Exp 2 IR SENSOR (HARDWARE AND SOFTWARE)

```
#include <IRremote.h>
const int RECV_PIN = 2;
void setup() {
  Serial.begin(9600);
  IrReceiver.begin(RECV_PIN);
  pinMode(13, OUTPUT);
  Serial.println("IR Receiver is ready...");
}
void loop() {
  if (IrReceiver.decode()) {
    Serial.print("IR Signal Received: ");
    Serial.println(IrReceiver.decodedIRData.decodedRawData, HEX);
    digitalWrite(13, HIGH);
    delay(2000);
    digitalWrite(13, LOW);
    IrReceiver.resume();
  }   }
```

Exp 3 iot application temp (SOFTWARE)

```
import dht
from machine import Pin
import time    import random
sensor = dht.DHT22(Pin(22))
led = Pin(5, Pin.OUT)
temp_threshold = 25
prev_temp = None
while True:
    try:
        sensor.measure()
        temperature = sensor.temperature() + random.uniform(-3, 3)
        if temperature != prev_temp:
            print(f"Temperature: {temperature:.2f}°C")
            print("-" * 30)
            prev_temp = temperature
        if temperature > temp_threshold:
            led.on()      time.sleep(0.2)
            led.off()     time.sleep(0.2)
        else:
            led.off()
    except OSError as e:
      print("Failed to read from DHT22 sensor:", e)         time.sleep(2)
```

exp 3 ldr iot application (SOFTWARE)

```
from machine import Pin, ADC

import time

ldr = ADC(26)     led = Pin(15, Pin.OUT)

threshold = 10000

while True:

   light_value = ldr.read_u16()


   print("LDR Value:", light_value)

   if light_value < threshold:

      print("Low light detected! Turning LED ON.")

      led.on()

   else:

      print("Bright light detected! Turning LED OFF.")

      led.off()

   time.sleep(1)
```

exp 5 TO CONNECT ARDUINO BOARD relay (SOFTWARE)

```
#include <Wire.h>       int myRelay = 8;
volatile byte relayState = LOW;
int myCount = 1;
void setup() {
   Serial.begin(9600);
   pinMode(myRelay, OUTPUT);
   digitalWrite(myRelay, LOW);
   Serial.println("MCP9808 Connected\nRELAY OFF");
}    void loop()    {
   float myTemp = 101;
   Serial.print(myCount++);
   Serial.print(" Temp: ");
   Serial.print(myTemp, 1);
   Serial.println("*F");
   if (myTemp > 100 && relayState == LOW) {
      digitalWrite(myRelay, HIGH);
      relayState = HIGH;
      Serial.println("RELAY ON");
      delay(2000);
      digitalWrite(myRelay, LOW);
      relayState = LOW;
      Serial.println("RELAY OFF");
}     delay(5000);     }
```

Exp 5 TO CONNECT ARDUINO BOARD   humidity (BOTH H & S)

```
#include <DHT.h>     #define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
int led = 13;
float hum;
void setup()
{
   Serial.begin(9600);
   dht.begin();
   pinMode(led, OUTPUT);
}
void loop() {
   hum = dht.readHumidity();
   Serial.print("Humidity: ");
   Serial.print(hum);
   Serial.println(" %");
   if (hum > 20)    {
      digitalWrite(led, HIGH);
      Serial.println(" HIGH HUMIDITY - LED ON");
   }    else   {
      digitalWrite(led, LOW);
      Serial.println(" NORMAL HUMIDITY - LED OFF");
   }   delay(2000);    }
```

Exp 5 TO CONNECT ARDUINO BOARD temp (SOFTWARE)

```
const int lm35Pin = A0;
const int buzzerPin = 2;    const int ledPin = 3;
void setup()  {
pinMode(lm35Pin, INPUT);
pinMode(buzzerPin, OUTPUT);
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}
void loop()
{
int sensorValue = analogRead(lm35Pin);
float temperature = (sensorValue * 0.48876);
Serial.print("Temperature: ");
Serial.print(temperature);
Serial.println(" °C");
if (temperature > 30.0) {

digitalWrite(buzzerPin, HIGH);
digitalWrite(ledPin, HIGH);
delay(2000);
digitalWrite(buzzerPin, LOW);
digitalWrite(ledPin, LOW);
}    delay(5000);    }
```

EXP 1 EXPLORE DIFFERENT COMMUNICATION METHODS (WRITTEN)

- **Interfacing Zigbee with Arduino**

```
void setup()    {
Serial.begin(9600);    }  void loop()  {
while (Serial.available() ) {
Serial.write(Serial.read());     }     }
```

- **Interfacing GSM Module with Arduino Code**

```
#include <SoftwareSerial.h>
SoftwareSerial SIM900A(10, 11);
void setup() {
   SIM900A.begin(9600);
   Serial.begin(9600);
   Serial.println("SIM900A Ready\nType 's' to send or 'r' to receive
an SMS");
}    void loop() {
   if (Serial.available()) {
      char cmd = Serial.read();
      if (cmd == 's') SendMessage();
      else if (cmd == 'r') ReceiveMessage();    }
   if (SIM900A.available()) Serial.write(SIM900A.read());      }
void SendMessage() {
   Serial.println("Sending Message...");
   SIM900A.println("AT+CMGF=1");     delay(500);
   SIM900A.println("AT+CMGS=\"911234567890\"");
```

```
  delay(500);
  SIM900A.println("Good morning, how are you doing?");
  SIM900A.write(26);
  delay(500);
  Serial.println("Message Sent");
}
void ReceiveMessage() {
  Serial.println("Reading SMS...");
  SIM900A.println("AT+CNMI=2,2,0,0,0");
  delay(500);   }
```

- <u>Arduino Bluetooth Controller</u>

```
char data = 0;
void setup()    {
Serial.begin(9600);
pinMode(13, OUTPUT); }
void loop()   {
if(Serial.available() > 0) {
data = Serial.read();
Serial.print(data);
Serial.print("\n");
if(data == '1')
digitalWrite(13, HIGH);
else if(data == '0')
digitalWrite(13, LOW);   }   }
```

4 b Blue Bot Platform (WRITTEN)

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11);
int buzzer = 13;
void setup() {
pinMode(buzzer, OUTPUT);
digitalWrite(buzzer, LOW);
BT.begin(9600);
}

void loop() {
if (BT.available()) {
char command = BT.read();
if (command == '1') {
digitalWrite(buzzer, HIGH);
}
else if (command == '0') {
digitalWrite(buzzer, LOW);
}
}
}
```

6. INTERFACE WITH ZIGBEE AND TRANSMIT SENSOR DATA TO OTHER NODE

```
#include <DHT.h>     (WRITTEN)
#define DHT_TYPE DHT11     #define DHT_PIN 2
void setup ()  {
// Initialize Zigbee module
// Setup Serial communication for Zigbee
}
void loop() {
float temperature = read Temperature ();
send Temperature Data(temperature);
delay (2000); // Adjust delay as needed
}     float read Temperature () {
// Implement temperature reading logic here using DHT library
}    void send Temperature Data (float temp) {
// Implement Zigbee transmission logic here
}
```

Here's a simple example:

```
void setup () {
Serial. Begin (9600);
}   void loop () {
if (Serial.available() > 0) {
char data = Serial.read();
Serial.print("Received: ");
Serial.println(data);     }   }
```

7. CONTROL YOUR HOME POWER OUTLET FROM ANYWHERE USING RASPBERRY PI, ZIGBEE AND ARDUINO (WRITTEN)

ARDUINO CODE:

```
#include <SoftwareSerial.h>
SoftwareSerial zigbeeSerial (2, 3);
void setup () {
zigbeeSerial.begin (9600);
pinmode (4, OUTPUT);
}
void loop () {
if (zigbeeSerial.available() > 0) {
char command = zigbeeSerial.read();
if (command == '1') {
digital Write (4, HIGH); // Turn on the relay
} else if (command == '0') {
Digital Write (4, LOW); // Turn off the relay
}   }  }
```

8. CONTROL YOUR HOME POWER OUTLET FROM ANYWHERE USING RASPBERRY PI, ZIGBEE AND ARDUINO (WRITTEN)

Device Registration (Node.js):

```
const Client = require('ibmiotf');
const deviceConfig = {
org: 'your-org-id',
id: 'your-device-id',
type: 'your-device-type',
authMethod: 'token',
authToken: 'your-auth-token'
};
const deviceClient = new Client.IotfDevice(device Config);
deviceClient.connect();
deviceClient.on('connect', function () {
console.log('Device connected to IBM IoT Platform');
});
deviceClient.on('command', function (commandName, format, payload, topic) {
console.log('Received command:', commandName, 'with payload:', payload);
});
```

Python to develop an IoT application

```
pip install ibmiotf
from ibmiotf import InternetOfThingsPlatform
```

```
org_id = "your-org-id"
device_type = "your-device-type"
device_id = "your-device-id"
auth_token = "your-auth-token"
# Create IoT Platform client
client = Internet of Things Platform ({
"org": org_id,
"type": device_type,
"id": device_id,
"auth-token": auth_token
})
# Connect to the IBM Watson IoT Platform
client. connect ()
# Define a function to handle incoming commands
def command_callback(cmd):
print("Received command: {} with payload: {}".format(cmd.command, cmd.payload))
# Subscribe to commands from the IoT Platform
client. command Callback = command_call back
# Send a sample event to the IoT Platform
data = {"temperature": 25, "humidity": 60}
client. publish Event ("sensorData", "json", data)
# Wait for incoming commands
client.deviceCommandLoop()
```