Experiment 8

Aim: To implement temperature sensor with Arduino/ ESP 32.

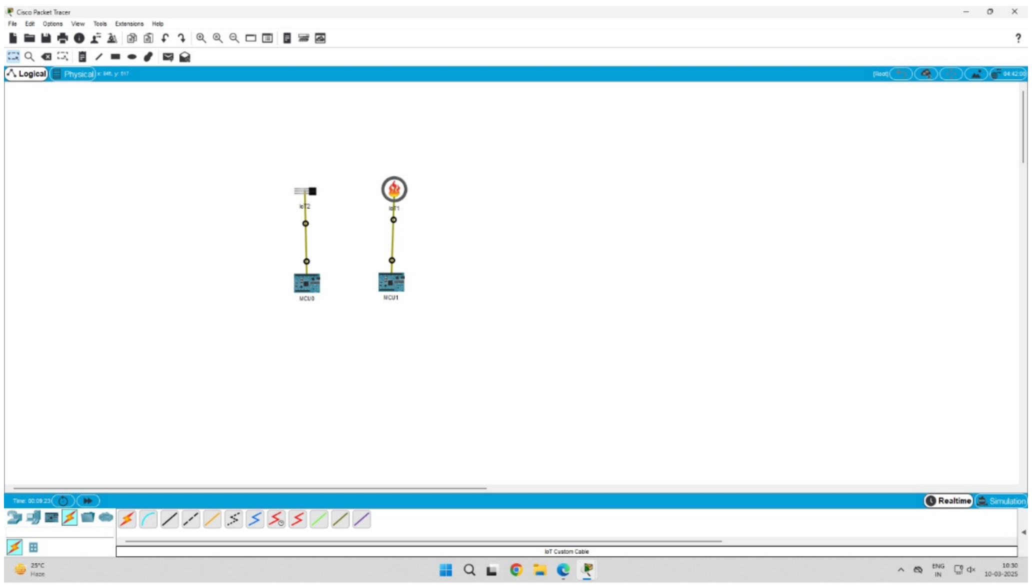
Theory: Temperature sensors are essential in various applications such as environmental monitoring, smart home automation, and healthcare systems. These sensors measure the ambient temperature and provide real-time data for processing. In this experiment, we implement a temperature sensor using Arduino or ESP32 within Cisco Packet Tracer, a network simulation tool that also supports IoT-based hardware simulations. The setup consists of a microcontroller unit (MCU) connected to a temperature sensor and a fire sensor, which detect environmental temperature changes and fire-related incidents, respectively.

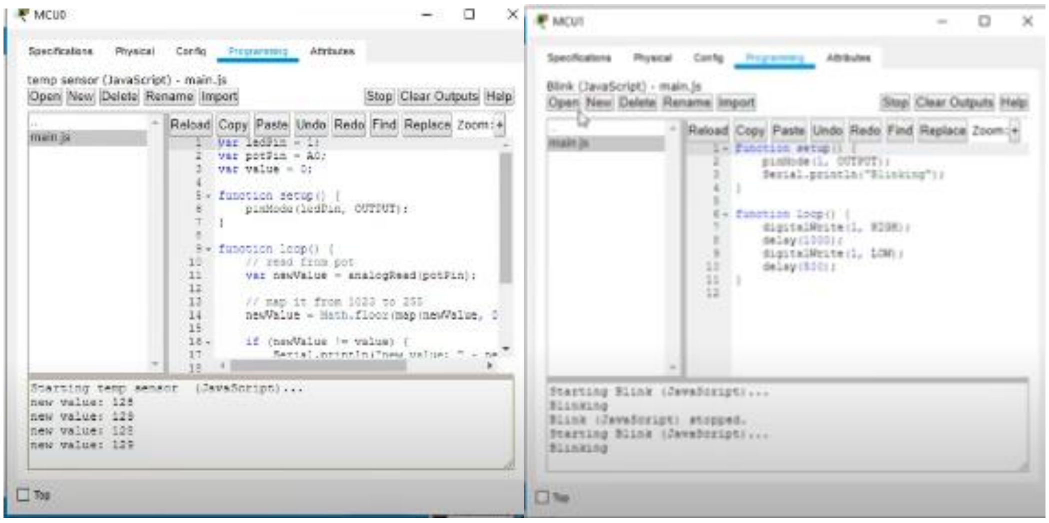
The experiment involves two types of sensors: a temperature sensor and a fire sensor. The temperature sensor detects the ambient temperature and transmits the data to the microcontroller for processing. This data can be visualized in the simulation for further analysis. The fire sensor, on the other hand, detects high-temperature conditions typically associated with fire hazards and sends a corresponding alert. Both sensors communicate with the microcontroller via a custom IoT cable, ensuring seamless data transfer.

The working principle of these sensors is based on detecting temperature variations using semiconductor-based elements. The temperature sensor provides real-time temperature readings, while the fire sensor triggers an alert when the temperature exceeds a critical threshold. The microcontroller processes the received data and can display it on an IoT dashboard or control panel within Cisco Packet Tracer.

In the simulation, the microcontroller acts as a central processing unit that gathers data from both sensors. This data can be further analyzed to implement automated responses, such as activating alarms or triggering safety mechanisms. The ability to integrate temperature sensors with IoT-enabled microcontrollers in Cisco Packet Tracer makes it a powerful tool for simulating real-world applications of IoT in smart environments.

Observations:



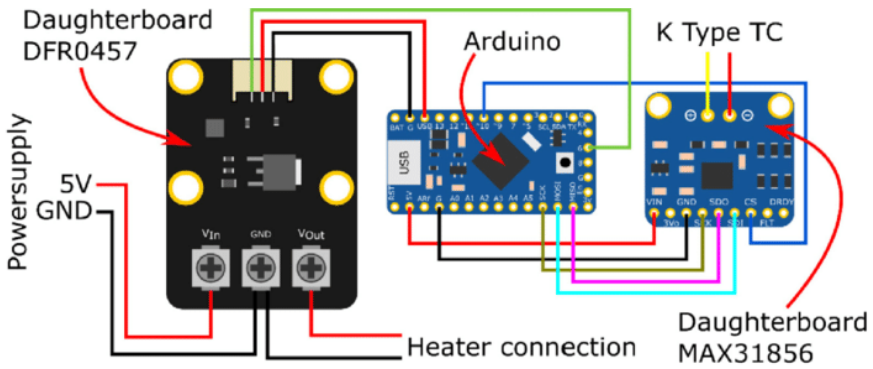


```
var ledPin = 1;
var potPin = A0;
var value = 0;

function setup() {
    pinMode(ledPin, OUTPUT);
}

function loop() {
    // read from pot
    var newValue = analogRead(potPin);

    // map it from 1023 to 255
    newValue = Math.floor(map(newValue, 0
    
    if (newValue != value) {
        Serial.println("new value: " + ne
```

```
Starting temp sensor  (JavaScript)...
new value: 128
new value: 129
new value: 128
new value: 129
```

```
function setup() {
    pinMode(1, OUTPUT);
    Serial.println("Blinking");
}

function loop() {
    digitalWrite(1, HIGH);
    delay(1000);
    digitalWrite(1, LOW);
    delay(500);
}
```

```
Starting Blink (JavaScript)...
Blinking
Blink (JavaScript) stopped.
Starting Blink (JavaScript)...
Blinking
```



Result: Successfully implemented temperature sensor with Arduino/ ESP 32

Experiment 9

Aim: To implement and study the LED blink using the Raspberry Pi

Theory: The blinking LED is one of the fundamental experiments when working with Raspberry Pi. It helps in understanding how the GPIO (General Purpose Input/Output) pins of the Raspberry Pi can be used to control external components. This experiment involves connecting an LED (Light Emitting Diode) to the Raspberry Pi and controlling its blinking pattern using a Python script.

Working Principle - The Raspberry Pi has multiple GPIO pins that can be programmed to act as either input or output. In this experiment, we configure a GPIO pin as an output to send a HIGH or LOW signal, which turns the LED ON and OFF, respectively. The Python programming language is used to write a script that alternates the state of the LED at a fixed time interval, creating a blinking effect.

When the GPIO pin is set to HIGH (1), the LED turns ON as current flows through it. When the GPIO pin is set to LOW (0), the LED turns OFF as the current flow stops. This process is repeated continuously in a loop to produce the blinking effect.

Circuit Components and Connections

1. Raspberry Pi – Acts as the microcontroller to control the LED.
2. LED – Emits light when powered.
3. Resistor (330Ω - 1kΩ) – Prevents excessive current flow, protecting the LED.
4. Jumper wires – Used for connections.
5. Breadboard – Helps in making circuit connections easily.

The LED’s anode (+) is connected to a GPIO pin, while the cathode (-) is connected to the ground (GND) through a resistor to complete the circuit.

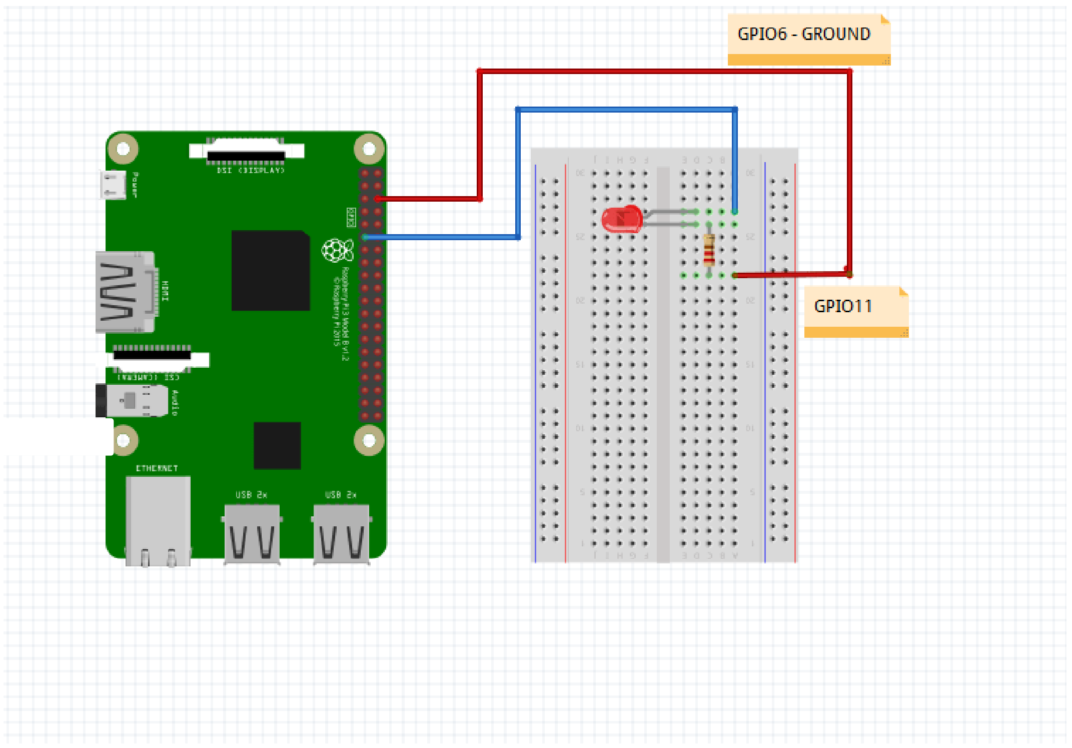
Programming the Raspberry Pi for LED Blinking

The Raspberry Pi is programmed using Python, with the RPi.GPIO or GPIO Zero library to control the GPIO pin. The script sets the GPIO pin as output and toggles its state in a loop with a delay to achieve the blinking effect.

Applications of LED Blinking with Raspberry Pi

- Basic IoT Applications – Used in status indicators.
- Home Automation – Can be used for smart lighting systems.
- Security Systems – Acts as an alert or indicator in response to sensor inputs.

Observations:



Result: Successfully implemented and studied the LED blink using the Raspberry Pi

Experiment 10

Aim: To implement and study the motion sensor using the Raspberry Pi

Theory: Motion sensors play a vital role in security systems, home automation, and IoT-based applications by detecting movement in a specified area. This experiment involves interfacing a PIR (Passive Infrared) motion sensor with a Raspberry Pi to detect motion and trigger an action, such as turning on an LED or sending an alert.

Working Principle - A PIR sensor works by detecting changes in infrared radiation, which is emitted by objects with heat, such as humans and animals. The sensor consists of two pyroelectric sensors and a Fresnel lens that focuses infrared radiation onto them. When an object moves within its range, the infrared levels change, and the sensor outputs a HIGH signal (1). When there is no movement, it remains LOW (0).

The Raspberry Pi reads this signal from the PIR sensor through a GPIO pin and processes it using a Python script. If motion is detected, the Raspberry Pi can trigger an action, such as turning on an LED, activating a buzzer, or sending an alert message.

Circuit Components and Connections

1. Raspberry Pi – Acts as the microcontroller.
2. PIR Motion Sensor – Detects movement.
3. LED/Buzzer (Optional) – Used as an indicator when motion is detected.
4. Jumper Wires – Connect components to the Raspberry Pi.

Connections

- VCC (Power) of PIR Sensor → 5V or 3.3V of Raspberry Pi
- GND (Ground) of PIR Sensor → GND of Raspberry Pi
- OUT (Data) of PIR Sensor → Any GPIO Pin of Raspberry Pi (e.g., GPIO17)

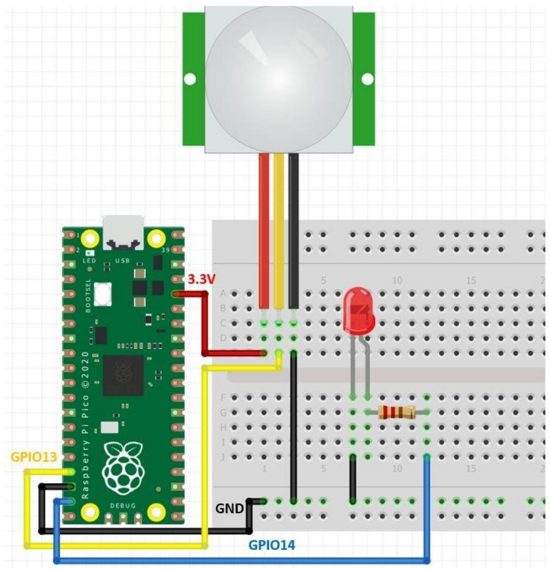
Programming the Raspberry Pi for Motion Detection

A Python script is used to read the PIR sensor’s output and perform an action when motion is detected. The RPi.GPIO or GPIO Zero library helps configure the GPIO pins and handle input signals.

Applications of Motion Sensors with Raspberry Pi

- Security Systems – Detects unauthorized movement and triggers an alarm.
- Home Automation – Turns on lights or appliances when motion is detected.
- Energy Conservation – Automates lighting based on occupancy detection.
- IoT-based Surveillance – Sends alerts or activates a camera when motion is detected.

Observations:



Result: Successfully implemented and studied the motion sensor using the Raspberry Pi