

Internet of Things Assignments

Project Title : Industry-specific intelligent fire management system

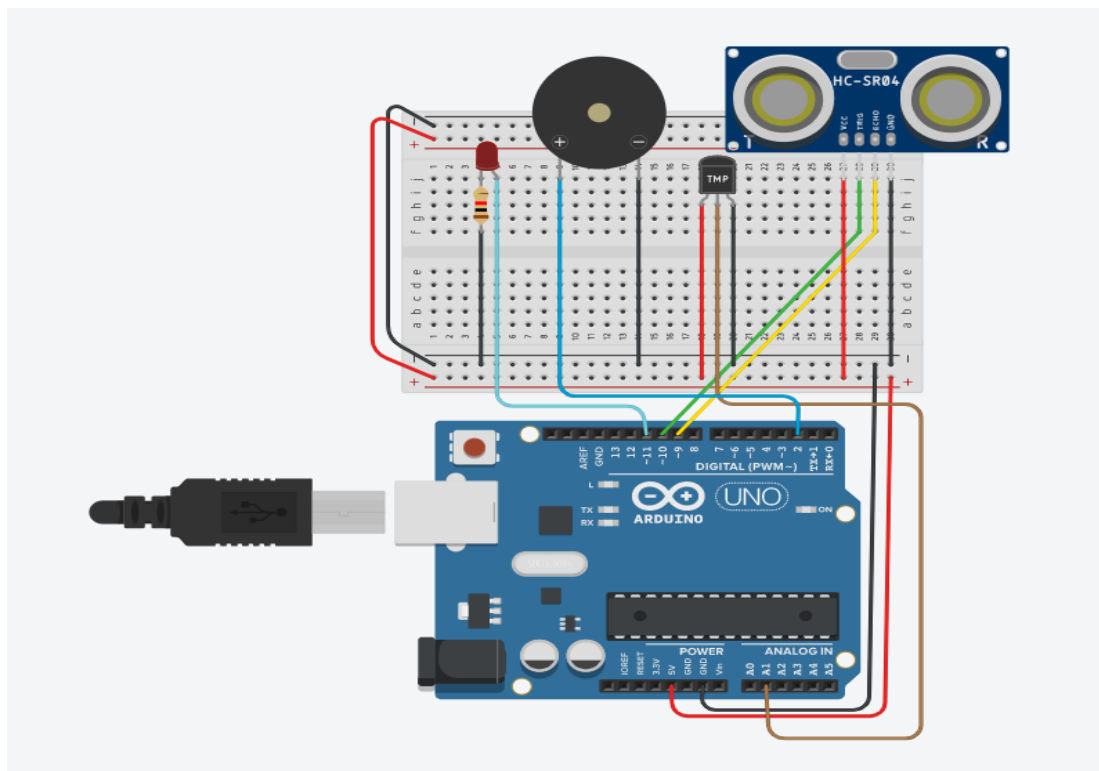
Team : **1901264**, 1901214, 1901267,1901071

Industry Mentor(s) Name : Mentor 11

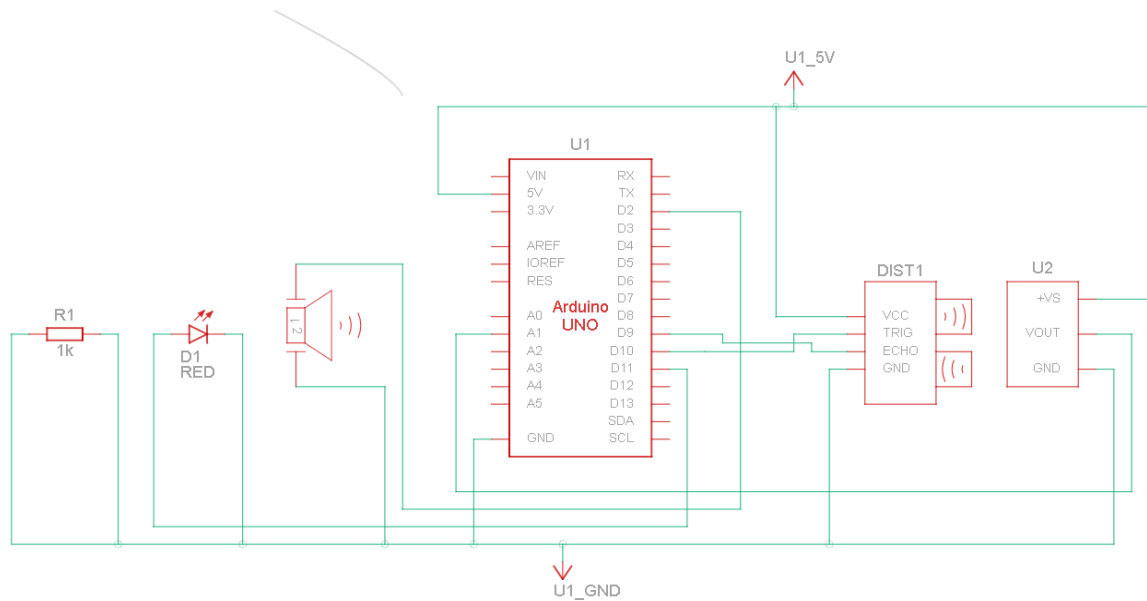
Faculty Mentor(s) Name : Mr SURESHKUMAR S

Assignment 1

Build a smart home in Tinkercad with 2 sensors, an Led, buzzer and submit it.



Schematic View -



Code-

```
int const trigPin = 10;
int const echoPin = 9;
int const buzzPin = 2;
int tempPin = A1;
#define led 11

void setup()
{
  pinMode(trigPin, OUTPUT); // trig pin will have pulses output
  pinMode(echoPin, INPUT); // echo pin should be input to get pulse width
  pinMode(buzzPin, OUTPUT); // buzz pin is output to control buzzing
  pinMode(led, OUTPUT);
}

void loop()
{
  int temp = map(((analogRead(tempPin) - 20) * 3.04), 0, 1023, -40, 125);
  Serial.print(temp);
  if (temp > 37){
    digitalWrite(led,HIGH);
  }
  else {
    digitalWrite(led,LOW);
  }
  int duration, distance;
  digitalWrite(trigPin, HIGH);
```

```
delay(1);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
if (distance <= 50 && distance >= 0) {
digitalWrite(buzzPin, HIGH);
} else {
digitalWrite(buzzPin, LOW);
}
delay(60);

}
```

Tinkercad Link –

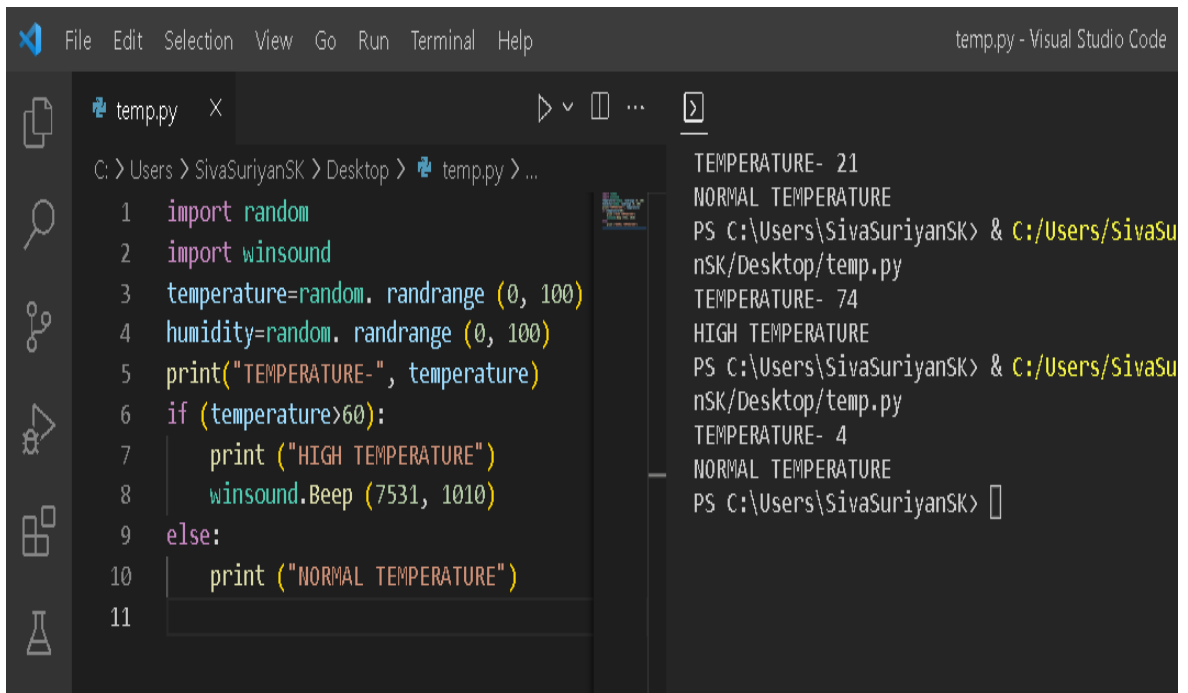
<https://www.tinkercad.com/things/7PKm2QtVuVT?sharecode=wCh-PjzANvXeK68Ez1rwh7Iv1Ado-7-jflbAGZxvN0o>

Assignment 2

Build a python code, assume you get temperature and humidity values (generated with a random function to a variable) and write a condition to detect an alarm in case of high temperature continuously.

Code –

```
import random
import winsound
temperature=random. randrange (0, 100)
humidity=random. randrange (0, 100)
print("TEMPERATURE-", temperature)
if (temperature>60):
    print ("HIGH TEMPERATURE")
    winsound.Beep (7531, 1010)
else:
    print ("NORMAL TEMPERATURE")
```



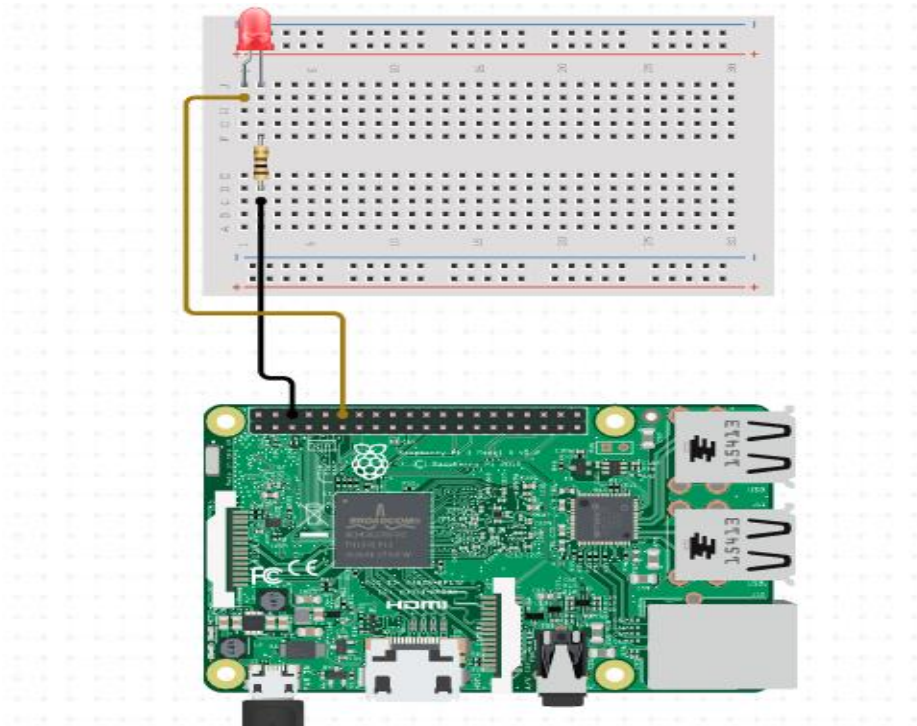
```
temp.py x
C: > Users > SivaSuriyanSK > Desktop > temp.py > ...
1 import random
2 import winsound
3 temperature=random. randrange (0, 100)
4 humidity=random. randrange (0, 100)
5 print("TEMPERATURE-", temperature)
6 if (temperature>60):
7     print ("HIGH TEMPERATURE")
8     winsound.Beep (7531, 1010)
9 else:
10    print ("NORMAL TEMPERATURE")
11

TEMPERATURE- 21
NORMAL TEMPERATURE
PS C:\Users\SivaSuriyanSK> & C:/Users/SivaSu
nSK/Desktop/temp.py
TEMPERATURE- 74
HIGH TEMPERATURE
PS C:\Users\SivaSuriyanSK> & C:/Users/SivaSu
nSK/Desktop/temp.py
TEMPERATURE- 4
NORMAL TEMPERATURE
PS C:\Users\SivaSuriyanSK> []
```

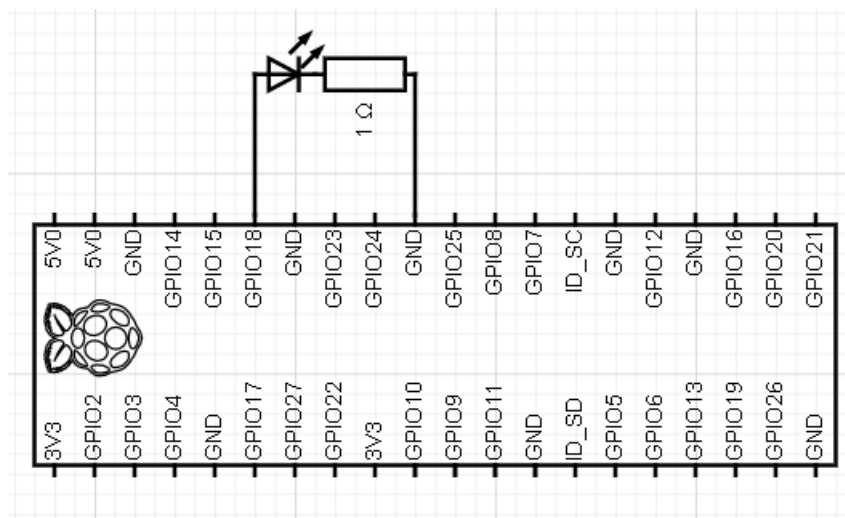
Assignment 3

Write python code for blinking LED and Traffic lights for Raspberry pi. Only python code is enough, no need to execute in raspberry pi. Note: you are allowed to use web search and complete the assignment.

Blinking LED –



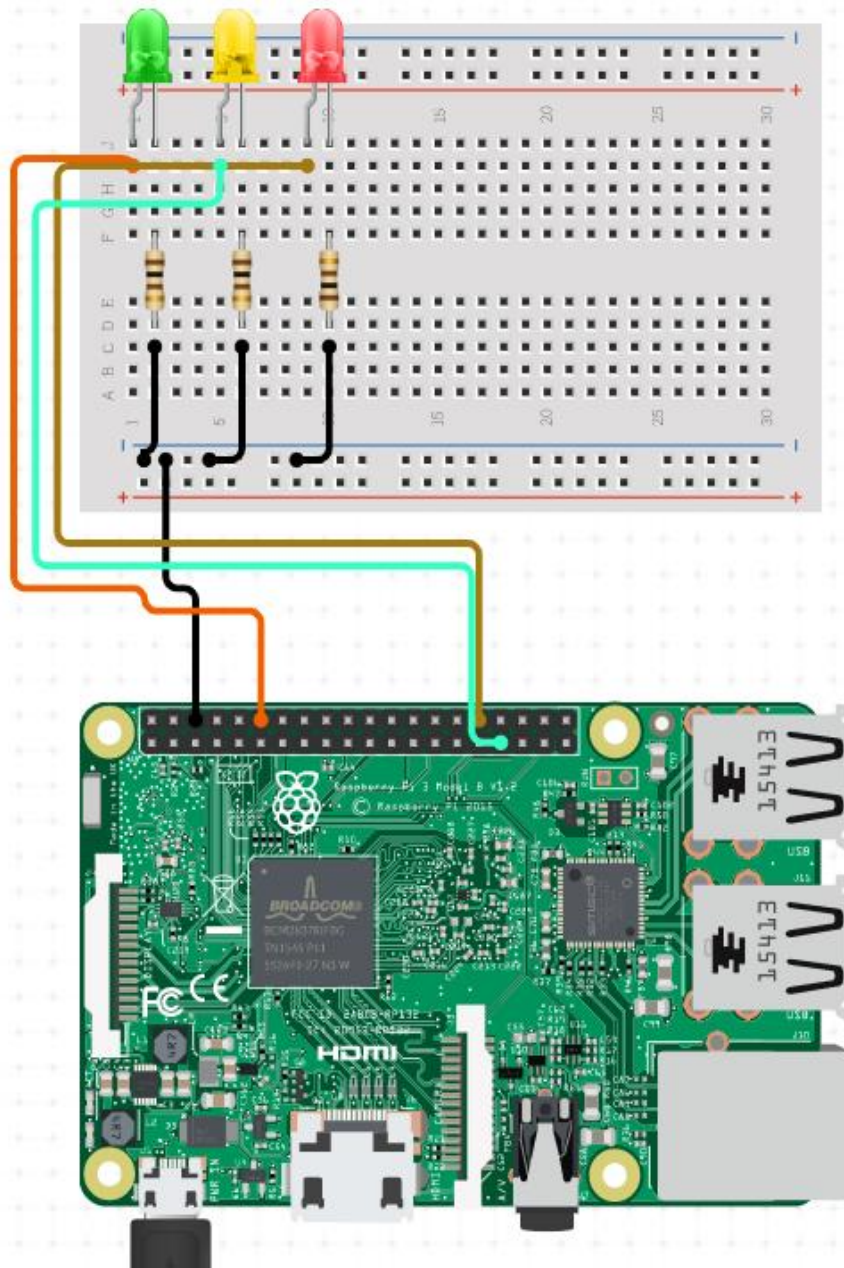
Schematic View -



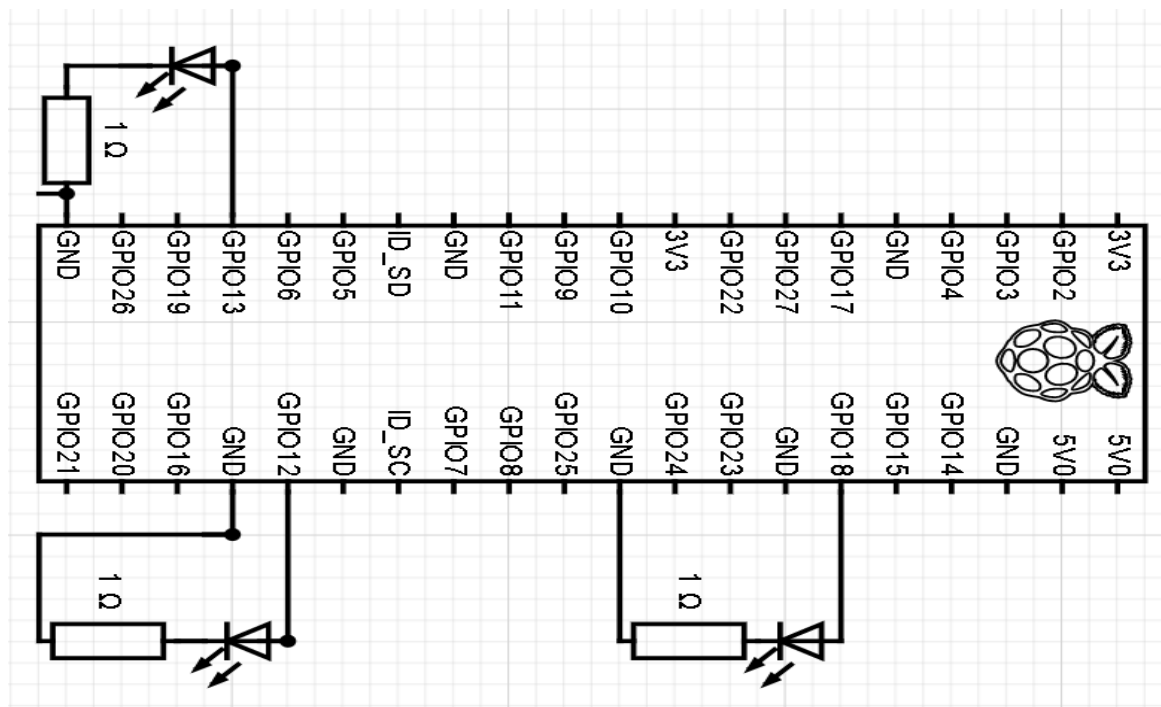
Code –

```
import RPi.GPIO as GPIO # RPi.GPIO can be referred as GPIO from now
import time
ledPin = 13 # pin22
def setup():
    GPIO.setmode(GPIO.BOARD) # GPIO Numbering of Pins
    GPIO.setup(ledPin, GPIO.OUT) # Set ledPin as output
    GPIO.output(ledPin, GPIO.LOW) # Set ledPin to LOW to turn Off the LED
def loop():
    while True:
        print 'LED on'
        GPIO.output(ledPin, GPIO.HIGH) # LED On
        time.sleep(1.0) # wait 1 sec
        print 'LED off'
        GPIO.output(ledPin, GPIO.LOW) # LED Off
        time.sleep(1.0) # wait 1 sec
def endprogram():
    GPIO.output(ledPin, GPIO.LOW) # LED Off
    GPIO.cleanup() # Release resources
if __name__ == '__main__': # Program starts from here
    setup()
    try:
        loop()
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the destroy() will be executed.
        endprogram()
```

Traffic Light –



Schematic View –



Code –

```
import RPi.GPIO as GPIO
import time
import signal
import sys

GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)

def allLightsOff(signal, frame):
    GPIO.output(9, False)
    GPIO.output(10, False)
    GPIO.output(11, False)
    GPIO.cleanup()
    sys.exit(0)
signal.signal(signal.SIGINT, allLightsOff)

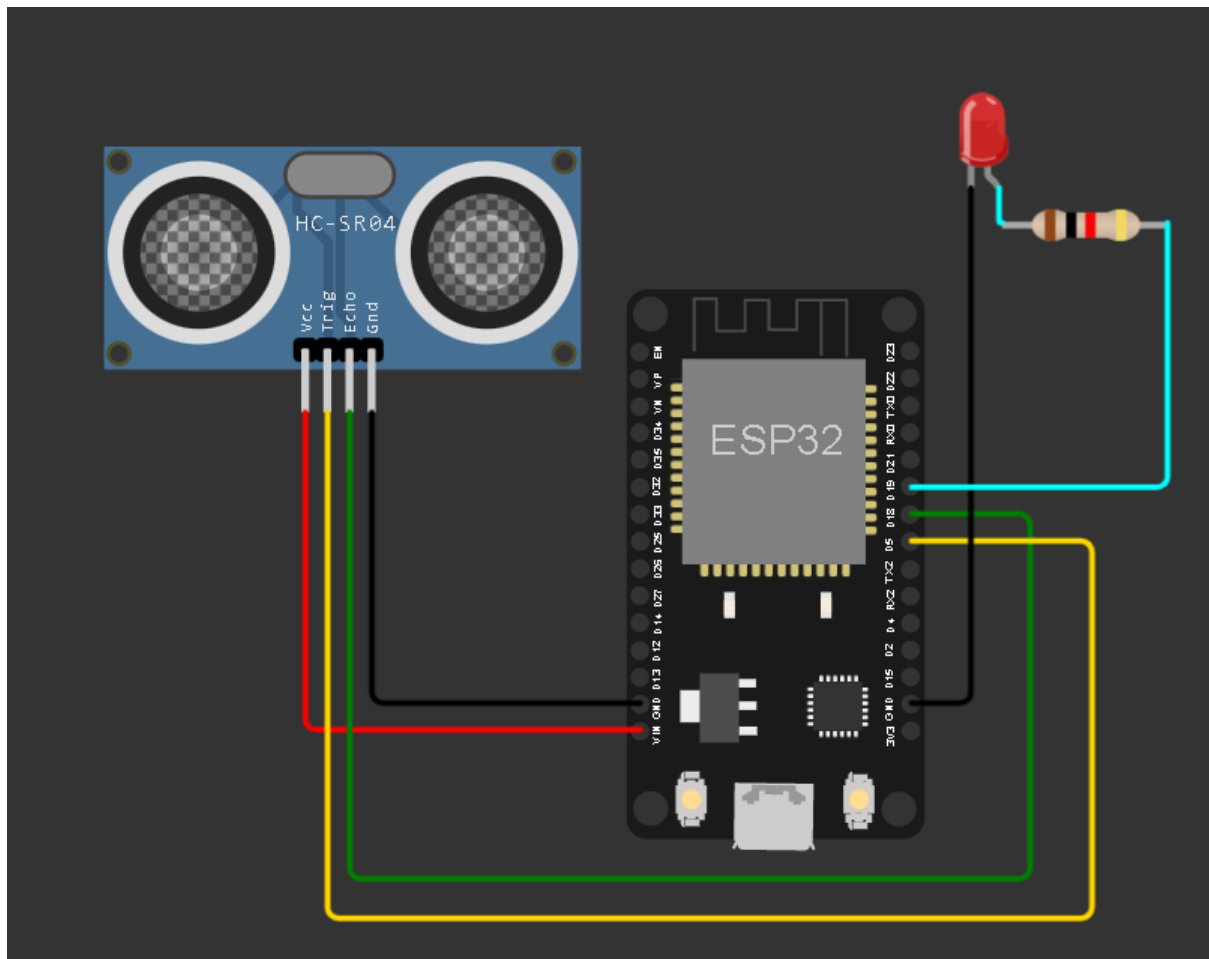
while True:
    GPIO.output(9, True)
    time.sleep(3)
    GPIO.output(10, True)
    time.sleep(1)
    GPIO.output(9, False)
    GPIO.output(10, False)
    GPIO.output(11, True)
    time.sleep(5)
```



```
GPIO.output(11, False)
    GPIO.output(10, True)
    time.sleep(2)
GPIO.output(10, False)
```

Assignment 4

Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events. Upload document with wokwi share link and images of IBM cloud.



Code –

```
#include "Ultrasonic.h"
#include <PubSubClient.h>
#include <WiFiClient.h>
#include <WiFi.h>

#define LED 19
#define ORG "9pdt70"
#define DEVICE_TYPE "abcd"
#define DEVICE_ID "12345"
#define TOKEN "123456789"
String data3;
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name  
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event  
perform and format in which data to be send
```

```
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT  
command type AND COMMAND IS TEST OF FORMAT STRING
```

```
char authMethod[] = "use-token-auth";// authentication method
```

```
char token[] = TOKEN;
```

```
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

```
WiFiClient wifiClient; // creating the instance for wificlient
```

```
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client  
id by passing parameter like server id,portand wificredential
```

```
Ultrasonic ultrasonic(5, 18);
```

```
int distance;
```

```
void setup() {  
  Serial.begin(9600);  
  wificonnect();  
  mqttconnect();  
}
```

```
void loop() {  
  // Pass INC as a parameter to get the distance in inches  
  //client.loop();
```

```
  distance = ultrasonic.read(CM);
```

```
  Serial.print("Distance in CM: ");  
  Serial.println(distance);  
  if(distance>100){  
    digitalWrite(LED,HIGH);  
    PublishData(distance);  
  }
```

```
  delay(1000);
```

```
}
```

```
void PublishData(int distance){  
  mqttconnect();
```

```
  String payload = "{\"distance\":";  
  payload += distance;  
  payload += "}";
```

```
  Serial.print("Sending payload: ");  
  Serial.println(payload);
```

```
  if (client.publish(publishTopic, (char*) payload.c_str())) {
```

```

    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will
    print publish ok in Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }
}

void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }

    // initManagedDevice();
    // Serial.println();
  }
}

void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

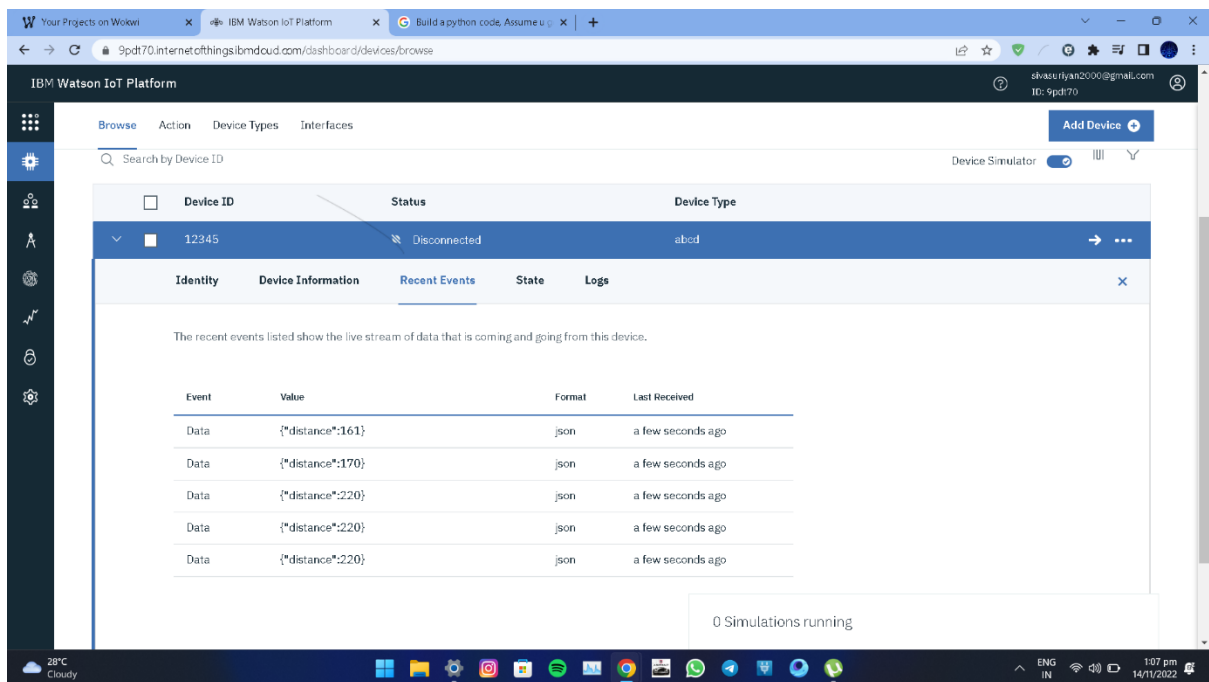
  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the
  connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {

```

```
Serial.println(data3);
digitalWrite(LED,HIGH);
}
else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}
data3="";
}
```

IBM Cloud –



The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar for 'Device ID' is present. The main content area displays a device with ID '12345' and status 'Disconnected'. The 'Recent Events' tab is selected, showing a table of events:

| Event | Value | Format | Last Received |
|-------|--------------------|--------|-------------------|
| Data | {\"distance\":161} | json | a few seconds ago |
| Data | {\"distance\":170} | json | a few seconds ago |
| Data | {\"distance\":220} | json | a few seconds ago |
| Data | {\"distance\":220} | json | a few seconds ago |
| Data | {\"distance\":220} | json | a few seconds ago |

At the bottom, a status bar indicates '0 Simulations running'.

Wokwi Link –

<https://wokwi.com/projects/348283645308437076>