****

**Internet Of Things (IoT)**

**Lab Experiment Report**

**Course Code: 24ECAC304**

**By**

**ADITYA KOLI 02FE22BCI004**

**Roll No. 03 Div: D**

**UNDER THE GUIDANCE OF**

**Prof. Salma Shahapur**



**DEPARTMENT OF**

**COMPUTER SCIENCE ENGINEERING**

**K. L. E. Dr. M. S. SHESHGIRI**

**COLLEGE OF ENGINEERING AND TECHNOLOGY,**

**BELAGAVI– 590008**

**2024-25**

****

**Roll No.: 22 USN No. 02FE22BCI025**

**Department of Computer Science and Engineering**

**2024-25**

**Laboratory Certificate**

**This is to certify that Mr. Soham Mali has satisfactorily completed the course of experiments in Internet of Things (IoT) practical prescribed by the Kle Technological University, for 5th Semester in the Laboratory of this University during the Year 2024-2025.**

**Staff-in Charge HOD**

**Final C.I.E. Marks: \_\_\_\_\_\_\_\_\_\_\_**

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **EXPERIMENT NUMBER** | **EXPERIMENT TOPIC** | **Page Number** |
| 1 | Experiment No. 1 | Blinking of a LED light | 04 |
| 2 | Experiment No. 2 | Relay Controlled Bulb | 06 |
| 3 | Experiment No.3 | Controlling the DC motor using L298 | 08 |
| 4 | Experiment No.4 | Integrating the IR sensor | 10 |
| 5 | Experiment No.5 | Passive Infrared Sensor (PIR) | 12 |
| 6 | Experiment No.6 | Ultrasonic Sensor for Distance Measurement (HC-SR04) | 14 |
| 7 | Experiment No.7 | DHT11/DHT22 Temperature and  Humidity Sensor | 16 |
| 8 | Experiment No.8 | Pulse Rate Sensor (Heart Rate  Sensor) | 18 |

### **Experiment 1: LED Blink**

#### **Aim**

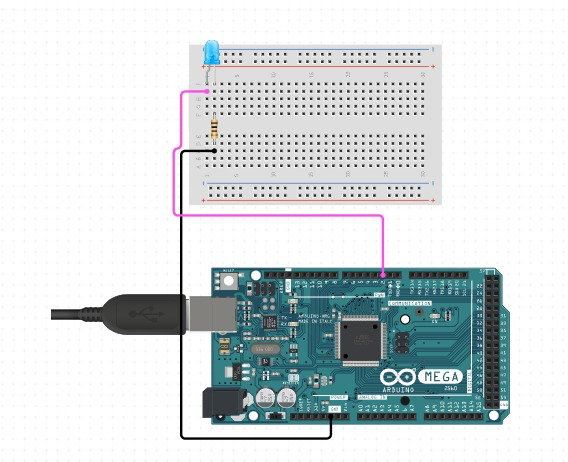
To interface an LED with a microcontroller and make it blink at regular intervals.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* LED
* Resistor (220Ω)
* Breadboard
* Jumper wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing an LED connecting to pin 13 of the microcontroller via a resistor and gorund.



* **Code**

// LED Blink Example Code

const int ledPin = 13; // Pin where LED is connected

void setup() {

pinMode(ledPin, OUTPUT); // Set pin as output

}

void loop() {

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

delay(1000); // Wait for 1 second

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

delay(1000); // Wait for 1 second

}

#### **Results/Conclusions**

The LED successfully blinked at an interval of 1 second, demonstrating the basic working of GPIO pins on a microcontroller.

### **Experiment 2: Relay Using Bulb**

#### **Aim**

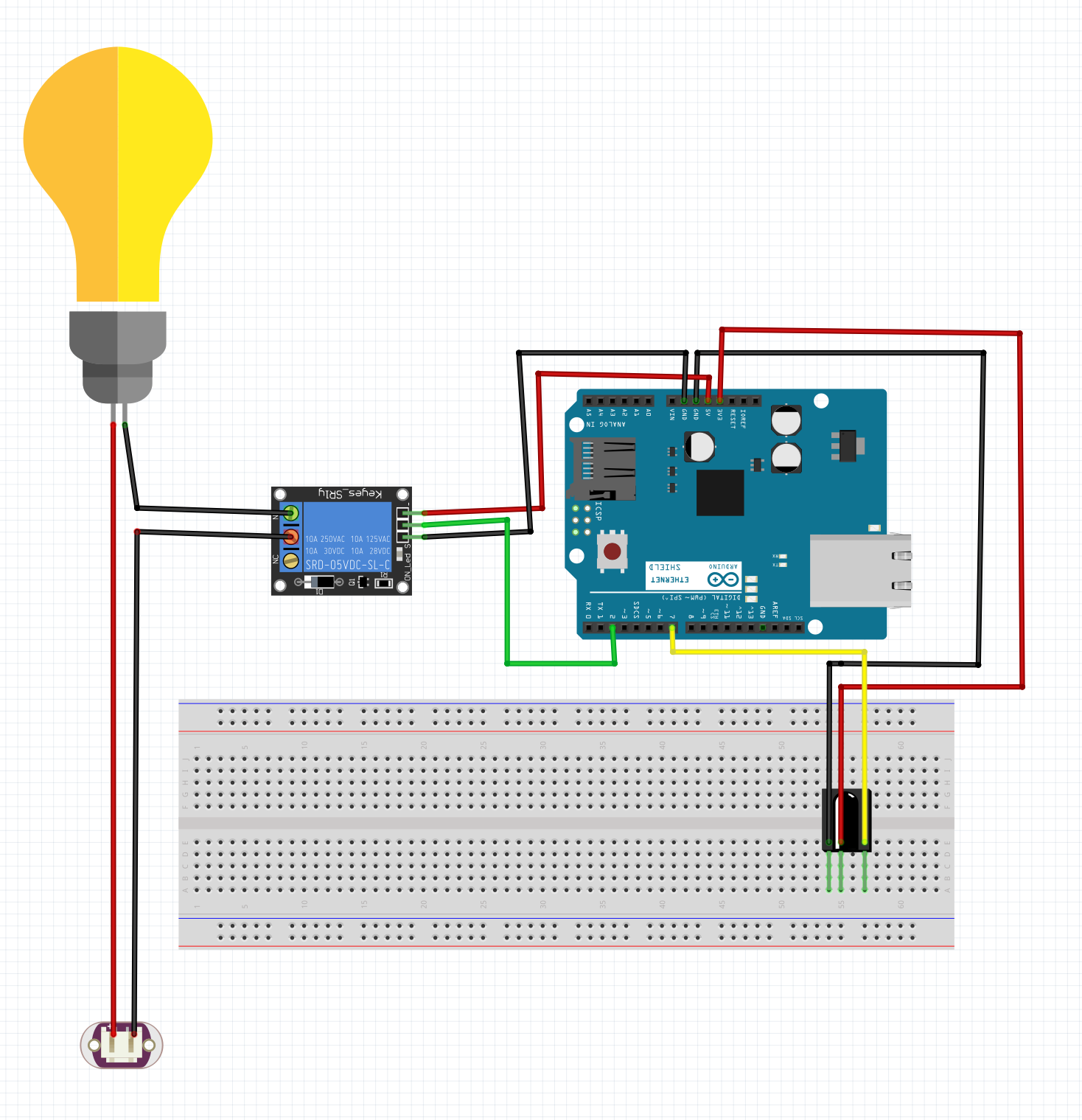
To control a bulb using a relay module interfaced with a microcontroller.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* 5V Relay Module
* 60W Bulb with Holder
* 220V AC Power Supply
* Jumper Wires
* Breadboard
* USB Cable for programming

#### **Circuit Diagram**

Circuit Diagram showing the relay module connected to the microcontroller’s GPIO pin, with its NO (Normally Open) terminal connected to the bulb and AC power source.

**

#### **Code**

// Relay Control Example Code

const int relayPin = 7; // Pin connected to relay module

void setup() {

pinMode(relayPin, OUTPUT); // Set relay pin as output

}

void loop() {

digitalWrite(relayPin, HIGH); // Turn relay ON (Bulb ON)

delay(5000); // Wait for 5 seconds

digitalWrite(relayPin, LOW); // Turn relay OFF (Bulb OFF)

delay(5000); // Wait for 5 seconds

}

#### **Results/Conclusions**

The relay successfully controlled the bulb, toggling it ON and OFF at 5-second intervals. This demonstrates the use of a relay module to interface AC devices with a microcontroller.

### **Experiment 3: 12V DC Motor Using L298**

#### **Aim**

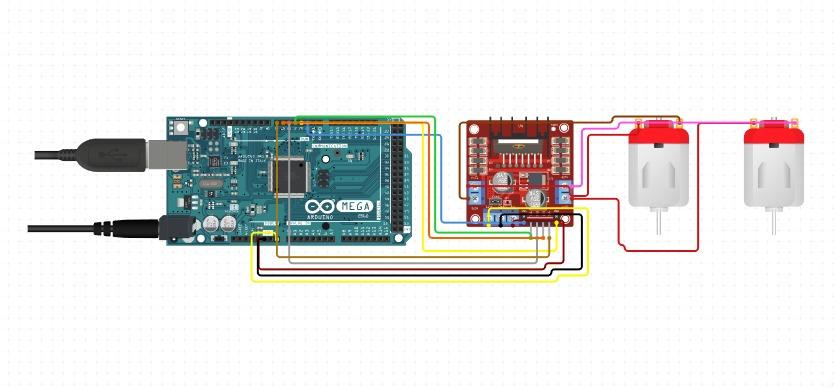
To control the rotation of a 12V DC motor using an L298 motor driver module and a microcontroller.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* L298 Motor Driver Module
* 12V DC Motor
* External Power Supply (12V)
* Breadboard
* Jumper Wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing the L298 motor driver connected to the microcontroller and DC motor with an external 12V power supply.

**

#### 

#### **Code**

// DC Motor Control using L298 Example Code

const int motorPin1 = 9; // Pin connected to IN1 on L298

const int motorPin2 = 10; // Pin connected to IN2 on L298

void setup() {

pinMode(motorPin1, OUTPUT); // Set motor control pins as output

pinMode(motorPin2, OUTPUT);

}

void loop() {

// Rotate motor clockwise

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, LOW);

delay(2000); // Run for 2 seconds

// Stop motor

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

delay(1000); // Wait for 1 second

// Rotate motor counterclockwise

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);

delay(2000); // Run for 2 seconds

// Stop motor

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

delay(1000); // Wait for 1 second

}

#### **Results/Conclusions**

The DC motor successfully rotated clockwise and counterclockwise as programmed, demonstrating motor control using the L298 motor driver module.

### 

### **Experiment 4: IR Sensor**

#### **Aim**

To interface an Infrared (IR) sensor with a microcontroller to detect the presence of an object.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* IR Sensor Module
* Breadboard
* Jumper Wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing the IR sensor's VCC and GND connected to the microcontroller's 5V and GND pins, and the OUT pin connected to a digital GPIO pin.

#### 

#### 

#### **Code**

// IR Sensor Detection Example Code

const int irSensorPin = 2; // Pin connected to the IR sensor's OUT pin

const int ledPin = 13; // Built-in LED to indicate detection

void setup() {

pinMode(irSensorPin, INPUT); // Set sensor pin as input

pinMode(ledPin, OUTPUT); // Set LED pin as output

Serial.begin(9600); // Start Serial communication

}

void loop() {

int sensorValue = digitalRead(irSensorPin); // Read sensor value

if (sensorValue == LOW) { // Object detected

digitalWrite(ledPin, HIGH); // Turn LED ON

Serial.println("Object Detected");

} else { // No object detected

digitalWrite(ledPin, LOW); // Turn LED OFF

Serial.println("No Object Detected");

}

delay(100); // Delay for stability

}

#### **Results/Conclusions**

The IR sensor successfully detected the presence of an object, and the built-in LED responded accordingly. This experiment demonstrates how to use an IR sensor for proximity detection.

### **Experiment 5: Passive Infrared (PIR) Sensor**

#### **Aim**

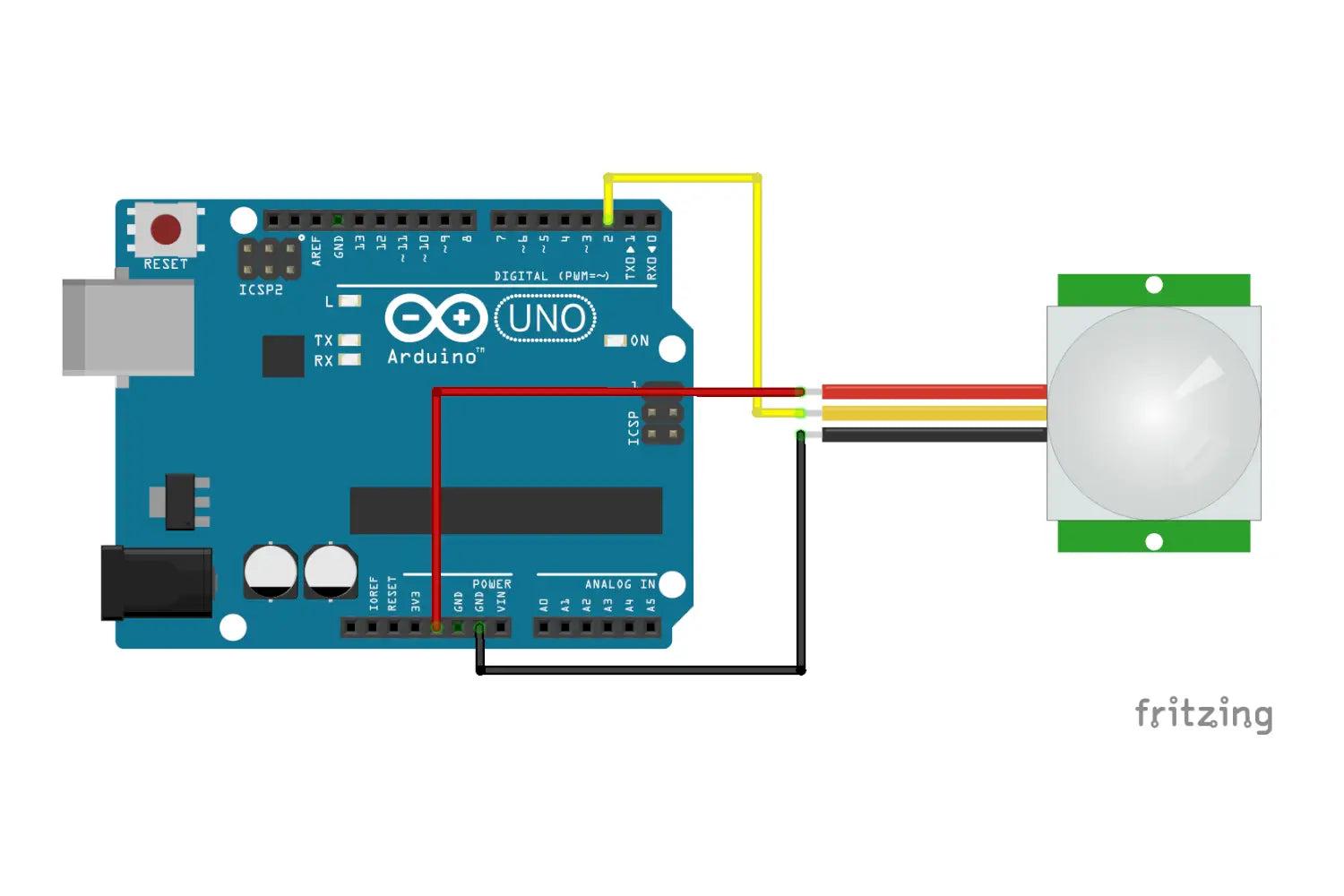
To interface a Passive Infrared (PIR) sensor with a microcontroller to detect human motion.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* PIR Sensor Module
* Breadboard
* Jumper Wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing the PIR sensor’s VCC and GND connected to the microcontroller’s 5V and GND pins, and the OUT pin connected to a digital GPIO pin.

**

#### **Code**

// PIR Sensor Motion Detection Example Code

const int pirSensorPin = 2; // Pin connected to the PIR sensor's OUT pin

const int ledPin = 13; // Built-in LED to indicate motion detection

void setup() {

pinMode(pirSensorPin, INPUT); // Set sensor pin as input

pinMode(ledPin, OUTPUT); // Set LED pin as output

Serial.begin(9600); // Start Serial communication

}

void loop() {

int motionDetected = digitalRead(pirSensorPin); // Read sensor value

if (motionDetected == HIGH) { // Motion detected

digitalWrite(ledPin, HIGH); // Turn LED ON

Serial.println("Motion Detected");

} else { // No motion detected

digitalWrite(ledPin, LOW); // Turn LED OFF

Serial.println("No Motion");

}

delay(100); // Delay for stability

}

#### **Results/Conclusions**

The PIR sensor successfully detected human motion, and the built-in LED responded accordingly. This demonstrates the use of a PIR sensor for motion detection applications.

### **Experiment 6: Ultrasonic Sensor**

#### **Aim**

To interface an ultrasonic sensor with a microcontroller to measure the distance of an object.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* Ultrasonic Sensor (e.g., HC-SR04)
* Breadboard
* Jumper Wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing the ultrasonic sensor's VCC and GND connected to the microcontroller's 5V and GND pins, TRIG pin connected to a GPIO pin, and ECHO pin connected to another GPIO pin.

#### 

#### 

#### **Code**

// Ultrasonic Sensor Distance Measurement Example Code

const int trigPin = 9; // Pin connected to TRIG pin

const int echoPin = 10; // Pin connected to ECHO pin

void setup() {

pinMode(trigPin, OUTPUT); // Set TRIG pin as output

pinMode(echoPin, INPUT); // Set ECHO pin as input

Serial.begin(9600); // Start Serial communication

}

void loop() {

long duration;

int distance;

// Send a 10µs pulse to trigger the sensor

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Read the echo pulse duration

duration = pulseIn(echoPin, HIGH);

// Calculate the distance in cm

distance = duration \* 0.034 / 2;

// Display the result

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

delay(500); // Wait before the next measurement

}

#### **Results/Conclusions**

The ultrasonic sensor successfully measured the distance to an object and displayed the values in centimeters on the Serial Monitor. This demonstrates the use of an ultrasonic sensor for distance measurement.

### **Experiment 7: DHT11 Sensor**

#### **Aim**

To interface a DHT11 sensor with a microcontroller to measure temperature and humidity.

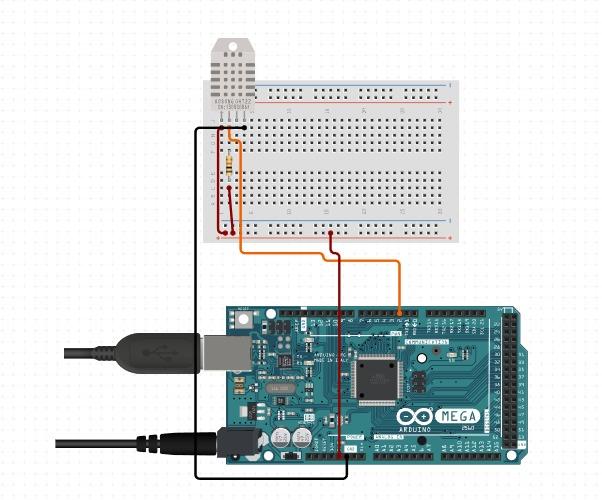
#### 

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* DHT11 Temperature and Humidity Sensor
* Breadboard
* Jumper Wires
* USB Cable for programming
* DHT library for Arduino (install via Library Manager: DHT Sensor Library by Adafruit)

#### **Circuit Diagram**

Circuit diagram showing the DHT11 sensor's VCC and GND connected to the microcontroller's 5V and GND pins, and the DATA pin connected to a GPIO pin.

**

#### **Code**

#include <DHT.h>

#define DHTPIN 2 // Pin connected to the DATA pin of DHT11

#define DHTTYPE DHT11 // Define sensor type as DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600); // Start Serial communication

dht.begin(); // Initialize DHT sensor

}

void loop() {

// Read temperature and humidity

float humidity = dht.readHumidity();

float temperature = dht.readTemperature();

// Check if the readings are valid

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

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

return;

}

// Display the results

Serial.print("Humidity: ");

Serial.print(humidity);

Serial.print("% Temperature: ");

Serial.print(temperature);

Serial.println("°C");

delay(2000); // Wait 2 seconds before the next reading

}

#### **Results/Conclusions**

The DHT11 sensor successfully measured and displayed temperature and humidity values. This demonstrates its use for environmental monitoring applications.

### **Experiment 8: Pulse Rate Sensor**

#### **Aim**

To interface a Pulse Rate Sensor with a microcontroller to measure the heart rate of a user.

#### **Requirements**

* Microcontroller (e.g., Arduino Uno)
* Pulse Rate Sensor (e.g., SEN-11574)
* Breadboard
* Jumper Wires
* USB Cable for programming

#### **Circuit Diagram**

Circuit diagram showing the Pulse Rate Sensor's VCC and GND connected to the microcontroller's 5V and GND pins, and the signal pin connected to an analog GPIO pin.

#### 

#### 

#### **Code**

// Pulse Rate Sensor Example Code

const int pulsePin = A0; // Analog pin connected to the Pulse Rate Sensor's signal pin

void setup() {

Serial.begin(9600); // Start Serial communication

}

void loop() {

int pulseValue = analogRead(pulsePin); // Read the analog value from the sensor

Serial.print("Pulse Sensor Value: ");

Serial.println(pulseValue); // Print the value to the Serial Monitor

delay(100); // Short delay for stability

}

#### **Results/Conclusions**

The Pulse Rate Sensor successfully detected the variations in the pulse, and the readings were displayed on the Serial Monitor. This demonstrates the use of a pulse rate sensor for basic health monitoring applications.