**1.LED Blinking using Arduino**

**Components:** Arduino, LED, 220Ω resistor  
**Pin Connections:**

* LED Anode (+) → Digital Pin 13
* LED Cathode (–) → 220Ω Resistor → GND

**Code:**

void setup() {

pinMode(13, OUTPUT);

}

void loop() {

digitalWrite(13, HIGH); // Turn LED on

delay(1000);

digitalWrite(13, LOW); // Turn LED off

delay(1000);

}

**2. Temperature Sensing using LM35**

**Components:** Arduino, LM35 Sensor  
**Pin Connections:**

* LM35 VCC → 5V
* LM35 GND → GND
* LM35 OUT → A0

**Code:**

void setup() {

Serial.begin(9600);

}

void loop() {

int reading = analogRead(A0);

float voltage = reading \* (5.0 / 1023.0);

float temperatureC = voltage \* 100; // 10mV per °C

Serial.print("Temp: ");

Serial.print(temperatureC);

Serial.println(" °C");

delay(1000);

}

**3. Motion Detection using PIR Sensor**

**Components:** PIR Sensor, Arduino  
**Pin Connections:**

* PIR VCC → 5V
* PIR GND → GND
* PIR OUT → Digital Pin 2

**Code:**

void setup() {

pinMode(2, INPUT);

Serial.begin(9600);

}

void loop() {

int motion = digitalRead(2);

if (motion == HIGH) {

Serial.println("Motion Detected!");

} else {

Serial.println("No Motion");

}

delay(1000);

}

**4. Temperature Monitoring using LM35 and Buzzer**

**Components:** LM35, Buzzer  
**Pin Connections:**

* LM35 VCC → 5V
* LM35 GND → GND
* LM35 OUT → A0
* Buzzer + → Digital Pin 8
* Buzzer – → GND

**Code:**

int lm35Pin = A0;

int buzzerPin = 8;

float temperature;

float threshold = 30.0;

void setup() {

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

int sensorValue = analogRead(lm35Pin);

float voltage = sensorValue \* (5.0 / 1023.0);

temperature = voltage \* 100;

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.println(" °C");

if (temperature > threshold) {

digitalWrite(buzzerPin, HIGH);

Serial.println("Alert! High Temperature");

} else {

digitalWrite(buzzerPin, LOW);

}

delay(1000); // Wait 1 second before next reading

}

**5. Motion Detection using PIR Sensor and Buzzer**

**Components:** PIR Sensor, Buzzer  
**Pin Connections:**

* PIR VCC → 5V
* PIR GND → GND
* PIR OUT → Digital Pin 2
* Buzzer + → Digital Pin 8
* Buzzer – → GND

**Code:**

int pirPin = 2;

int buzzerPin = 8;

void setup() {

pinMode(pirPin, INPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

int motion = digitalRead(pirPin);

if (motion == HIGH) {

Serial.println("Motion Detected!");

digitalWrite(buzzerPin, HIGH);

delay(1000);

digitalWrite(buzzerPin, LOW);

} else {

Serial.println("No Motion.");

digitalWrite(buzzerPin, LOW);

}

delay(500);

}**6. Distance Measurement using Ultrasonic Sensor and Buzzer**

**Experiment Name:** Ultrasonic Distance Alert  
**Components:** HC-SR04 Ultrasonic Sensor, Buzzer  
**Pin Connections:**

* VCC → 5V
* GND → GND
* Trig → Digital Pin 9
* Echo → Digital Pin 10
* Buzzer + → Digital Pin 8
* Buzzer – → GND

**Code:**

long duration;

int distance;

void setup() {

pinMode(9, OUTPUT);

pinMode(10, INPUT);

pinMode(8, OUTPUT);

Serial.begin(9600);

}

void loop() {

digitalWrite(9, LOW);

delayMicroseconds(2);

digitalWrite(9, HIGH);

delayMicroseconds(10);

digitalWrite(9, LOW);

duration = pulseIn(10, HIGH);

distance = duration \* 0.034 / 2;

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (distance < 20) {

digitalWrite(8, HIGH);

} else {

digitalWrite(8, LOW);

}

delay(500);

}

**7. PIR and Ultrasonic Sensor Integration**

**Components:** PIR Sensor, HC-SR04, Buzzer  
**Pin Connections:**

* PIR VCC → 5V
* PIR GND → GND
* PIR OUT → Digital Pin 2
* HC-SR04 Trig → Pin 9
* HC-SR04 Echo → Pin 10
* HC-SR04 VCC → 5V
* HC-SR04 GND → GND
* Buzzer + → Pin 8
* Buzzer – → GND

**Code:**

int pirPin = 2;

int trigPin = 9;

int echoPin = 10;

int buzzerPin = 8;

int distanceThreshold = 20;

void setup() {

pinMode(pirPin, INPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

}

void loop() {

int motionDetected = digitalRead(pirPin);

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH);

int distance = duration \* 0.034 / 2;

Serial.print("Motion: ");

Serial.print(motionDetected);

Serial.print(" | Distance: ");

Serial.print(distance);

Serial.println(" cm");

if (motionDetected == HIGH || distance < distanceThreshold) {

digitalWrite(buzzerPin, HIGH);

} else {

digitalWrite(buzzerPin, LOW);

}

delay(500); // Wait before next check

}

**8.i. LED Fading using Raspberry Pi Pico**

**Components:** Raspberry Pi Pico, LED, 330Ω resistor  
**Pin Connections:**

* LED Anode → GP15 (PWM capable pin)
* LED Cathode → Resistor → GND

**Code :**

from machine import Pin, PWM

from time import sleep

led = PWM(Pin(15))

led.freq(1000)

while True:

for duty in range(0, 65535, 500):

led.duty\_u16(duty)

sleep(0.01)

for duty in range(65535, 0, -500):

led.duty\_u16(duty)

sleep(0.01)

**8.ii. LED Blinking using Loop (Multiple Blinks)**

**Components:** Raspberry Pi Pico, LED, 330Ω resistor  
**Pin Connections:**

* LED Anode → GP14
* LED Cathode → Resistor → GND

**Code :**

from machine import Pin

from time import sleep

led = Pin(14, Pin.OUT)

for i in range(10):

led.toggle()

sleep(0.5)

**8.iii. LED Blinking using Raspberry Pi Pico**

**Components:** Raspberry Pi Pico, LED, 330Ω resistor  
**Pin Connections:**

* LED Anode → GP13
* LED Cathode → Resistor → GND

**Code :**

from machine import Pin

from time import sleep

led = Pin(13, Pin.OUT)

while True:

led.on()

sleep(1)

led.off()

sleep(1)

**All in One**

**Pin Connections:**

* LED 1 Anode → GP15
* LED 2 Anode → GP14
* LED 3 Anode → GP13
* LED Cathode → GND

**Code :**

from machine import Pin, PWM

from time import sleep

led\_fade = PWM(Pin(15))

led\_blink\_loop = Pin(14, Pin.OUT)

led\_blink = Pin(13, Pin.OUT)

led\_fade.freq(1000)

print("Starting LED Fading on GP15...")

for i in range(2):

for duty in range(0, 65535, 1000):

led\_fade.duty\_u16(duty)

sleep(0.01)

for duty in range(65535, 0, -1000):

led\_fade.duty\_u16(duty)

sleep(0.01)

led\_fade.duty\_u16(0)

print("Blinking LED on GP14 - 10 times...")

for i in range(10):

led\_blink\_loop.toggle()

sleep(0.5)

led\_blink\_loop.value(0)

print("Starting continuous blinking on GP13...")

while True:

led\_blink.on()

sleep(1)

led\_blink.off()

sleep(1)

**9. Motion Detection using PIR Sensor and LED**

**Components:** PIR Sensor, LED  
**Pin Connections:**

* PIR VCC → 3.3V
* PIR GND → GND
* PIR OUT → GP14
* LED Anode → GP25
* LED Cathode → Resistor → GND

**Code :**

from machine import Pin

from time import sleep

pir = Pin(14, Pin.IN)

led = Pin(25, Pin.OUT)

while True:

if pir.value() == 1:

print("Motion Detected!")

led.on()

else:

print("No Motion")

led.off()

sleep(0.5)

**10. Ultrasonic Sensor and LED using Raspberry Pi Pico**

**Components:** HC-SR04 Ultrasonic Sensor, LED  
**Pin Connections:**

* HC-SR04 VCC → 5V
* GND → GND
* Trig → GP3
* Echo → GP2
* LED Anode → GP4
* LED Cathode → Resistor → GND

**Code :**

from machine import Pin, time\_pulse\_us

from time import sleep

trigger = Pin(3, Pin.OUT)

echo = Pin(2, Pin.IN)

led = Pin(4, Pin.OUT)

def get\_distance():

trigger.low()

sleep(0.002)

trigger.high()

sleep(0.00001)

trigger.low()

duration = time\_pulse\_us(echo, 1, 30000)

distance\_cm = (duration / 2) / 29.1

return distance\_cm

while True:

dist = get\_distance()

print("Distance:", dist, "cm")

if dist < 20:

led.on()

else:

led.off()

sleep(0.5)

**11. Simple DHT11/DHT22 Code with ESP8266**

**Connections (DHT11/DHT22 to ESP8266):**

| DHT Pin | Connect to ESP8266 |
| --- | --- |
| VCC | 3.3V |
| GND | GND |
| DATA | D4 (GPIO 2) |

**Code:**

#include <DHT.h>

#define DHTPIN D4

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

}

void loop() {

float humidity = dht.readHumidity();

float temperature = dht.readTemperature();

if (isnan(humidity) || isnan(temperature)) {

Serial.println("Failed to read from DHT sensor!");

} else {

Serial.print("Temperature: ");

Serial.print(temperature);

Serial.print(" °C, Humidity: ");

Serial.print(humidity);

Serial.println(" %");

}

delay(2000);

}

**12. LED Control with ESP8266 and Blynk (New Blynk 2.0)**

**Pin Connections:**

| LED Pin | ESP8266 (NodeMCU) |
| --- | --- |
| Anode (+) | D2 (with 330Ω resistor) |
| Cathode (–) | GND |

**Code:**

#define BLYNK\_TEMPLATE\_ID "YourTemplateID"

#define BLYNK\_TEMPLATE\_NAME "YourProjectName"

#define BLYNK\_AUTH\_TOKEN "YourAuthToken"

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

char auth[] = BLYNK\_AUTH\_TOKEN;

char ssid[] = "YourWiFiName";

char pass[] = "YourWiFiPassword";

int ledPin = D2; // GPIO 4 (D2 on NodeMCU)

BLYNK\_WRITE(V0) {

int pinValue = param.asInt();

digitalWrite(ledPin, pinValue);

}

void setup() {

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

Serial.begin(9600);

Blynk.begin(auth, ssid, pass);

}

void loop() {

Blynk.run();

}

**13. MQTT Send & Receive Code (ESP8266)**

**The LED connected to pin D1 (GPIO 5)**

**Code:**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

const char\* ssid = "YourWiFiName";

const char\* password = "YourWiFiPassword";

const char\* mqtt\_server = "broker.hivemq.com";

WiFiClient espClient;

PubSubClient client(espClient);

const int ledPin = 5; // D1 on NodeMCU

void callback(char\* topic, byte\* payload, unsigned int length) {

String message;

for (int i = 0; i < length; i++) {

message += (char)payload[i];

}

Serial.print("Message received: ");

Serial.println(message);

if (message == "ON") {

digitalWrite(ledPin, HIGH);

} else if (message == "OFF") {

digitalWrite(ledPin, LOW);

}

}

void setup\_wifi() {

delay(10);

Serial.println("Connecting to WiFi...");

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWiFi connected");

}

void reconnect() {

while (!client.connected()) {

Serial.print("Connecting to MQTT...");

if (client.connect("ESPClient")) {

Serial.println("connected");

client.subscribe("led/control");

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

delay(2000);

}

}

}

void setup() {

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

Serial.begin(9600);

setup\_wifi();

client.setServer(mqtt\_server, 1883);

client.setCallback(callback);

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

}

**14. Offline Emergency Alert Bracelet Simulation**

**🔧 Circuit:**

| **Component** | **Arduino Pin** |
| --- | --- |
| **Button** | **D2** |
| **LED** | **D3** |
| **Buzzer** | **D4** |

**Code:**

const int buttonPin = 2;

const int ledPin = 3;

const int buzzerPin = 4;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

digitalWrite(ledPin, LOW);

digitalWrite(buzzerPin, LOW);

}

void loop() {

int buttonState = digitalRead(buttonPin);

if (buttonState == LOW) {

digitalWrite(ledPin, HIGH);

digitalWrite(buzzerPin, HIGH);

delay(1000);

} else {

digitalWrite(ledPin, LOW);

digitalWrite(buzzerPin, LOW);

}

}