IOT Assignment

**Sensors used**

1. **IR Sensor with Raspberry pi:**
   1. **Components:**
      * Raspberry Pi 4
      * IR sensor Module
      * LED light
      * Breadboard
      * Connecting wires
   2. **Connection:**
      * VCC ---------- 5V (Pin 2)
      * GND -------- GND (Pin 6)
      * DATA ------ GPIO 23 (Pin 11)
      * Cathode of LED ------- GND
      * Anode of LED -------- GPIO 24 (Pin 18)
   3. **Code:**

import RPi.GPIO as GPIO

import time

sensor = 16

led = 18

GPIO.setmode(GPIO.BOARD)

GPIO.setup(sensor,GPIO.IN)

GPIO.setup(led,GPIO.OUT)

try:

while True:

if GPIO.input(sensor):

GPIO.output(led, False)  
  
while GPIO.input(sensor):  
  
 time.sleep(0.2)

else:  
  
 GPIO.output(led,True)

except KeyboardInterrupt:

GPIO.cleanup()

* 1. **Result:** The LED lights up when an object is detected and it is off when no object is detected.

1. **Sound Sensor:**
   1. **Components used:**
      * Raspberry pi
      * Sound Sensor
      * Jumper wires
   2. **Connection:**
      * **VCC ---------- 5V**
      * **GND ----------- GND (Pin 6)**
      * **D0 ------- GPIO 12**
   3. **Code:**

**Import RPi.GPIO as GPIO**

**Import time**

**SENSOR = 12**

**GPIO.setmode(GPIO.BOARD)**

**GPIO.setup(SENSOR,GPIO.IN)**

**try:**

while True:

if GPIO.input ( SENSOR ) :

print (“Sound Detected”)

else:

print (“No Sound Detected” )

except KeyboardInterrupt :

GPIO.cleanup()

* 1. **Result:** The code prints “Sound Detected” when there is sound and prints “No Sound Detected” when there is no sound.

1. **LDR with Raspberry pi:**
   1. **Components used:**
      * Raspberry pi
      * LDR
      * Jumper Wires
   2. **Connection:**
      * VCC ------------ 5V
      * GND ------------- GND(Pin 6)
      * DATA ------------- Pin 16
   3. **Code:**

import RPi.GPIO as GPIO

sensor=16

GPIO.setmode(GPIO.BOARD)

GPIO.setup(sensor,GPIO.IN)

while True:

print( GPIO.input(sensor))

**Result:** The code gives output 1 when it is light and 0 when it is dark.

1. **Ultrasonic sensor:**
   1. **Components used:**
      * Raspberry pi 4
      * Ultrasonic sensor
      * Breadboard
      * Resistance
      * Jumper wires
   2. **Connection:**
      * Trig Pin ----- GPIO 11(pin 23)
      * Echo Pin ------ GPIO 12(Pin 32)
      * 5V --------------- 5V
      * GND ----------- GND
   3. **Code:**

#Raspberry Pi 4.0 code for interfacing with ultrasonic sensor

import RPI.GPIO as GPIO

import time

GPIO.setmode (GPIO.BCM)

TRIG PIN=11

ECHO PIN=12

GPIO.setup(TRIG\_PIN,GPIO.OUT)

GPIO.setup(ECHO\_PIN,GPIO.IN)

GPIO.OUTPUT(TRIG\_PIN,GPIO.LOW)

time.sleep(2)

GPIO.output(TRIG\_PIN,GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(TRIG\_PIN,GPIO.LOW)

while GPIO.input(ECHO\_PIN)==0:

pulse\_send=time.time()

while GPIO.input(ECHO\_PIN)==1:

pulse\_received=time.time()

pulse\_duration=pulse\_received - pulse\_send

distance = round(pulse\_duration \* 17150, 2)

print (f"Distance: {distance} cm")

GPIO.cleanup()

* 1. **Result:** It gives the distance of an object from the sensor.

1. **PIR Sensor:**
   1. **Components:**
      * Raspberry pi 4
      * PIR sensor
      * Jumper wires
   2. **Connection:**
      * VCC ----------- 5V (Pin 2)
      * DATA --------- GPIO 17 (Pin 11)
      * GND ----------- GND (Pin 6)
      * Cathode of LED ------ GND
      * Anode of LED --------- GPIO 2 (Pin 3)
   3. **Code:**

import RPi.GPIO as GPIO

import time

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11,GPIO.IN)

GPIO.setup(3,GPIO.OUT)

while True:

i=GPIO.input(11)

if i==0:

print(“No intruders”,i)

GPIO.output(3,0)

Time.sleep(0.1)

elif i==1:

print(“intruder detected”,i)

GPIO.output(3,1)

time.sleep(0.1)

* 1. **Result:** “Intruder detected” message is displayed movement is detected.

1. **Rain Sensor:**
   1. **Components:**
      * Raspberry pi 4
      * rain sensor
      * Jumper wires
   2. **Connection:**
   3. **Code:**

# raindrop sensor DO connected to GPI018 # HIGH = no rain, LOW = rain detected # Buzzer on GPI013

from time import sleep

from gpiozero import Buzzer, Input Device

buzz = Buzzer (13)

no\_rain = Input Device(18)

def buzz\_now(iterations):

for x in range(iterations):

buzz.on()

sleep(0.1)

buzz.off()

sleep(0.1)

while True:

if not no\_rain.is\_active:

print("It's raining - get the washing in!")

buzz\_now(5)

sleep(1)

* 1. **Result:** When water is sensed the message “Its raining - get the washing in!” is displayed.

1. **Sending data from Raspberry pi to AWS account**
   1. **Using EC2:**
      * Connect the raspberry pi with ssh. Write the following commands on Windows Power Shell terminal :

ssh <ip of raspberry pi>

ssh <username>

password

Through ssh, we are inside the raspberry pi

* + - auto\_update.html file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Auto update page</title>

</head>

<body>

    <center>

        <h1>Distance</h1>

        <h2 id = 'rondechaka'>0</h2>

    </center>

    <script>

        var element = document.getElementById("rondechaka");

        setInterval(function() {

            fetch('get\_data')

                .then(res => res.json())

                .then(data => element.innerHTML=data.distance)

        }, 500);

    </script>

</body>

</html>

* + - app.py file:

from flask import Flask,request,jsonify,render\_template

app = Flask(\_\_name\_\_)

gloabl\_variable = 0

@app.route("/")

def hello\_world():

    return render\_template('auto\_update.html')

@app.route("/get\_data")

def get\_data():

    return jsonify({'distance': gloabl\_variable})

@app.route('/set\_data',methods=['GET','POST'])

def set\_data():

    print(request.args.to\_dict())

    global gloabl\_variable

    gloabl\_variable = request.args.get('distance')

    return 'thank you'

if \_\_name\_\_=='\_\_main\_\_':

    app.run(host='0.0.0.0',debug= True)

* + - requesting.py file:

import requests

import random

import time

for i in range(100):

    x = requests.get(f'http://localhost:5000/set\_data?distance={random.randint(0,99)}')

    print(x)

    time.sleep(1)

* + - index.html file

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>simple web page</title>

</head>

<body>

    <h1>Hello world</h1>

</body>

</html>

Ultrasonic sensor.py file:

import RPi.GPIO as GPIO

import time

import requests

GPIO.setmode(GPIO.BCM)

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

GPIO.setup(GPIO\_TRIGGER , GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

    GPIO.output(GPIO\_TRIGGER, True )

    time.sleep(0.00001)

    GPIO.output(GPIO\_TRIGGER, False )

    starttime = time.time()

    stoptime = time.time()

    while GPIO.input(GPIO\_ECHO)==0:

        starttime = time.time()

    while GPIO.input(GPIO\_ECHO)==1:

        stoptime = time.time()

    timeescaped = stoptime - starttime

    distance = (timeescaped \* 34300) /2

    return distance

if \_\_name\_\_ == '\_\_main\_\_':

    try:

        while True:

            dist = distance()

            print(f'Measured distance = {round(dist)}')

            dictionary={'distance':dist}

            requests.get(f'http://34.222.69.68:5000/set\_data',params=dictionary)

            # add the requets code here + imported

            time.sleep(0.2)

    except KeyboardInterrupt:

        print('why you stopped?')

        GPIO.cleanup()

* + - Create a folder named IOT LAB. Copy the files index.html and ultrasonic sensor.py in that folder. Create a folder named flask\_app inside the IOT LAB. In the flask\_app folder paste the files app.py, requesting.py, auto\_update.html
    - Transfer the IOT LAB folder in Raspberry pi. Go to the IOT LAB directory. Go to flask\_app dir. Run the following commands:

sudo apt install python3-pip

pip 3 install flask

* + - login to AWS account----> Go to EC2 ----> Launch instance ----> Ubuntu ----> create a new key-pair(.pem)----> Create----> Launch instance
    - In AWS go to Security----> click on security group ----> inbound rules ----> edit inbound rules----> add rules TCP/IP ----> 0.0.0.0
    - Copy the IVP4 address and in localhost paste it and give the port (5000)
    - Write the command:

scp -i <keypair name> -r <filename> connect to instance using public DNS

python3 app.py

**Result:** Gives ultrasonic sensor data in EC2

**Using IOT Core Service:**

* Log in to AWS account
* Go to IOT Core Services
* Manage ----> All Device ----> Things -----> create things ----> create single thing---> give a thing name(Raspberry)-----> Create thing type---> give a name of thing type(pi) ----> next----> auto generate a new certificate ----> next ---> create policy ----> give a policy name(raspberrypolicy)----> Policy effect(allow)-----> Policy action(\*)----> Policy resource(\*) ----> Create ----> Create thing---> download Device Certificate, Public Key, Private Key ---> Done
* Create a folder named aws in desktop ----> paste the public key, private key and device certificate
* Go to Security ----> Certificates ----> select the certificate ----> Action ----> Attach to thing ----> select the thing that we want to attach ----> Attach to thing ----> select the certificate ----> Action ----> Attach policy ----> from dropdown select the policy ----> Attach policy
* Go to MQTT test client
* Go to Settings ----> in Device data endpoint copy the REST API URL
* We write a python code named pipython.py and keep it in aws folder created in desktop ----> paste the rootCA.pem file in aws folder.
* pipython.py file:

import time

import paho.mqtt.client as mqtt  
import ssl  
import json  
import thread  
import RPi.GPIO as GPIO  
  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(21, GPIO.OUT)  
  
def on\_connect(client, userdata, flags, rc):  
    print("Connected with result code "+str(rc))  
  
client = mqtt.Client()  
client.on\_connect = on\_connect  
client.tls\_set(ca\_certs='./rootCA.pem', certfile='./certificate name', keyfile='./private key file name', tls\_version=ssl.PROTOCOL\_SSLv23)  
client.tls\_insecure\_set(True)  
client.connect(“The REST API endpoint", 8883, 60)   
  
def intrusionDetector(Dummy):  
    while (1):  
        x=GPIO.input(21)  
        if (x==0):  
            print("Just Awesome")  
            client.publish("device/data", payload="Hello from BinaryUpdates!!" , qos=0, retain=False)  
        time.sleep(5)  
  
thread.start\_new\_thread(intrusionDetector,("Create intrusion Thread",))  
     
client.loop\_forever()

* Log into raspberry pi using ssh. Go to Windows Power Shell terminal and write : ssh <ip of raspberry pi> , ssh <username>, password of raspberry pi
* Transfer the aws folder in the raspberry pi and run the python script to send the message “Hello from BinaryUpdates!!” from Raspberry pi to AWS account
* Run the command sudo pip install paho-mqtt. It installs the mqtt library on raspberry pi
* Run the python code using the command : python pipython.py. Then Just Awesome message gets printed in every 5 sec.
* Go to MQTT test client in AWS ----> Subscribe to a topic ----> device/data ----> Subscribe
* We get Output: “Hello from BinaryUpdates!!” in each 5 sec.

1. **Interfacing NRF with Arduino:**
   1. **Connection for transmitter:**
      * 3.3 V ----- 3.3 V
      * GND ------ GND
      * CSN ------ Pin 10
      * CE ------- Pin 9
      * MOSI ----- Pin 11
      * SCK ------ Pin 13
      * MISO ------ Pin 12
   2. **Code for Transmitter**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte address[6] = "00001";

int button\_pin = 2;

boolean button\_state = 0;

void setup() {

pinMode(button\_pin, INPUT);

radio.begin();

radio.openWritingPipe(address);

radio.setPALevel(RF24\_PA\_MIN);

radio.stopListening();

}

void loop()

{

button\_state = digitalRead(button\_pin);

if(button\_state == HIGH)

{

const char text[] = "Your Button State is HIGH";

radio.write(&text, sizeof(text));

}

else

{

const char text[] = "Your Button State is LOW";

radio.write(&text, sizeof(text));

}

radio.write(&button\_state, sizeof(button\_state));

delay(1000); }

* 1. **Connection for Receiver:**
     + 3.3 V ----- 3.3 V
     + GND ------ GND
     + CSN ------ Pin 10
     + CE ------- Pin 9
     + MOSI ----- Pin 11
     + SCK ------ Pin 13
     + MISO ------ Pin 12
  2. **Code for Receiver:**

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

RF24 radio(9, 10); // CE, CSN

const byte address[6] = "00001";

boolean button\_state = 0;

int led\_pin = 3;

void setup() {

pinMode(6, OUTPUT);

Serial.begin(9600);

radio.begin();

radio.openReadingPipe(0, address);

radio.setPALevel(RF24\_PA\_MIN);

radio.startListening();

}

void loop()

{

if (radio.available())

{

char text[32] = "";

radio.read(&text, sizeof(text));

radio.read(&button\_state, sizeof(button\_state));

if(button\_state == HIGH)

{

digitalWrite(6, HIGH);

Serial.println(text);

}

else

{

digitalWrite(6, LOW);

Serial.println(text);}

}

delay(5);

}

1. **LED blinking using ESP8266:**
   1. **Components:**
      * ESP8266
      * Arduino IDE
   2. **Code:**
   3. #define LED D0 // Led in NodeMCU at pin GPIO16 (D0)
   4. void setup() {
   5. pinMode(LED, OUTPUT); // LED pin as output.
   6. }
   7. void loop() {
   8. digitalWrite(LED, HIGH);// turn the LED off.(Note that LOW is the voltage level but actually
   9. //the LED is on; this is because it is acive low on the ESP8266.
   10. delay(1000); // wait for 1 second.
   11. digitalWrite(LED, LOW); // turn the LED on.
   12. delay(1000); // wait for 1 second.
   13. }
   14. **Result:** The LED bulb blinks for 1 second and then turns off for another second and the process repeats itself.

# **Seven Segment Display with Esp8266:**

* 1. **Components:**
     + ESP8266 development board
     + Jumper cables
     + Arduino IDE
  2. **Connection:**
     + A ----- D6
     + B ----- GND
     + C ----- D5
     + D ----- D4
     + E ----- D3
     + F ------ D2
     + G ----- D1
     + SEG1 ----- RX
     + SEG2 ---- D0
  3. **Code:**

void setup() {

// initialize digital pin LED\_BUILTIN as an output.

pinMode(D0, OUTPUT);

pinMode(D1, OUTPUT);

pinMode(D3, OUTPUT);

pinMode(D4, OUTPUT);

pinMode(D2, OUTPUT);

pinMode(D5,OUTPUT);

pinMode(D6,OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

digitalWrite(D0, HIGH);

digitalWrite(D1, HIGH);

digitalWrite(D2, HIGH);

digitalWrite(D3, HIGH);

digitalWrite(D4, HIGH);

digitalWrite(D5, HIGH);

digitalWrite(D6, HIGH);// turn the LED on (HIGH is the voltage level)

delay(250); // wait for a second

digitalWrite(D0, LOW);

digitalWrite(D1, LOW);

digitalWrite(D2, LOW);

digitalWrite(D3, LOW);

digitalWrite(D4, LOW);

digitalWrite(D5, LOW);

digitalWrite(D6, LOW);

// turn the LED off by making the voltage LOW

delay(250);

// wait for a second

}

* 1. **Result:** We can see 8 blinking in the seven-segment display.