# Tempreture Sensor Using Thinkspeak

```
import network
import urequests
import time
import machine
import dht

# Wi-Fi credentials
SSID = 'Redmi'
PASSWORD = '11234567'

# ThingSpeak API key
THINGSPEAK_API_KEY = 'R4NXW9F7PN6XL8B0'  # Replace with your actual ThingSpeak API key
THINGSPEAK_URL = 'https://api.thingspeak.com/update'

# Set up the DHT sensor on GPIO15
sensor = dht.DHT11(machine.Pin(15))  # Use dht.DHT22 if you're using a DHT22 sensor

# Function to connect to Wi-Fi
def connect_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(SSID, PASSWORD)

    # Wait until connected
    while not wlan.isconnected():
        print("Connecting to Wi-Fi...")
        time.sleep(1)
    print("Connected to Wi-Fi")
    print("IP address:", wlan.ifconfig()[0])  # Display IP address

# Function to send data to ThingSpeak
def send_to_thingspeak(temperature, humidity):
    try:
        # Prepare HTTP GET request with ThingSpeak API and sensor data
```

```
        response = urequests.get(THINGSPEAK_URL +
                                 '?api_key=' + THINGSPEAK_API_KEY +
                                 '&field1=' + str(temperature) +
                                 '&field2=' + str(humidity))
        print("Data sent to ThingSpeak: ", response.text)
        response.close()
    except Exception as e:
        print("Failed to send data to ThingSpeak:", e)

# Main loop
def main():
    connect_wifi()

    while True:
        try:
            # Read the temperature and humidity from the DHT sensor
            sensor.measure()
            temperature = sensor.temperature()  # Temperature in Celsius
            humidity = sensor.humidity()       # Humidity percentage

            # Print values to the console
            print("Temperature: {}°C, Humidity: {}%".format(temperature,
humidity))

            # Send data to ThingSpeak
            send_to_thingspeak(temperature, humidity)

        except OSError as e:
            print("Failed to read sensor:", e)

        # Wait 15 seconds before the next reading
        time.sleep(15)

# Run the main function
main()
```

IDE:Thony pytho
conection

| ultrasonic sensor | pico |
|---|---|
| Dout | gp15 |
| Gnd | gnd |

# RASBERRY PI Blink light

**Program:**

```
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
GPIO.setnode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT, initial=GPIO.LOW)
while True:
GPIO.output(11,GPIO.HIGH)

time.sleep(1)
GPIO.output(11.GPIO.LOW)
Time.sleep(1).
```

**Using Rasberry pi OS**

**Connection**

| board | led |
|---|---|
| GPIO 6 | - |
| GPIO 11 | + |

# IR SESOR USING AURDINO WITHOUT THIKSPEAK

```
from machine import Pin
import time


ir_sensor = Pin(15, Pin.IN)
buzzer = Pin(16, Pin.OUT)

while True:
    if ir_sensor.value() == 0:
        print("Object detected!")
```

```
        buzzer.value(1)  # Turn the buzzer on
    else:
        print("No object detected")
        buzzer.value(0)  # Turn the buzzer off

    time.sleep(0.5)
```

**IDE:Thony python**
**Connectio**

| IR | PICO |
|---|---|
| Gnd | Gnd |
| D2 | GPIO15 |

| BUZZER | PICO |
|---|---|
| D3 | GPIO16 |
| Gnd | Gnd |

# HTTP REQUEST BLINK LIGHT

```
#include <ESP8266WiFi.h>

const char* ssid = "yourNetwork"; // Your WiFi SSID
const char* password = "yourPassword"; // Your WiFi password

const int LED = 16; // GPIO pin for the LED (D0 on most ESP8266 boards)

WiFiServer server(80); // Initialize the server on port 80

void setup() {
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW); // Turn the LED off initially

  // Connect to WiFi
  Serial.println("Connecting to WiFi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
```

```
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP())

  server.begin(); // Start the server
  Serial.println("Server started");
}

void loop() {
  WiFiClient client = server.available(); // Listen for incoming clients
  if (!client) {
    return;
  }

  Serial.println("New client connected");
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Check the request and control the LED
  int value = LOW;
  if (request.indexOf("/LED=ON") != -1) {
    digitalWrite(LED, HIGH);
    value = HIGH;
  }
  if (request.indexOf("/LED=OFF") != -1) {
    digitalWrite(LED, LOW);
    value = LOW;
  }

  // Send the response to the client
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println();
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.print("LED is: ");
  if (value == HIGH) {
    client.print("ON");
  } else {
    client.print("OFF");
  }
  client.println("<br>");
  client.println("<a href=\"/LED=ON\">Turn ON</a><br>");
  client.println("<a href=\"/LED=OFF\">Turn OFF</a><br>");
  client.println("</html>");

client.stop();
```

**Serial.println("Client disconnected");**
**}**

# Hardware Connections:

- **LED**:
  - **Positive** of the LED to **(D0)** on the ESP8266.
  - **Negative** of the LED to **GND** through an appropriate **current-limiting resistor** (e.g., 220Ω).

### Detailed Connection Steps:

1. **LED Anode** (long leg) to **GPIO16 (D0)** on the ESP8266.
2. **LED Cathode** (short leg) to one end of the **220Ω resistor**.
3. The other end of the **220Ω resistor** to **GND** on the ESP8266.

### Explanation:

- **GPIO16 (D0)** is configured as an output pin to control the LED.
- The resistor prevents excessive current from damaging the LED.

### WiFi Network Configuration:

- Replace `"yourNetwork"` with the SSID of your WiFi network.
- Replace `"yourPassword"` with the password for your WiFi network.

### Working of the Code:

1. The ESP8266 connects to the specified WiFi network and starts an HTTP server on port 80.
2. When a client connects and sends an HTTP request, the ESP8266 reads the request and checks for the `/LED=ON` or `/LED=OFF` command.
3. If `/LED=ON` is found in the request, the LED turns on. If `/LED=OFF` is found, the LED turns off.
4. The server responds with an HTML page containing links to turn the LED on and off.

### Accessing the Web Page:

- Once the ESP8266 is connected to WiFi, the IP address will be printed in the Serial Monitor.
- Open a web browser and type the IP address to access the control web page (e.g., `http://192.168.1.100`).
- Click the **"Turn ON"** or **"Turn OFF"** links to control the LED.

# SOIL MOISTURE SENSOR USING AURDINO

**void setup() {**

```
  pinMode(A0, INPUT);        // Configuring A0 as an input for analog reading
  pinMode(6, OUTPUT);        // Configuring pin 6 as an output for the LED or other device
  Serial.begin(9600);        // Initializing serial communication at 9600 baud
}

void loop() {
  int x = analogRead(A0);    // Reading the analog value from A0

  Serial.println(x);         // Printing the value to the serial monitor
  delay(1000);               // 1-second delay

  if (x < 300) {
    digitalWrite(6, HIGH);   // Turn on the LED if the reading is less than 300
  } else {
    digitalWrite(6, LOW);    // Turn off the LED otherwise
  }
}
```

## IDE: Aurdino

## Connection

**VCC of the sensor to 5V or 3.3V on the Arduino.**

- **GND** of the sensor to **GND** on the Arduino.
- **Output** of the sensor to **A0** (analog input) on the Arduino.

# INFRARED USING THINKSPEAK

```
#include <ESP8266WiFi.h>        // Include the ESP8266 Wi-Fi library
#include <ThingSpeak.h>         // Include the ThingSpeak library

// Replace with your network credentials
const char* ssid = "RAM";
const char* password = "123456789";

// Replace with your ThingSpeak channel details
const char* apiKey = "ZJZGS6I1LRZ90K7N";
const unsigned long channelID = 2635255;  // Replace with your channel ID
```

```
// Define pin constants
#define IR_SENSOR_PIN  11  // Digital input pin for the IR sensor

WiFiClient client;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize IR sensor pin as an input
  pinMode(IR_SENSOR_PIN, INPUT);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("Connected to Wi-Fi");

  // Initialize ThingSpeak
  ThingSpeak.begin(client);
}

void loop() {
  // Read the digital output from the IR sensor
  bool sensorValue = digitalRead(IR_SENSOR_PIN);

  // Print the sensor value to the serial monitor
  Serial.print("IR Sensor Output: ");
  Serial.println(sensorValue);

  // Update ThingSpeak
  ThingSpeak.setField(1, sensorValue);

  // Write to ThingSpeak
  int responseCode = ThingSpeak.writeFields(channelID, apiKey);

  if (responseCode == 200) {
    Serial.println("Data sent successfully.");
  } else {
    Serial.print("Failed to send data. Response code: ");
    Serial.println(responseCode);
  }

  // Add a delay before the next update
  delay(20000);  // ThingSpeak allows updates every 15 seconds, so 20 seconds is safe
}
```

**IDE: Aurdino Board**

**Connection:**

**Dout => ~11**
**Gnd => Gnd**

# UlTRASONIC SENSOR USING AURDINO

```
#include <Arduino.h>
const int TRIGGER_PIN = 10;
const int ECHO_PIN = 11;
const float SPEED_OF_SOUND = 343.0;
void setup() {
  Serial.begin(9600);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}
void loop() {
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  float distance = (duration * 0.5 * SPEED_OF_SOUND) / 10000.0;

 Serial.print("Distance: ");
  Serial.print(distance, 2);
  Serial.println(" cm");

  delay(1000);
}
```

**IDE:Aurdino**

**Board:Audino**

**connnection:**

**trig => ~11**
**echo => ~10**