temperature and Humidity Sensor
- 1 PIR Motion Sensor
- 1 Buzzer
- 1 Raspberry Pi camera (piCam)
- 3 Buttons
- 3 LEDs
- 3 10K ohms resistors
- 3 330 ohms resistors
- Plenty of male and female wires

Section 3

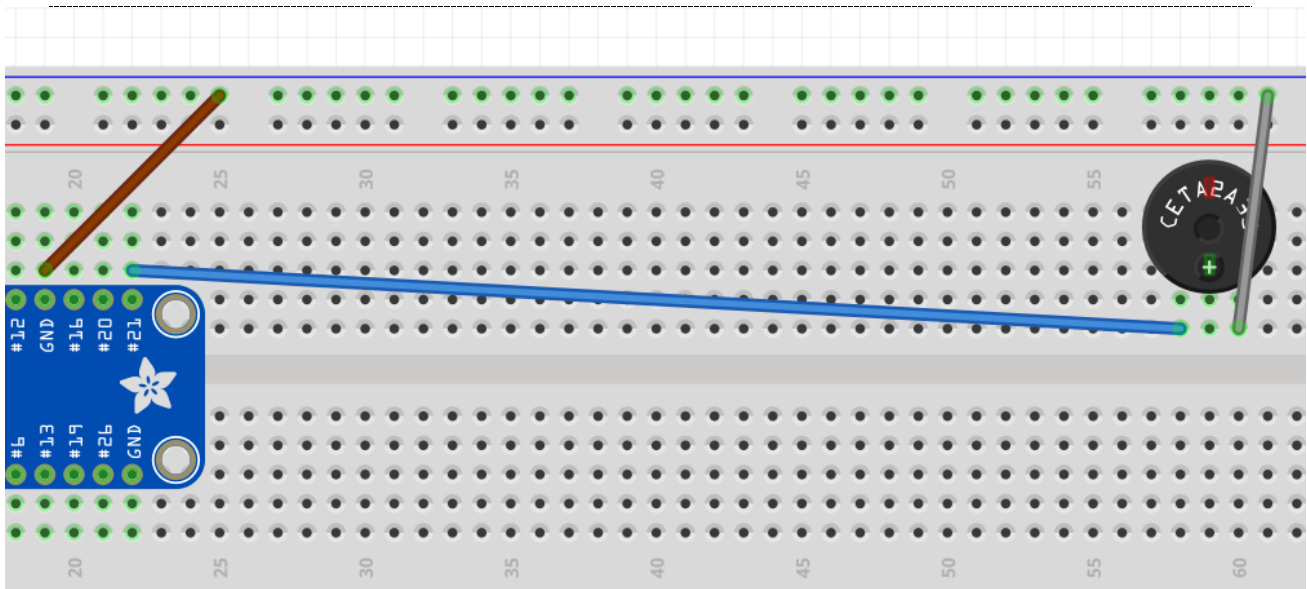
Doorbell for the smart door system

In this section, we will complete the doorbell portion of the smart door system.

A. Completed Fritzing Diagram



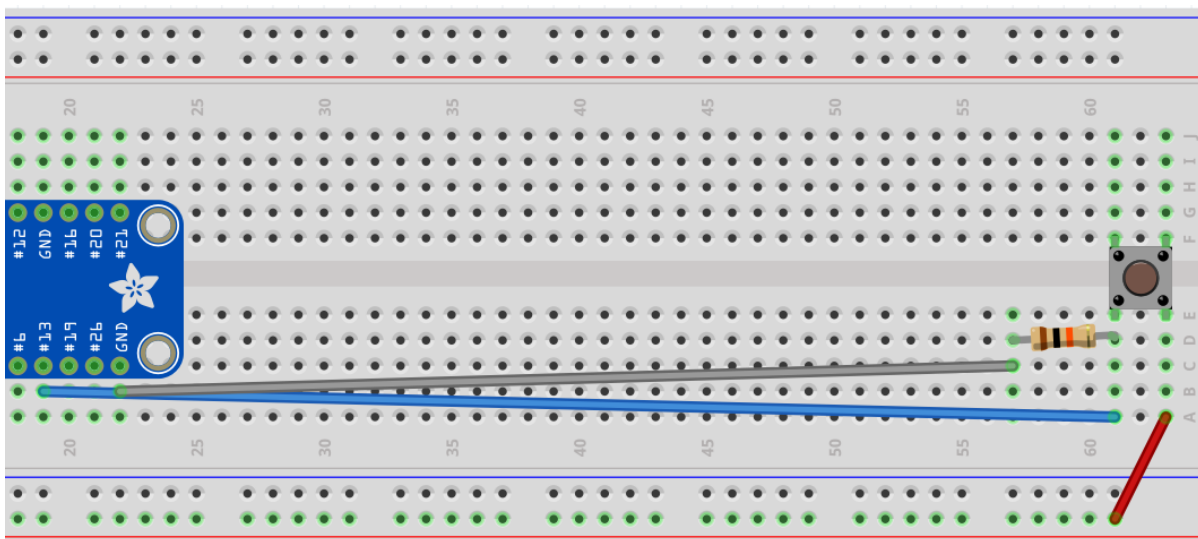
B. Connect the Buzzer



Connect the Buzzer with the RPi as follows:

Jumper color	PIR pin	RPi pin
Black	GND	GND
Blue	VOUT	BCM21

C. Connect the Button



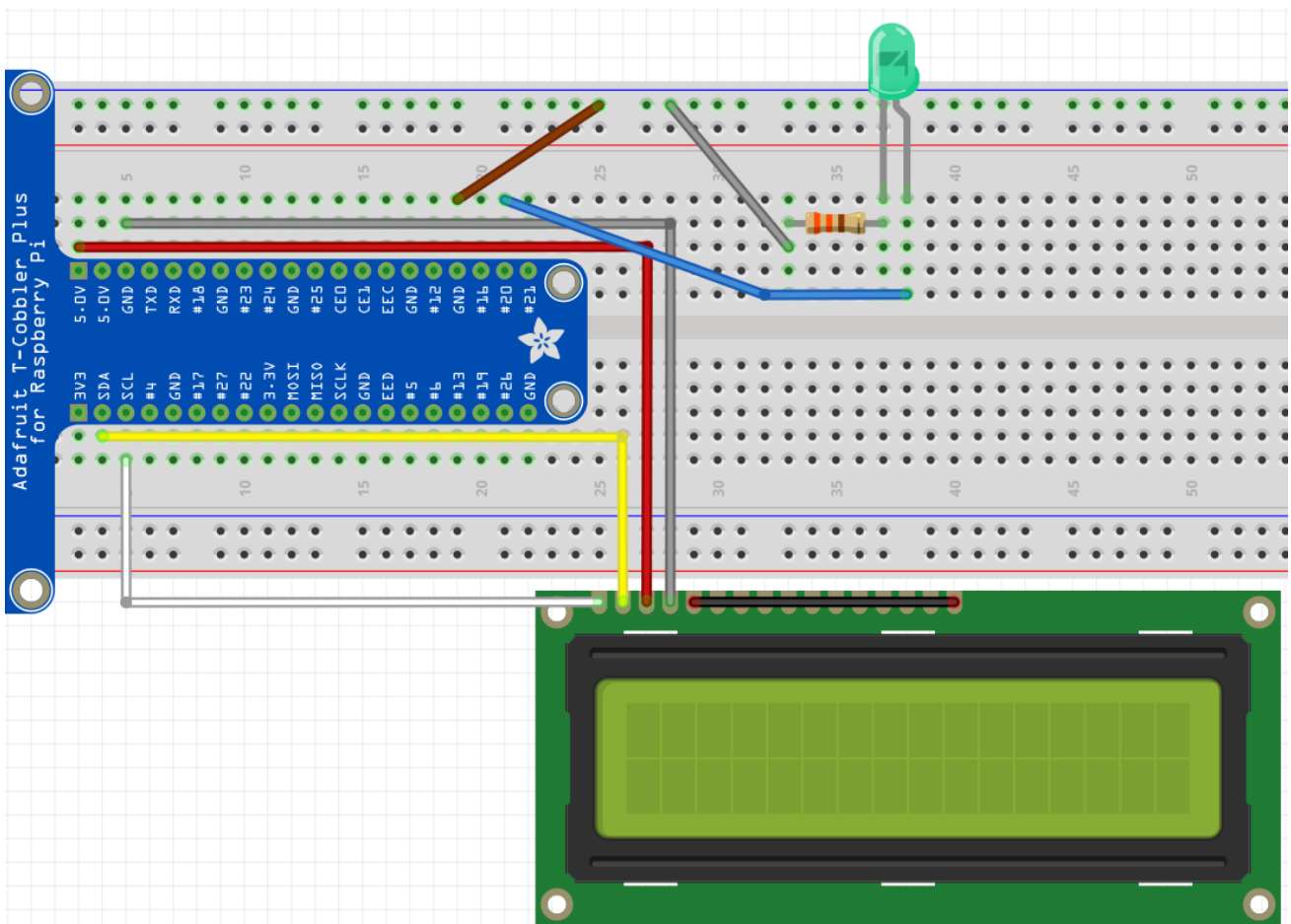
1. Plug in the button on the breadboard as shown.
2. Plug in the **10K Ω** resistor into the breadboard as shown. One end of the resistor should be in the same column as the left pin of the button.
3. Use a **red** cable to connect the right pin of the push button to 3.3V. In the diagram above, the wire is only connected to the side of the board which is connected by another wire to 3.3V.
4. Use a **blue** cable to connect the left pin of the push button to **BCM13**

Section 4

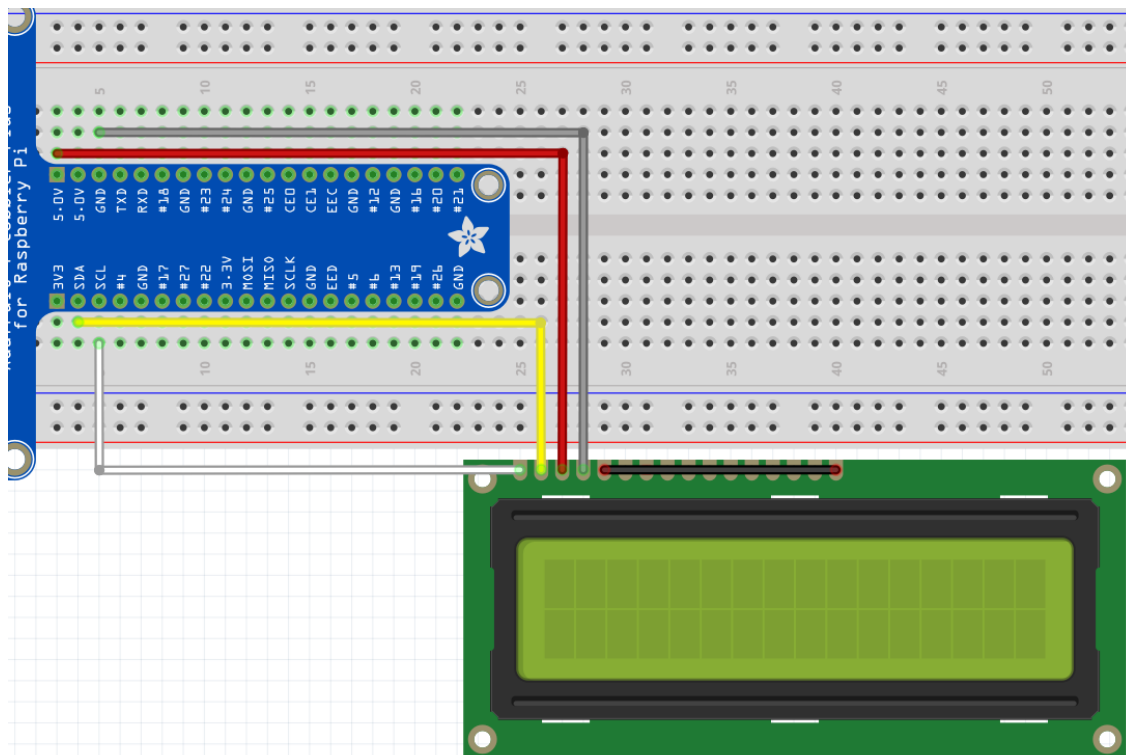
LCD Display for the smart door system

In this section, we will complete the LCD display the smart door system.

A. Completed Fritzing Diagram



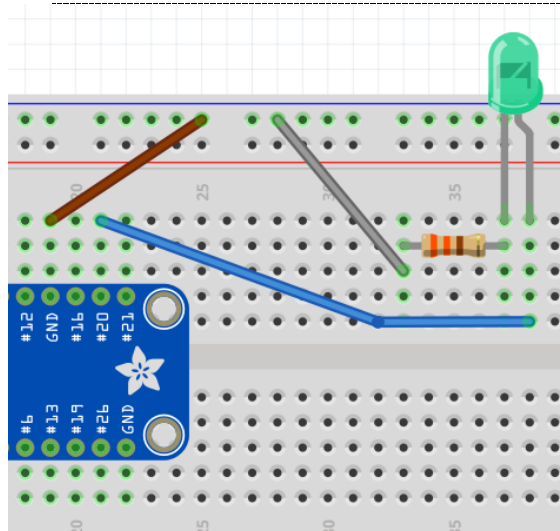
B. Connect the LCD Display



Connect the LCD Display with the RPi as follows:

Jumper color	LCD pin	RPi pin
White	SCL	SCL
Yellow	SDA	SDA
Black	GND	GND
Red	Vcc	5V

C. Connect the LED Indicator



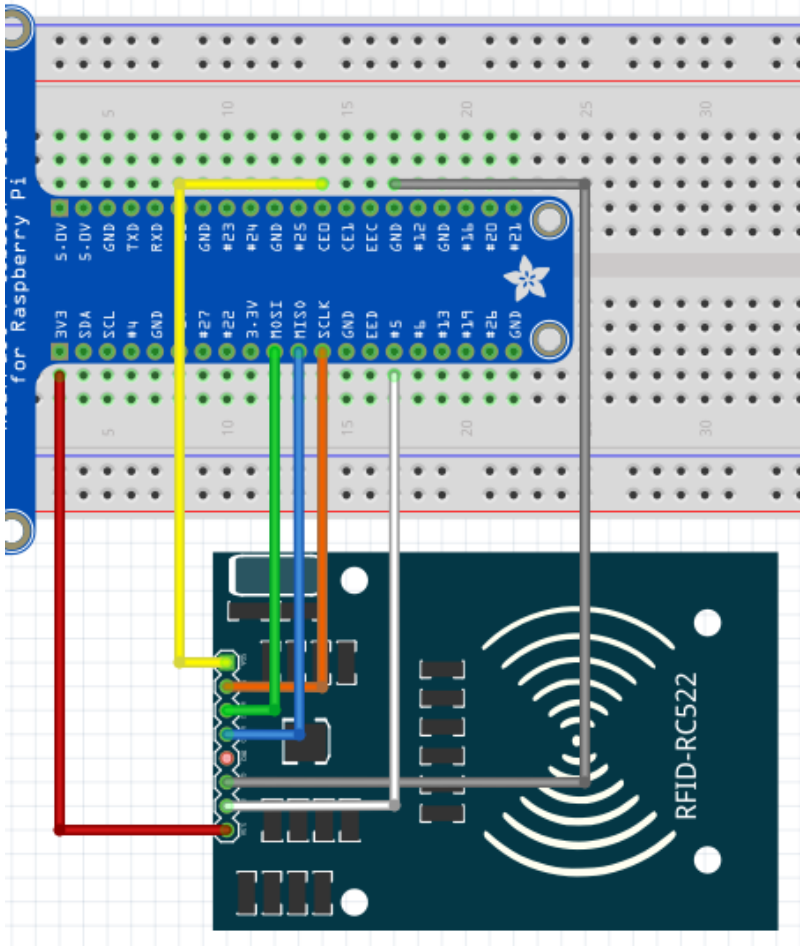
1. Plug in the LED into the breadboard as shown.
2. Plug in the **330 Ω** resistor into the breadboard as shown. One end of the resistor should be in the same column as the short leg of the LED.
3. Using a **blue** cable, connect the long end of the LED to **BCM20**
4. Using a **black** cable, connect the free end of the 330 Ω resistor to GND

Section 5

NFC/ RFID Reader to read access card

In this section, we will complete the NFC/ RFID Reader of the smart door system.

A. Completed Fritzing Diagram



Connect the pins on the MFRC522 card reader to the RPi as indicated below.

Jumper color	MFRC522pin	RPi pin
Yellow	SDA	CE0
Orange	SCK	SCLK
Green	MOSI	MOSI
Blue	MISO	MISO
	IDR	
Black	GND	GND
White	RST	GPIO5
Red	3.3V	3.3V
	5V	

Section 6

Picam Livestream

In this section, we will complete the picam livestream setup. We are using uv4l to livestream.

A. Installing uv4l

Task

- a) Add the following line to the file /etc/apt/sources.list

```
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ jessie main
```
- b) Update apt-get with the following line

```
sudo apt-get update
```
- c) Next, install the required packages with the following lines

```
sudo apt-get install uv4l uv4l-raspicam  
sudo apt-get install uv4l-raspicam-extras
```
- d) Install the additional drivers for uv4l

```
sudo apt-get install uv4l-server uv4l-uvic uv4l-xscreen  
uv4l-mjpegstream uv4l-dummy uv4l-rapidisp
```

B. Installing ffmpeg

Now, we will be installing ffmpeg, a free software suite with many video handling libraries and programs included.

Task

- a) Add the following line to the file /etc/apt/sources.list

```
deb http://www.deb-multimedia.org jessie main non-free
```
- b) Update apt-get with the following line

```
sudo apt-get update
```

Task

- c) Finally, install ffmpeg
`sudo apt-get install ffmpeg`

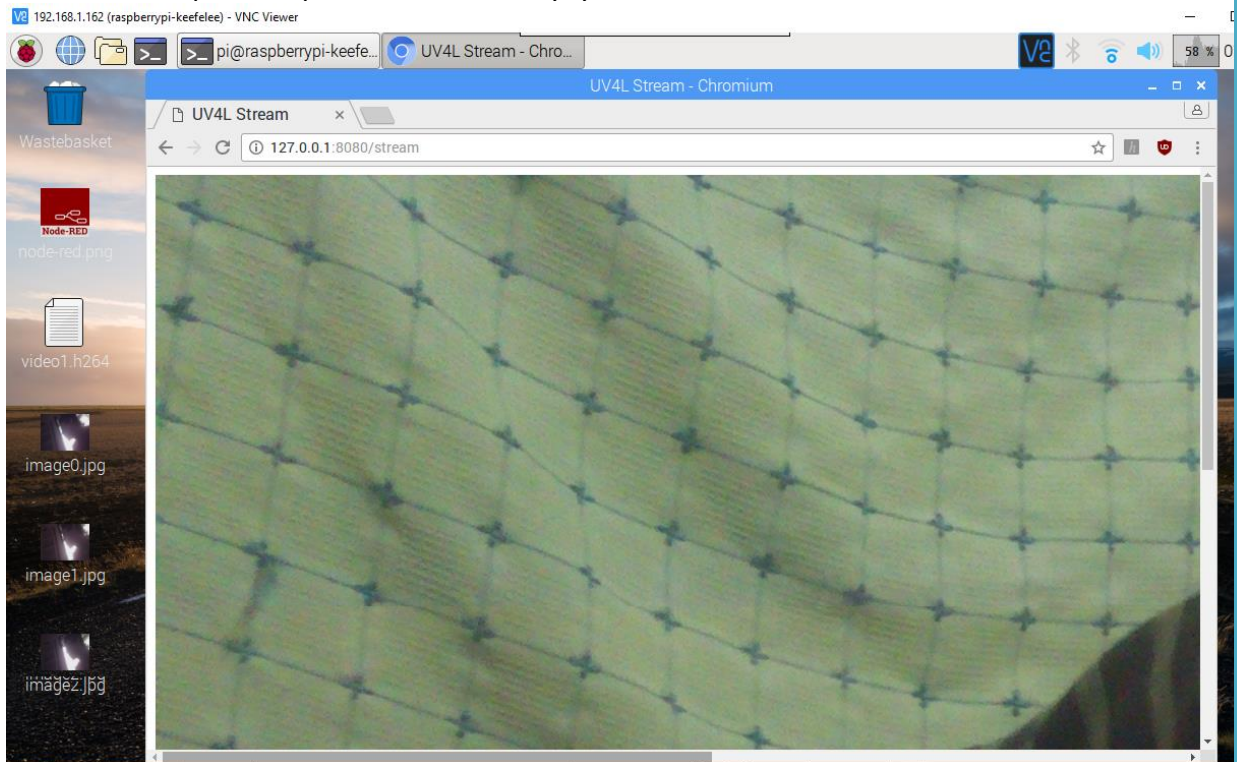
C. Testing the livestream

With uv4l and ffmpeg successfully installed, we can now begin the livestream.

Task

- a) Run the following command to begin the livestream
`uv4l --driver raspicam --auto-video_nr --encoding mjpeg --width 512 --height 288 --enable-server on`
- b) If you are using the raspberry pi browser, go to 127.0.0.1:8080/stream to view the livestream. If you are using some other browser such as Google Chrome on your computer, go to <RaspberryPi's IP address>:8080/stream.

- c) If all of the steps were performed correctly, you should be able to see something like this

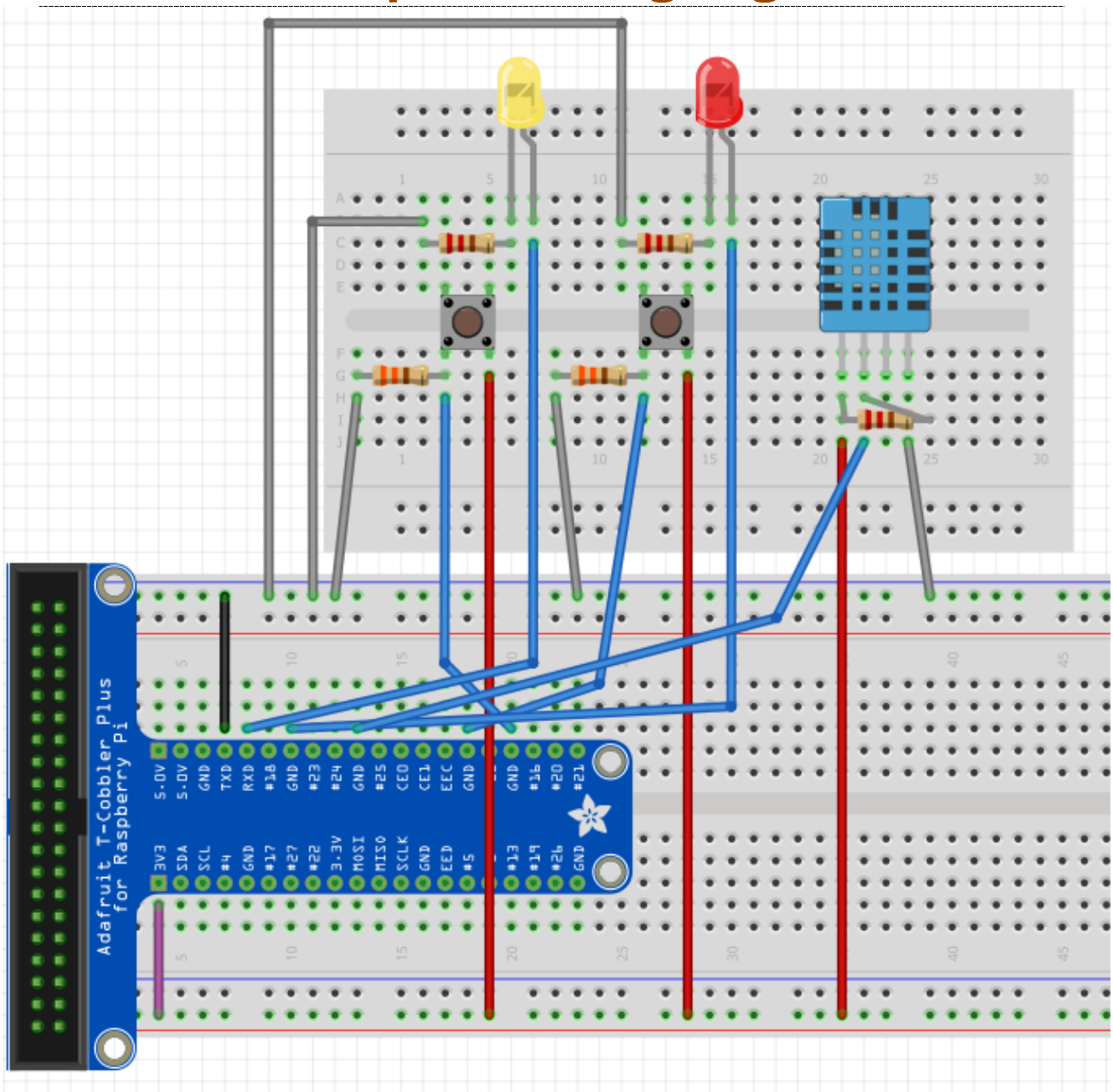


Section 7

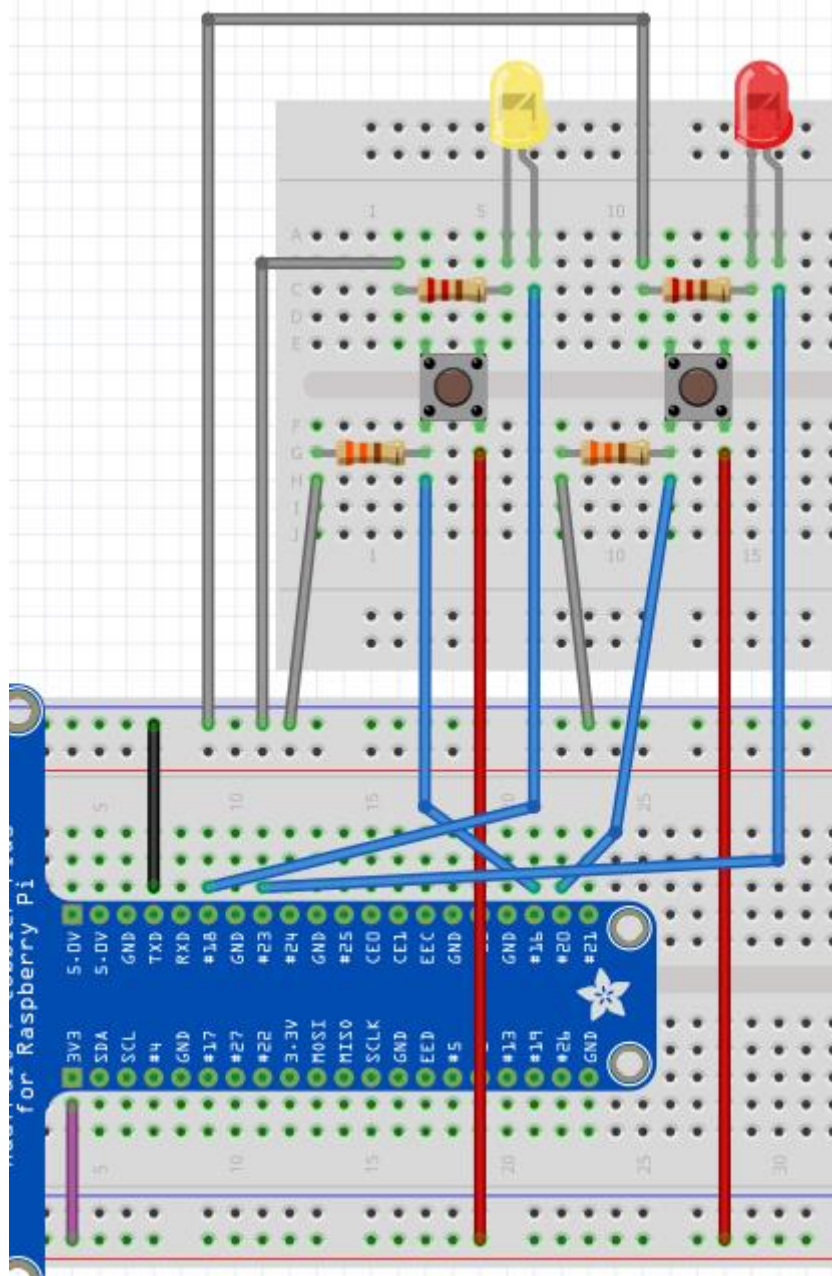
Smart Home System

In this section, we will complete the Smart Home System.

A. Completed Fritzing Diagram



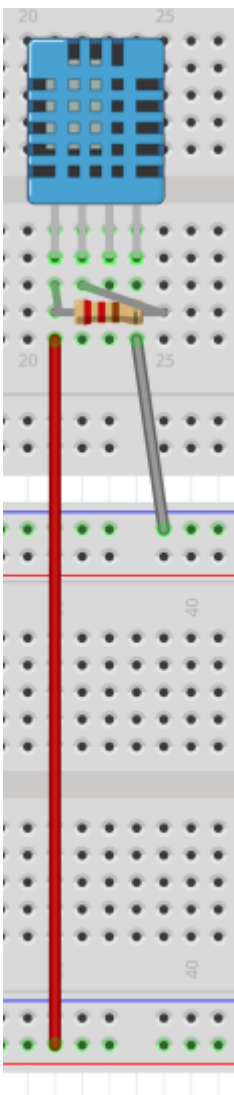
B. LEDs for the room and the air-conditioning






1. Plug in the LED into the breadboard as shown.
2. Plug in the **330 Ω** resistor into the breadboard as shown. One end of the resistor should be in the same column as the short leg of the LED.
3. Using a **blue** cable, connect the long end of the LED to **BCM18**
4. Using a **black** cable, connect the free end of the 330 Ω resistor to GND
5. Plug in the button into the breadboard as shown.

- 6. Plug in the **10K Ω** resistor into the breadboard as shown. One end of the resistor should be in the same column as the left pin of the button.
- 7. Use a **red** cable to connect the right pin of the push button to 3.3V. In the diagram above, the wire is only connected to the side of the board which is connected by another wire to 3.3V.
- 8. Use a **blue** cable to connect the left pin of the push button to **BCM16**
- 9. Repeat for the other LED using **BCM23** for the LED and **BCM20** for the button.

C. Temperature Sensor



Connect the DHT_11 temperature sensor with the RPi as follows:

<i>DHT11 Sensor</i>	<i>Raspberry Pi</i>	<i>Jumper Color</i>
VCC	3V3	Red 
DATA	GPIO4 / BCM4	Blue 
NC		
GND	GND	Black 

Section 8

Coding the programs

A. Code the Main Program

In this section, we will write the program to allow all the sensors to collect data and the webpage to be hosted.

Task

- d) Create a python script **Indoor.py** with the code below

```
sudo nano ~/ca2/Indoor.py
```

- e) Remember to create the captures folder in the ca2 folder to allow the images to be stored properly:

```
mkdir ~/ca2/captures
```

- f)
- ```
from __future__ import print_function # Python 2/3 compatibility
import RPi.GPIO as GPIO
import signal
from gpiozero import LED
import time
import sys
import Adafruit_DHT
from multiprocessing import Process
import datetime
import gevent
import gevent.monkey
from gevent.pywsgi import WSGIServer
import MySQLdb
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from flask import Flask, request, Response, render_template
import boto3
import json
import decimal
from boto3.dynamodb.conditions import Key, Attr
#Patch gevent monkey
gevent.monkey.patch_all()

#Set mode to board
#GPIO.setmode(GPIO.BOARD)

#Disable warnings
#GPIO.setwarnings(False)

#Connecting to database
try:
 db = MySQLdb.connect("localhost", "root", "root", "ca1DB")
 curs = db.cursor()
 print("Successfully connected to database!")
except:
 print("Error connecting to mySQL database")
```

## Task

```
def customCallback(client, userdata, message):
 print("Received a new message: ")
 print(message.payload)
 print("from topic: ")
 print(message.topic)
 print("-----\n")

#Setting up connection to AWS
try:
 host = "aav1qj8erc0f4.iot.us-west-2.amazonaws.com"
 rootCAPath = "rootca.pem"
 certificatePath = "certificate.pem.crt"
 privateKeyPath = "private.pem.key"

 my_rpi = AWSIoTMQTTClient("Indoor")
 my_rpi.configureEndpoint(host, 8883)
 my_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

 my_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing
 my_rpi.configureDrainingFrequency(2) # Draining: 2 Hz
 my_rpi.configureConnectDisconnectTimeout(10) # 10 sec
 my_rpi.configureMQTTOperationTimeout(5) # 5 sec

 # Connect and subscribe to AWS IoT
 my_rpi.connect()
 my_rpi.subscribe("sensors/temperature", 1, customCallback)
 my_rpi.subscribe("security/door", 1, customCallback)
 my_rpi.subscribe("security/motion", 1, customCallback)
 print("Connected to AWS!")

except Exception as e:
 print(e)
 print("Can't connect to AWS")

#Setting up all the sensors and actuators
#Room light button
GPIO.setup(16, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#Room light LED
GPIO.setup(18, GPIO.OUT)
#Air-Conditioning button
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#Air-Conditioning LED
GPIO.setup(23, GPIO.OUT)
continue_checking = True

def roomled():
 print("Room light: Ready")
 while continue_checking:
 button_state = GPIO.input(16)
 if button_state:
 if check_status(18) == "On":
 GPIO.output(18, False)
 time.sleep(0.2)
 else:
 GPIO.output(18, True)
 time.sleep(0.2)
 else:
 pass

def aircon():
 print("Air-Conditioning: Ready")
```

## Task

```

while continue_checking:
 button_state = GPIO.input(20)
 if button_state:
 if check_status(23) == "On":
 GPIO.output(23,False)
 time.sleep(0.2)
 else:
 GPIO.output(23,True)
 time.sleep(0.2)
 else:
 pass

def temp_sensor():
 print("Temperature sensor: On")
 while continue_checking:
 humidity, temperature = Adafruit_DHT.read_retry(11,4)
 print('Temp: {:.1f} C'.format(temperature))
 print('Humidity: {:.1f}'.format(humidity))
 timestamp = time.strftime("%Y/%m/%d %H:%M:%S", time.localtime())
 mqtt_message = "{\"timestamp\":\"" + str(timestamp) +
 "\",\"temperature\":\"" + str(temperature) + "\""
 my_rpi.publish("sensors/temperature", mqtt_message, 1)
 try:
 sql = "INSERT into house_temperature (temperature,humidity) VALUES
 (%f,%f)" % (temperature,humidity)
 curs.execute(sql)
 db.commit()
 except MySQLdb.Error as e:
 print e
 time.sleep(3)

def ledOn(gpio):
 GPIO.output(gpio,True)
 if gpio == 16:
 return "Air-Conditioning switched on"
 elif gpio == 20:
 return "Room light switched on"
 else:
 return 0

def ledOff(gpio):
 GPIO.output(gpio,False)
 if gpio == 16:
 return "Air-Conditioning switched off"
 elif gpio == 20:
 return "Room light switched off"
 else:
 return 0

def check_status(gpio):
 if GPIO.input(gpio):
 return 'On'
 else:
 return 'Off'

#Create the process to handle the home LEDs
home_leds = Process(target=roomled)
#Start the homeled process
home_leds.start()
#Create the process to handle the air-conditioning
air_con = Process(target=aircon)
#Start the air-conditioning process

```

## Task

```
air_con.start()
#Create the process to handle the temperature sensor
temp_sense = Process(target=temp_sensor)
#Start the temperature sensor process
temp_sense.start()
#Create the process to handle the publishing of light data
#publish_light = Process(target=pub_light, args=(adc,))
#Start the publish_light process
#publish_light.start()

app = Flask(__name__)

@app.route("/")
def index():
 return render_template('index.html')

@app.route("/checkLED1")
def checkled1():

 response = check_status(18)

 templateData = {
 'title' : 'Status of LED: ',
 'response' : response
 }

 return render_template('pin.html', **templateData)

@app.route("/checkLED2")
def checkled2():

 response = check_status(23)

 templateData = {
 'title' : 'Status of LED: ',
 'response' : response
 }

 return render_template('pin.html', **templateData)

@app.route("/readLED1/")
def readPin1():

 response = check_status(23)

 templateData = {
 'title' : 'Status of LED: ',
 'response' : response
 }

 return render_template('pin.html', **templateData)

@app.route("/writeLED1/<status>")
def writePin1(status):

 if status == 'On':
 response = ledOn(18)
 else:
 response = ledOff(18)
```

## Task

```

templateData = {
 'title' : 'Status of LED',
 'response' : response
}

return render_template('pin.html', **templateData)

@app.route("/readLED2/")
def readPin2():

 response = check_status(23)

 templateData = {
 'title' : 'Status of LED: ',
 'response' : response
 }

 return render_template('pin.html', **templateData)

@app.route("/writeLED2/<status>")
def writePin2(status):

 if status == 'On':
 response = ledOn(23)
 else:
 response = ledOff(23)

 templateData = {
 'title' : 'Status of LED',
 'response' : response
 }

 return render_template('pin.html', **templateData)

@app.route("/tempvalue")
def chart():
 import mysql.connector
 u, pw, h, db = 'root', 'root', 'localhost', 'calDB'
 data = []
 con = mysql.connector.connect(user=u,password=pw,host=h,database=db)
 print("Database successfully connected")
 cur = con.cursor()
 query = "SELECT datetimevalue, temperature FROM house_temperature ORDER BY
datetimevalue DESC LIMIT 10"
 cur.execute(query)
 for (datetimevalue, temperature) in cur:
 d = []
 d.append("{:%H:%M:%S}".format(datetimevalue))
 d.append(temperature)
 data.append(d)
 print(data)
 data_reversed = data[::-1]

 return render_template('tempvalue.html', data=data_reversed)

@app.route("/dooraccess")
def table():
 dynamodb = boto3.resource('dynamodb', aws_access_key_id="AKIAIIOPTSUQVWVDKYNNA",
aws_secret_access_key="Jtp2RBa9x9gTiAwPlzy2bHH+WtFpmAkFau0dx5O9", region_name='us-
west-2')

```

## Task

```

table = dynamodb.Table('DoorAccess')

response = table.scan()
value = []
for i in response['Items']:
 v = []
 v.append(json.dumps(i, cls=DecimalEncoder))
 value.append(v)
templateData = {
 'title': 'Status of LED',
 'response': value
}
return render_template('dooraccess.html', **templateData)

if __name__ == '__main__':
 print("Welcome to Jon's Smart Home System! Please wait.....")
 try:
 http_server = WSGIServer(('0.0.0.0', 8001), app)
 app.debug = True
 print("Website: Online")
 http_server.serve_forever()

 except:
 print("Website: Error!")
#Capture Keyboard Interrupt
def end_read(signal, frame):
 global continue_reading
 global continue_time
 global capture_motion
 global doorbell_check
 global continue_checking
 print("Stopping.....")
 curs.close()
 db.close()
 GPIO.cleanup()

signal.signal(signal.SIGINT, end_read)

```

## Task

- g) Create a python script **Outdoor.py** with the code below

```
sudo nano ~/ca2/Indoor.py
```

- h)
- ```

import RPi.GPIO as GPIO
import MFRC522
import signal
from gpiozero import LED
import time
import sys
from rpi_lcd import LCD
import picamera
from multiprocessing import Process
import datetime
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

```

Task

```
#Set mode to board
GPIO.setmode(GPIO.BOARD)

#Disable warnings
GPIO.setwarnings(False)

# Custom MQTT message callback
def customCallback(client, userdata, message):
    print("Received a new message: ")
    print(message.payload)
    print("from topic: ")
    print(message.topic)
    print("-----\n")

# Setting up connection to AWS
try:
    host = "aav1qj8erc0f4.iot.us-west-2.amazonaws.com"
    rootCAPath = "rootca.pem"
    certificatePath = "certificate.pem.crt"
    privateKeyPath = "private.pem.key"

    my_rpi = AWSIoTMQTTClient("Outdoor")
    my_rpi.configureEndpoint(host, 8883)
    my_rpi.configureCredentials(rootCAPath, privateKeyPath, certificatePath)

    my_rpi.configureOfflinePublishQueueing(-1) # Infinite offline Publish queueing
    my_rpi.configureDrainingFrequency(2) # Draining: 2 Hz
    my_rpi.configureConnectDisconnectTimeout(10) # 10 sec
    my_rpi.configureMQTTOperationTimeout(5) # 5 sec

    # Connect and subscribe to AWS IoT
    my_rpi.connect()
    my_rpi.subscribe("security/door", 1, customCallback)
    print("Connected to AWS!")

except:
    print("Can't connect to AWS")

#Setting up all the sensors and actuators
#Buzzer
GPIO.setup(40, GPIO.OUT)
doorbell_check = True
#Button
GPIO.setup(33, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#Motion Sensor
GPIO.setup(7, GPIO.IN)
capture_motion = True
#Set the motion value to 0
motion = 0
#LCD
lcd = LCD()
#Display welcome message
lcd.text("12W0L's Residence",1)
continue_time = True
#NFC Reader
mfrc522 = MFRC522.MFRC522()
continue_reading = True
#Door access LED
GPIO.setup(38, GPIO.OUT)

#LCD Clock Time Display
def lcd_time():
```

Task

```

print("LCD Display: On")
while continue_time:
    timestring = time.strftime("%H:%M:%S", time.gmtime())
    lcd.text(timestring, 2)
    time.sleep(1)

def motion_capture():
    print("Motion Sensor: On")
    camera = picamera.PiCamera()
    print("PiCamera: On")
    while capture_motion:
        motion = GPIO.input(7)
        if motion == 1:
            timestring = time.strftime("%Y-%m-%dT%H:%M:%S", time.gmtime())
            camera.capture('/home/pi/ca2/captures/photo_'+timestring+'.jpg')
            mqtt_message = "Motion Detected outside the house at " + timestring + " "
            my_rpi.publish("security/motion", mqtt_message, 1)
            print("Motion detected, image captured: " + timestring)
            time.sleep(5)

def doorbell():
    print("Doorbell: On")
    while doorbell_check:
        button_state = GPIO.input(33)
        if button_state:
            print("Somebody pressed the doorbell!")
            GPIO.output(40, True)
            time.sleep(1)
            GPIO.output(40, False)
        else:
            GPIO.output(40, False)

#Create process for the LCD time
lcd_timer = Process(target=lcd_time)
#Start the process to display the time
lcd_timer.start()
#Create process for the motion sensor
sense_motion = Process(target=motion_capture)
#Start the process to capture motion
sense_motion.start()
#Create process for the doorbell
door_bell = Process(target=doorbell)
#Start the doorbell process
door_bell.start()

print("Door Access System: On")
while continue_reading:
    (status, TagType) = mfrc522.MFRC522_Request(mfrc522.PICC_REQIDL)

    if status == mfrc522.MI_OK:
        (status, uid) = mfrc522.MFRC522_Anticoll()
        if uid == [136, 4, 91, 212, 3]:
            timestamp = time.strftime("%Y/%m/%d %H:%M:%S", time.localtime())
            mqtt_message = "{\"timestamp\": \"" + str(timestamp) +
            "\", \"doorAccess\": \"" + "granted" + "\"}"
            my_rpi.publish("security/door", mqtt_message, 1)
            print("Access Granted")
            lcd.text("Unlocked", 1)
            GPIO.output(38, GPIO.HIGH)
            time.sleep(1.5)
            GPIO.output(38, GPIO.LOW)
            lcd.text("Jon's Residence", 1)

```


Task

```

else:
    timestamp = time.strftime("%Y/%m/%d %H:%M:%S", time.localtime())
    mqtt_message = "{\"timestamp\": \"" + str(timestamp) +
    "\", \"doorAccess\": \"" + "denied" + "\"}"
    my_rpi.publish("security/door", mqtt_message, 1)
    lcd.text("Access Denied",1)
    time.sleep(1.5)

if __name__ == '__main__':
    print("Welcome to 12W0L's Smart Home System! Please wait.....")

#Capture Keyboard Interrupt
def end_read(signal, frame):
    global continue_reading
    global continue_time
    global capture_motion
    global doorbell_check
    global continue_checking
    print("Stopping.....")
    continue_reading = False
    continue_time = False
    capture_motion = False
    doorbell_check = False
    lcd.clear()
    GPIO.cleanup()

signal.signal(signal.SIGINT, end_read)

```

Task

- a) Extract all the files in the certs.zip file and put them in the ca2 folder using filezilla.

B. Coding the website

Task

- i) Make a new folder called templates in the ca2 folder.

```
mkdir ~/ca2/templates
```

Make a new folder called static in the ca2 folder.

```
mkdir ~/ca2/static
```

Create a html file in the templates folder called **index.html**.

```
Sudo nano ~/ca2/templates/index.html
```

Task

j)

```
<!DOCTYPE html>
<head>
  <link rel= "stylesheet" type= "text/css" href=
  "{{ url_for('static',filename='styles/main.css') }}">
  <title>{{ title }}</title>
  <script
  src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
  <script>

    function turnon1(){
      $.ajax({url: "writeLED1/On",
        success: function(result){
          $("#status1").html(result);
        }
      })
    }

    function turnoff1(){
      $.ajax({url: "writeLED1/Off",
        success: function(result){
          $("#status1").html(result);
        }
      })
    }

    function turnon2(){
      $.ajax({url: "writeLED2/On",
        success: function(result){
          $("#status2").html(result);
        }
      })
    }

    function turnoff2(){
      $.ajax({url: "writeLED2/Off",
        success: function(result){
          $("#status2").html(result);
        }
      })
    }

    function checkled1(){
      $.ajax({url: "checkLED1",
        success: function(result){
          $("#status1").html(result);
        }
      })
    }

    function checkled2(){
      $.ajax({url: "checkLED2",
        success: function(result){
          $("#status2").html(result);
        }
      })
    }

    $(document).ready(function(){
      $("#b1").click(function(){
```

Task

```

        turnon1();
    });
    $("#b2").click(function(){
        turnoff1();
    });

    $("#b3").click(function(){
        turnon2();
    });
    $("#b4").click(function(){
        turnoff2();
    });
});
checked1()
checked2()
</script>
</head>

<header>
<h1>Jon's Smart Home Web Interface</h1>
<div id="buttons">
<form action="tempvalue">
    <input type="submit" value="View Temperature Values" />
</form>
    <form action="dooraccess">
        <input type="submit" value="View Door Access attempts"/>
    </form>
</div>
</header>

<body>

<h2>Light switch</h2>
<button id="b1">Turn on lights</button>
<button id="b2">Turn off lights</button>
<h2 id="status1"></h2>
</br>
<h2>Air-con switch</h2>
<button id="b3">Turn on air-con</button>
<button id="b4">Turn off air-con</button>
<h2 id="status2"></h2>

</body>
</html>

```

Task

- a) Create a html file **tempvalue.html** with the code below
sudo nano ~/ca2/templates/tempvalue.html

- b)
- ```

<!doctype html>
<head>
 <link rel= "stylesheet" type= "text/css" href=
 "{{ url_for('static',filename='styles/main.css') }}">
 <title>Temperature Chart</title>
 <script type="text/javascript" src="https://code.jquery.com/jquery-

```

## Task

```

3.2.1.js"></script>

<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
 google.load('visualization', '1', {'packages':['corechart']});
 google.setOnLoadCallback(drawChart);

 function drawChart() {
 var data = new google.visualization.DataTable();
 data.addColumn('string', 'Time');
 data.addColumn('number', 'Temperature');
 data.addRows([
 {%- for timeinfo, valueinfo in data %}
 ['{{ timeinfo }}', {{ valueinfo }}],
 {%- endfor %}
]);

 var chart = new google.visualization.LineChart(
 document.getElementById('chart_div'));
 chart.draw(data, {legend: 'none', vAxis: {baseline: 0},
 colors: ['#A0D100']});
 }
</script>
</head>
<header>
 <h1>Jon's Smart Home Web Interface</h1>
 <div id="buttons">
 <form action="/">
 <input type="submit" value="Home" />
 </form>
 </div>
</header>
<body>

 <div id="content">
 <h1>Temperature values captured</h1>
 <div id="chart_div"></div>

 <div>
 <table border="1">
 {%- for timeinfo, valueinfo in data %}
 <tr>
 <td>{{ timeinfo }}</td><td>{{ valueinfo }}</td>
 </tr>
 {%- endfor %}
 </table>
 </div>
</div>
</body>
</html>

```

## Task

- a) Create a html file **dooraccess.html** with the code below

Task

```
sudo nano ~/ca2/templates/dooraccess.html
```

```
b) <!doctype html>
<head>
 <link rel= "stylesheet" type= "text/css" href=
 "{{ url_for('static',filename='styles/main.css') }}">
 <title>Door Access</title>
</head>
<header>
 <h1>12W0L's Smart Home Web Interface</h1>
 <div id="buttons">
 <form action="/">
 <input type="submit" value="Home" />
 </form>
 </div>
</header>
<body>

 <div id="content">
 <h1>Door Access values captured</h1>
 {{ response }}
 </div>
</body>
</html>
```

Task

a) Create a html file **pin.html** with the code below

```
sudo nano ~/ca2/templates/pin.html
```

```
b) <!DOCTYPE html>
<head>
 <title>{{ title }}</title>
</head>

<body>
 {{ response }}
</body>
</html>
```

## C. Adding styles to the website

Task

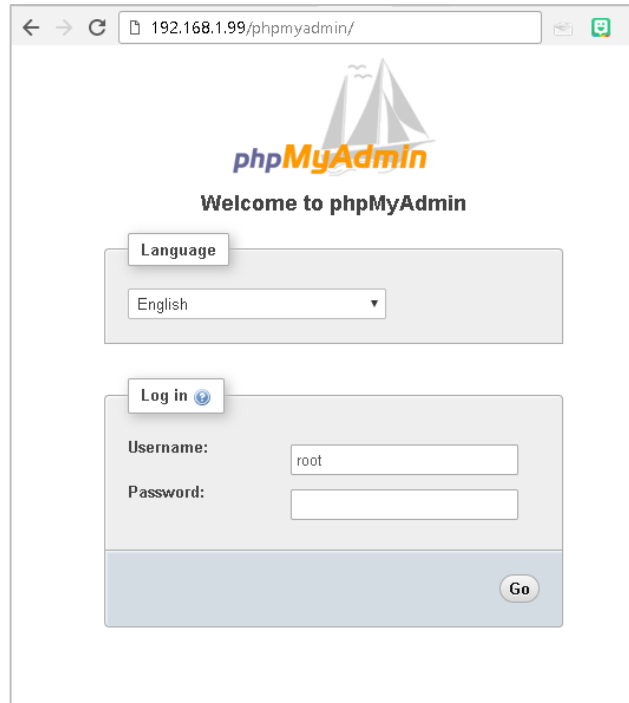
a) Extract all the files in the css.zip file and put them in the static folder using filezilla.

## D. Preparing the database

### Task

- a) Open a browser on your laptop and type in the following URL to access the **phpmyadmin** web interface where x.x.x.x represents the IP address of your Raspberry Pi.

x.x.x.x/phpmyadmin



### Task

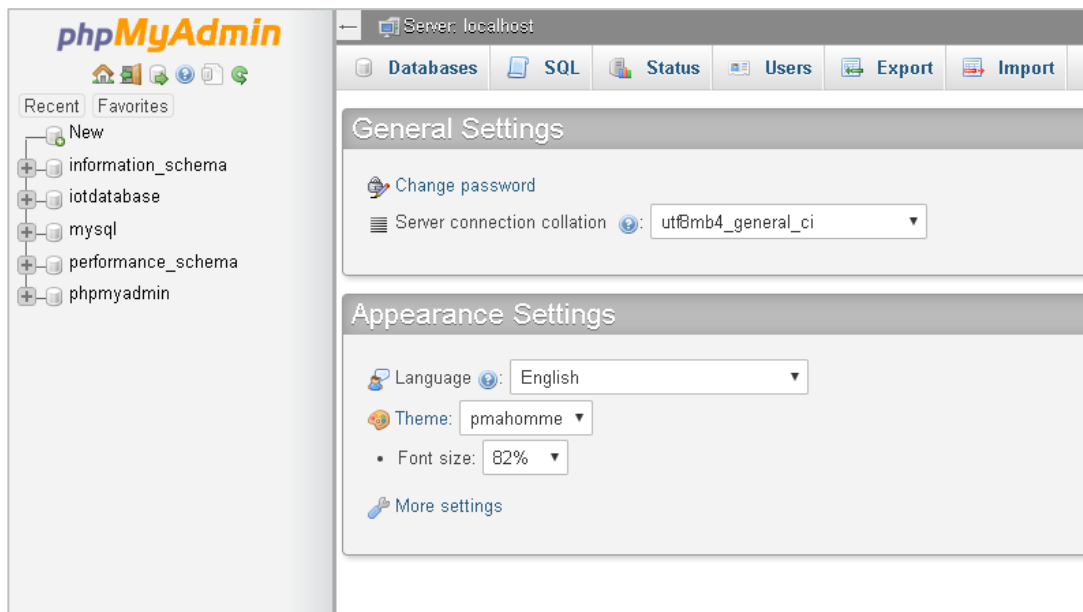
- b) Log-in the interface by using the password that was setup as follows

<b>Username</b>	root
<b>Password</b>	dmitiot

### Task

- c) Once you have logged in, you should see the phpmyadmin dashboard.

Task

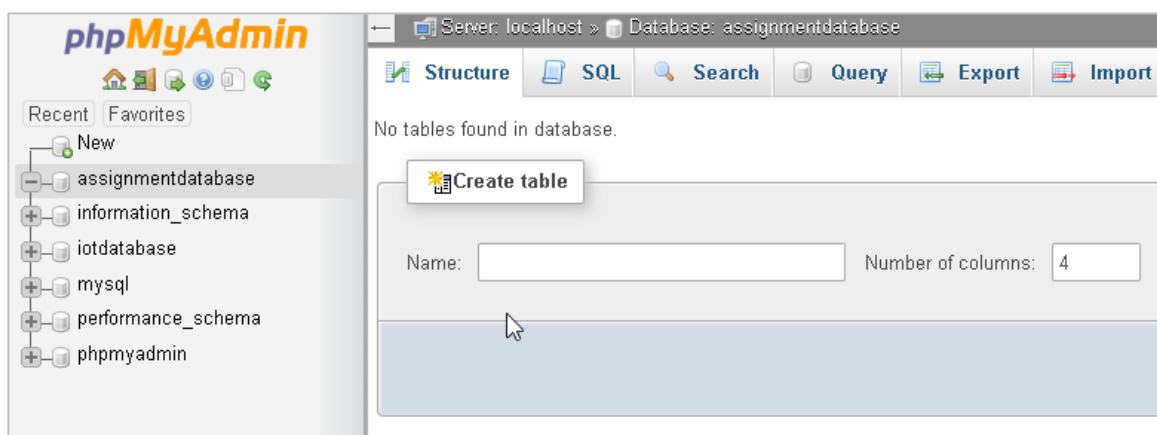


d Click on the “New button”  
)

e) Enter the name ca1DB for the database name and press ‘create’. If successful, you will be shown a success message.

f) After creating the database, you may proceed to create a new table inside this database.

Click on the new database on the navigation bar on the left. You will be shown a “Create table” page similar to this.



Task

g) Create the table house\_temperature and add in 3 columns.

Task

h ) The first column should be called "temperature", using decimal datatype. The second column should be called "humidity", using decimal datatype. The last column should be called datetimevalue and should have a datatype of timestamp.

<input type="checkbox"/>	1	<b>temperature</b>	decimal(10,0)	No	No
<input type="checkbox"/>	2	<b>humidity</b>	decimal(10,0)	No	No
<input type="checkbox"/>	3	<b>datetimevalue</b>	timestamp	on update CURRENT_TIMESTAMP	No

Task

i ) Under "Privileges", click "Add user". For username, type "ca1user", anyhost, for password, type "robots1234" then, make sure that the user is only granted all privileges on database "ca1DB", then click "go"

User	Host	Type	Privileges	Grant	Action
ca1user	%	database-specific	ALL PRIVILEGES	No	Edit Privileges

## Section 9 Running the programs

### A. Run the program

Task

a) Type in the following command to run your Python program

```
sudo python ~/ca2/Indoor.py
sudo python ~/ca2/Outdoor.py
```



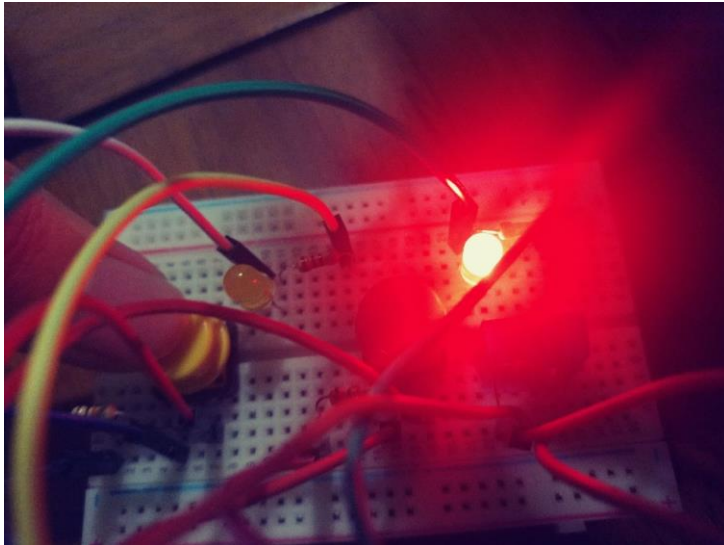
## B. Output of program

### Task

- a) When run, the program should display a series of text to show that each component is being run properly.

```
pi@raspberrypi-1625679-jontan:~ $ sudo python ~/ca1/ca1.py
Welcome to Jon's Smart Home System! Please wait.....
Successfully connected to database!
LCD Display: On
Website: Online
Door Access System: On
Doorbell: On
Motion Sensor: On
Temperature sensor: On
Room light: Ready
Air-Conditioning: Ready
PiCamera: On
```

- b) The Buzzer and home LEDs should be able to respond to the button presses.

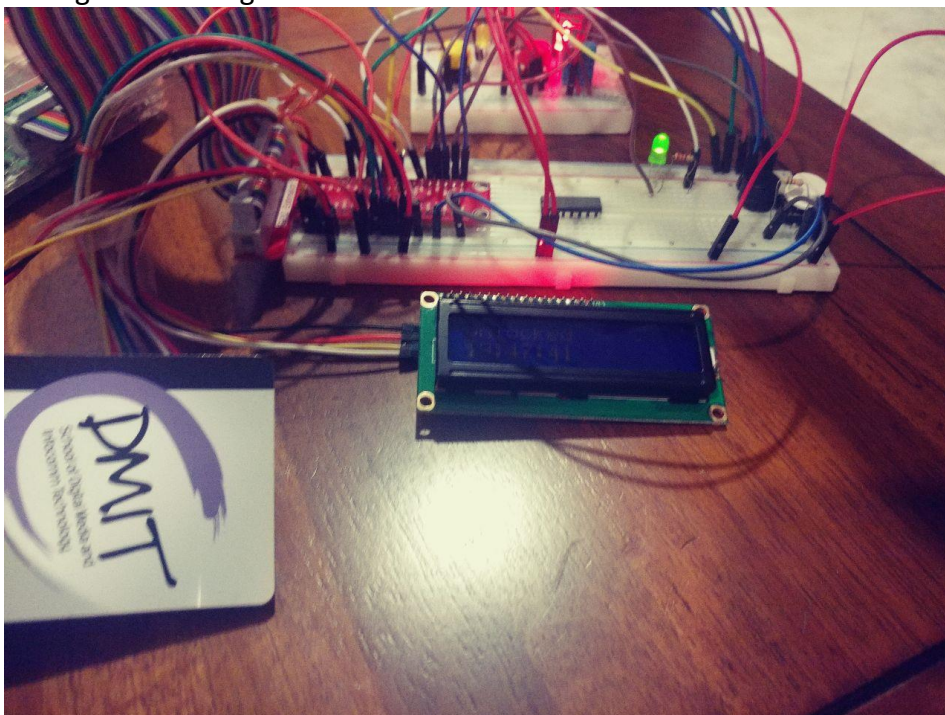


- c) The LCD Display should show “Jon’s Residence” and the time.

Task



- d) When the access card is scanned, the LCD screen should display “Unlocked” and the green LED light will turn green.



- e) The webpage should be up and running!

## Task

Index.html:

# JON'S SMART HOME WEB INTERFACE

View Temperature Values

## LIGHT SWITCH

Turn on lights

Turn off lights

OFF

## AIR-CON SWITCH

Turn on air-con

Turn off air-con

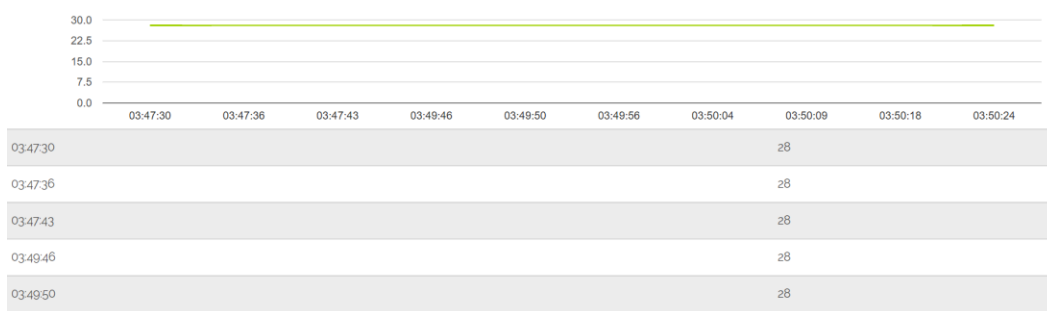
OFF

tempvalue.html

# JON'S SMART HOME WEB INTERFACE

Home

## TEMPERATURE VALUES CAPTURED



-- End of Smart Home System Step-by-step tutorial --