Sensors used

1. IR Sensor with Raspberry pi:

Components:

- Raspberry Pi 4
- IR sensor Module
- LED light
- Breadboard
- Connecting wires

Connection:

- VCC ---------- 5V (Pin 2/Pin 4)
- GND -------- GND (Pin 6)
- DATA ------ GPIO 23 (Pin 16)
- Cathode of LED ------- GND (Pin 20)
- Anode of LED -------- GPIO 24 (Pin 18)

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, True)



                else:

                        GPIO.output(led,False)

                time.sleep(0.2)
```

except KeyboardInterrupt:

GPIO.cleanup()

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

2. Sound Sensor:

Components used:

- Raspberry pi
- Sound Sensor
- Jumper wires

Connection:

- VCC ---------- 5V (Pin 2/Pin 4)
- GND ---------- GND (Pin 6)
- D0 ------- GPIO 23 (Pin 16)

Code:

```
Import RPi.GPIO as GPIO

Import time

SENSOR = 16

GPIO.setmode(GPIO.BOARD)

GPIO.setup(SENSOR,GPIO.IN)

try:

    while True:

        if  GPIO.input ( SENSOR ) :

            print (“Sound Detected”)

        else:

            print (“No Sound Detected” )
```

```
            time.sleep(0.2)
except KeyboardInterrupt :
    GPIO.cleanup()
```

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

3. LDR with Raspberry pi:

Components used:

- Raspberry pi
- LDR
- Jumper Wires

Connection:

- VCC ---------- 5V
- GND ---------- GND(Pin 6)
- DATA ---------- Pin 16

Code:

```
import RPi.GPIO as GPIO
import time
sensor=16
def rc_time (sensor):
    count = 0
    GPIO.setup
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.

4. Ultrasonic sensor:

Components used:

- Raspberry pi 4
- Ultrasonic sensor
- Breadboard
- Resistance
- Jumper wires

Connection:

- Trig Pin ----- GPIO 11(pin 23)
- Echo Pin ------ GPIO 12(Pin 32)
- 5V ---------- 5V
- GND ---------- GND

Code:

```
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()
```

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

5. PIR Sensor:

Components:

- Raspberry pi 4
- PIR sensor
- Jumper wires

Connection:

- VCC ---------- 5V (Pin 2)
- DATA --------- GPIO 23 (Pin 16)
- GND ---------- GND (Pin 6)
- Cathode of LED ------ GND
- Anode of LED --------- GPIO 2 (Pin 3)

Code:

```
import RPi.GPIO as GPIO
import time
sensor = 16
led = 3
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor,GPIO.IN)
GPIO.setup(led,GPIO.OUT)
while True:
        i=GPIO.input(sensor)
        if i==0:
```

```
        print("No intruders",i)
        GPIO.output(led,0)
        Time.sleep(0.1)
    elif i==1:
        print("intruder detected",i)
        GPIO.output(led,1)
        time.sleep(0.1)
```

Result: "Intruder detected" message is displayed movement is detected.

6. Rain Sensor:

Components:

- Raspberry pi 4
- rain sensor
- Jumper wires

Connection:

- VCC ---------- 5V (Pin 2/Pin 4)
- GND -------- GND (Pin 6)
- DATA ------ GPIO 23 (Pin 16)
- Cathode of Buzzer ------- GND (Pin 20)
- Anode of Buzzer -------- GPIO 24 (Pin 18)

Code:

```
import RPi.GPIO as GPIO
import time

sensor = 16
buzzer = 18

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(sensor, GPIO.IN)
```

```
GPIO.setup(buzzer, GPIO.OUT)


try:
    while True:
        i = GPIO.input(sensor)


        if i == 0:
            print("No rain", i)
            GPIO.output(buzzer, 0)
            time.sleep(0.1)


        elif i == 1:
            print("Rain detected", i)
            GPIO.output(buzzer, 1)
            time.sleep(0.1)


except KeyboardInterrupt:
    GPIO.cleanup()
```

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

7. Sending data from Raspberry pi to AWS account

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 ----> Actions ----> Attach to things ----> select the thing that we want to attach(raspberry) ----> Attach to thing ----> select the certificate ----> Actions ----> Attach policy ----> from dropdown select the policy (raspberrypolicy)----> Attach policies
- 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) #change the name of server

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.

8. Interfacing NRF with Arduino:

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

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); }
```

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

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);
}
```

9. LED blinking using ESP8266:

Components:

- ESP8266
- Arduino IDE

Steps:

- Open Arduino IDE, Open **preferences** window from Arduino IDE. Go to File -> Preferences.
- Enter the URL "http://arduino.esp8266.com/stable/package_esp8266com_index.json" into Additional Board Manager URLs field and click the "OK" button
- Open Boards Manager. Go to Tools -> Board -> Boards Manager
- Search for **ESP8266** and press install button for the "**ESP8266 by ESP8266 Community** "
- Choose your ESP8266 board from Tools > Board > Generic ESP8266 Module

Code:

```
#define LED D0 // Led in NodeMCU at pin GPIO16 (D0)

void setup() {
pinMode(LED, OUTPUT);    // LED pin as output.
}
void loop() {
digitalWrite(LED, HIGH);// turn the LED off.(Note that LOW is the voltage level but actually
                        //the LED is on; this is because it is acive low on the ESP8266.
```

delay(1000); // wait for 1 second.
digitalWrite(LED, LOW); // turn the LED on.
delay(1000); // wait for 1 second.
}

Result: The LED bulb blinks for 1 second and then turns off for another second and the process repeats itself.

10.Seven Segment Display with Esp8266:

Components:

- ESP8266 development board
- Jumper cables
- Arduino IDE

Connection:

- A ----- D6
- B ----- GND
- C ----- D5
- D ----- D4
- E ----- D3
- F ------ D2
- G ----- D1
- SEG1 ----- RX
- SEG2 ---- D0

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
}
```

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