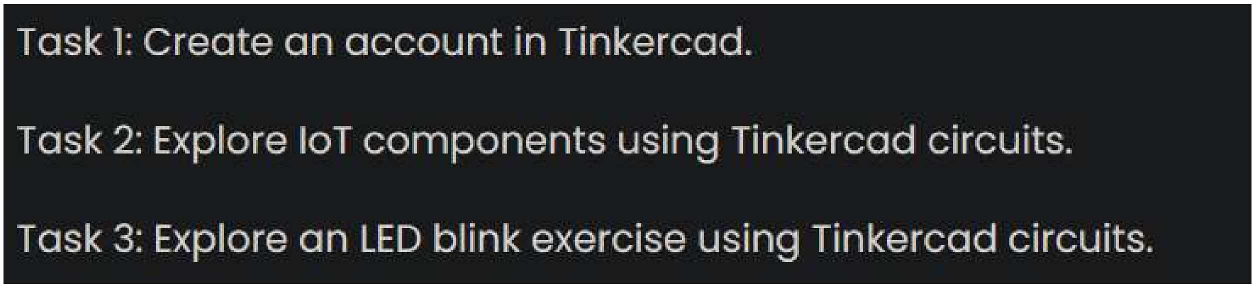
Lab 1

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



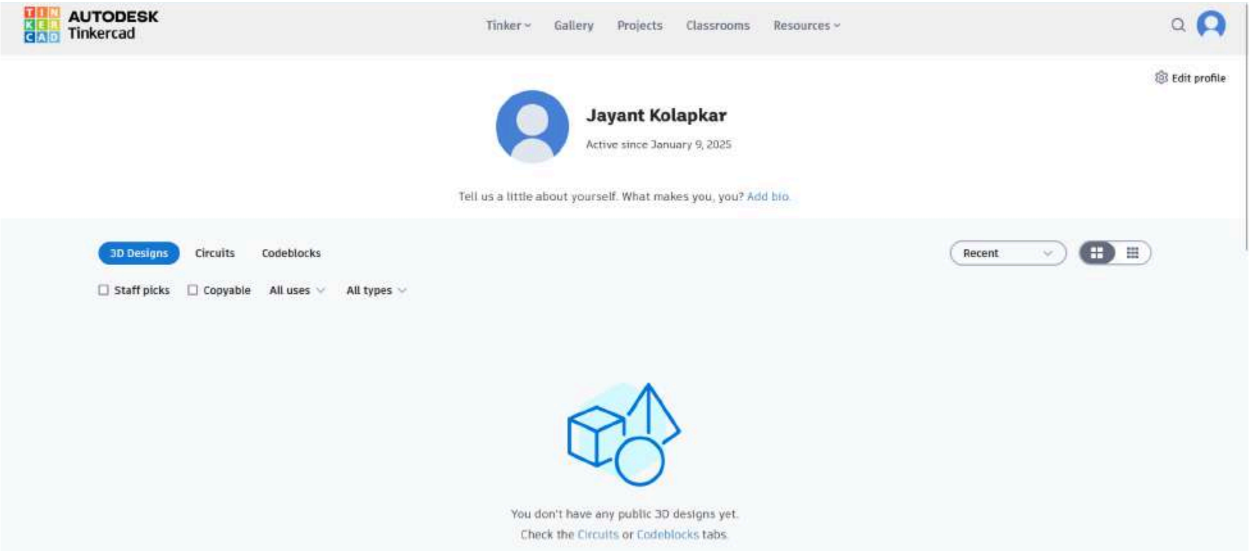
Task 1: Create an account in Tinkercad.

Task 2: Explore IoT components using Tinkercad circuits.

Task 3: Explore an LED blink exercise using Tinkercad circuits.

Task 1

Sign up to TinkerCAD using email.



Jayant Kolapkar

Active since January 9, 2025

Tell us a little about yourself. What makes you, you? Add bio.

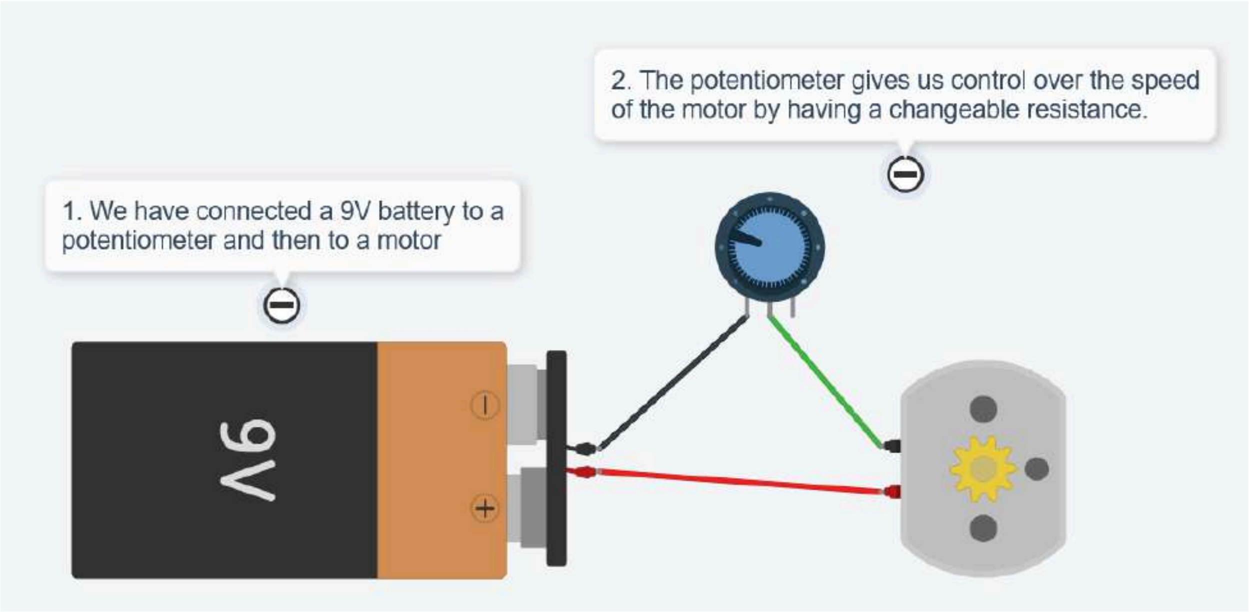
You don't have any public 3D designs yet.
Check the Circuits or Codeblocks tabs.

Task 2

We made a simple circuit to control the speed of a DC motor using a potentiometer.

1. **Power Source:** A 9V battery provides the electrical energy to power the motor.
2. **Potentiometer:** This variable resistor acts as a voltage divider. By turning the knob, you change the resistance, which in turn adjusts the voltage supplied to the motor.
3. **Motor:** The DC motor receives the variable voltage from the potentiometer. As the voltage increases, the motor spins faster. Conversely, decreasing the voltage slows down the motor.

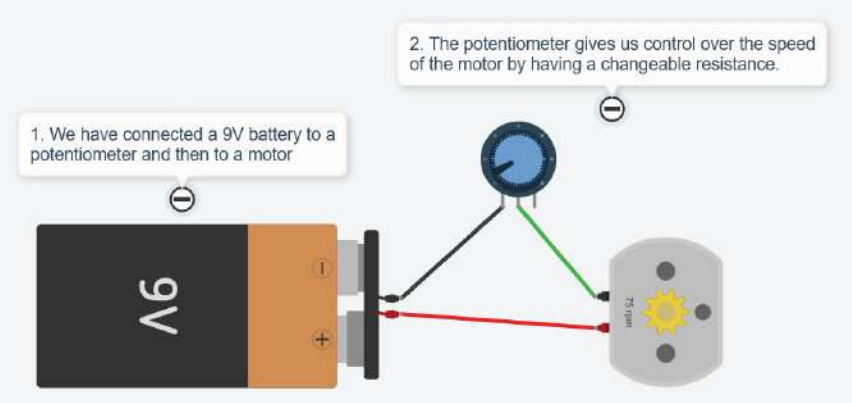
In essence, the potentiometer acts as a speed controller for the motor, allowing us to fine-tune its rotation speed.



2. The potentiometer gives us control over the speed of the motor by having a changeable resistance.

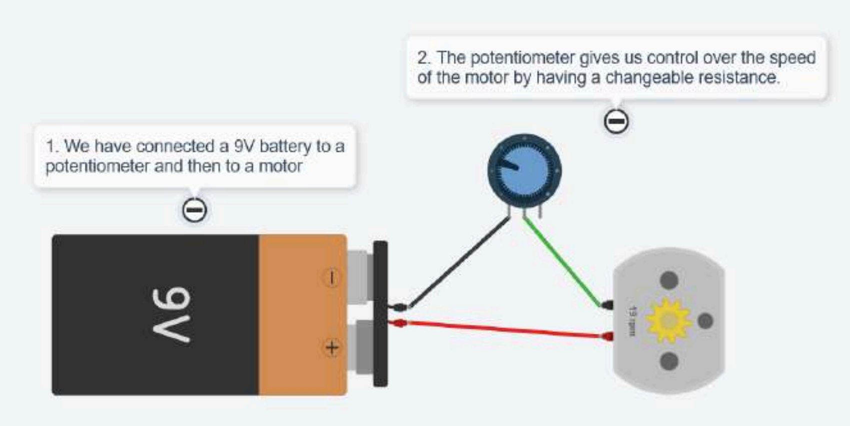
1. We have connected a 9V battery to a potentiometer and then to a motor

Output for various settings



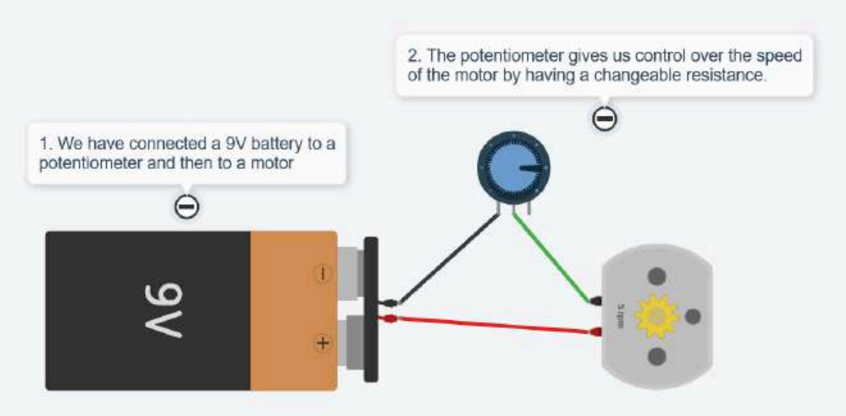
1. We have connected a 9V battery to a potentiometer and then to a motor

2. The potentiometer gives us control over the speed of the motor by having a changeable resistance.



1. We have connected a 9V battery to a potentiometer and then to a motor

2. The potentiometer gives us control over the speed of the motor by having a changeable resistance.



1. We have connected a 9V battery to a potentiometer and then to a motor

2. The potentiometer gives us control over the speed of the motor by having a changeable resistance.

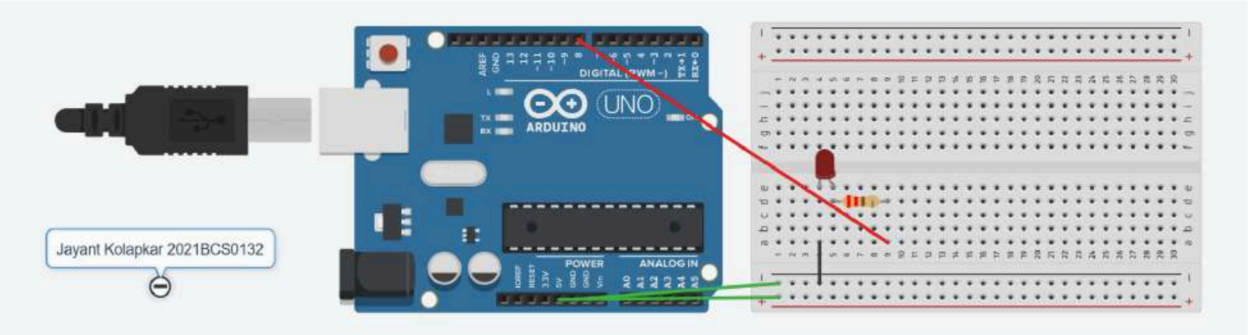
Task 3

We use Arduino UNO, breadboard and LED to make the LED light up and turn off repeatedly. We connect the LED to the Arduino board using wires and a resistor. The Arduino has a tiny program that tells it to

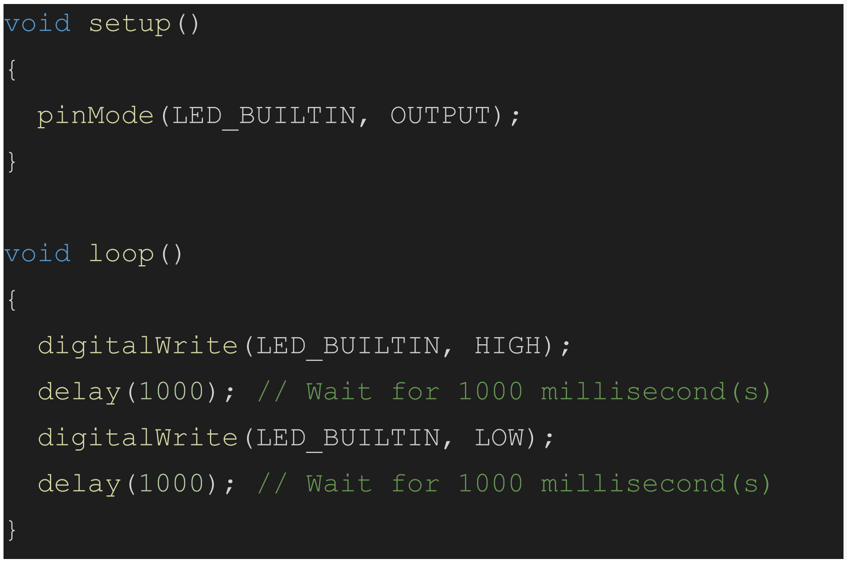
- Turn the LED on for a short time.
- Turn the LED off for a short time.
- Repeat these steps over and over.

The resistor protects the LED from getting too much electricity.

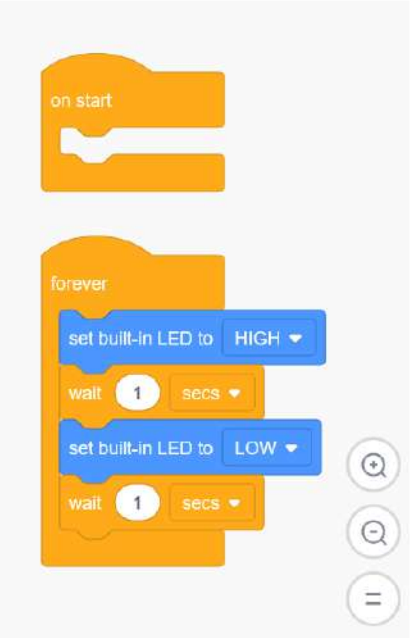
In essence, the Arduino acts like a tiny brain that controls the LED's on/off behavior.



Code



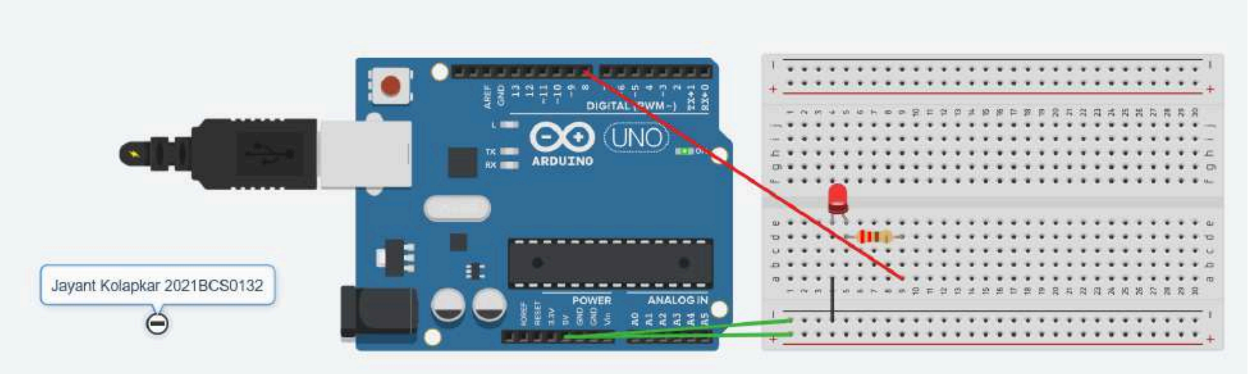
```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```



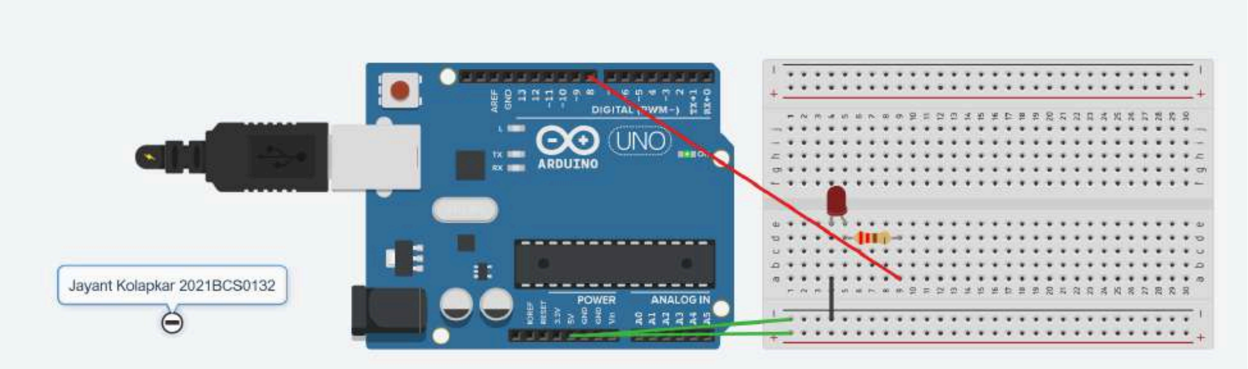
```
on start

forever
  set built-in LED to HIGH
  wait 1 secs
  set built-in LED to LOW
  wait 1 secs
```

Output

Blinking on



Blinking off

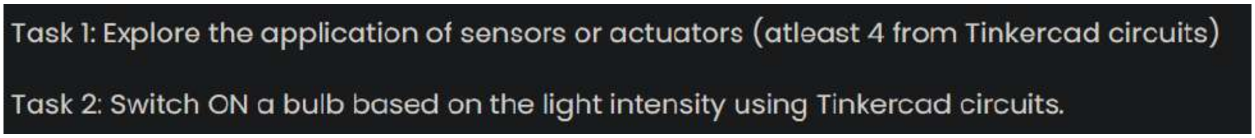


Lab 2

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



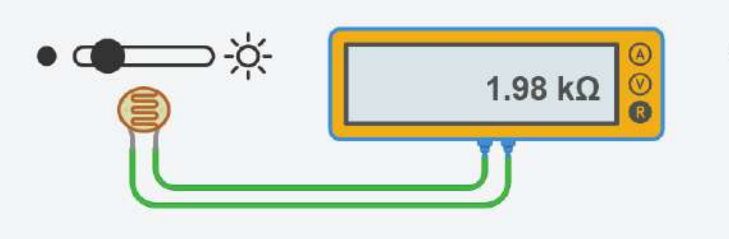
Task 1: Explore the application of sensors or actuators (atleast 4 from Tinkercad circuits)

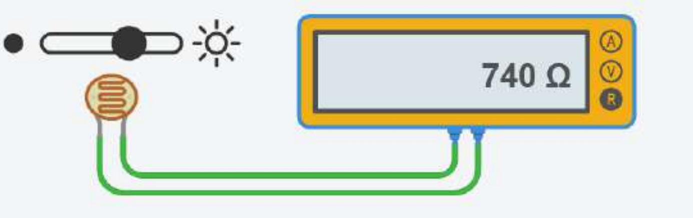
Task 2: Switch ON a bulb based on the light intensity using Tinkercad circuits.

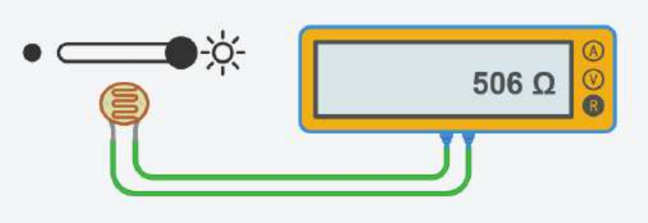
Task 1

1. Light Dependent Resistor (LDR) - Sensor

Description: The LDR detects light intensity and adjusts circuit behavior based on ambient lighting conditions. It is used for energy-saving devices like automatic streetlights.
Application: In this circuit, the LDR turns on a bulb when the light intensity falls below a threshold and turns it off when the intensity increases.

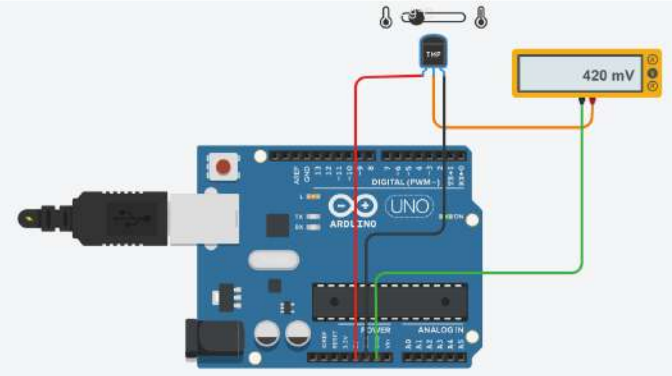


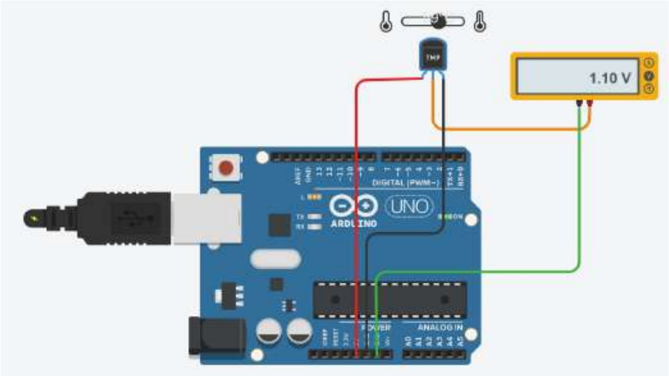


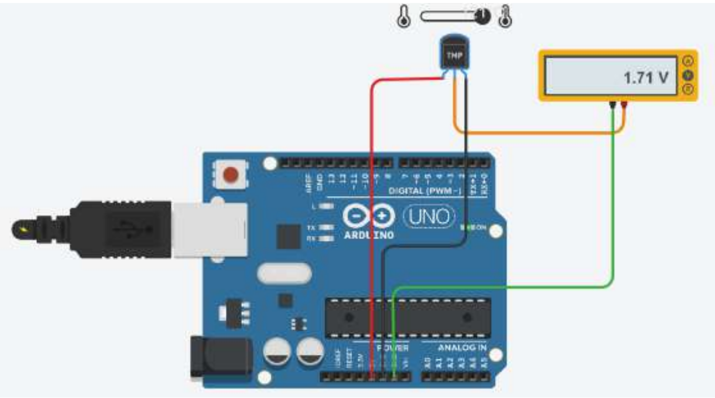


2. Temperature Sensor - Sensor

Description: A temperature sensor detects the surrounding temperature and converts it into a readable analog signal. It's useful in thermostats and weather monitoring systems.
Application: In this circuit, the temperature sensor provides analog readings to indicate environmental temperature, which can control heating or cooling devices.

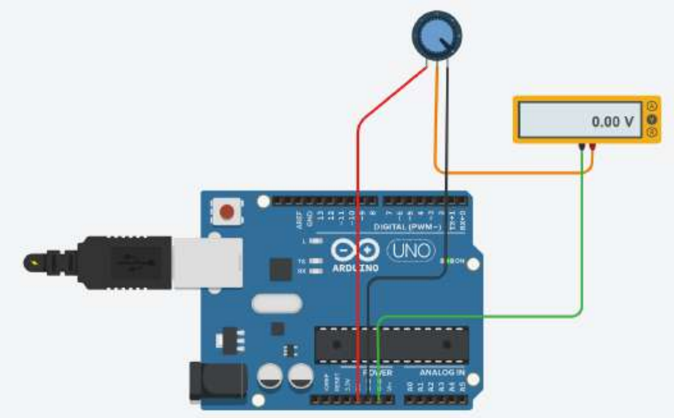


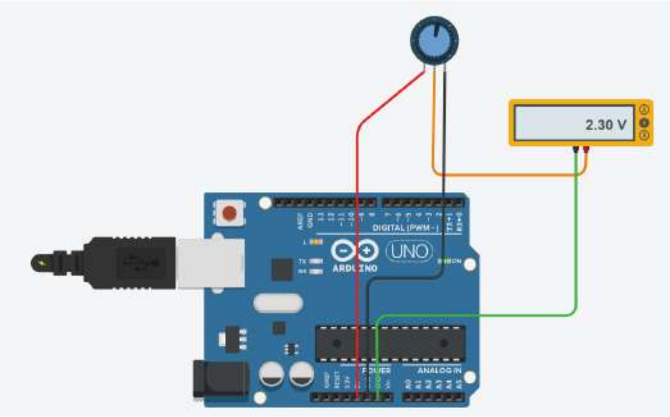


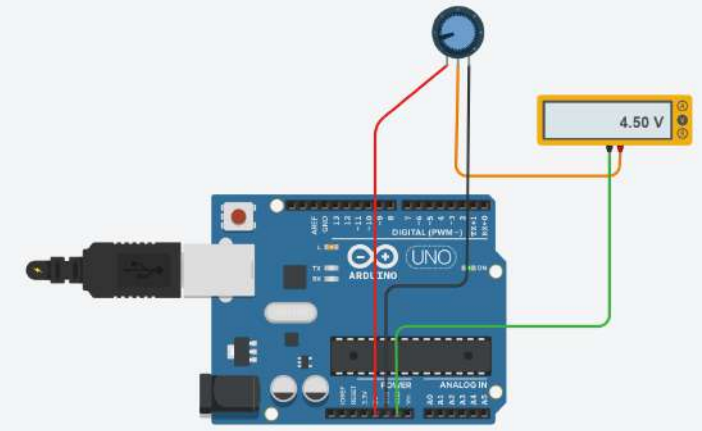


3. Potentiometer - Sensor/Controller

Description: A potentiometer is a variable resistor used to manually adjust the resistance in a circuit, providing control over voltage or current. It's commonly used in devices like volume knobs or brightness controls.
Application: In this setup, the potentiometer simulates an adjustable analog input, which can control the brightness of an LED.





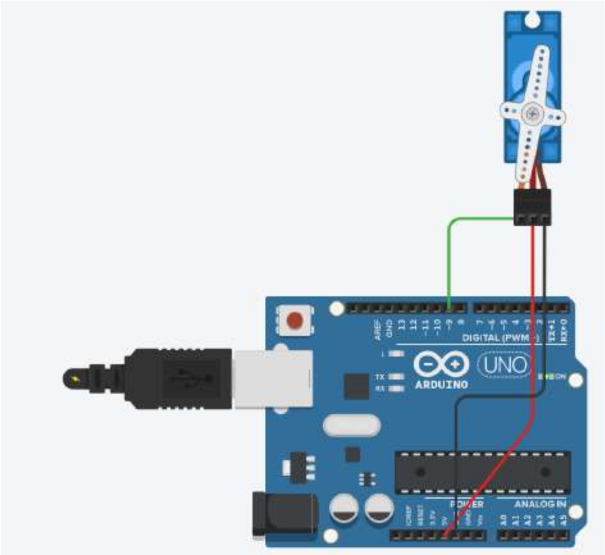


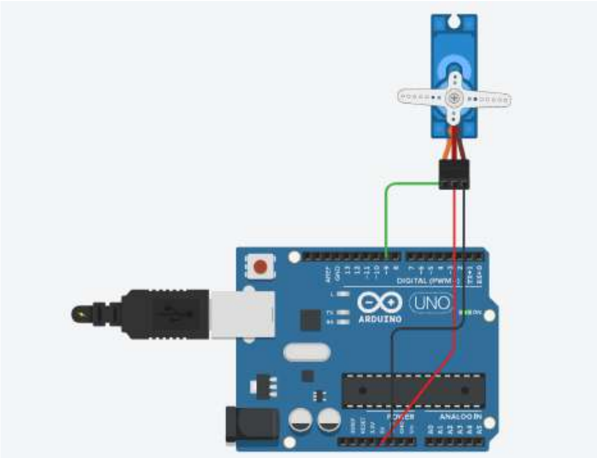
4. Servo Motor - Actuator

Description: A servo motor is an actuator capable of precise angular movement. It's widely used in robotics, home automation, and control systems.
Application: The servo motor rotates to specific angles (0°, 90°, and 180°) to demonstrate control and movement capabilities. This can simulate turning dials or opening/closing mechanisms.

Code

```
#include <Servo.h>
Servo myServo;

void setup() {
  myServo.attach(9); // Attach servo to pin 9
}

void loop() {
  myServo.write(0);    // Rotate to 0 degrees
  delay(1000);
  myServo.write(90);   // Rotate to 90 degrees
  delay(1000);
  myServo.write(180); // Rotate to 180 degrees
  delay(1000);
}
```





Task 2

In this lab, we will simulate an **Arduino-based light intensity lamp** using Tinkercad Circuits. The objective is to turn on a light bulb when the light intensity is low and turn it off when the light intensity is high, helping conserve energy.

Overview

The system uses an **LDR (Light Dependent Resistor)** to detect light intensity and control a light bulb accordingly. A relay is employed to handle the power difference between the bulb (120V) and the Arduino (5V).

Required Components

1. **LDR** (Light Dependent Resistor): To detect light and dark conditions.
2. **Arduino Microcontroller**: To process the input from the LDR and control the output.
3. **Light Bulb**: The output device that turns ON or OFF based on light intensity.
4. **Relay**: To switch the light bulb since it operates at a higher voltage.
5. **Power Source**: To supply power to the circuit.
6. **Breadboard** (Optional but recommended): For easier and cleaner wiring.

Instructions

Step 1: Setup the Circuit

- Open Tinkercad and create a new circuit project.
- Drag all the required components (LDR, Arduino, relay, light bulb, power source, and optional breadboard) into the workspace.

Step 2: Arrange Components

- Place the components on the breadboard for better organization and easy wiring. Position the Arduino, LDR, relay, and light bulb logically to minimize clutter.

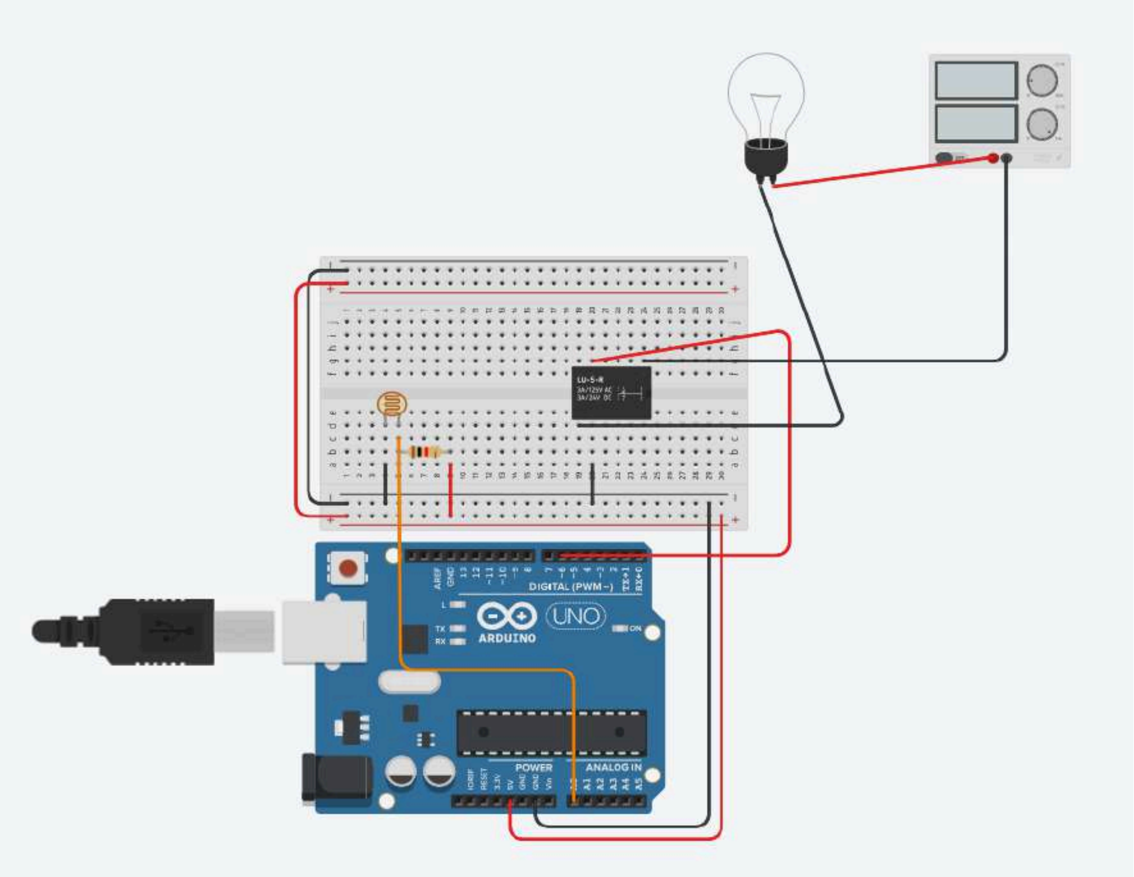
Step 3: Connect the Wires

A. Wiring the Relay and Light Bulb

1. Connect the **5V pin** of the Arduino to the **power rail** on the breadboard.
2. Connect the **GND pin** of the Arduino to the **ground rail**.
3. Attach the **positive terminal** of the power source to **Terminal 2** of the light bulb.
4. Connect the **negative terminal** of the power source to **Terminal 1** of the relay.
5. Link the **negative terminal** of the light bulb to **Terminal 7** of the relay.
6. Connect **Terminal 8** of the relay to the **ground rail**.
7. Finally, connect **Terminal 5** of the relay to any **digital pin** on the Arduino.

B. Wiring the Photoresistor

1. Connect **Terminal 1** of the LDR to the **ground rail**.
2. Connect **Terminal 2** of the LDR to an **analog pin** on the Arduino (e.g., A0) and to the **power rail** using a **1kΩ resistor**.

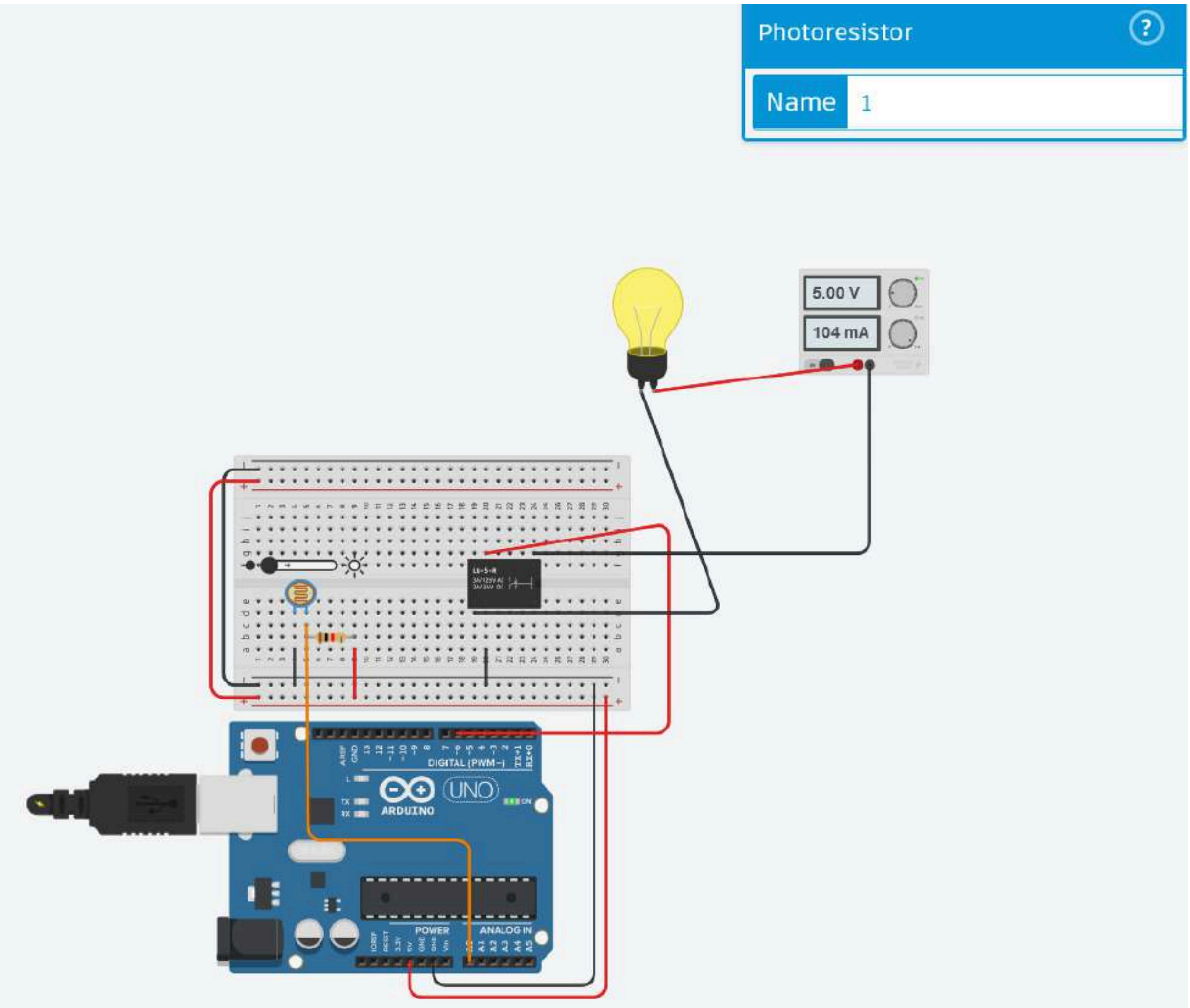


Code

```
void setup()
{
  pinMode(A0, INPUT);
  pinMode(6, OUTPUT);
}

void loop()
{
    Serial.println(analogRead(A0));
  if (analogRead(A0) > 500)
    {
        digitalWrite(6, LOW);
    }
  else
    {
      digitalWrite(6, HIGH);
    }
    delay(10);
}
```

Output

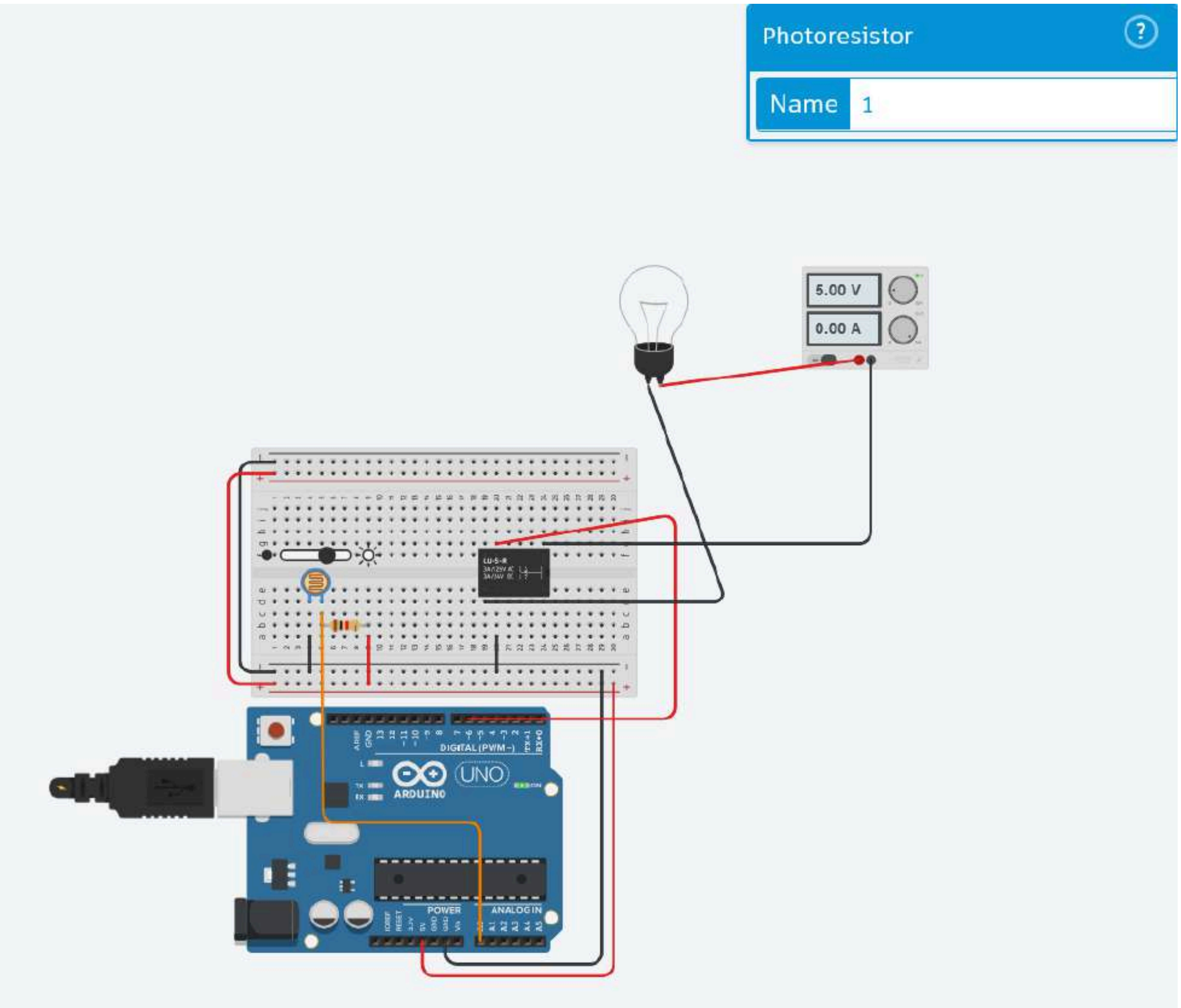
LDR in low light



Photoresistor

Name 1

LDR in high light



Photoresistor

Name 1

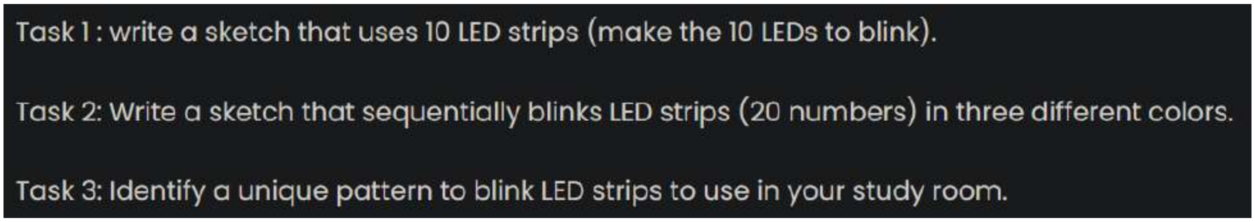
We can see the behavior of the circuit for different light conditions. The light bulb will automatically turn ON when the light intensity drops and turn OFF when the light intensity rises, simulating an energy-efficient system for real-world applications.

Lab 3

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



Task 1 : write a sketch that uses 10 LED strips (make the 10 LEDs to blink).

Task 2: Write a sketch that sequentially blinks LED strips (20 numbers) in three different colors.

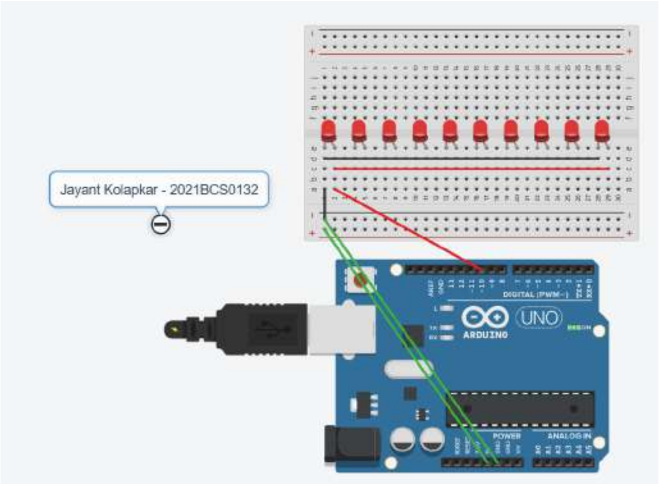
Task 3: Identify a unique pattern to blink LED strips to use in your study room.

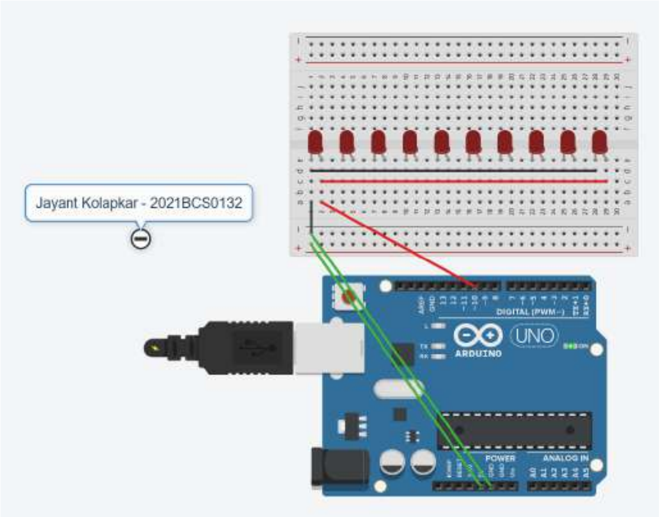
Task 1

Code

```
int ledPin=10; //definition digital 8 pins as pin to control the LED
void setup()
{
    pinMode(ledPin,OUTPUT);     //Set the digital 8 port mode, OUTPUT:
Output mode
}
void loop()
{
    digitalWrite(ledPin,HIGH); //HIGH is set to about 5V PIN8
    delay(1000);               //Set the delay time, 1000 = 1S
    digitalWrite(ledPin,LOW);  //LOW is set to about 5V PIN8
    delay(1000);               //Set the delay time, 1000 = 1S
}
```

Output



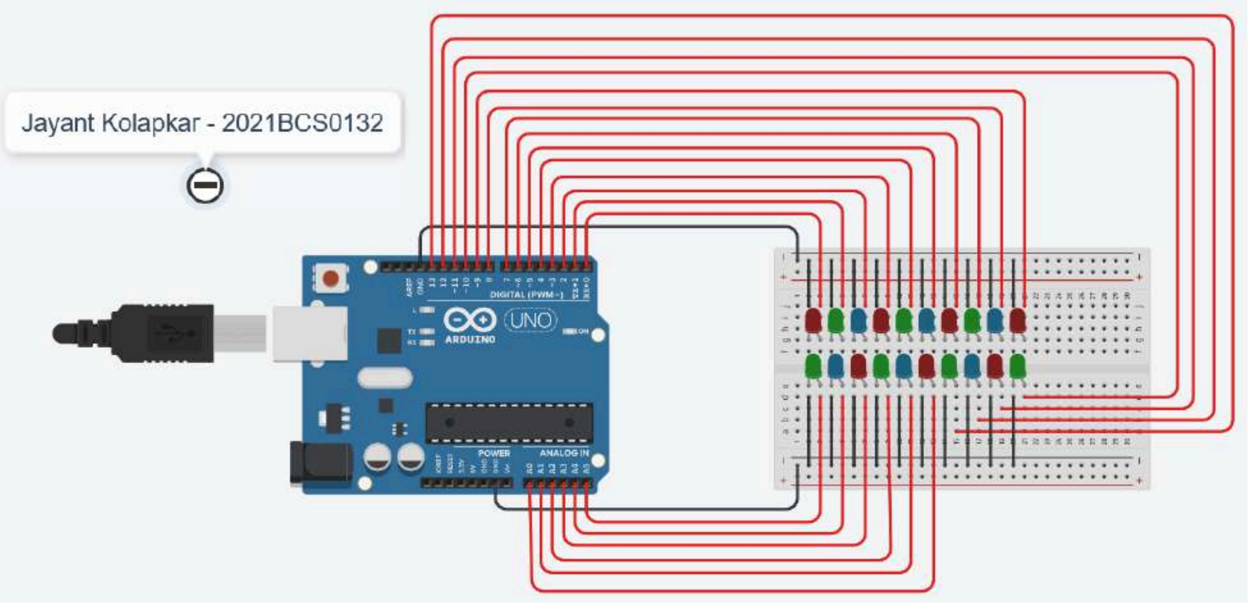


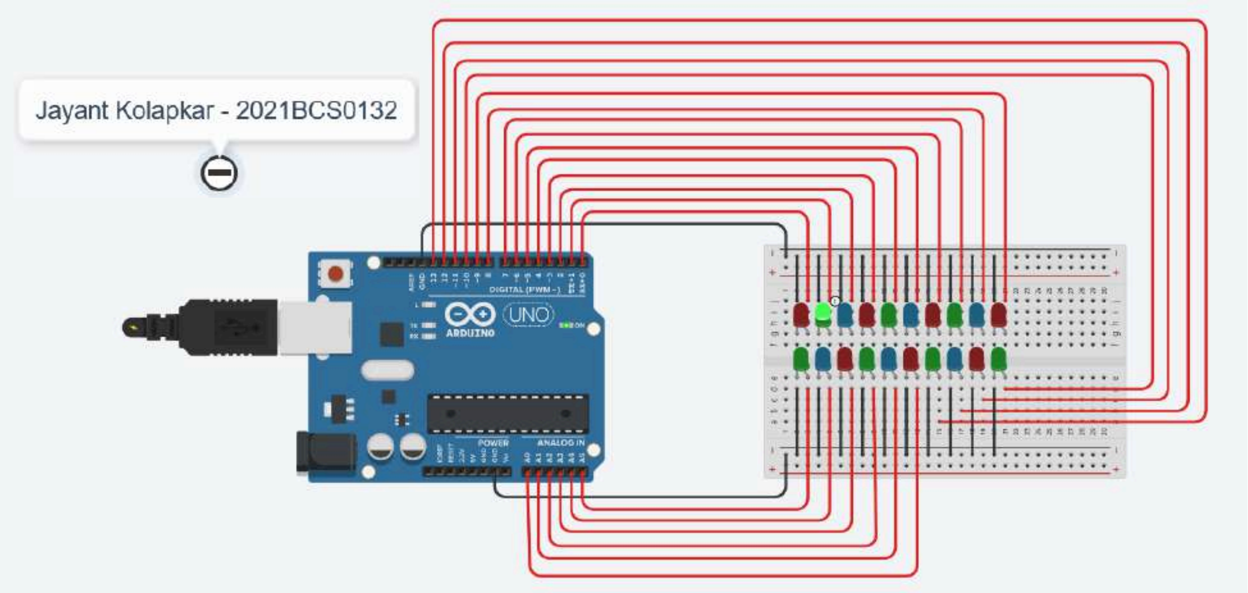
Task 2

Code

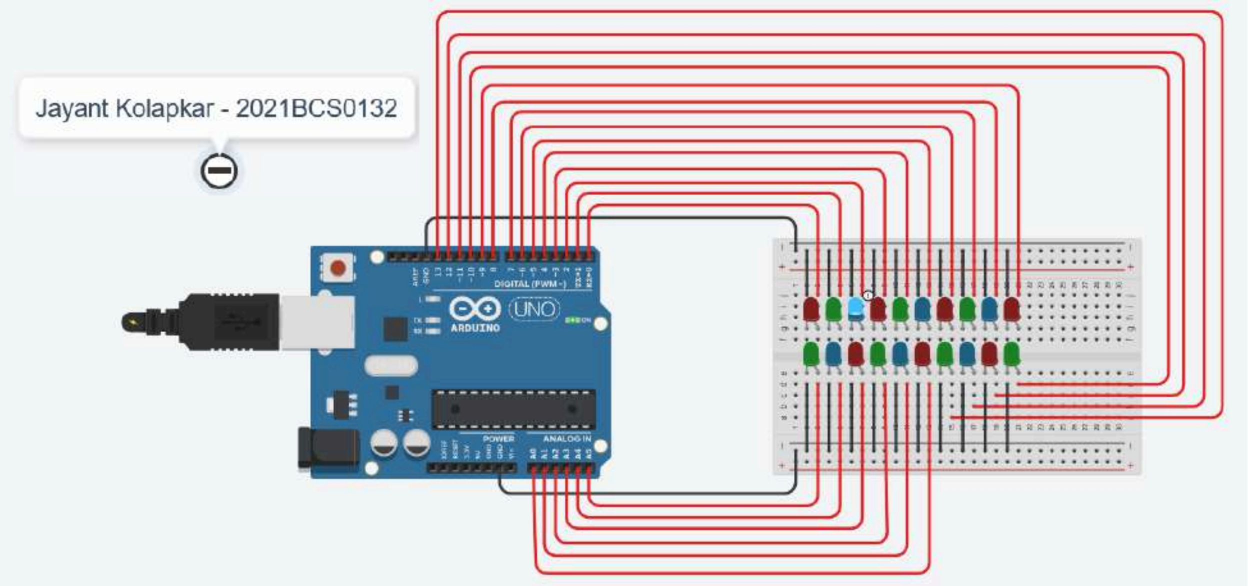
```
void setup()
{
  for (int pin = 0; pin <= 13; pin++)
  {
    pinMode(pin, OUTPUT);
  }
  for (int pin = A0; pin <= A5; pin++)
  {
    pinMode(pin, OUTPUT);
  }
}
void loop()
{
  for (int pin = 0; pin <= 13; pin++)
  {
    digitalWrite(pin, HIGH);
    delay(1000);
    digitalWrite(pin, LOW);
  }
  for (int pin = A0; pin <= A5; pin++)
  {
    digitalWrite(pin, HIGH);
    delay(1000);
    digitalWrite(pin, LOW);
  }
}
```

Output

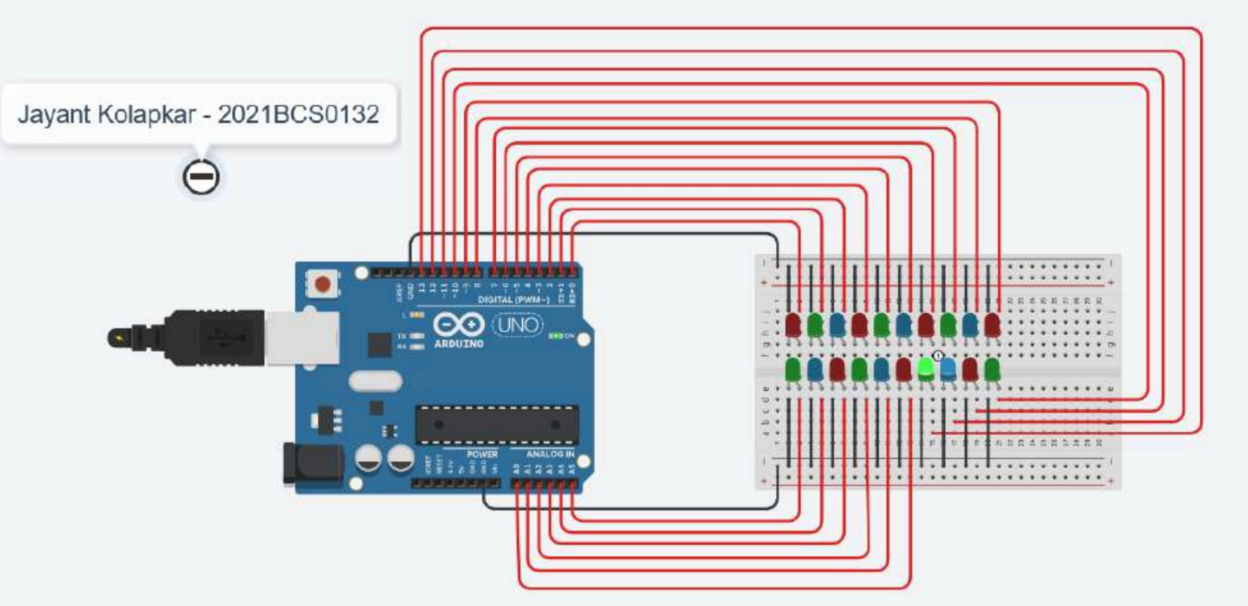




Jayant Kolapkar - 2021BCS0132



Jayant Kolapkar - 2021BCS0132



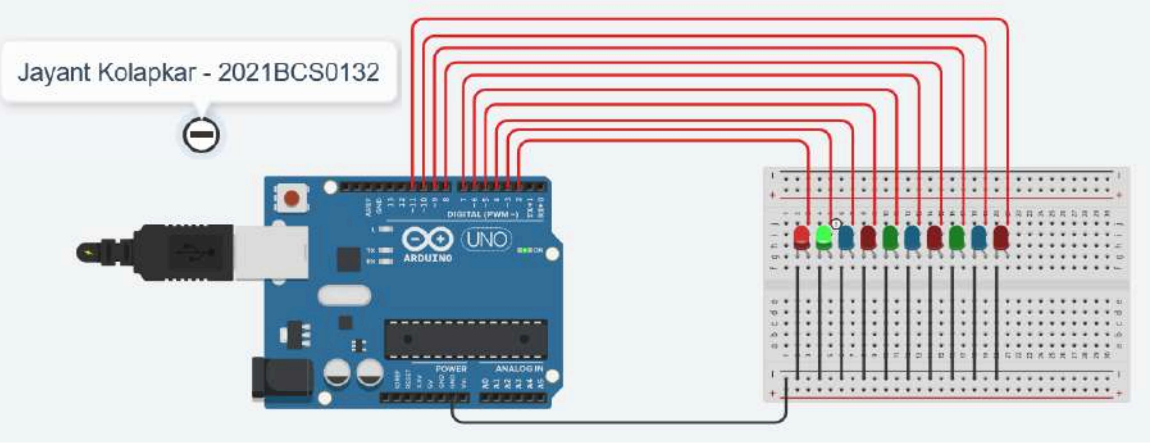
Jayant Kolapkar - 2021BCS0132
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

Task 3

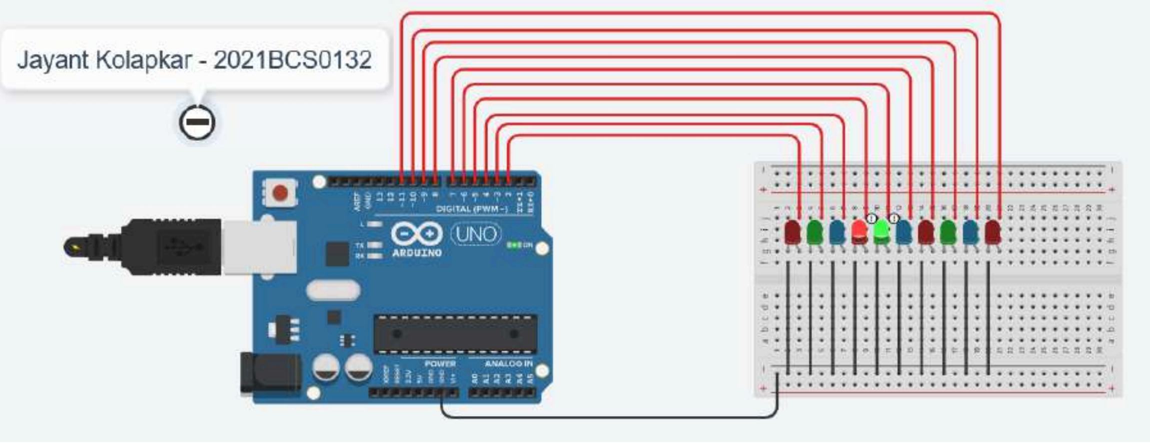
Code

```
void setup()
{
  for (int i = 2; i < 12; i++)
  {
    pinMode(i, OUTPUT);
  }
}
void loop()
{
  for (int k = 0; k < 10; k++)
  {
    for (int i = 2; i < 12; i++)
    {
      for (int j = i; j <= min(i + k, 11); j++)
        digitalWrite(j, 1);
      delay(500);
      for (int j = i; j <= min(i + k, 11); j++)
        digitalWrite(j, 0);
    }
  }
}
```

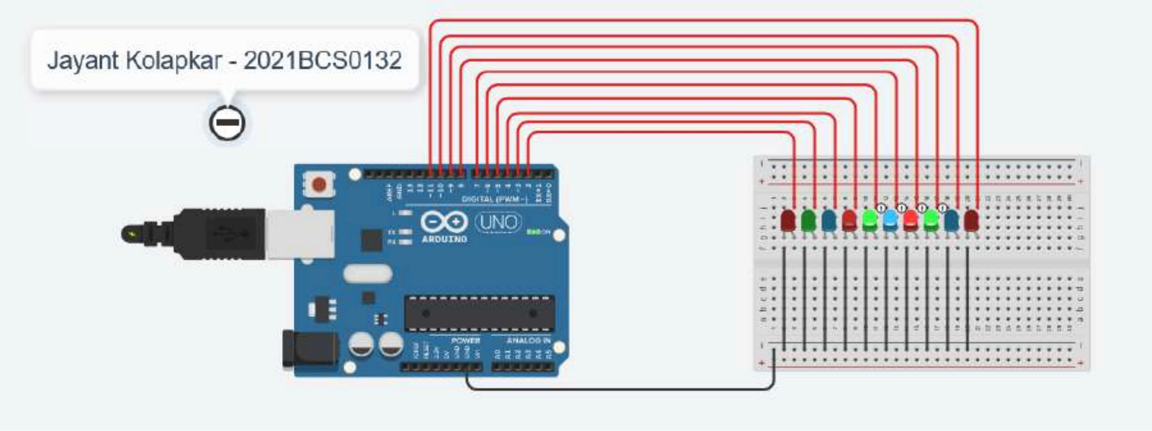
Output



Jayant Kolapkar - 2021BCS0132



Jayant Kolapkar - 2021BCS0132



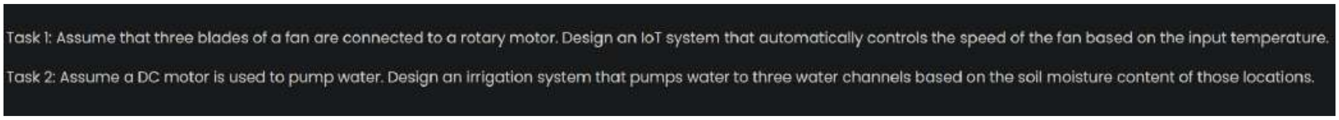
Jayant Kolapkar - 2021BCS0132

Lab 4

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question

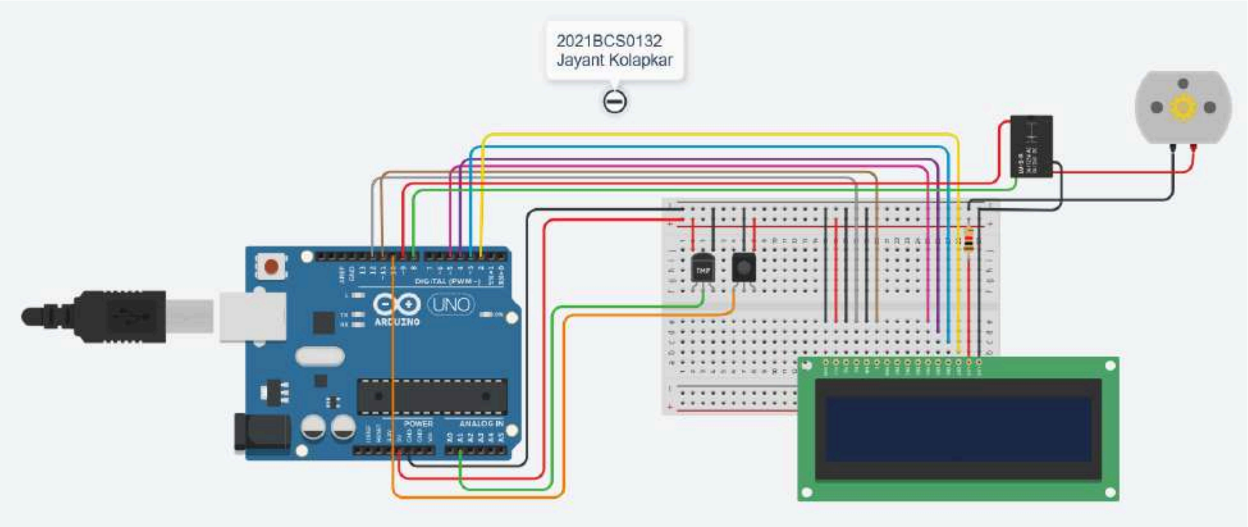


Task 1: Assume that three blades of a fan are connected to a rotary motor. Design an IoT system that automatically controls the speed of the fan based on the input temperature.

Task 2: Assume a DC motor is used to pump water. Design an irrigation system that pumps water to three water channels based on the soil moisture content of those locations.

Task 1

Diagram

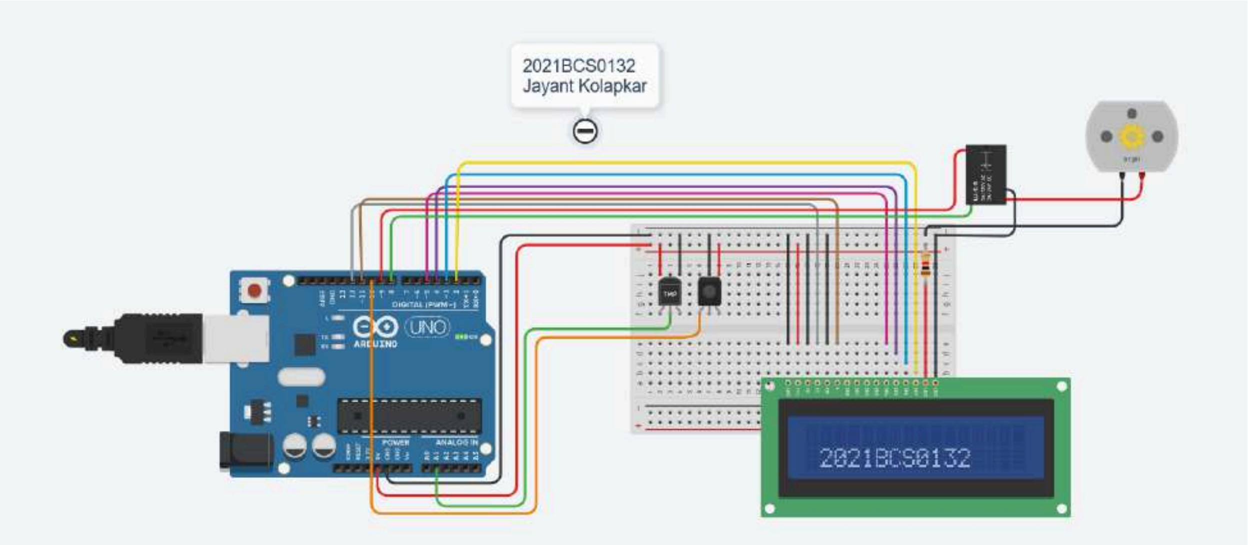


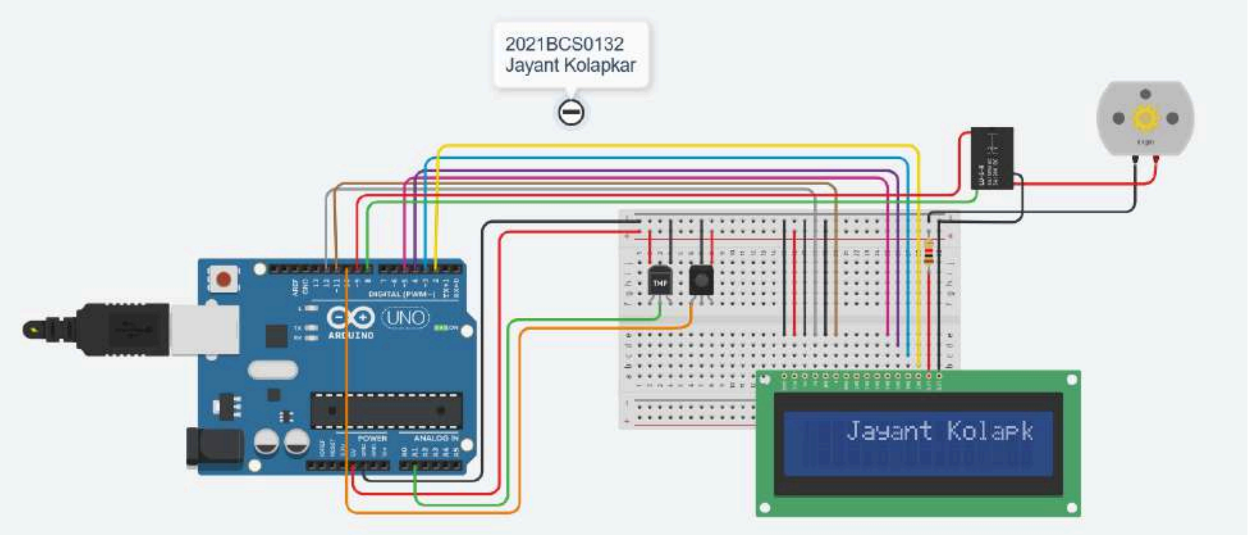
Code

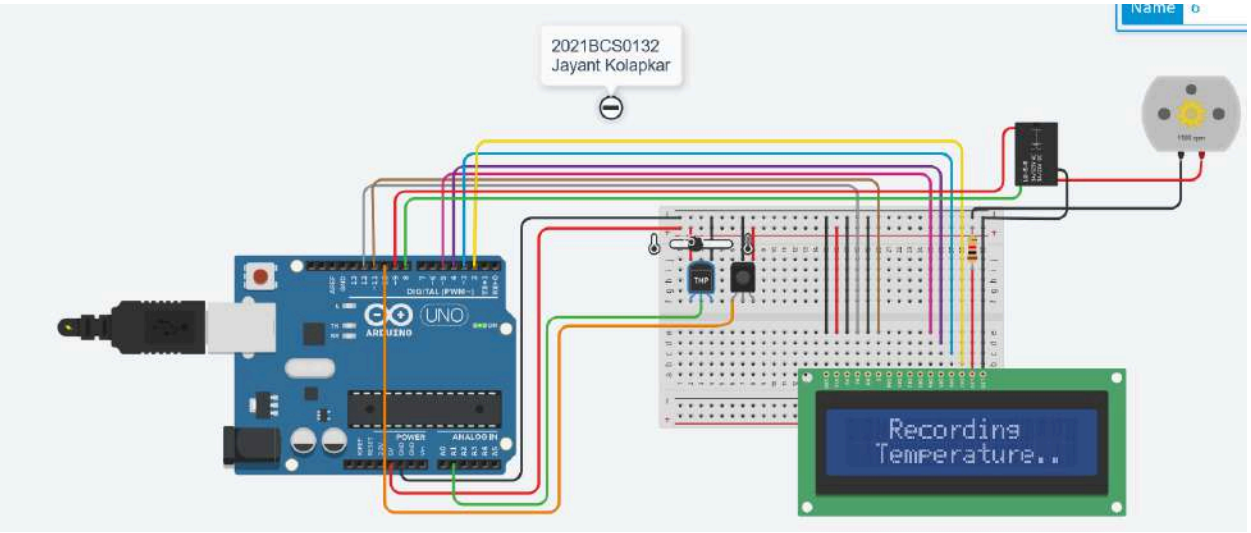
```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

float temp;
int tempPin = A1;
int relayPin = 8;

#define fan 9

void setup(){
  pinMode(fan, OUTPUT);
    pinMode(relayPin, OUTPUT);

    lcd.begin(16, 3);

    lcd.setCursor(1, 1);
    lcd.print("2021BCS0132");
    delay(1000);
    lcd.clear();
    lcd.setCursor(3,0);
    lcd.print("Jayant Kolapkar");
    delay(1000);
    lcd.clear();
    lcd.print("AUTO TEMPERATURE");
    delay(2000);
    lcd.clear();

}

void loop()
{
  lcd.setCursor(3,0);
    lcd.print("Recording");
  lcd.setCursor(2, 1);
  lcd.print("Temperature..");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0,2);
  temp = analogRead(tempPin);
  //temp = temp*0.48828125;
  float voltage = temp * 5.0;
  voltage /= 1024.0;

  // print out the voltage
  lcd.print(voltage); lcd.println(" volts");

  // now print out the temperature
  float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit 500 mV offset
                                                //to degrees ((voltage - 500mV) times 100)

  lcd.setCursor(0, 0);
  lcd.print("Temperature = ");
  lcd.setCursor(2,1);
  //lcd.print(temp);
  lcd.print(temperatureC); lcd.println(" degrees C");
  delay(3000);
  lcd.clear();

  if(temperatureC >= 20)
  {
    poweronRelay();
    if(temperatureC >= 20 && temperatureC <= 25)
    {
      analogWrite(fan,51);
      lcd.print("Fan Speed: 20% ");
      delay(2000);
      lcd.clear();
    }
    else if(temperatureC <= 35)
    {
      analogWrite(fan,102);
      lcd.print("Fan Speed: 40% ");
      delay(2000);
      lcd.clear();
    }
    else if(temperatureC <= 40)
    {
      analogWrite(fan,153);
      lcd.print("Fan Speed: 60% ");
      delay(2000);
      lcd.clear();
    }
    else if(temperatureC <= 44)
    {
      analogWrite(fan,200);
      lcd.print("Fan Speed: 80% ");
      delay(2000);
      lcd.clear();
    }
    else if(temperatureC >= 45)
    {
      analogWrite(fan,255);
      lcd.print("Fan Speed: 100% ");
      delay(2000);
      lcd.clear();
    }
  }
  else if(temperatureC < 20)
  {
    poweroffRelay();
  }


}

void poweronRelay()
  {
    digitalWrite(relayPin, HIGH);
    lcd.print("Fan ON");
    delay(2000);
    lcd.clear();
  }

void poweroffRelay()
  {
    digitalWrite(relayPin, LOW);
    analogWrite(fan,0);
    lcd.print("Fan OFF");
    delay(2000);
    lcd.clear();
  }
```

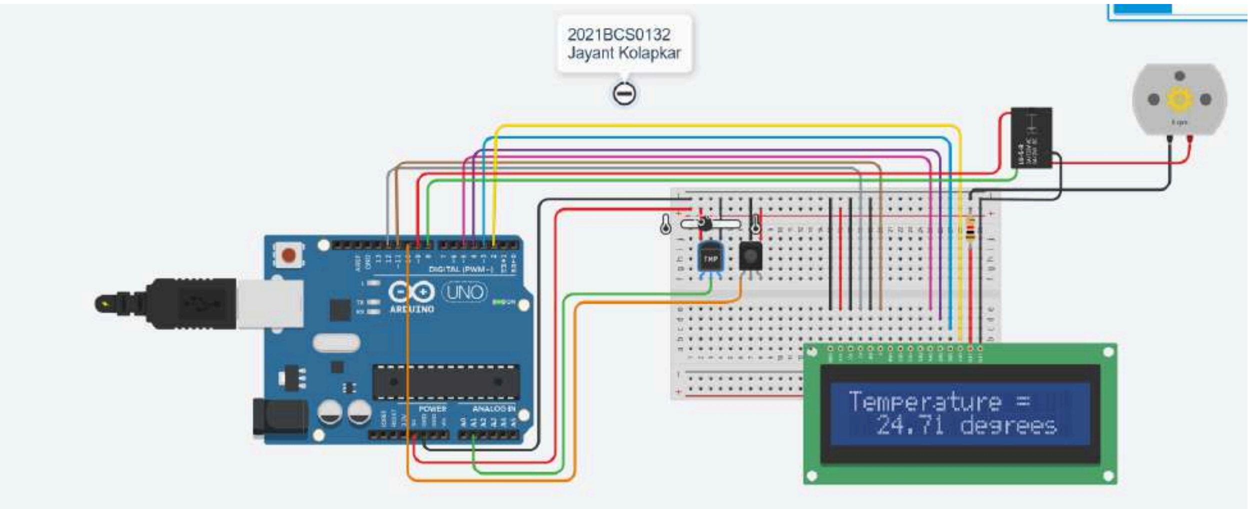
The system continuously reads temperature data from the analog sensor connected to pin A1. The raw analog values are converted into voltage, which is further translated into temperature in Celsius. The measured temperature is displayed on the LCD along with real-time updates. Based on the recorded temperature, the system adjusts the fan speed accordingly. If the temperature is **20°C or higher**, the relay is activated, and the fan is turned on with varying speeds. The fan operates at **20% speed** between **20°C and 25°C**, **40% speed** between **25°C and 35°C**, **60% speed** between **35°C and 40°C**, **80% speed** between **40°C and 44°C**, and at **full speed (100%)** for temperatures **45°C and above**. Each speed change is displayed on the LCD before being cleared for the next reading.

Output

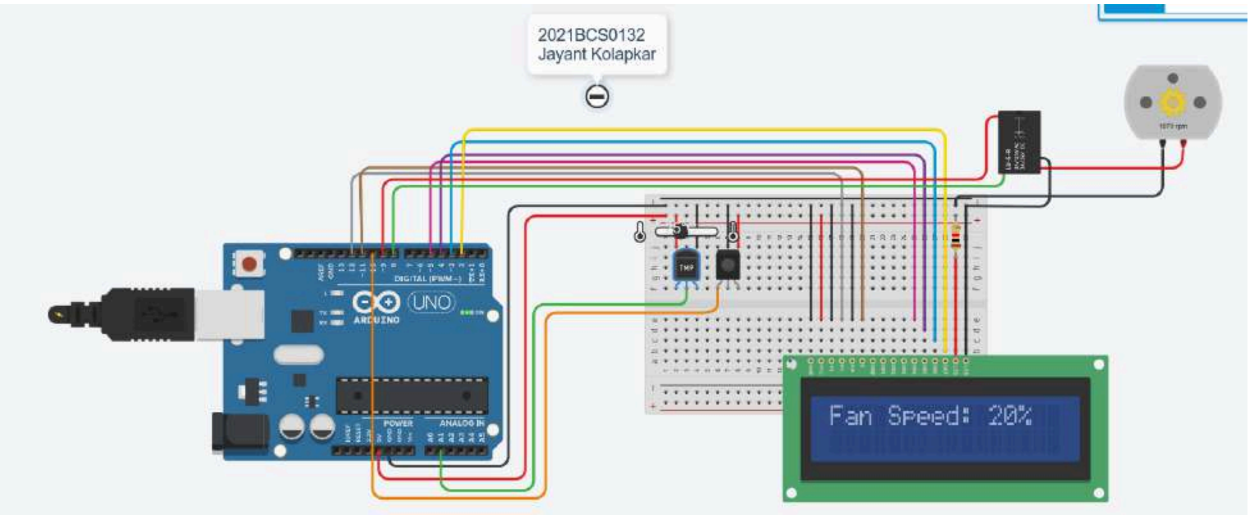



2021BCS0132
Jayant Kolapkar
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
TMP
Jayant Kolapk


Name
2021BCS0132
Jayant Kolapkar
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
TMP
Recording
Temperature..

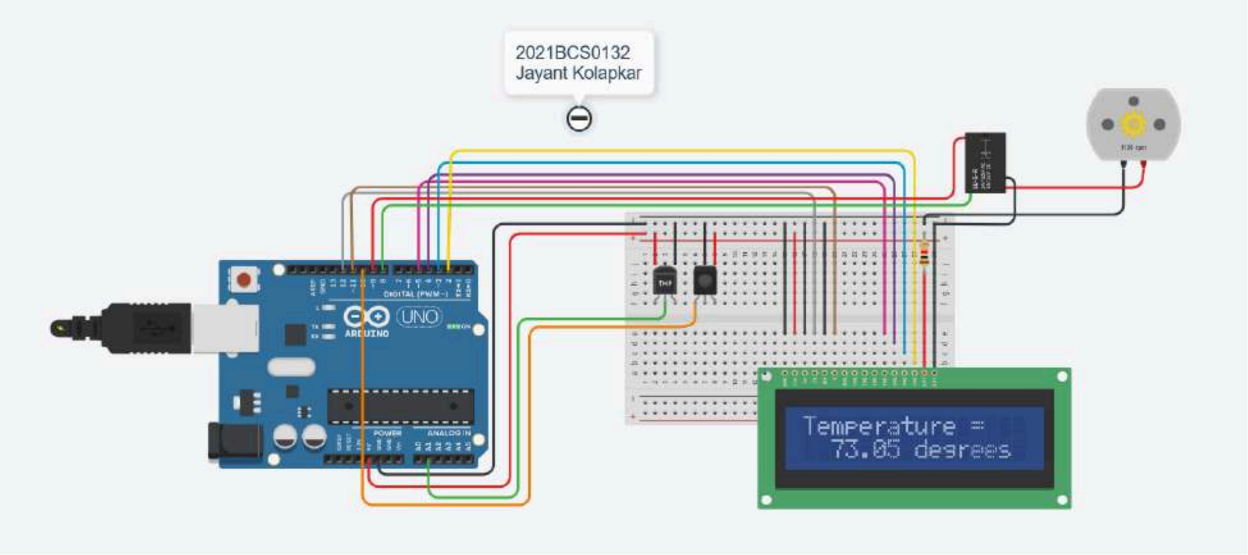


```
Temperature =
  24.71 degrees
```

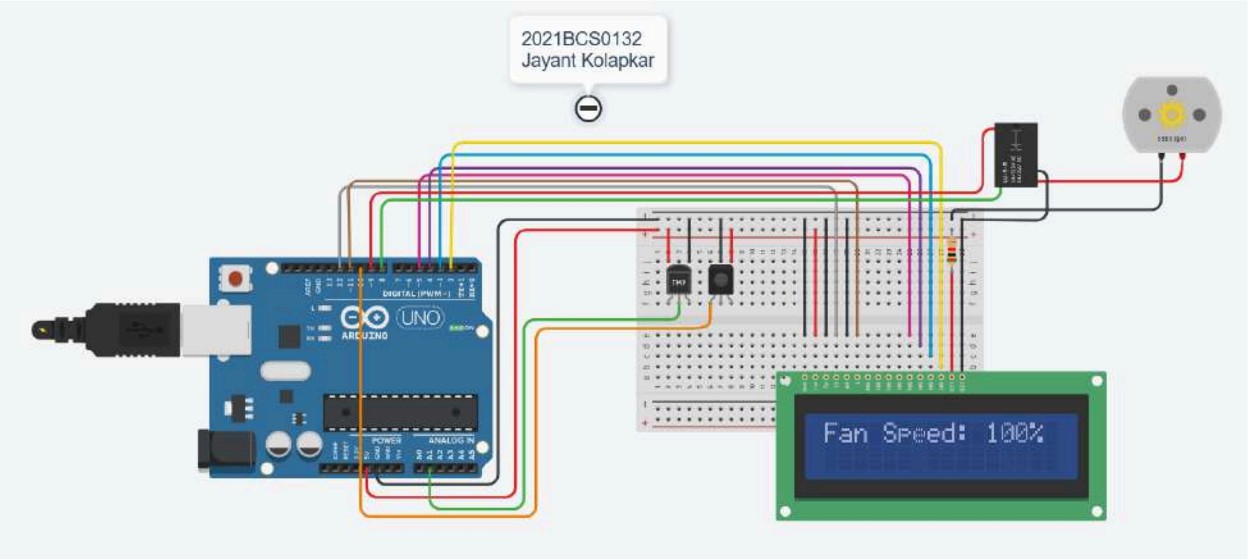


```
Fan Speed: 20%
```

Now we change the temperature.



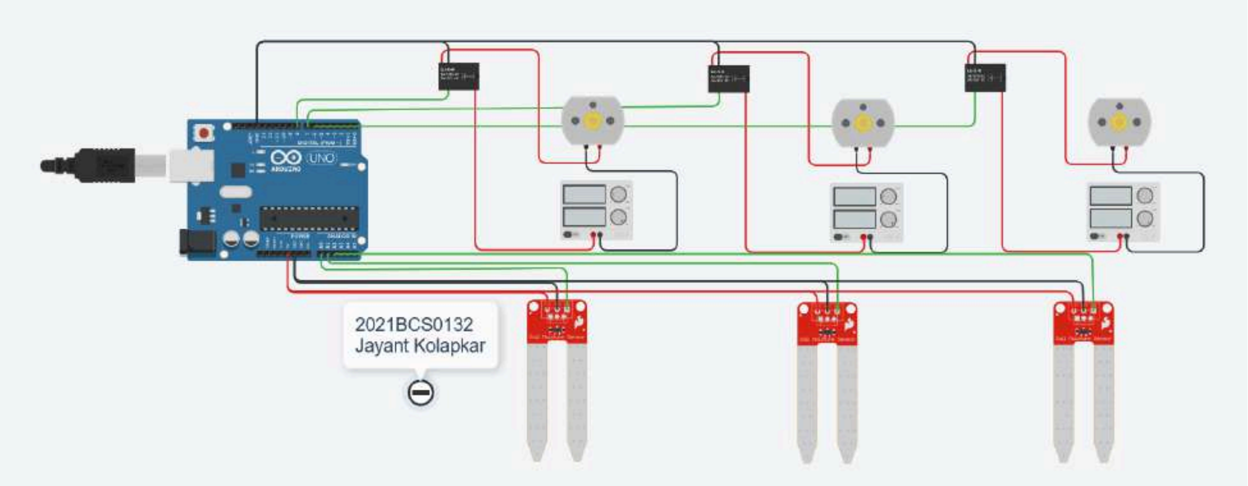
2021BCS0132
Jayant Kolapkar
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
Temperature =
73.05 degrees



2021BCS0132
Jayant Kolapkar
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
Fan Speed: 100%

Task 2

Diagram

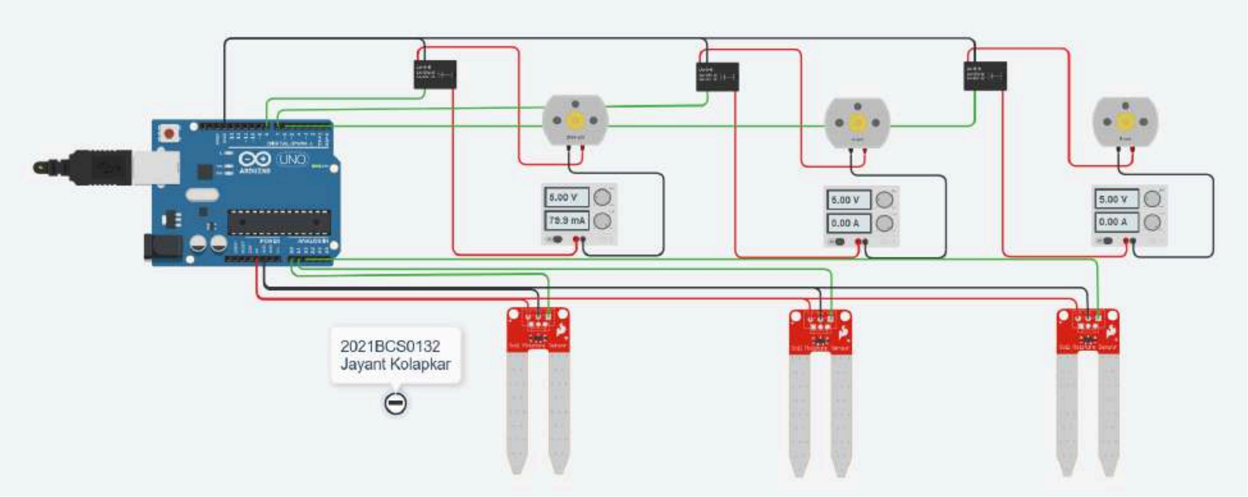


Code

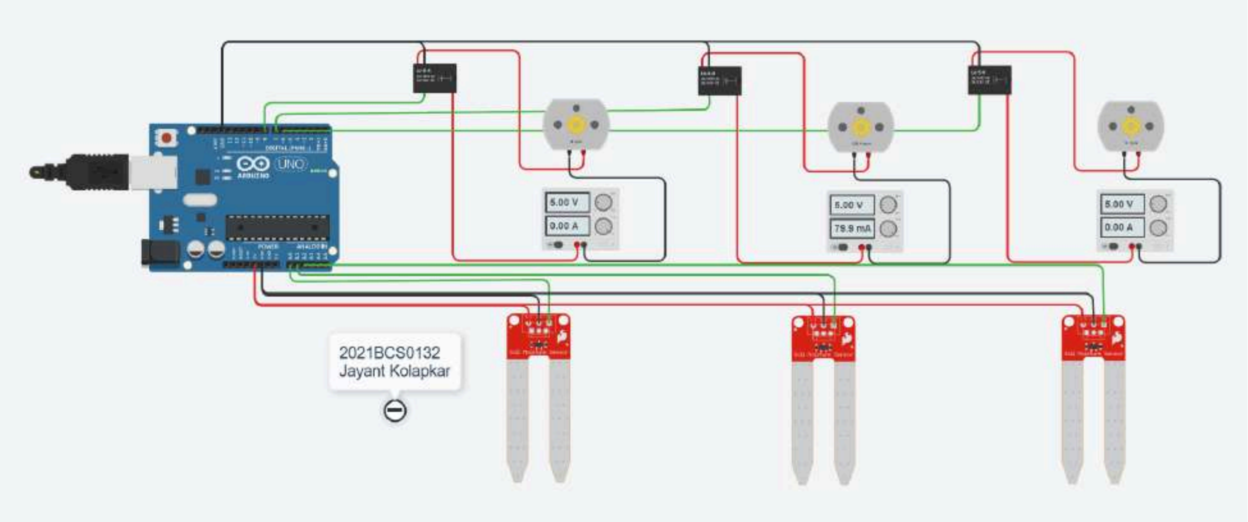
```
const int moistureSensorPins[] = {A0, A1, A2}; // Moisture sensors
const int relayPins[] = {8, 7, 6};             // Relay-controlled motors
const int moistureThreshold = 300;            // Threshold for dryness

void setup() {
  Serial.begin(9600);

  for (int i = 0; i < 3; i++) {
    pinMode(relayPins[i], OUTPUT);
    digitalWrite(relayPins[i], LOW); // Ensure motors are off initially
  }
}

void loop() {
  for (int i = 0; i < 3; i++) {
    int moistureLevel = analogRead(moistureSensorPins[i]);
    Serial.print("Soil Moisture Level (Sensor ");
    Serial.print(i + 1);
    Serial.print("): ");
    Serial.println(moistureLevel);

    if (moistureLevel < moistureThreshold) {
      Serial.print("Soil is dry at Sensor ");
      Serial.print(i + 1);
      Serial.println("! Watering the plant...");
      digitalWrite(relayPins[i], HIGH);
      delay(5000);
      digitalWrite(relayPins[i], LOW);
      Serial.println("Watering complete.");
    } else {
      Serial.print("Soil moisture is sufficient at Sensor ");
      Serial.println(i + 1);
    }
  }

  delay(10000); // Wait before the next reading
}
```

This code will check the moisture levels of three sensors and activate the corresponding motor if the soil is too dry. Each motor runs independently for 5 seconds when needed, and the system waits 10 seconds before checking again.

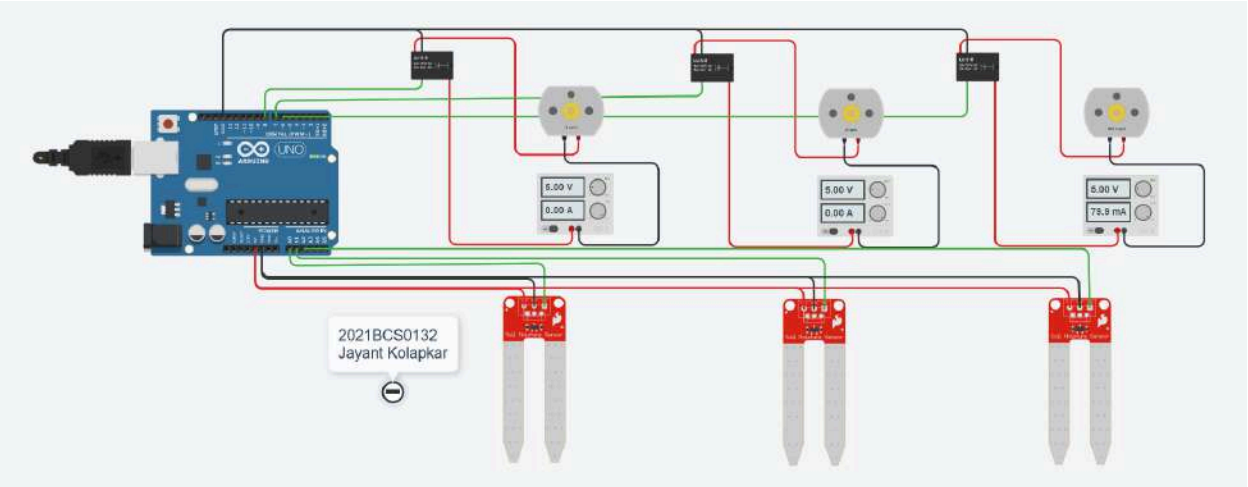
Output



First motor runs as it’s soil moisture level is low.



Then the 2nd sensor is checked and the motor is run.



At last the 3rd sensor is checked and the motor is run.

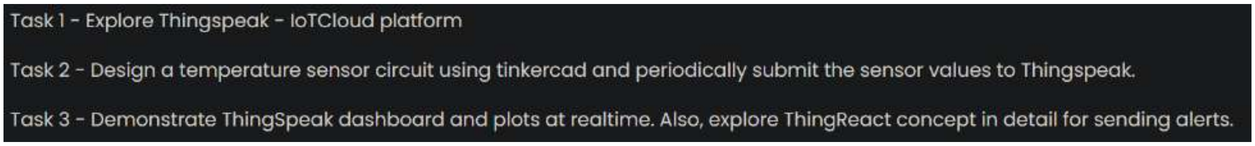
The loop then repeats infinitely, and if the soil moisture level is sufficient, that channel is skipped.

Lab 5

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



Task 1 - Explore Thingspeak - IoTCloud platform

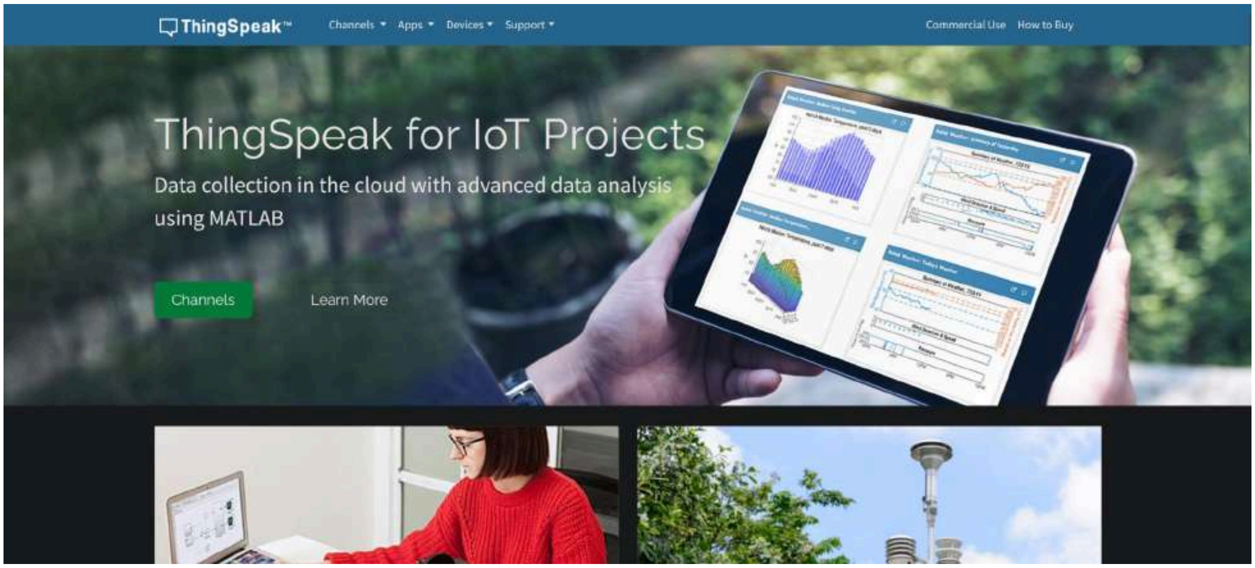
Task 2 - Design a temperature sensor circuit using tinkercad and periodically submit the sensor values to Thingspeak.

Task 3 - Demonstrate ThingSpeak dashboard and plots at realtime. Also, explore ThingReact concept in detail for sending alerts.

Task 1

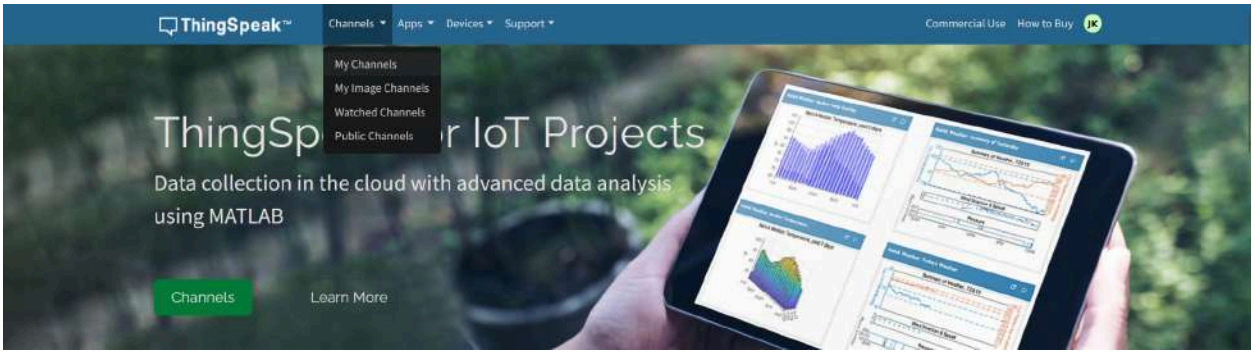
Introduction to ThingSpeak - IoT Cloud Platform

1. **Cloud-Based IoT Analytics** – ThingSpeak is an open-source IoT platform designed for real-time collection, analysis, and visualization of sensor data.
2. **MATLAB Integration** – It seamlessly integrates with MATLAB, enabling advanced data processing, machine learning, and predictive analytics.
3. **RESTful API Support** – ThingSpeak offers RESTful APIs for reading and writing data, allowing easy connectivity with IoT devices, web applications, and cloud services.
4. **Data Storage & Visualization** – Users can store sensor data in channels and visualize it through built-in graphs and charts for deeper insights.
5. **Triggers & Alerts** – It supports event-driven automation, allowing users to trigger actions such as notifications or device control based on sensor readings.



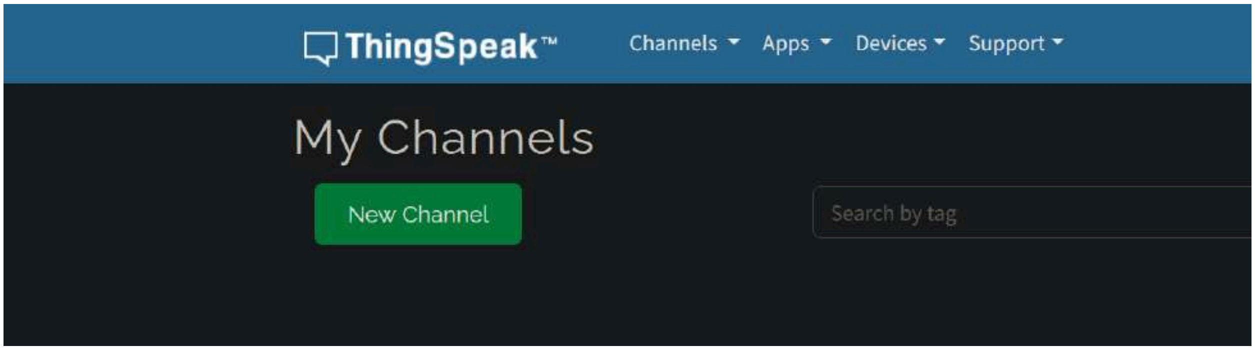
ThingSpeak for IoT Projects

Data collection in the cloud with advanced data analysis using MATLAB

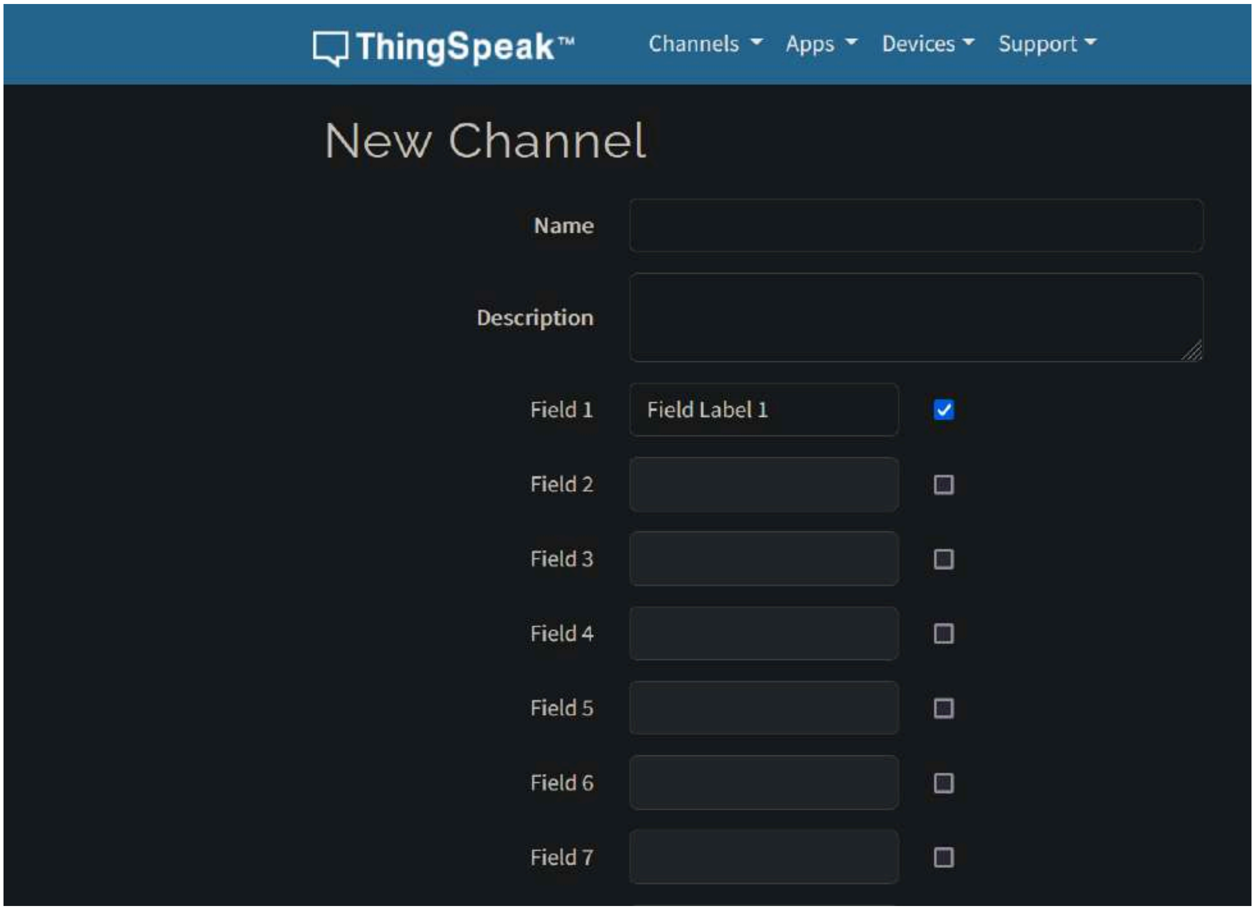


ThingSp r IoT Projects

Data collection in the cloud with advanced data analysis using MATLAB



My Channels



New Channel

Name		
Description		
Field 1	Field Label 1	☑
Field 2		□
Field 3		□
Field 4		□
Field 5		□
Field 6		□
Field 7		□

Task 2

Design a Temperature Sensor Circuit Using Tinkercad & Transmit Data to ThingSpeak

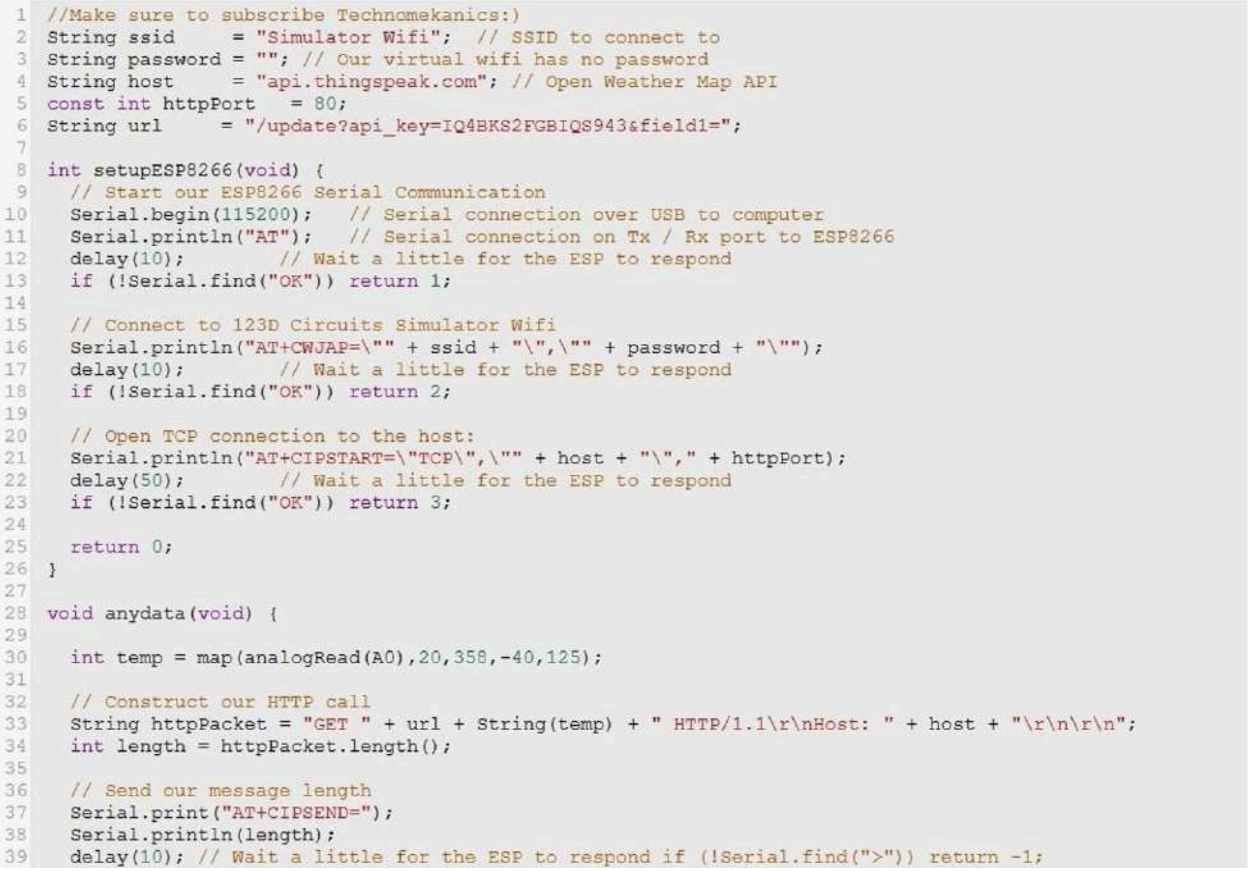
Algorithm:

1. **Initialize Serial Communication & Wi-Fi Connection**
   - Start serial communication at 115200 baud rate.
   - Send AT commands to verify ESP8266 response.
   - Connect to the Wi-Fi network using the `AT+CWJAP` command.
2. **Establish Connection with ThingSpeak**

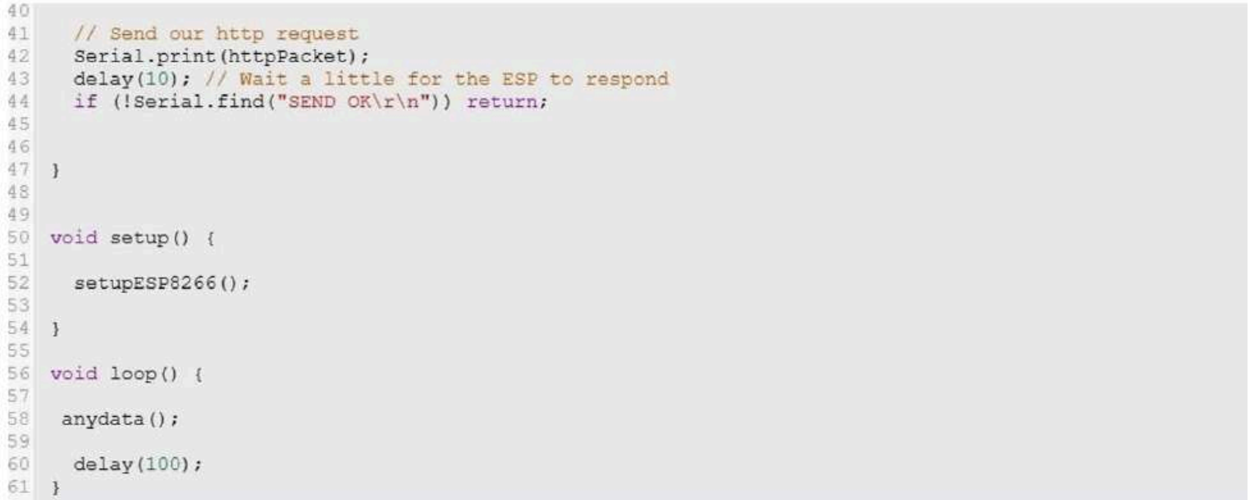
   - Use the `AT+CIPSTART` command to create a TCP connection to ThingSpeak's API server (`api.thingspeak.com`) on port 80.

3. **Read Sensor Data**
   - Capture analog input from pin A0, corresponding to the temperature sensor.
   - Convert the analog reading to a temperature value using the `map()` function.
4. **Transmit Data to ThingSpeak**
   - Construct an HTTP GET request containing the API key and temperature value.
   - Send the request using `AT+CIPSEND`, followed by the actual HTTP request.
5. **Repeat Data Transmission**
   - Continuously send sensor readings within the `loop()` function at intervals of 100 milliseconds.

Code:



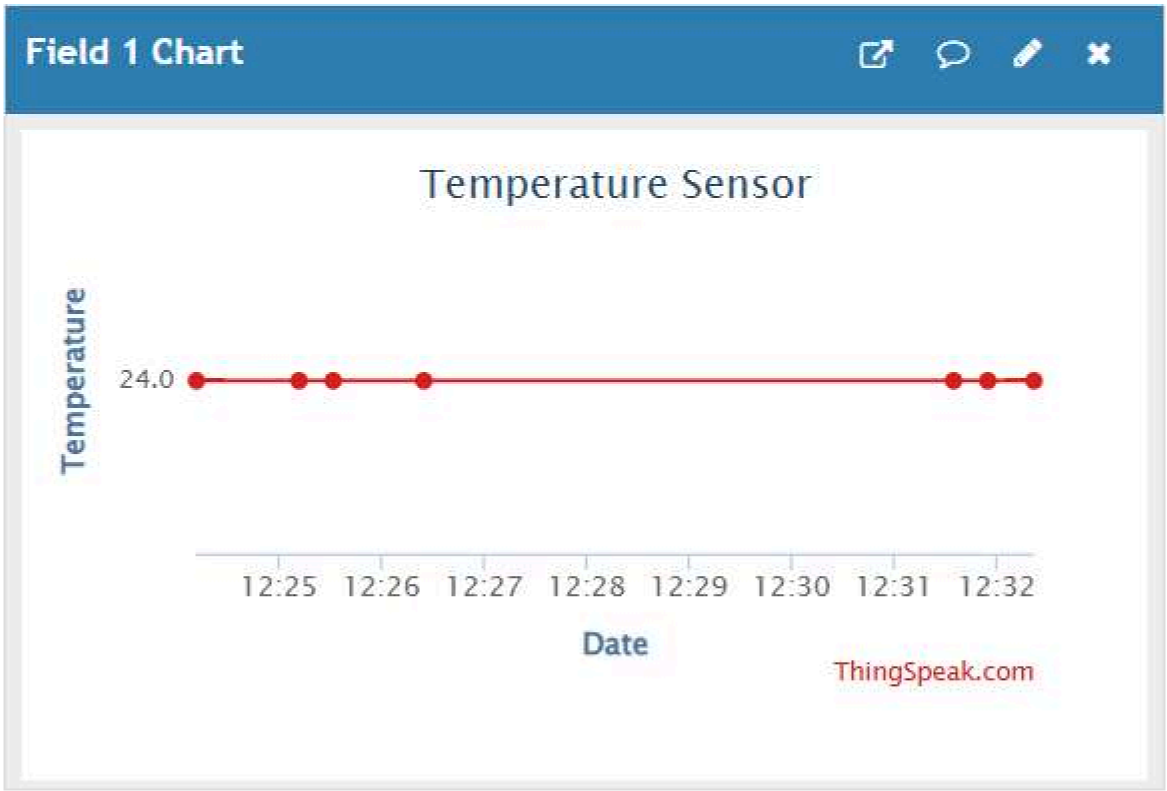
```
//Make sure to subscribe Technomekanics:)
String ssid     = "Simulator Wifi";  // SSID to connect to
String password = ""; // Our virtual wifi has no password
String host     = "api.thingspeak.com"; // Open Weather Map API
const int httpPort   = 80;
String url     = "/update?api_key=IQ4BKS2FGBIQS943&field1=";

int setupESP8266(void) {
  // Start our ESP8266 Serial Communication
  Serial.begin(115200);   // Serial connection over USB to computer
  Serial.println("AT");   // Serial connection on Tx / Rx port to ESP8266
  delay(10);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 1;

  // Connect to 123D Circuits Simulator Wifi
  Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
  delay(10);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 2;

  // Open TCP connection to the host:
  Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
  delay(50);        // Wait a little for the ESP to respond
  if (!Serial.find("OK")) return 3;

  return 0;
}

void anydata(void) {

  int temp = map(analogRead(A0),20,358,-40,125);

  // Construct our HTTP call
  String httpPacket = "GET " + url + String(temp) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
  int length = httpPacket.length();

  // Send our message length
  Serial.print("AT+CIPSEND=");
  Serial.println(length);
  delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;
```



```

  // Send our http request
  Serial.print(httpPacket);
  delay(10); // Wait a little for the ESP to respond
  if (!Serial.find("SEND OK\r\n")) return;


}


void setup() {

  setupESP8266();

}

void loop() {

 anydata();

  delay(100);
}
```

Output:

- Temperature data is periodically submitted to ThingSpeak for real-time monitoring.



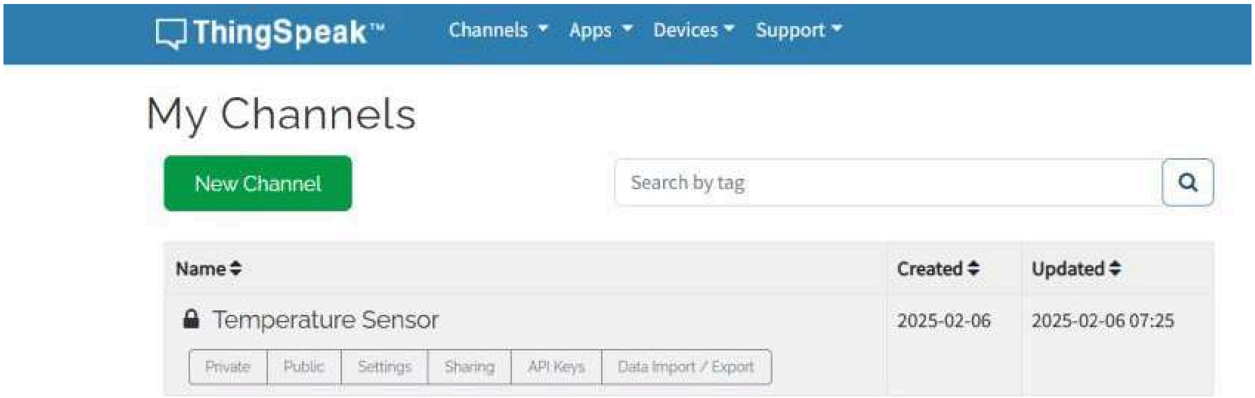
Task 3

Real-Time Data Visualization & ThingReact Alerts

ThingSpeak enables real-time visualization of sensor data through customizable dashboards.

Steps to Create a ThingSpeak Channel:

1. **Create a New Channel** – Define a name and specify fields (e.g., Temperature, Humidity).
2. **Send Sensor Data** – Use an ESP8266, ESP32, or Arduino to transmit data using an API key.
3. **Visualize Data** – Access the “Private View” or “Public View” tab to observe live graphs.
4. **Customize Widgets** – Utilize widgets like line charts, gauges, and histograms to analyze trends.



Name	Created	Updated
Temperature Sensor	2025-02-06	2025-02-06 07:25

ThingReact – Event-Based Trigger System

ThingReact enables automated responses based on sensor data thresholds.

How ThingReact Works:

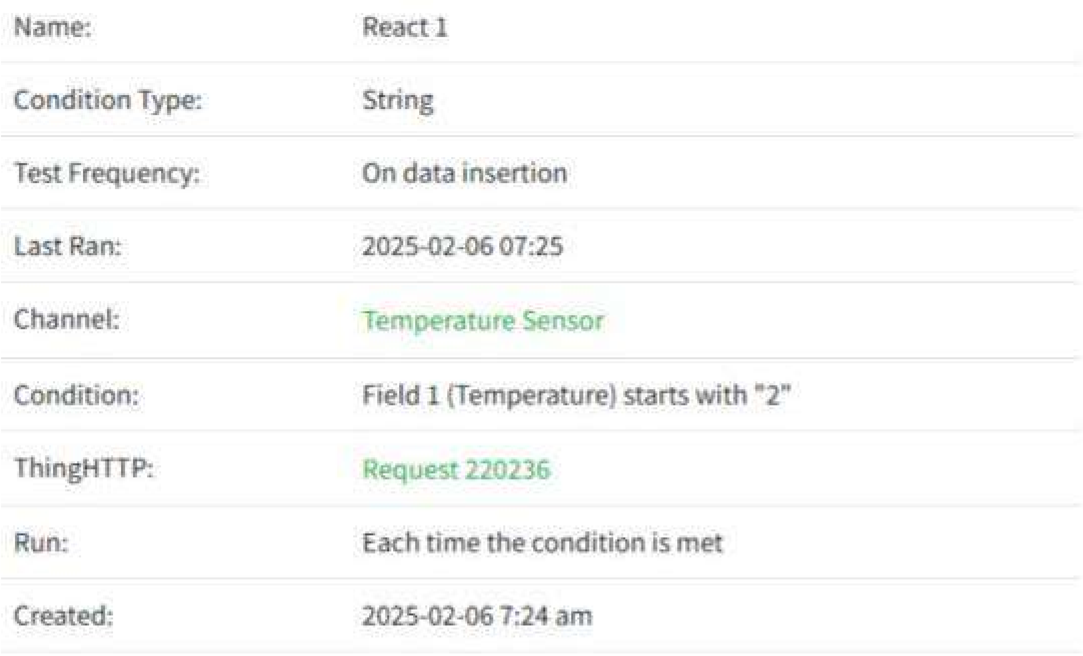
- Continuously monitors real-time data from a ThingSpeak channel.
- Triggers a response when a predefined condition is met (e.g., temperature surpassing 50°C).
- Sends notifications via email, tweets, or HTTP requests to external services like IFTTT or Telegram.

Steps to Use ThingReact:

1. Navigate to **Apps → ThingReact** on ThingSpeak.
2. Click **Create New Reaction** and set conditions:
   - **Example Condition:** If Temperature (Field1) starts with “2”.
   - **Example Action:** Send an email alert or trigger a webhook.
3. Select an Action Type (HTTP Request, Twitter Post, Email Notification).
4. Enable the Reaction and monitor real-time alerts.

Practical Use Cases:

- **Smart Agriculture:** Send SMS alerts when soil moisture drops below a threshold.
- **Industrial Safety:** Trigger an alarm if gas levels exceed safe limits.
- **Home Automation:** Automatically switch on a fan when room temperature crosses a set value.



Name:	React 1
Condition Type:	String
Test Frequency:	On data insertion
Last Ran:	2025-02-06 07:25
Channel:	Temperature Sensor
Condition:	Field 1 (Temperature) starts with "2"
ThingHTTP:	Request 220236
Run:	Each time the condition is met
Created:	2025-02-06 7:24 am

Lab 6

ICS 423: Internet Of Things

Jayant Kolapkar - 2021BCS0132

Task 1: Design an alarm system that triggers warning if a package gets damaged. Use force sensor and a buzzer alarm.

Algorithm

1. Initialize Components

- Define force sensor pin (Analog A0).
- Define buzzer pin (Digital 9).
- Define LED pin (Digital 7).
- Set the threshold value for the force sensor.

2. Setup Function (Runs Once)

- · Start the Serial Monitor for debugging.
- Set LED and Buzzer as OUTPUT.

3. Loop Function (Repeats Continuously)

- Read the force sensor value from A0.
- Print the sensor value to the Serial Monitor.
- Compare the sensor value with the threshold:
  - If the value is greater than the threshold:
    - Turn ON the LED.
    - Turn ON the Buzzer.
  - Else:
    - Turn OFF the LED.
    - Turn OFF the Buzzer.

- Add a small delay for stability.

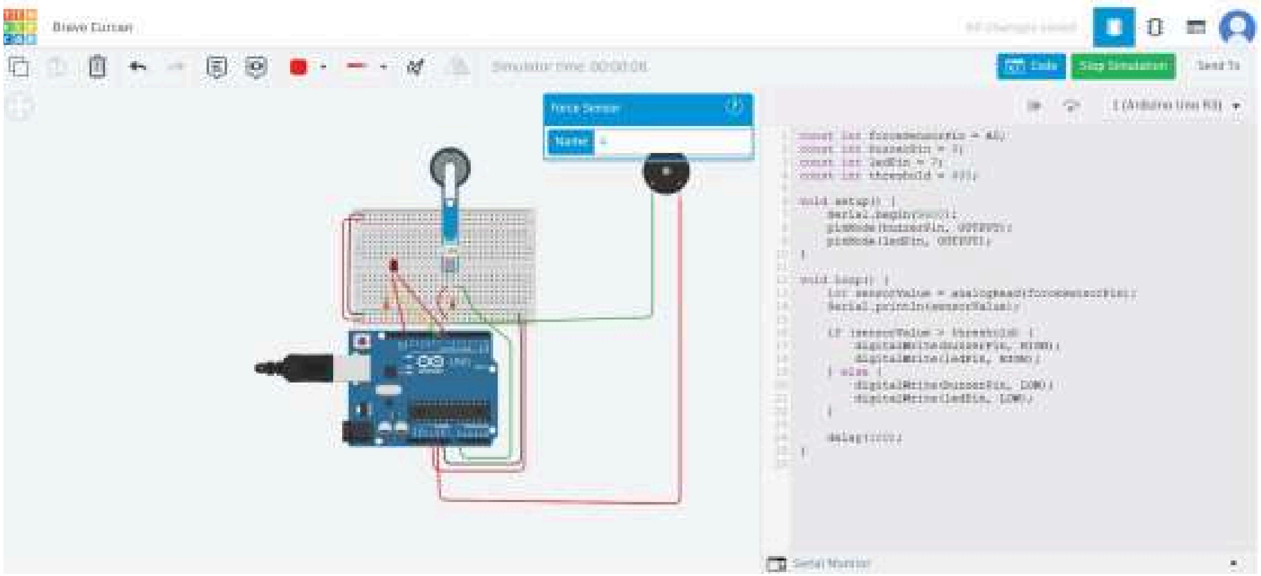
Code

```
const int forceSensorPin = A0;
const int buzzerPin = 9;
const int ledPin = 7;
const int threshold = 400;

void setup() {
	Serial.begin(9600);
	pinMode(buzzerPin, OUTPUT);
	pinMode(ledPin, OUTPUT);
}

void loop() {
	int sensorValue = analogRead(forceSensorPin);
	Serial.println(sensorValue);

	if (sensorValue > threshold) {
	digitalWrite(buzzerPin, HIGH);
```

```
        digitalWrite(ledPin, HIGH);

        } else {

        digitalWrite(buzzerPin, LOW);

        digitalWrite(ledPin, LOW);

        }



        delay(100);

}
```

Output

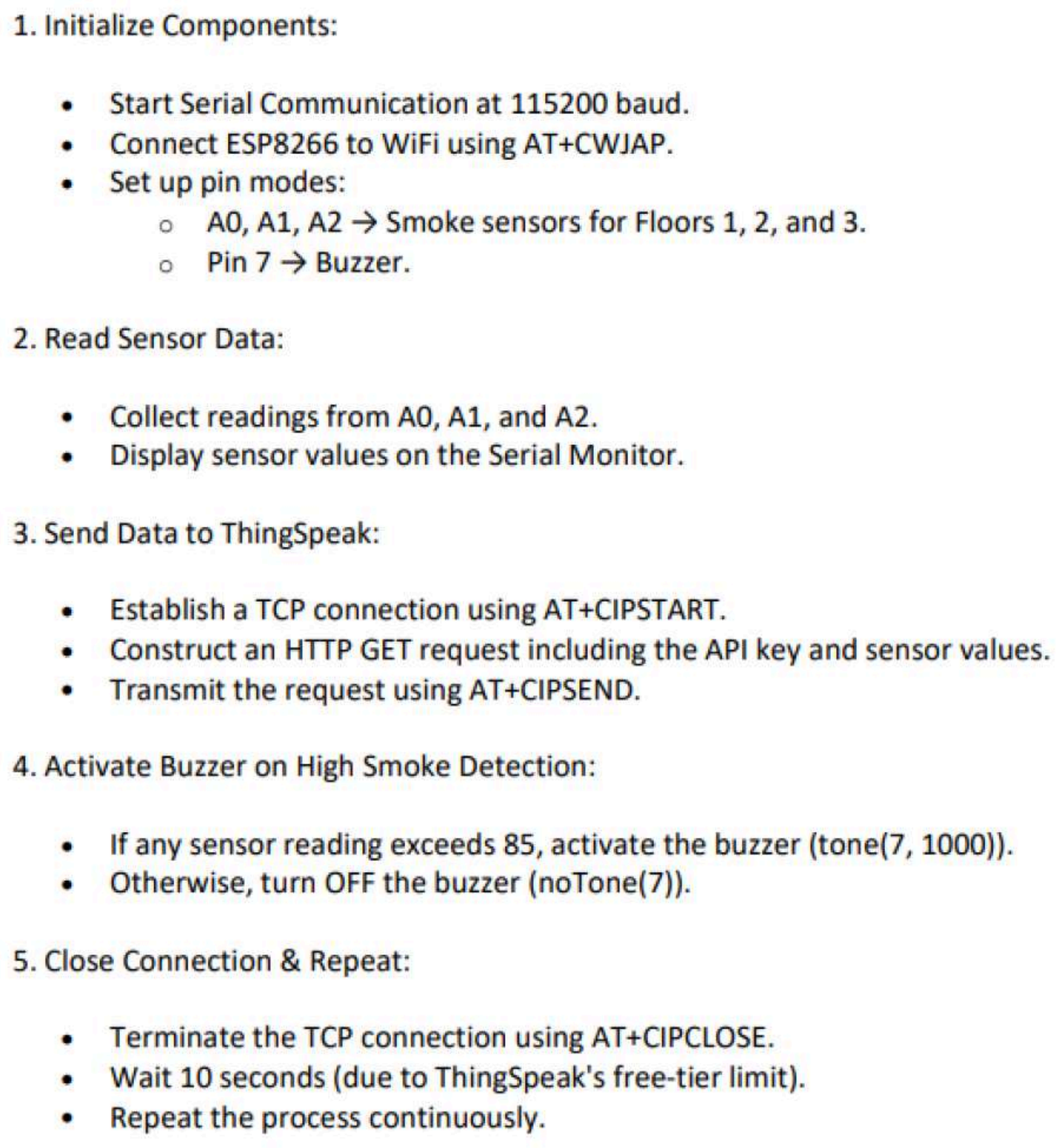
The alarm and the LED bulb are off when the force applied is below the threshold.



```
const int forceSensorPin = A0;
const int buzzerPin = [illegible];
const int ledPin = 7;
const int threshold = [illegible];

void setup() {
    Serial.begin([illegible]);
    pinMode(buzzerPin, OUTPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    int sensorValue = analogRead(forceSensorPin);
    Serial.println(sensorValue);

    if (sensorValue > threshold) {
        digitalWrite(buzzerPin, HIGH);
        digitalWrite(ledPin, HIGH);
    } else {
        digitalWrite(buzzerPin, LOW);
        digitalWrite(ledPin, LOW);
    }

    delay([illegible]);
}
```

The alarm rings and the LED bulb glows when the force crosses the threshold limit.

Task 2: Develop a smoke detecting system for a building consisting of three floors. Notify the real-time value of the floors to thingspeak IoT cloud platform.



1. Initialize Components:

- Start Serial Communication at 115200 baud.
- Connect ESP8266 to WiFi using AT+CWJAP.
- Set up pin modes:
  - A0, A1, A2 → Smoke sensors for Floors 1, 2, and 3.
  - Pin 7 → Buzzer.

2. Read Sensor Data:

- Collect readings from A0, A1, and A2.
- Display sensor values on the Serial Monitor.

3. Send Data to ThingSpeak:

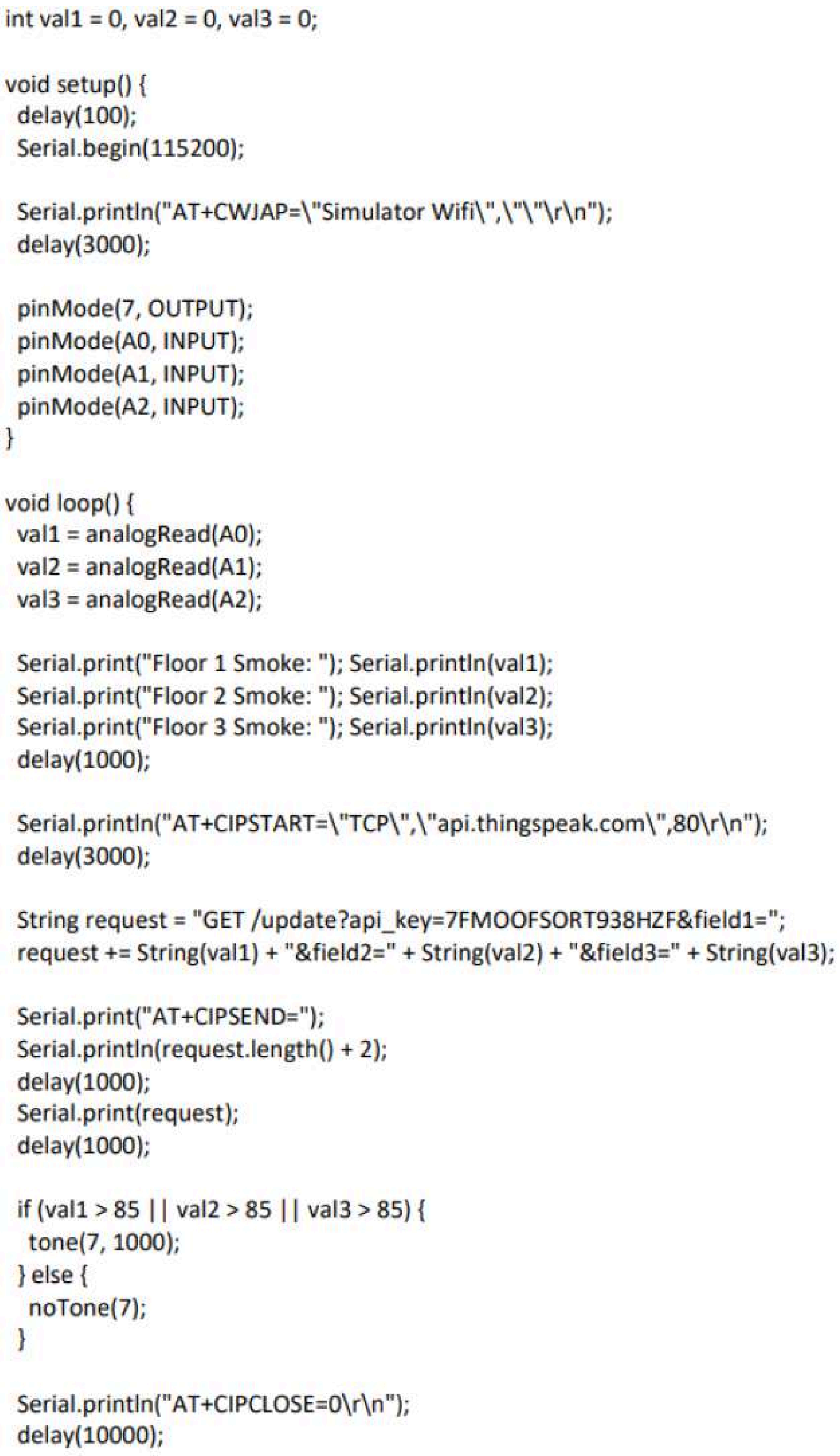
- Establish a TCP connection using AT+CIPSTART.
- Construct an HTTP GET request including the API key and sensor values.
- Transmit the request using AT+CIPSEND.

4. Activate Buzzer on High Smoke Detection:

- If any sensor reading exceeds 85, activate the buzzer (tone(7, 1000)).
- Otherwise, turn OFF the buzzer (noTone(7)).

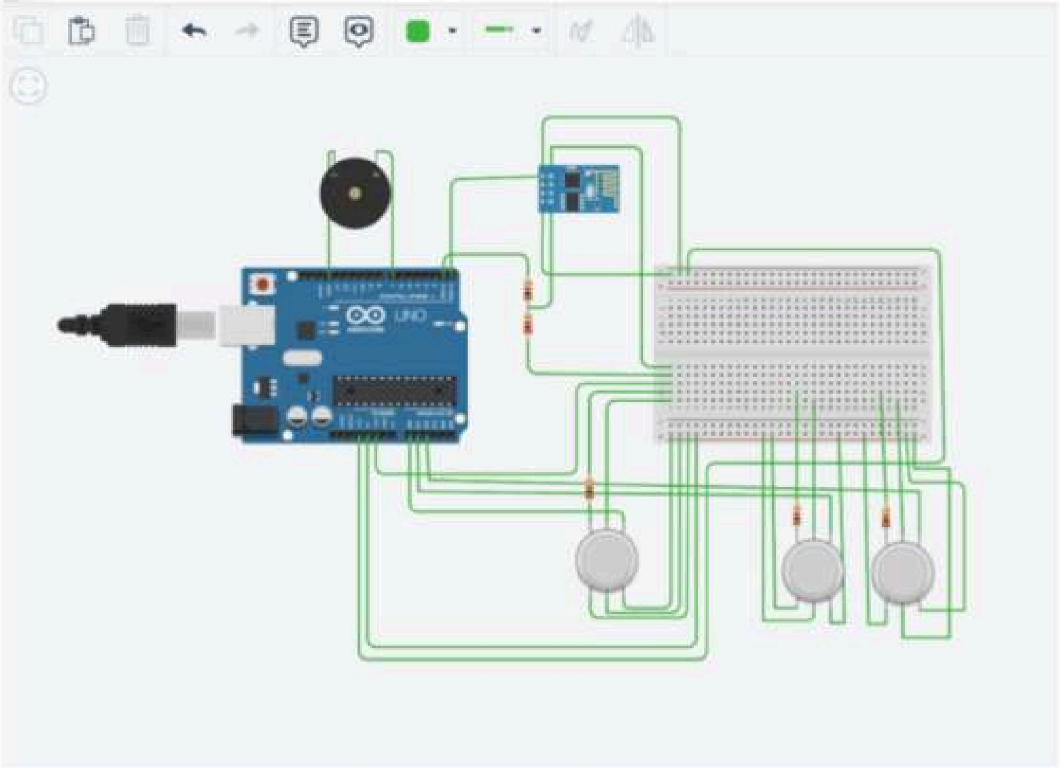
5. Close Connection & Repeat:

- Terminate the TCP connection using AT+CIPCLOSE.
- Wait 10 seconds (due to ThingSpeak's free-tier limit).
- Repeat the process continuously.

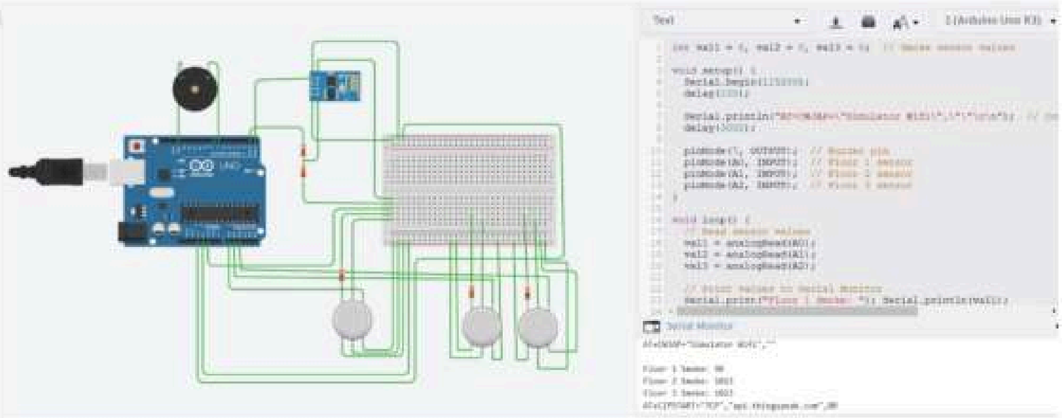


```
int val1 = 0, val2 = 0, val3 = 0;

void setup() {
  delay(100);
  Serial.begin(115200);

  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n");
  delay(3000);

  pinMode(7, OUTPUT);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
}

void loop() {
  val1 = analogRead(A0);
  val2 = analogRead(A1);
  val3 = analogRead(A2);

  Serial.print("Floor 1 Smoke: "); Serial.println(val1);
  Serial.print("Floor 2 Smoke: "); Serial.println(val2);
  Serial.print("Floor 3 Smoke: "); Serial.println(val3);
  delay(1000);

  Serial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");
  delay(3000);

  String request = "GET /update?api_key=7FMOOFSORT938HZF&field1=";
  request += String(val1) + "&field2=" + String(val2) + "&field3=" + String(val3);

  Serial.print("AT+CIPSEND=");
  Serial.println(request.length() + 2);
  delay(1000);
  Serial.print(request);
  delay(1000);

  if (val1 > 85 || val2 > 85 || val3 > 85) {
    tone(7, 1000);
  } else {
    noTone(7);
  }

  Serial.println("AT+CIPCLOSE=0\r\n");
  delay(10000);
```

Output



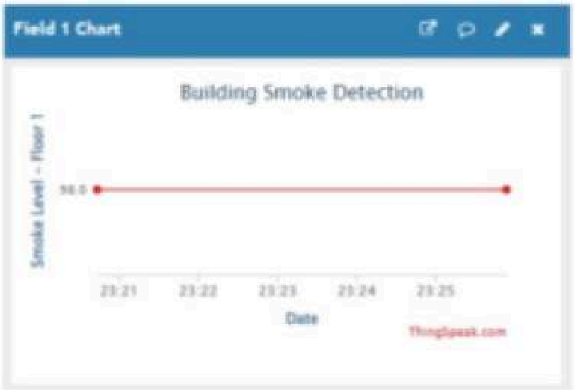
Simulation

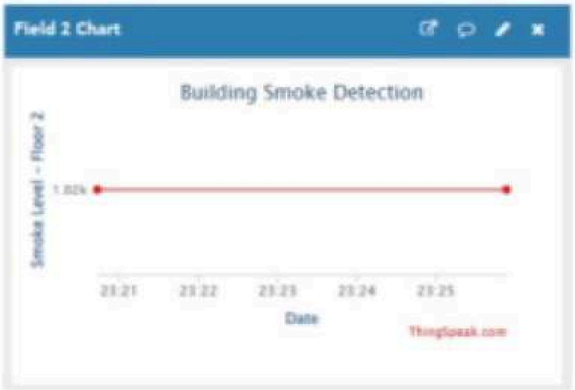


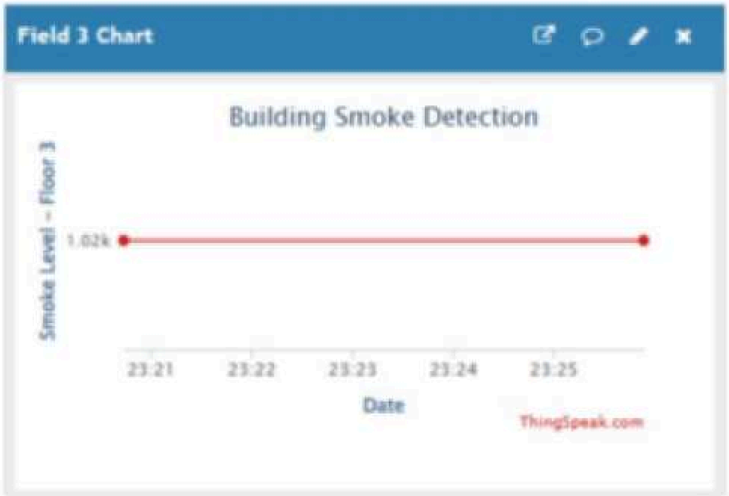
```
int val1 = 0, val2 = 0, val3 = 0;  // Smoke sensor values

void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println("AT+CWJAP=\"Simulator Wifi\",\"\"\r\n"); // Co
  delay(3000);

  pinMode(7, OUTPUT);  // Buzzer pin
  pinMode(A0, INPUT);  // Floor 1 sensor
  pinMode(A1, INPUT);  // Floor 2 sensor
  pinMode(A2, INPUT);  // Floor 3 sensor
}

void loop() {
  // Read sensor values
  val1 = analogRead(A0);
  val2 = analogRead(A1);
  val3 = analogRead(A2);

  // Print values to Serial Monitor
  Serial.print("Floor 1 Smoke: "); Serial.println(val1);
```

```
AT+CWJAP="Simulator Wifi",""

Floor 1 Smoke: 98
Floor 2 Smoke: 1023
Floor 3 Smoke: 1023
AT+CIPSTART="TCP","api.thingspeak.com",80
```

Graph visualization on ThingSpeak


Field 1 Chart
Building Smoke Detection
Smoke Level - Floor 1
98.0
23:21
23:22
23:23
23:24
23:25
Date
ThingSpeak.com


Field 2 Chart
Building Smoke Detection
Smoke Level - Floor 2
1.02k
23:21
23:22
23:23
23:24
23:25
Date
ThingSpeak.com

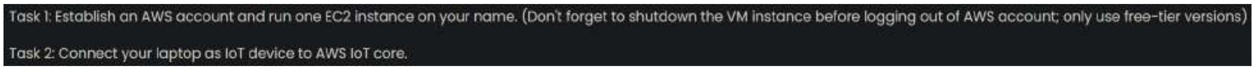

Field 3 Chart
Building Smoke Detection
Smoke Level - Floor 3
1.02k
23:21
23:22
23:23
23:24
23:25
Date
ThingSpeak.com

Lab 7

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question

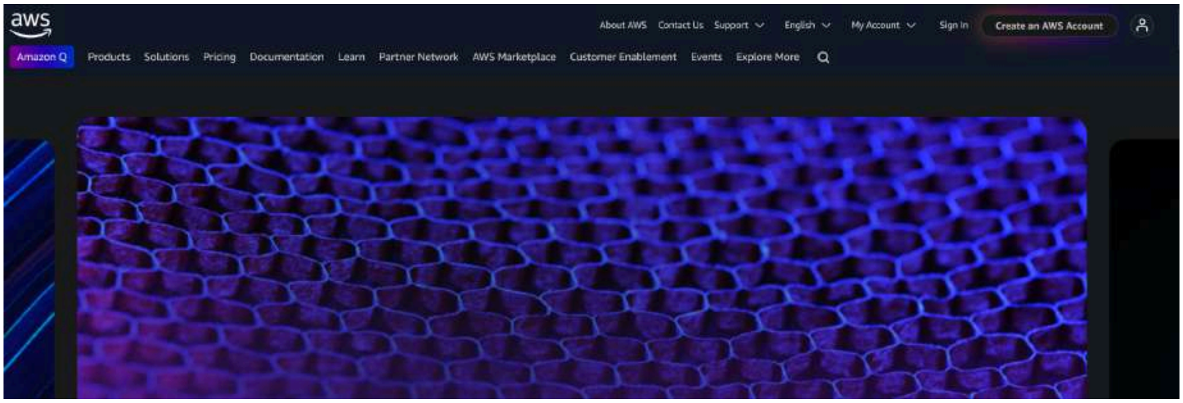


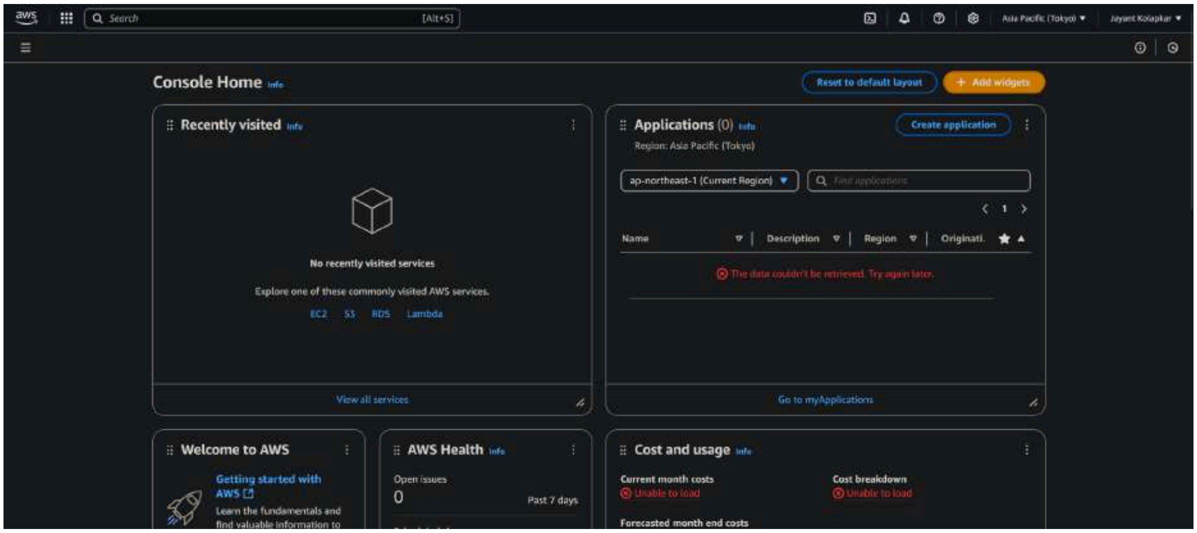
Task 1: Establish an AWS account and run one EC2 instance on your name. (Don't forget to shutdown the VM instance before logging out of AWS account; only use free-tier versions)

Task 2: Connect your laptop as IoT device to AWS IoT core.

Task 1

Create an AWS account and sign up for the free plan





No recently visited services

Explore one of these commonly visited AWS services.

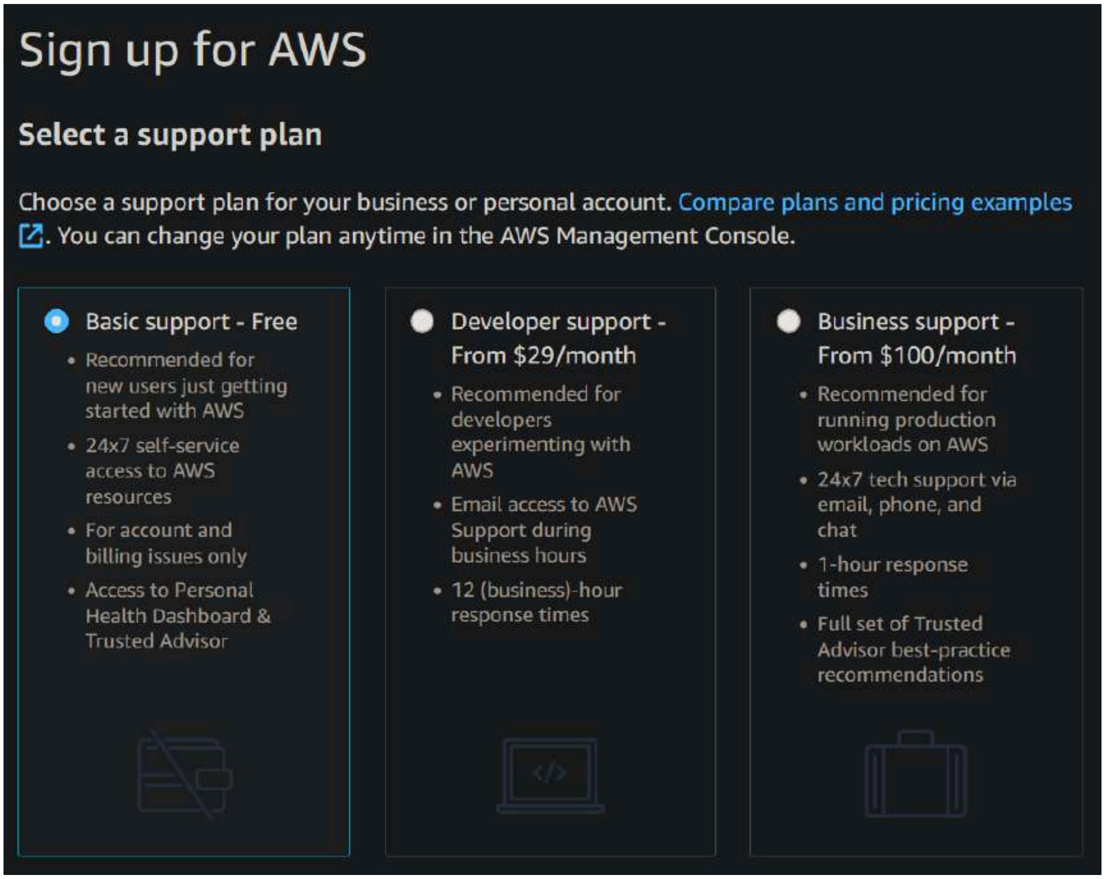
Region: Asia Pacific (Tokyo)

The data couldn't be retrieved. Try again later.

Learn the fundamentals and find valuable information to

Unable to load

Unable to load



Sign up for AWS

Select a support plan

Choose a support plan for your business or personal account. Compare plans and pricing examples. You can change your plan anytime in the AWS Management Console.

● Basic support - Free

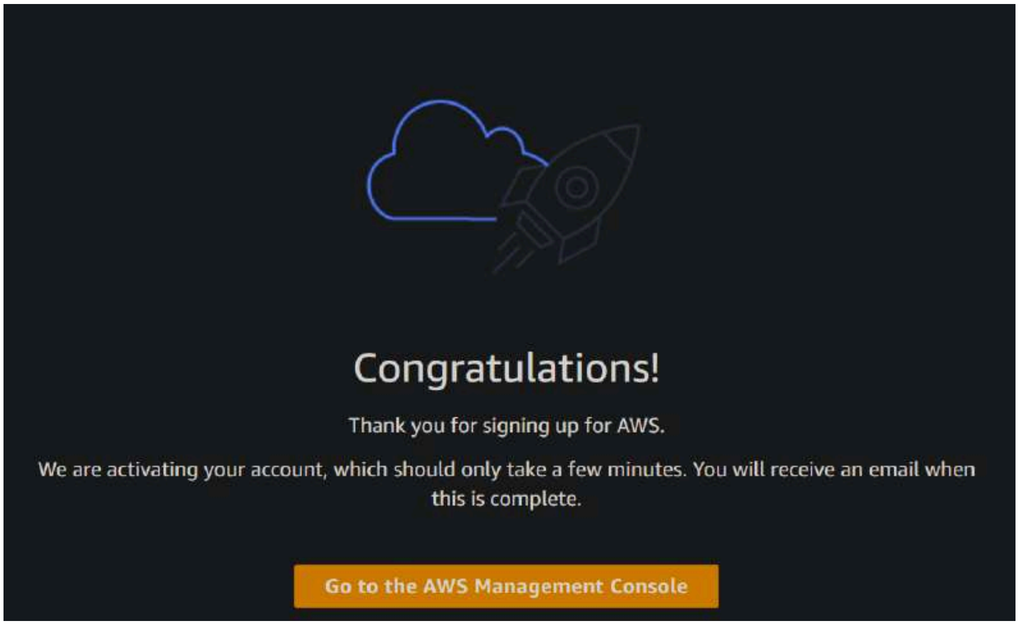
- Recommended for new users just getting started with AWS
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor

○ Developer support - From $29/month

- Recommended for developers experimenting with AWS
- Email access to AWS Support during business hours
- 12 (business)-hour response times

○ Business support - From $100/month

- Recommended for running production workloads on AWS
- 24x7 tech support via email, phone, and chat
- 1-hour response times
- Full set of Trusted Advisor best-practice recommendations

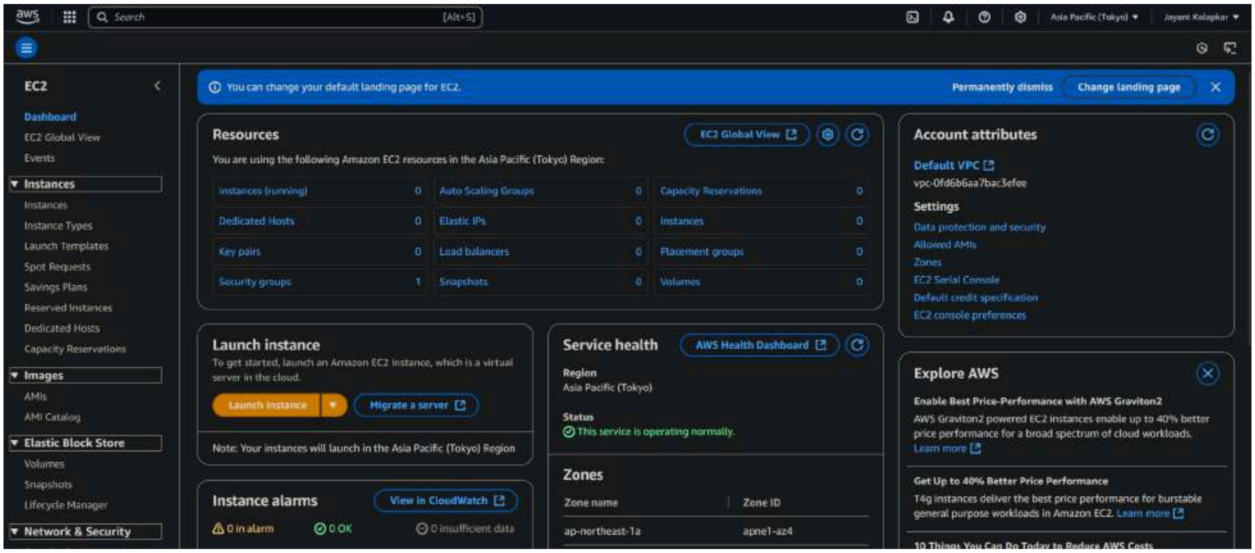


Congratulations!

Thank you for signing up for AWS.

We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

Launch a new instance using the given attributes



You can change your default landing page for EC2.

Resources

You are using the following Amazon EC2 resources in the Asia Pacific (Tokyo) Region:

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	0
Key pairs	0	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the Asia Pacific (Tokyo) Region

Instance alarms

⚠ 0 in alarm ✓ 0 OK ○ 0 insufficient data

Service health

Region
Asia Pacific (Tokyo)

Status
✓ This service is operating normally.

Zones

Zone name	Zone ID
ap-northeast-1a	apne1-az4

Account attributes

Default VPC
vpc-0fd6b6aa7bac3efee

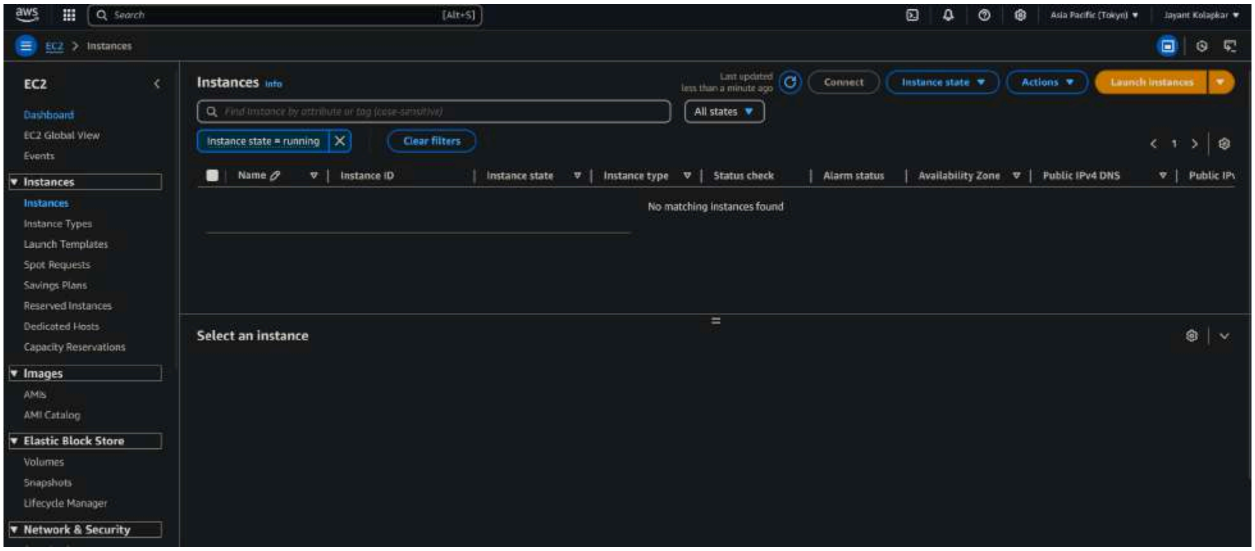
Settings
Data protection and security
Allowed AMIs
Zones
EC2 Serial Console
Default credit specification
EC2 console preferences

Explore AWS

Enable Best Price-Performance with AWS Graviton2
AWS Graviton2 powered EC2 instances enable up to 40% better price performance for a broad spectrum of cloud workloads. Learn more

Get Up to 40% Better Price Performance
T4g instances deliver the best price performance for burstable general purpose workloads in Amazon EC2. Learn more

10 Things You Can Do Today to Reduce AWS Costs



Instances

Last updated
less than a minute ago

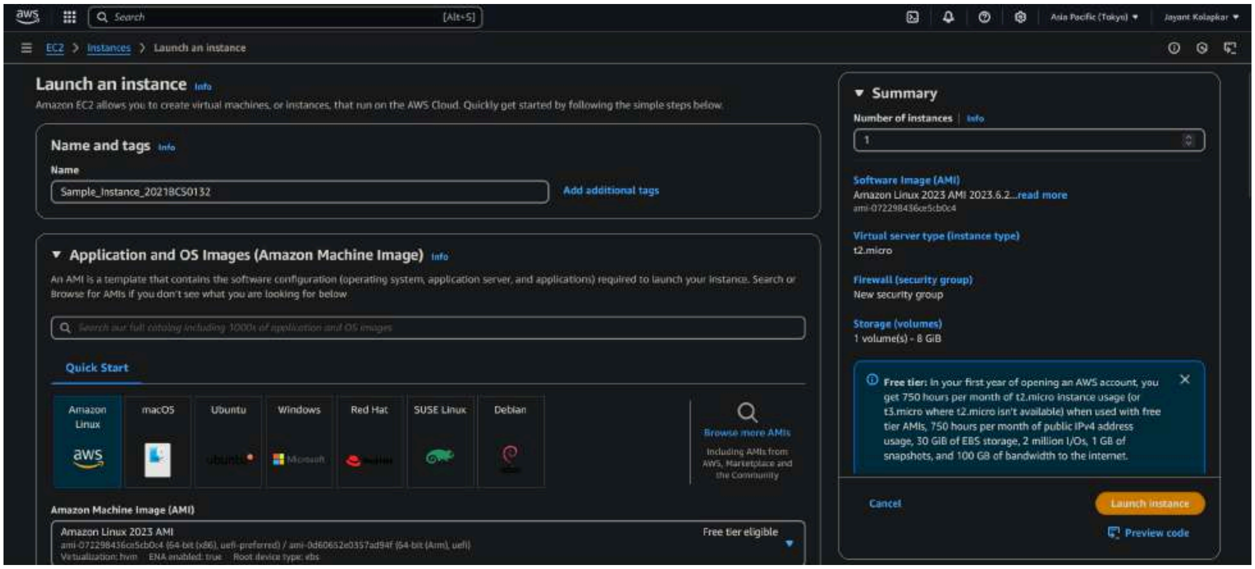
Find instance by attribute or tag (case-sensitive)

Instance state = running ✕

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP\

No matching instances found

Select an instance



EC2 > Instances > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: Sample_Instance_20218CS0132

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Quick Start: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, Debian

Browse more AMIs: Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-072298436ce5cb0c4 (64-bit (x86), uefi-preferred) / ami-0d60652e0357ad94f (64-bit (Arm), uefi)
Virtualization: hvm   ENA enabled: true   Root device type: ebs

Summary

Number of instances: 1

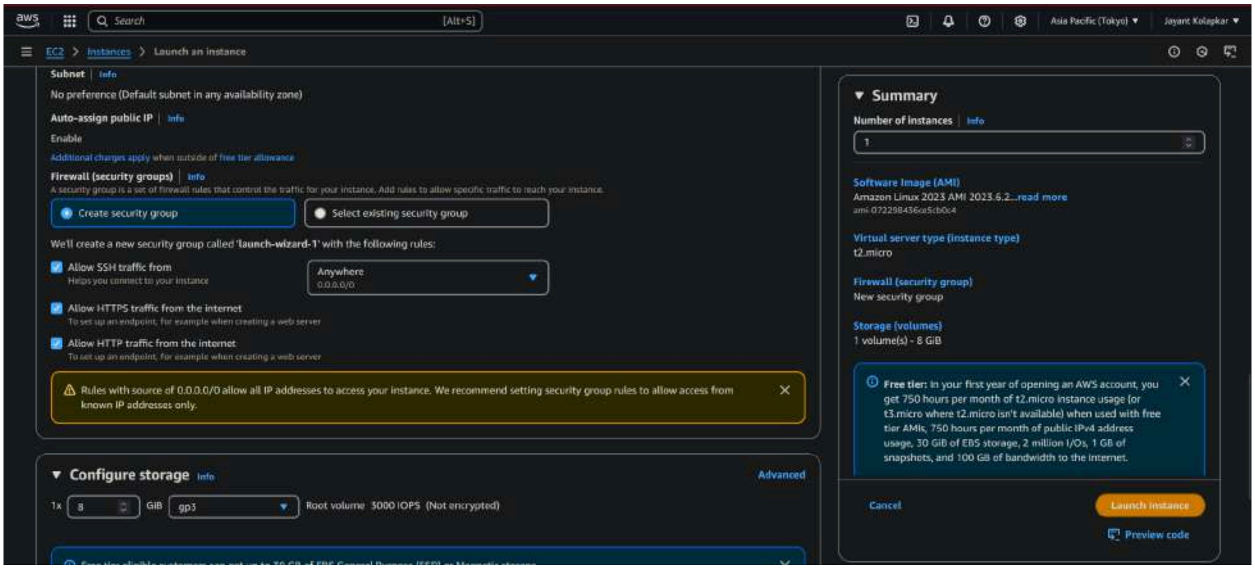
Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-072298436ce5cb0c4

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.



EC2 > Instances > Launch an instance

Subnet
No preference (Default subnet in any availability zone)

Auto-assign public IP
Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group   ○ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

☑ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Configure storage

1x 8 GiB gp3 Root volume 3000 IOPS (Not encrypted)

Summary

Number of instances: 1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...read more
ami-072298436ce5cb0c4

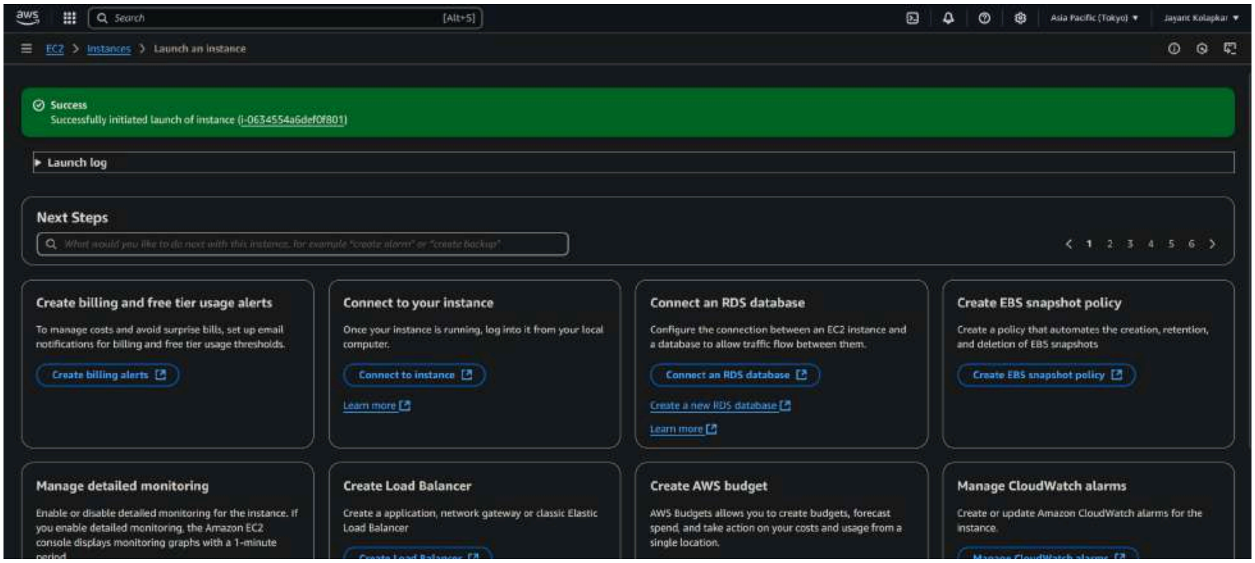
Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

Here we see that our instance has been created



Success
Successfully initiated launch of instance (i-0634554a6def0f801)

▸ Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create backup"

‹ 1 2 3 4 5 6 ›

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

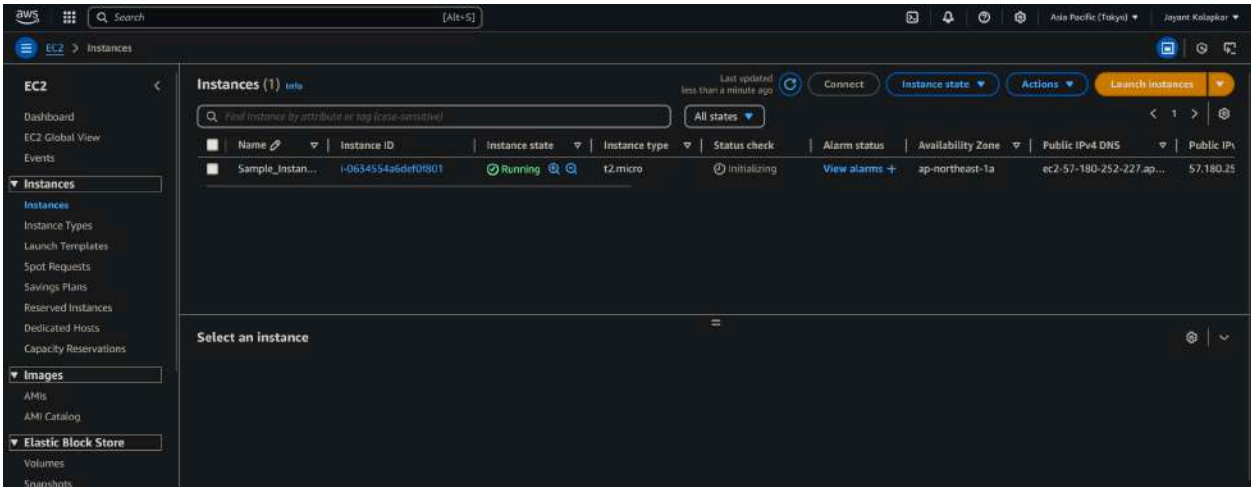
Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy

Manage detailed monitoring
Enable or disable detailed monitoring for the instance. If you enable detailed monitoring, the Amazon EC2 console displays monitoring graphs with a 1-minute period.

Create Load Balancer
Create a application, network gateway or classic Elastic Load Balancer
Create Load Balancer

Create AWS budget
AWS Budgets allows you to create budgets, forecast spend, and take action on your costs and usage from a single location.

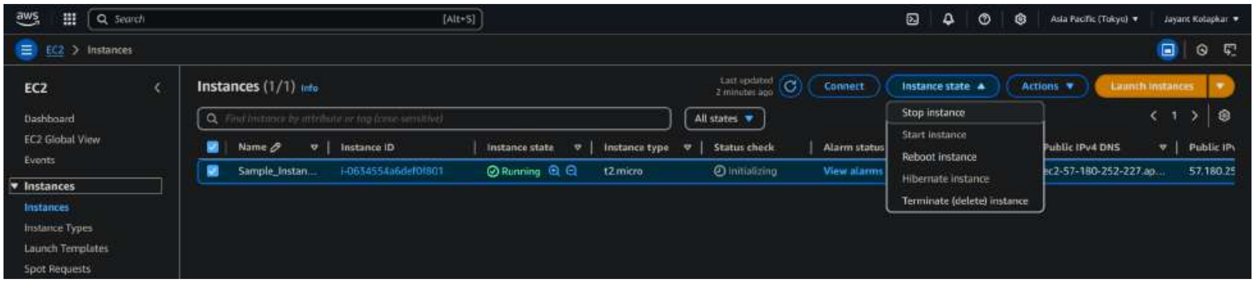
Manage CloudWatch alarms
Create or update Amazon CloudWatch alarms for the instance.
Manage CloudWatch alarms



Instances (1) Info

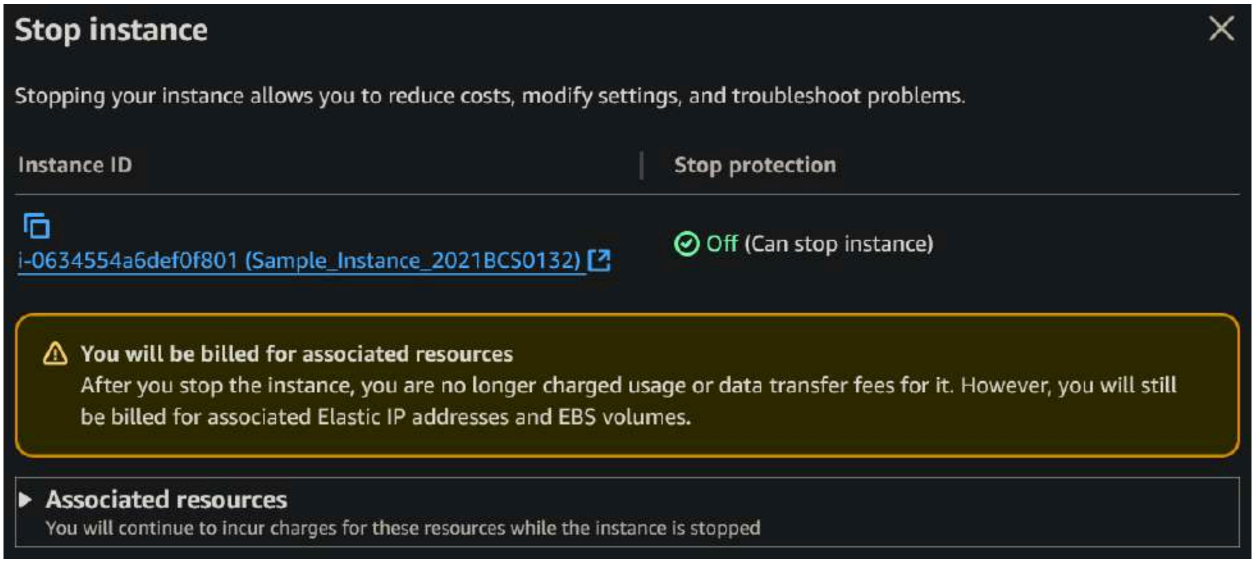
□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IP…
□	Sample_Instan...	i-0634554a6def0f801	Running	t2.micro	Initializing	View alarms +	ap-northeast-1a	ec2-57-180-252-227.ap...	57.180.25

Select an instance



Instances (1/1) Info

☑	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Public IPv4 DNS	Public IP…
☑	Sample_Instan...	i-0634554a6def0f801	Running	t2.micro	Initializing	View alarms	ec2-57-180-252-227.ap...	57.180.25



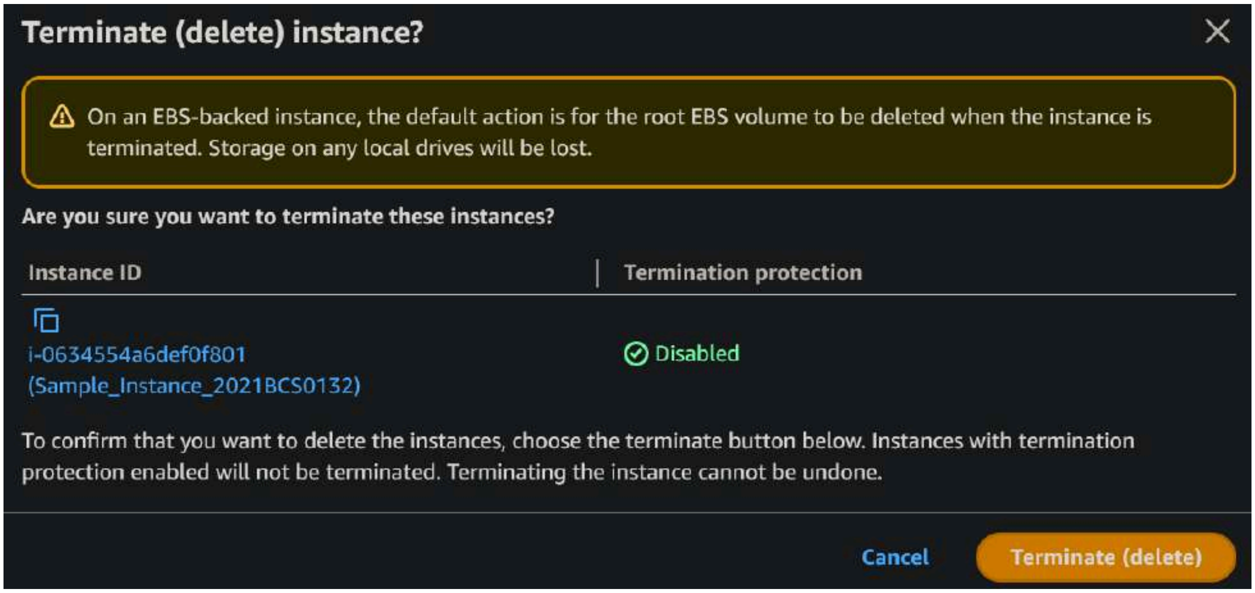
Stop instance

Stopping your instance allows you to reduce costs, modify settings, and troubleshoot problems.

Instance ID	Stop protection
i-0634554a6def0f801 (Sample_Instance_2021BCS0132)	✓ Off (Can stop instance)

⚠ **You will be billed for associated resources**
After you stop the instance, you are no longer charged usage or data transfer fees for it. However, you will still be billed for associated Elastic IP addresses and EBS volumes.

▶ **Associated resources**
You will continue to incur charges for these resources while the instance is stopped



Terminate (delete) instance?

⚠ On an EBS-backed instance, the default action is for the root EBS volume to be deleted when the instance is terminated. Storage on any local drives will be lost.

Are you sure you want to terminate these instances?

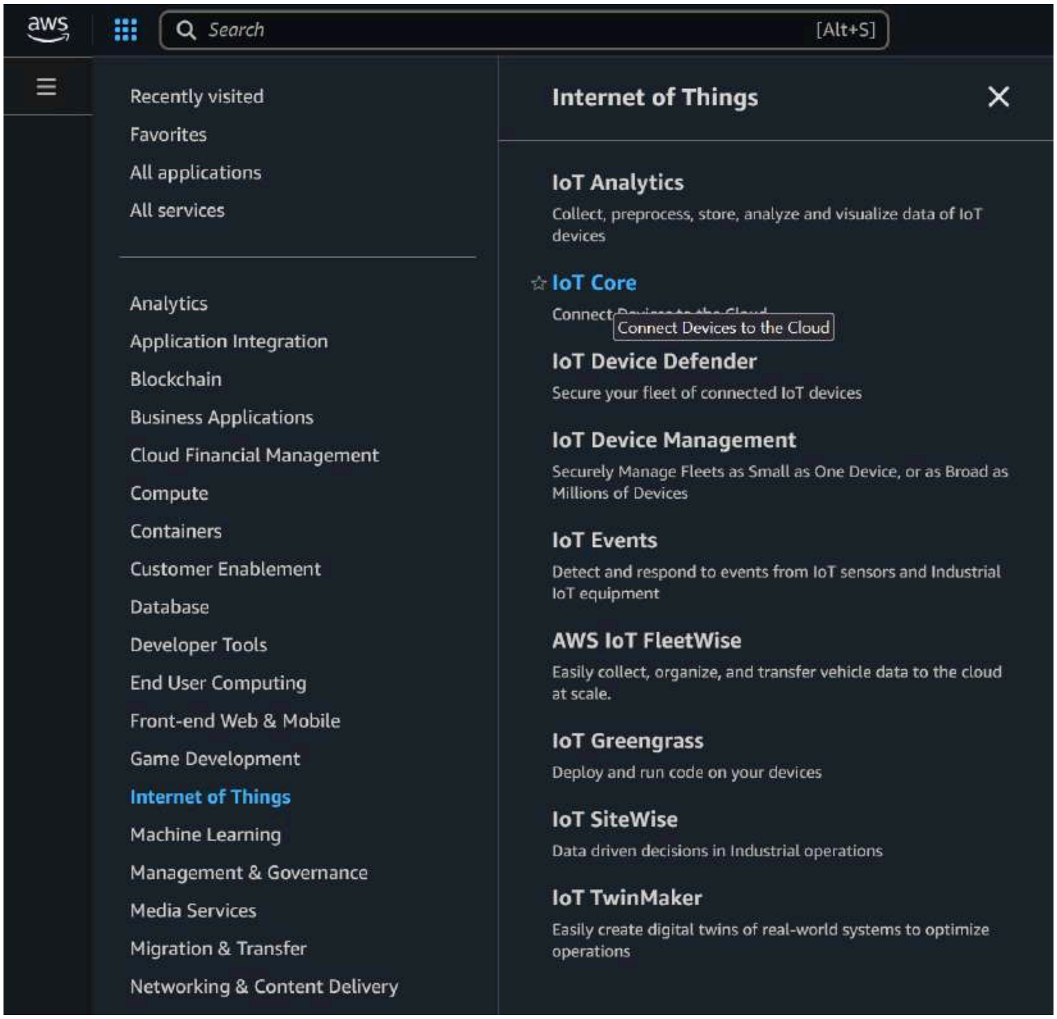
Instance ID	Termination protection
i-0634554a6def0f801 (Sample_Instance_2021BCS0132)	✓ Disabled

To confirm that you want to delete the instances, choose the terminate button below. Instances with termination protection enabled will not be terminated. Terminating the instance cannot be undone.

The instance is now terminated and will not be billed.

Task 2

Go to IoT core and create a new ‘Thing’



aws

Search [Alt+S]

Recently visited
Favorites
All applications
All services

Analytics
Application Integration
Blockchain
Business Applications
Cloud Financial Management
Compute
Containers
Customer Enablement
Database
Developer Tools
End User Computing
Front-end Web & Mobile
Game Development
Internet of Things
Machine Learning
Management & Governance
Media Services
Migration & Transfer
Networking & Content Delivery

Internet of Things

IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices

☆ **IoT Core**
Connect Devices to the Cloud

IoT Device Defender
Secure your fleet of connected IoT devices

IoT Device Management
Securely Manage Fleets as Small as One Device, or as Broad as Millions of Devices

IoT Events
Detect and respond to events from IoT sensors and Industrial IoT equipment

AWS IoT FleetWise
Easily collect, organize, and transfer vehicle data to the cloud at scale.

IoT Greengrass
Deploy and run code on your devices

IoT SiteWise
Data driven decisions in Industrial operations

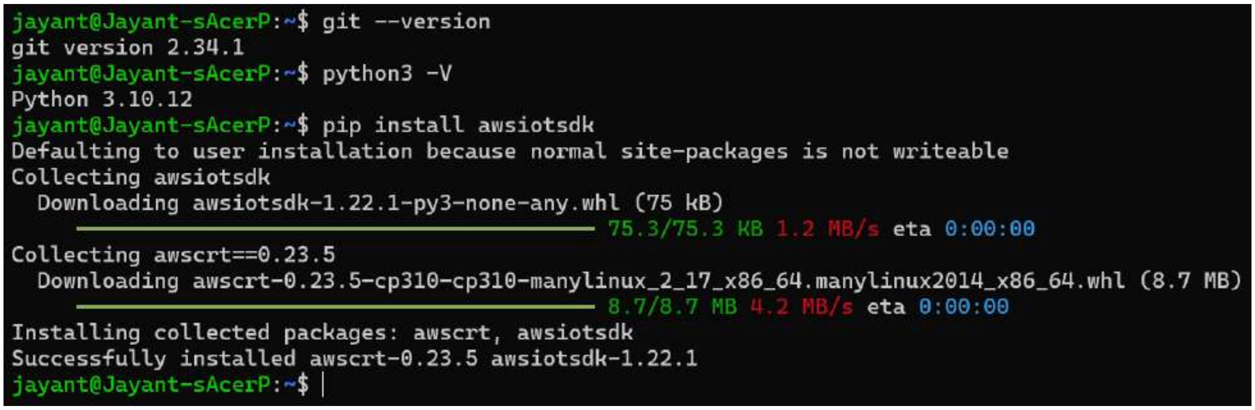
IoT TwinMaker
Easily create digital twins of real-world systems to optimize operations

1. Set Up Your Laptop

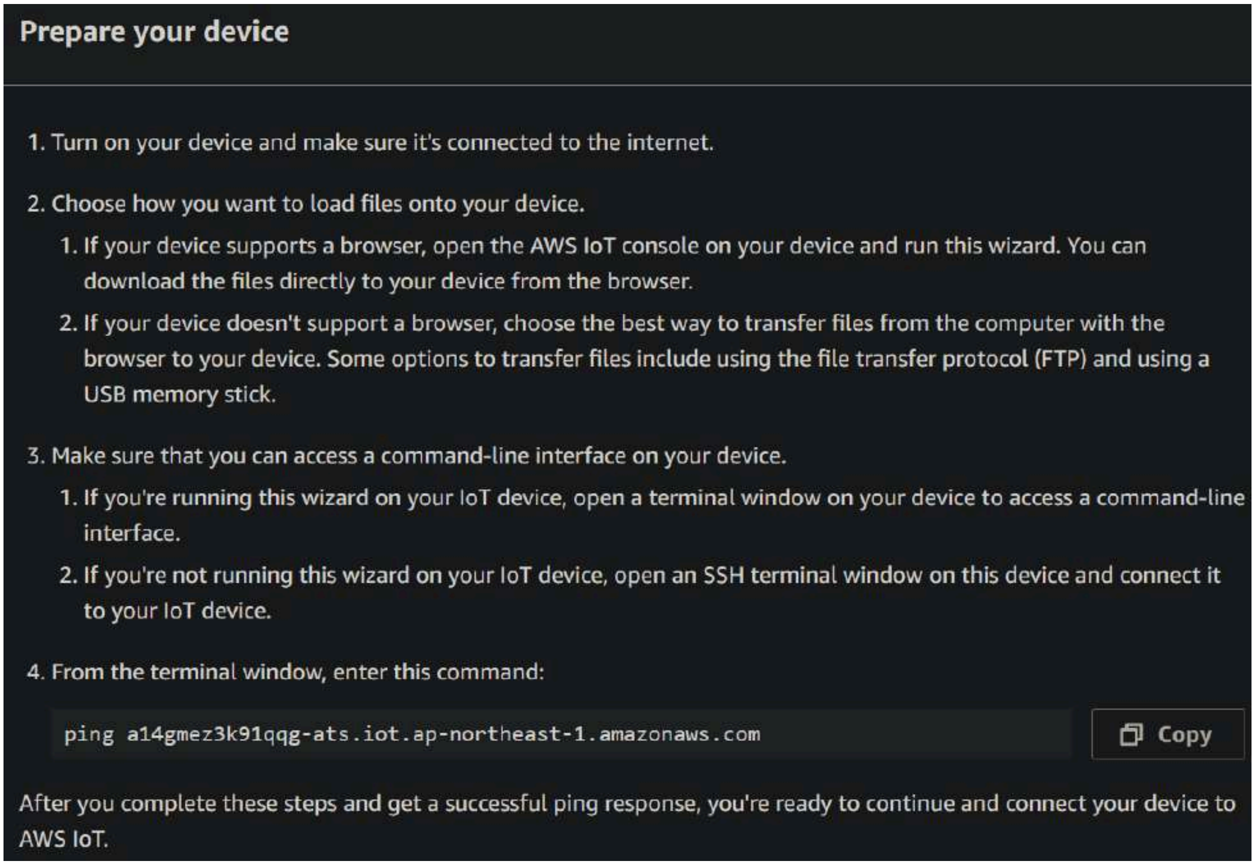
- **Install Git**: Verify if Git is installed by running `git --version` in your command line. If not installed, download and install it from the Git website.
- **Install Python**: Check for Python installation with `python -V`. AWS IoT Core requires Python version 3.7 or later. If not installed, download and install the latest version from the Python website.

2. Install the AWS IoT Device SDK for Python

The AWS IoT Device SDK for Python enables your laptop to communicate with AWS IoT Core using MQTT.



```
jayant@Jayant-sAcerP:~$ git --version
git version 2.34.1
jayant@Jayant-sAcerP:~$ python3 -V
Python 3.10.12
jayant@Jayant-sAcerP:~$ pip install awsiotsdk
Defaulting to user installation because normal site-packages is not writeable
Collecting awsiotsdk
  Downloading awsiotsdk-1.22.1-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.3/75.3 KB 1.2 MB/s eta 0:00:00
Collecting awscrt==0.23.5
  Downloading awscrt-0.23.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.7/8.7 MB 4.2 MB/s eta 0:00:00
Installing collected packages: awscrt, awsiotsdk
Successfully installed awscrt-0.23.5 awsiotsdk-1.22.1
jayant@Jayant-sAcerP:~$ 
```

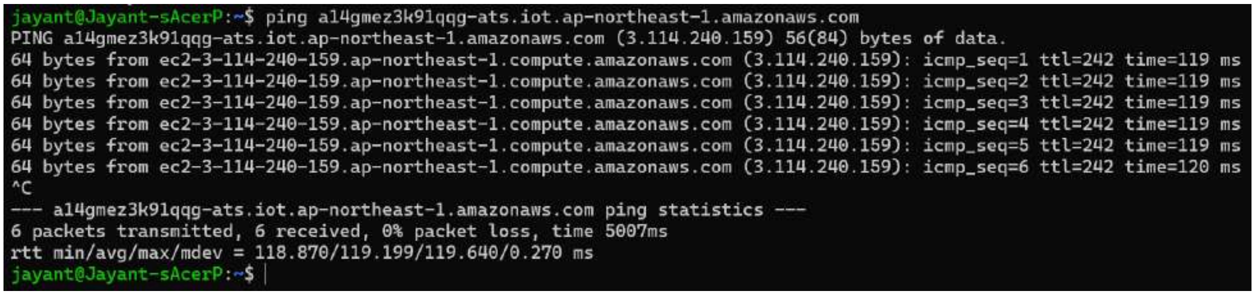


Prepare your device

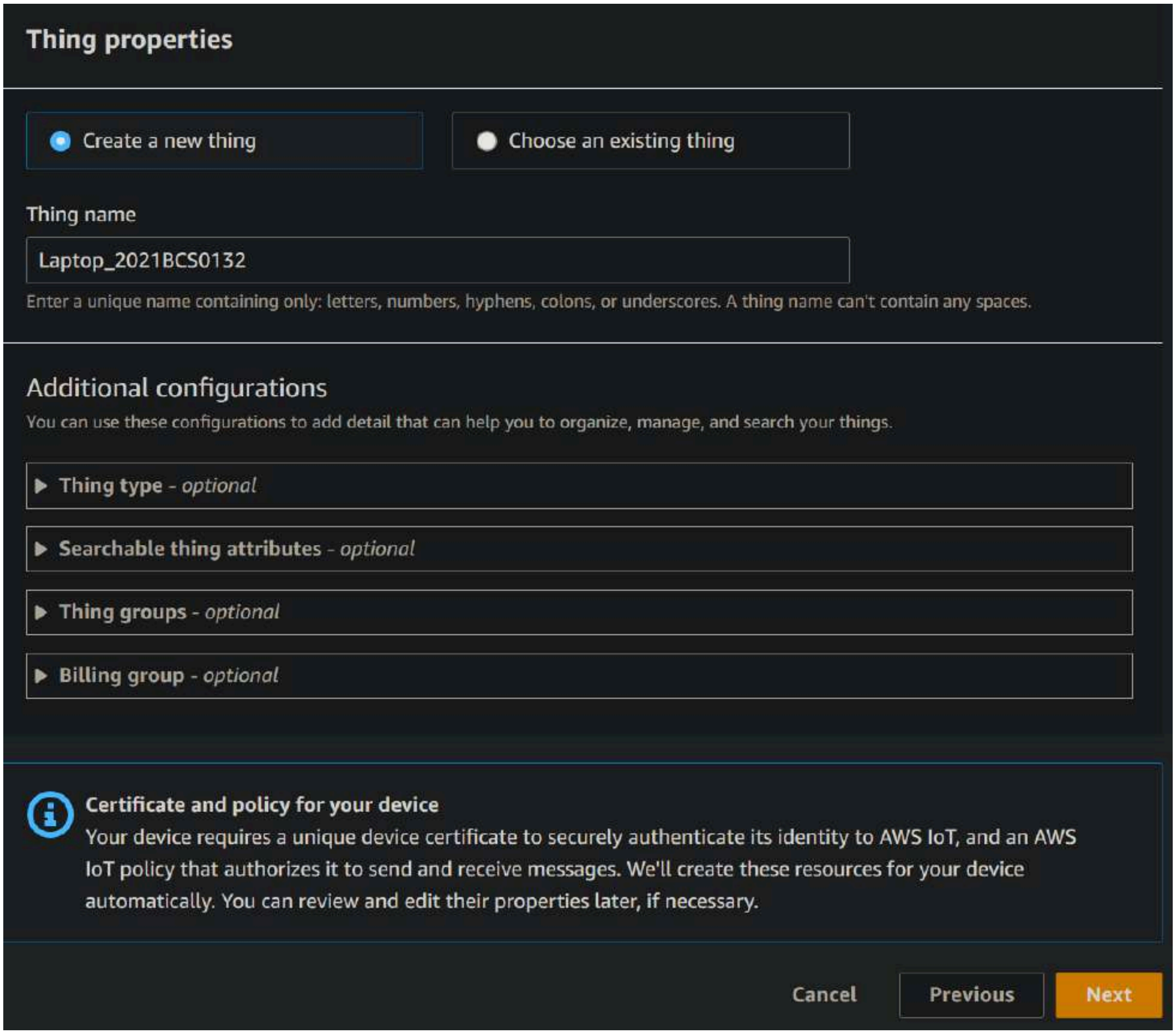
1. Turn on your device and make sure it's connected to the internet.
2. Choose how you want to load files onto your device.
    1. If your device supports a browser, open the AWS IoT console on your device and run this wizard. You can download the files directly to your device from the browser.
    2. If your device doesn't support a browser, choose the best way to transfer files from the computer with the browser to your device. Some options to transfer files include using the file transfer protocol (FTP) and using a USB memory stick.
3. Make sure that you can access a command-line interface on your device.
    1. If you're running this wizard on your IoT device, open a terminal window on your device to access a command-line interface.
    2. If you're not running this wizard on your IoT device, open an SSH terminal window on this device and connect it to your IoT device.
4. From the terminal window, enter this command:

```
ping a14gmez3k91qqg-ats.iot.ap-northeast-1.amazonaws.com
```

After you complete these steps and get a successful ping response, you're ready to continue and connect your device to AWS IoT.



```
jayant@Jayant-sAcerP:~$ ping a14gmez3k91qqg-ats.iot.ap-northeast-1.amazonaws.com
PING a14gmez3k91qqg-ats.iot.ap-northeast-1.amazonaws.com (3.114.240.159) 56(84) bytes of data.
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=1 ttl=242 time=119 ms
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=2 ttl=242 time=119 ms
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=3 ttl=242 time=119 ms
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=4 ttl=242 time=119 ms
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=5 ttl=242 time=119 ms
64 bytes from ec2-3-114-240-159.ap-northeast-1.compute.amazonaws.com (3.114.240.159): icmp_seq=6 ttl=242 time=120 ms
^C
--- a14gmez3k91qqg-ats.iot.ap-northeast-1.amazonaws.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 118.870/119.199/119.640/0.270 ms
jayant@Jayant-sAcerP:~$ 
```



Thing properties

● Create a new thing   ○ Choose an existing thing

Thing name

Laptop_2021BCS0132

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

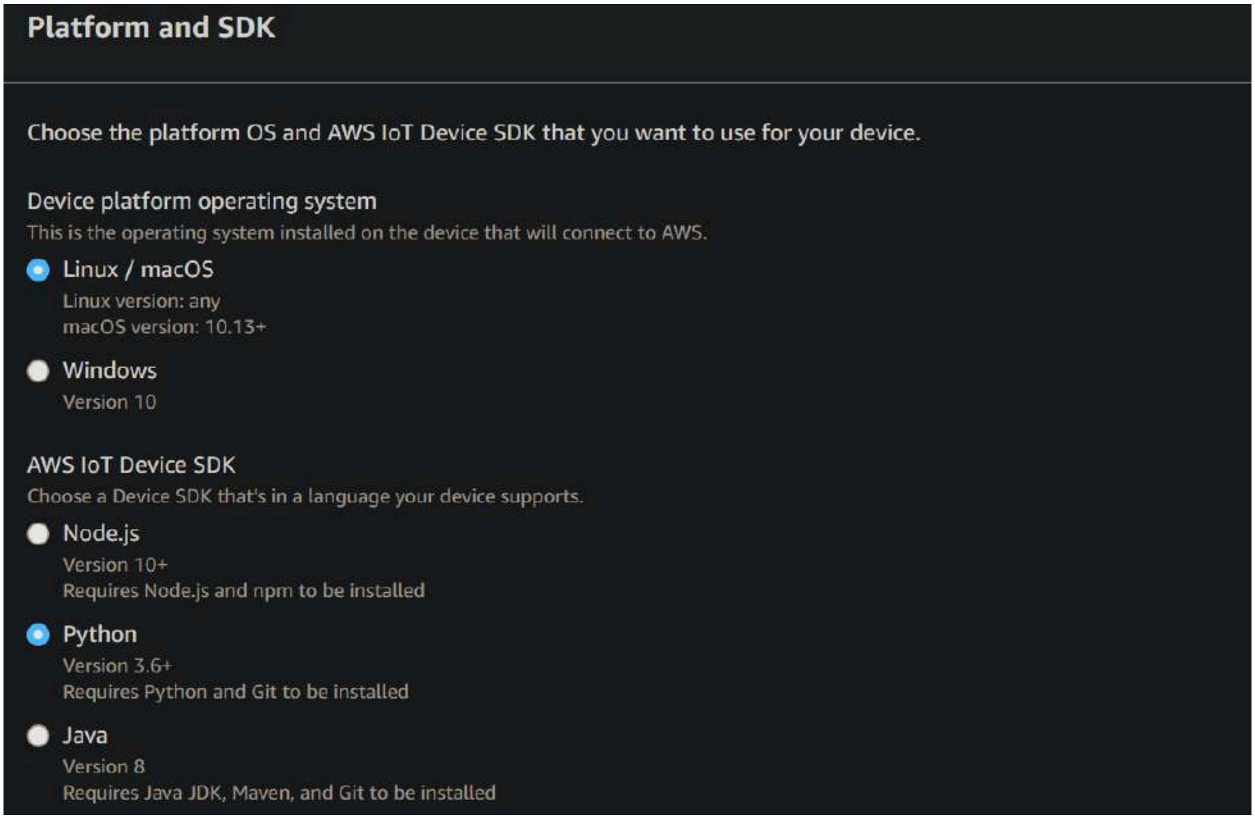
Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

▶ Thing type - optional
▶ Searchable thing attributes - optional
▶ Thing groups - optional
▶ Billing group - optional

Certificate and policy for your device

Your device requires a unique device certificate to securely authenticate its identity to AWS IoT, and an AWS IoT policy that authorizes it to send and receive messages. We'll create these resources for your device automatically. You can review and edit their properties later, if necessary.



Platform and SDK

Choose the platform OS and AWS IoT Device SDK that you want to use for your device.

Device platform operating system
This is the operating system installed on the device that will connect to AWS.

◉ Linux / macOS
Linux version: any
macOS version: 10.13+

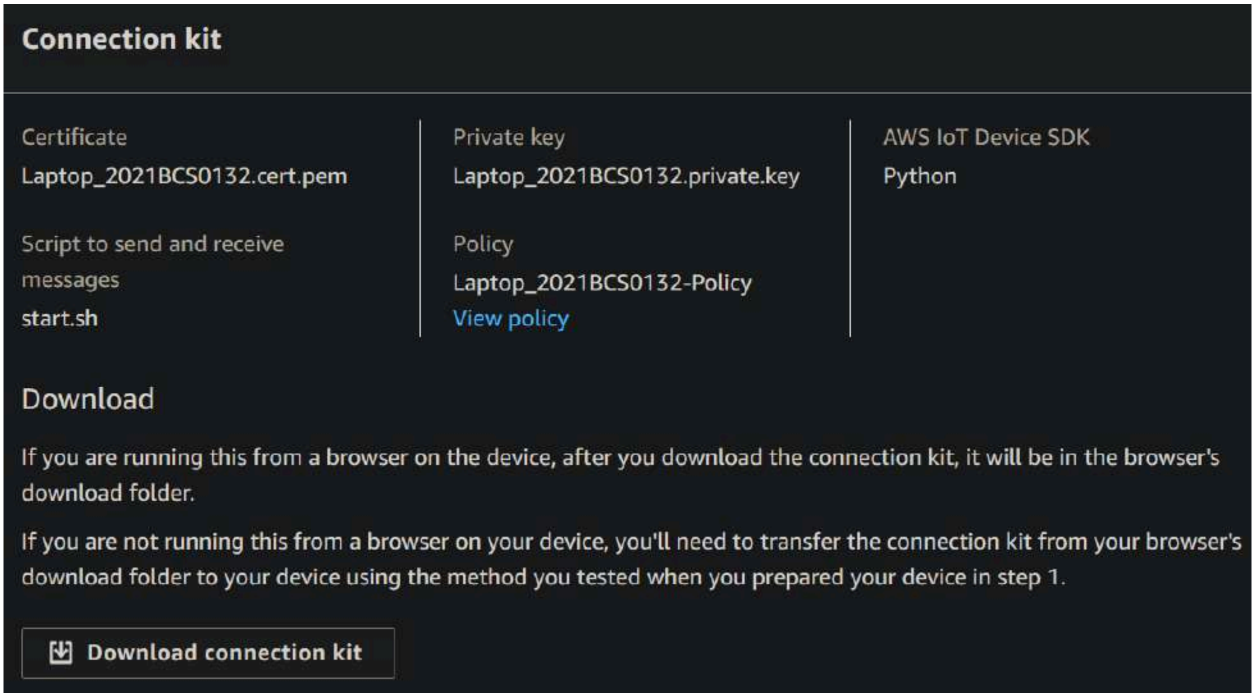
○ Windows
Version 10

AWS IoT Device SDK
Choose a Device SDK that's in a language your device supports.

○ Node.js
Version 10+
Requires Node.js and npm to be installed

◉ Python
Version 3.6+
Requires Python and Git to be installed

○ Java
Version 8
Requires Java JDK, Maven, and Git to be installed



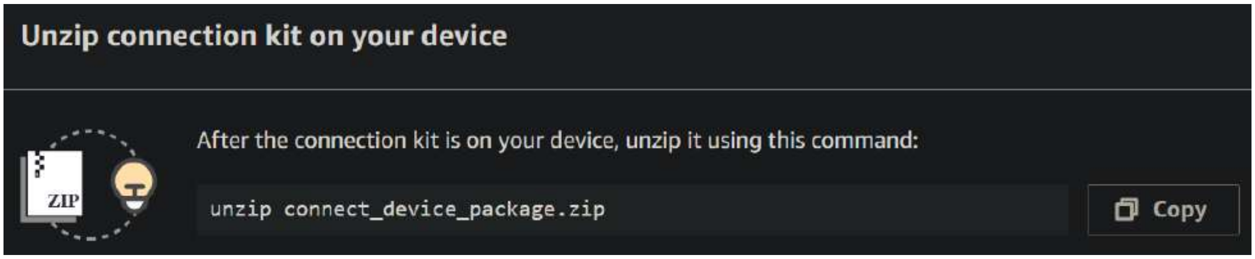
Connection kit

Certificate	Private key	AWS IoT Device SDK
Laptop_2021BCS0132.cert.pem	Laptop_2021BCS0132.private.key	Python
Script to send and receive messages	Policy	
start.sh	Laptop_2021BCS0132-Policy View policy	

Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

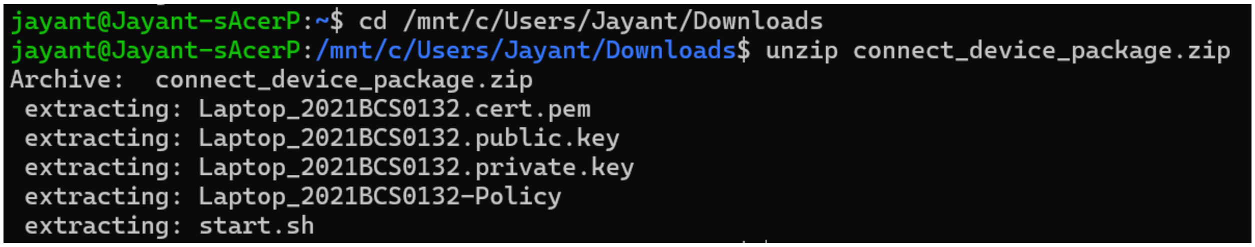
If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.



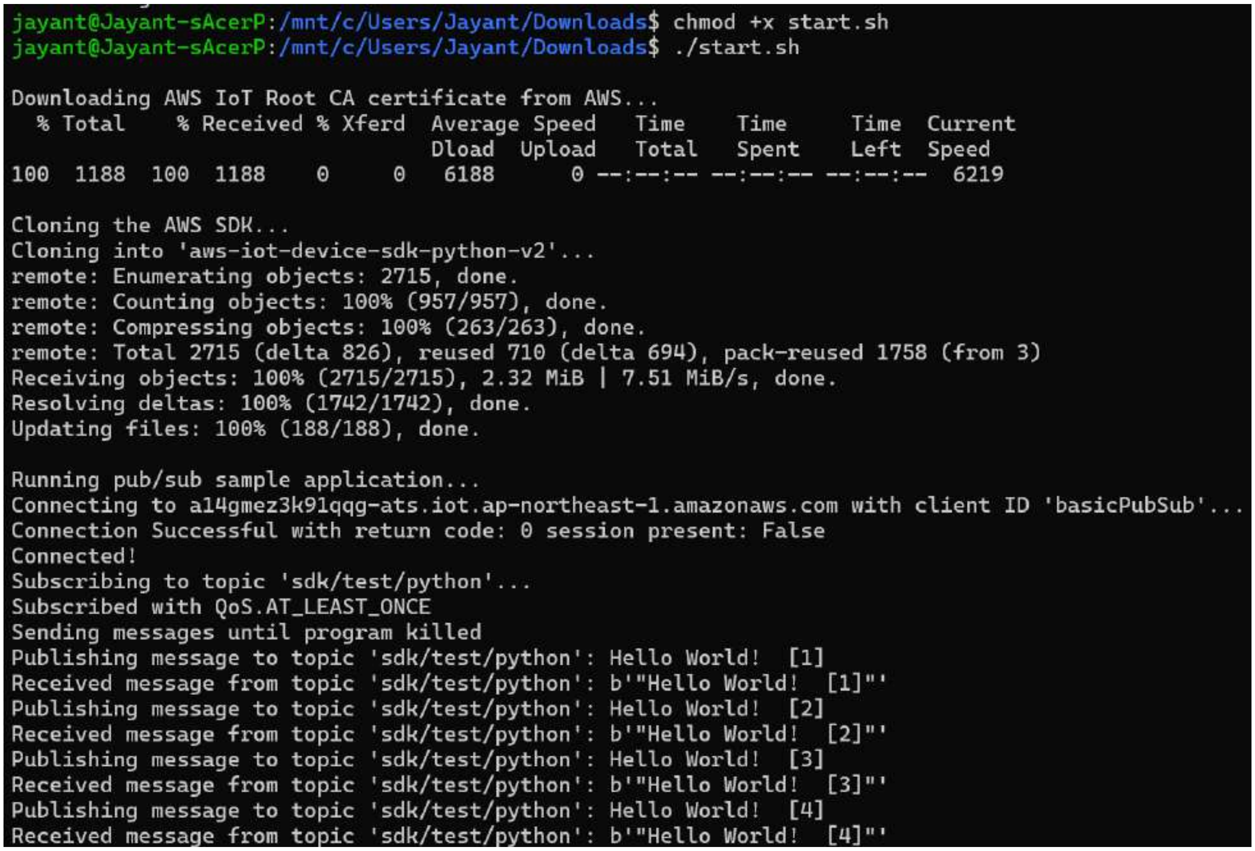
Unzip connection kit on your device

After the connection kit is on your device, unzip it using this command:

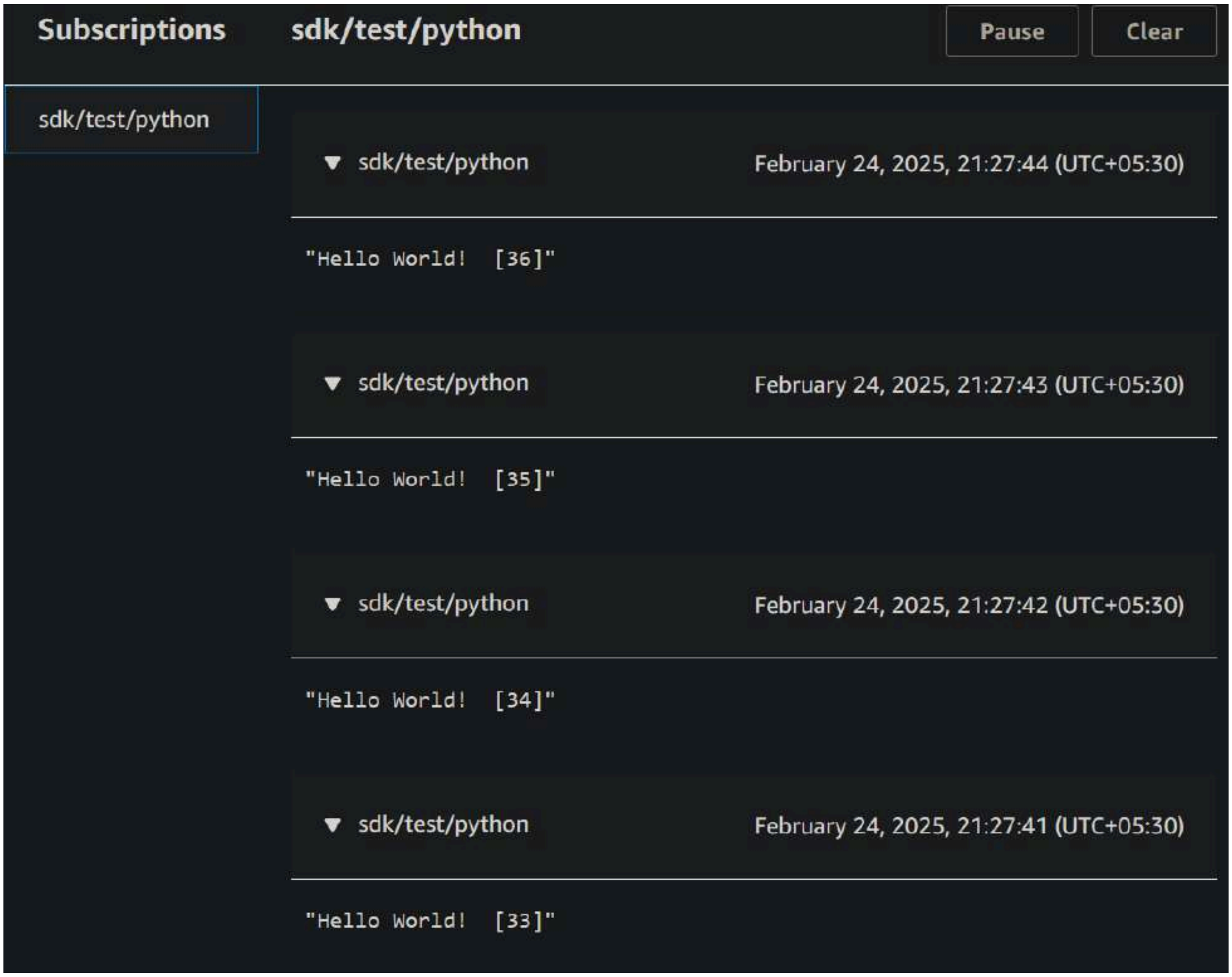
```
unzip connect_device_package.zip
```



```
jayant@Jayant-sAcerP:~$ cd /mnt/c/Users/Jayant/Downloads
jayant@Jayant-sAcerP:/mnt/c/Users/Jayant/Downloads$ unzip connect_device_package.zip
Archive:  connect_device_package.zip
 extracting: Laptop_2021BCS0132.cert.pem
 extracting: Laptop_2021BCS0132.public.key
 extracting: Laptop_2021BCS0132.private.key
 extracting: Laptop_2021BCS0132-Policy
 extracting: start.sh
```



```
jayant@Jayant-sAcerP:/mnt/c/Users/Jayant/Downloads$ chmod +x start.sh
jayant@Jayant-sAcerP:/mnt/c/Users/Jayant/Downloads$ ./start.sh

Downloading AWS IoT Root CA certificate from AWS...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1188  100  1188    0     0   6188      0 --:--:-- --:--:-- --:--:--  6219

Cloning the AWS SDK...
Cloning into 'aws-iot-device-sdk-python-v2'...
remote: Enumerating objects: 2715, done.
remote: Counting objects: 100% (957/957), done.
remote: Compressing objects: 100% (263/263), done.
remote: Total 2715 (delta 826), reused 710 (delta 694), pack-reused 1758 (from 3)
Receiving objects: 100% (2715/2715), 2.32 MiB | 7.51 MiB/s, done.
Resolving deltas: 100% (1742/1742), done.
Updating files: 100% (188/188), done.

Running pub/sub sample application...
Connecting to a14gmez3k91qqg-ats.iot.ap-northeast-1.amazonaws.com with client ID 'basicPubSub'...
Connection Successful with return code: 0 session present: False
Connected!
Subscribing to topic 'sdk/test/python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/python': Hello World!  [1]
Received message from topic 'sdk/test/python': b'"Hello World!  [1]"'
Publishing message to topic 'sdk/test/python': Hello World!  [2]
Received message from topic 'sdk/test/python': b'"Hello World!  [2]"'
Publishing message to topic 'sdk/test/python': Hello World!  [3]
Received message from topic 'sdk/test/python': b'"Hello World!  [3]"'
Publishing message to topic 'sdk/test/python': Hello World!  [4]
Received message from topic 'sdk/test/python': b'"Hello World!  [4]"'
```



▼ sdk/test/python February 24, 2025, 21:27:44 (UTC+05:30)

```
"Hello World!  [36]"
```

▼ sdk/test/python February 24, 2025, 21:27:43 (UTC+05:30)

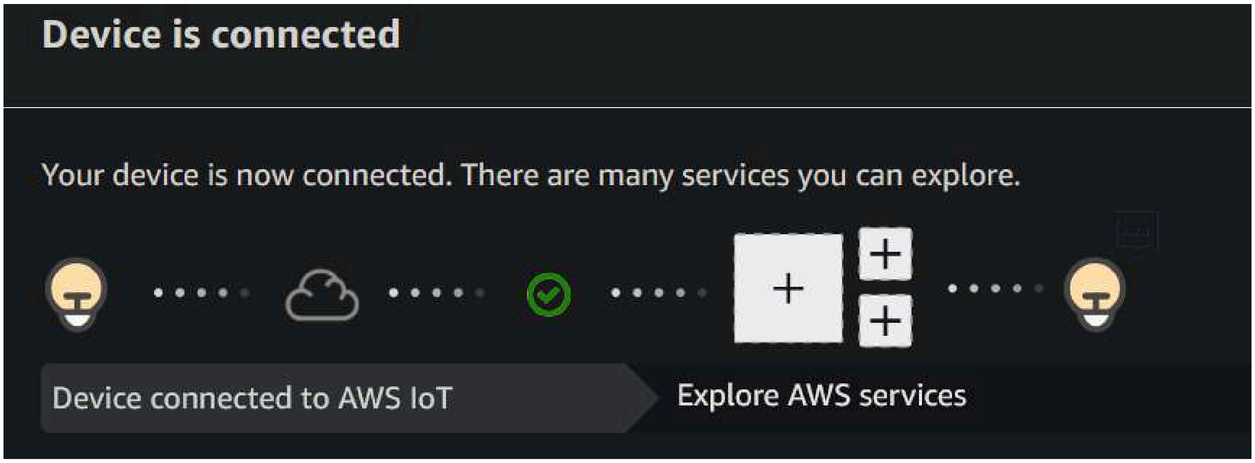
```
"Hello World!  [35]"
```

▼ sdk/test/python February 24, 2025, 21:27:42 (UTC+05:30)

```
"Hello World!  [34]"
```

▼ sdk/test/python February 24, 2025, 21:27:41 (UTC+05:30)

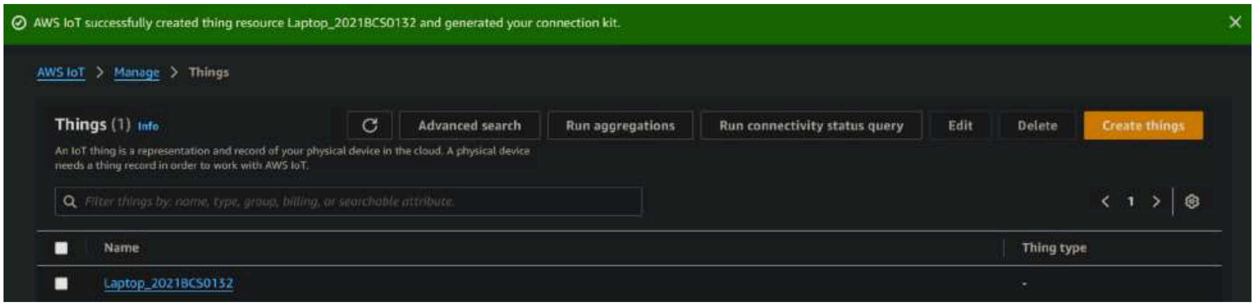
```
"Hello World!  [33]"
```



Device is connected

Your device is now connected. There are many services you can explore.

As we see, the laptop is connected as a ‘Thing’ under AWS IoT Core



AWS IoT successfully created thing resource Laptop_2021BCS0132 and generated your connection kit.

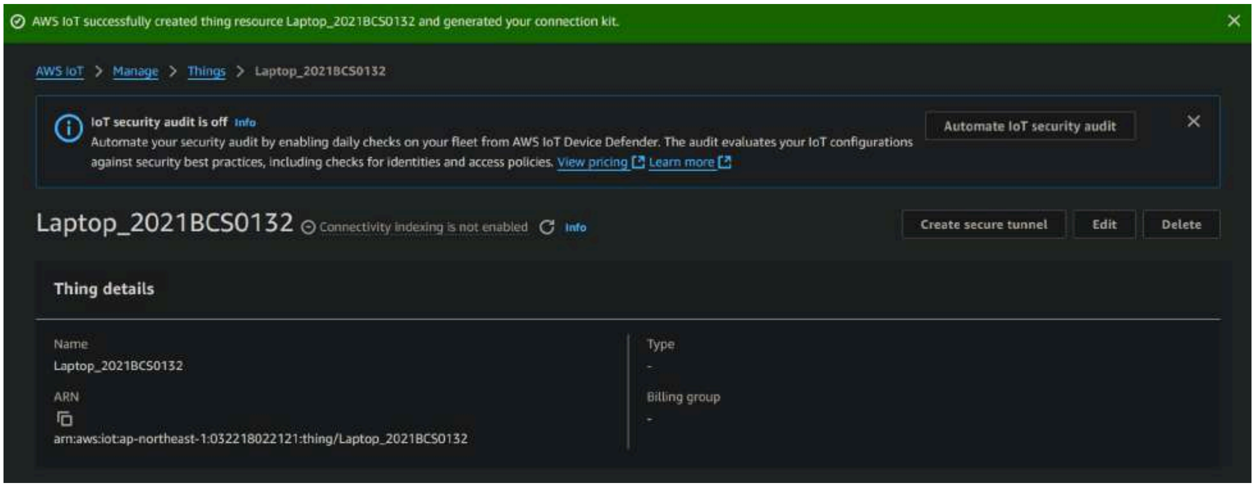
AWS IoT > Manage > Things

Things (1) Info

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

Name	Thing type
Laptop_2021BCS0132	-

Advanced search
Run aggregations
Run connectivity status query
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
1



AWS IoT successfully created thing resource Laptop_2021BCS0132 and generated your connection kit.

AWS IoT > Manage > Things > Laptop_2021BCS0132

IoT security audit is off Info

Automate your security audit by enabling daily checks on your fleet from AWS IoT Device Defender. The audit evaluates your IoT configurations against security best practices, including checks for identities and access policies. View pricing Learn more

Laptop_2021BCS0132 Connectivity indexing is not enabled Info

Thing details

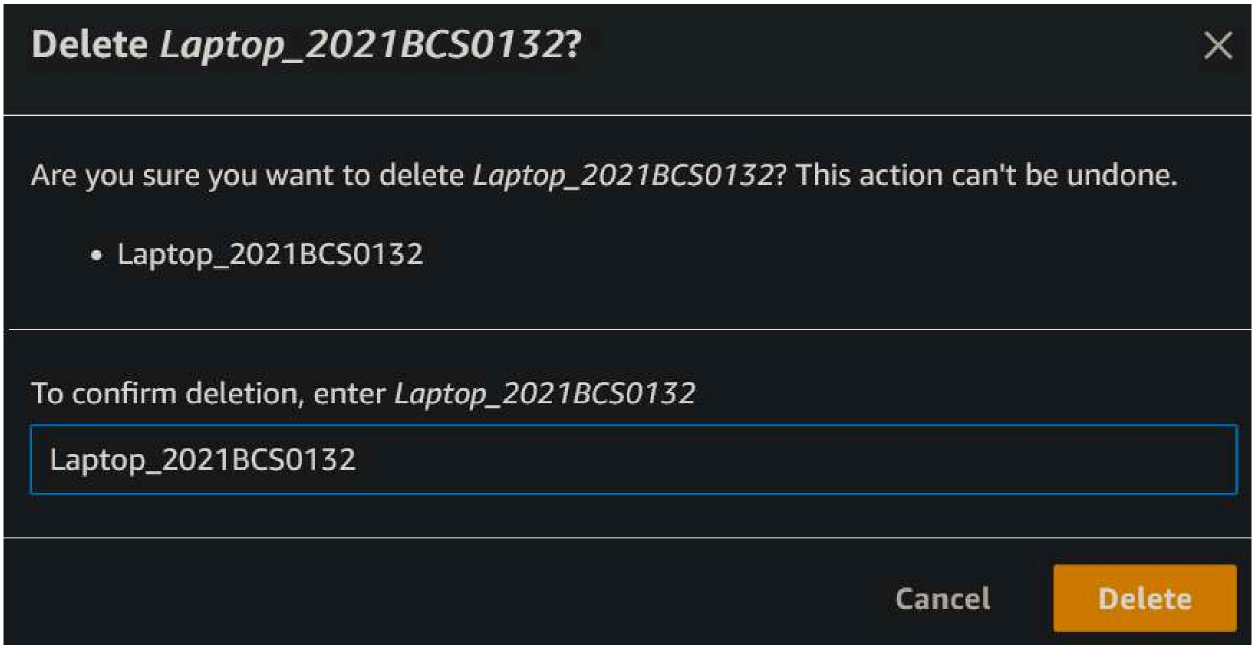
Name: Laptop_2021BCS0132

ARN: arn:aws:iot:ap-northeast-1:032218022121:thing/Laptop_2021BCS0132

Type: -

Billing group: -

To delete the thing, follow these steps



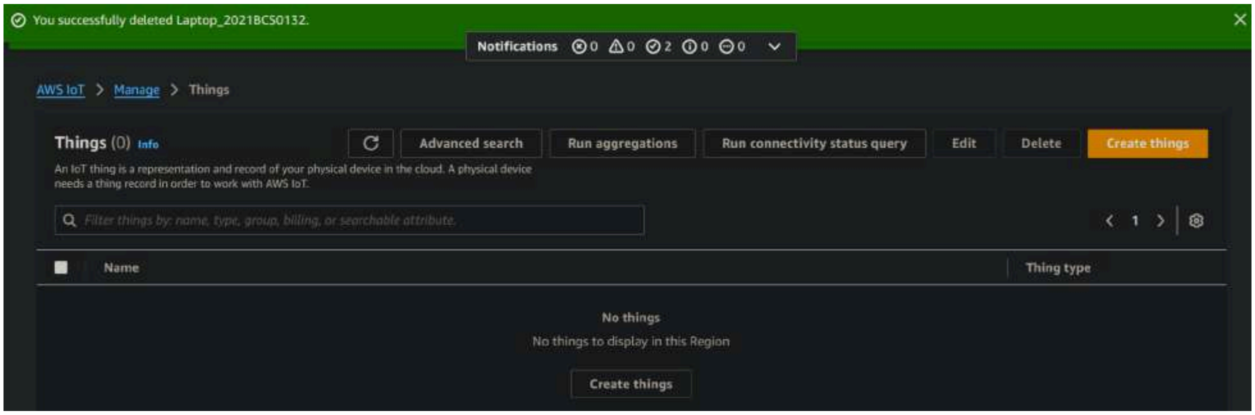
Delete *Laptop_2021BCS0132*?

Are you sure you want to delete *Laptop_2021BCS0132*? This action can't be undone.

- Laptop_2021BCS0132

To confirm deletion, enter *Laptop_2021BCS0132*

Laptop_2021BCS0132



You successfully deleted Laptop_2021BCS0132.

Notifications ⊗ 0 ⚠ 0 ✓ 2 ⓘ 0 ⊖ 0
AWS IoT > Manage > Things
Things (0) Info
Advanced search
Run aggregations
Run connectivity status query
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
1
Name
Thing type
Create things

An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

No things

No things to display in this Region

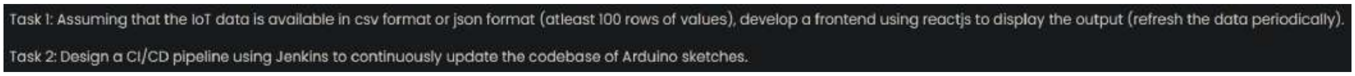
The Thing has been deleted and no longer exists.

Lab 8

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



Task 1: Assuming that the IoT data is available in csv format or json format (atleast 100 rows of values), develop a frontend using reactjs to display the output (refresh the data periodically).

Task 2: Design a CI/CD pipeline using Jenkins to continuously update the codebase of Arduino sketches.

Task 1

Algorithm and Code

1. Create a node server which generates the IOT (temperature data) periodically and stores the data in a CSV file.

Code: server.js

```
const express = require("express");
const fs = require("fs");
const cors = require("cors");
const fastCsv = require("fast-csv");
const app = express();
const PORT = 5000;
const FILE_PATH = "iot_data.csv";
```

```
// Enable CORS
app.use(cors());

// Function to generate random IoT data
const generateData = () => {
    const timestamp = new Date().toISOString();
    const temperature = (Math.random() * 15 + 20).toFixed(2); // Random temp between 20-35°C
    return `${timestamp},${temperature}\n`;
};

// Function to write data to CSV file every 2 seconds
const appendData = () => {
    const header = "timestamp,temperature\n";
    if (!fs.existsSync(FILE_PATH)) {
        fs.writeFileSync(FILE_PATH, header);
    }
    setInterval(() => {
    fs.appendFileSync(FILE_PATH, generateData());
    }, 2000);
};

// API endpoint to get the last 100 rows
app.get("/data", (req, res) => {
    const rows = [];
    fs.createReadStream(FILE_PATH)
        .pipe(fastCsv.parse({ headers: true }))
        .on("data", (row) => rows.push(row))
        .on("end", () => {
            res.json(rows.slice(-100)); // Send last 100 rows
        });
});

// Start the server and begin data generation
app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
    appendData();
});
```

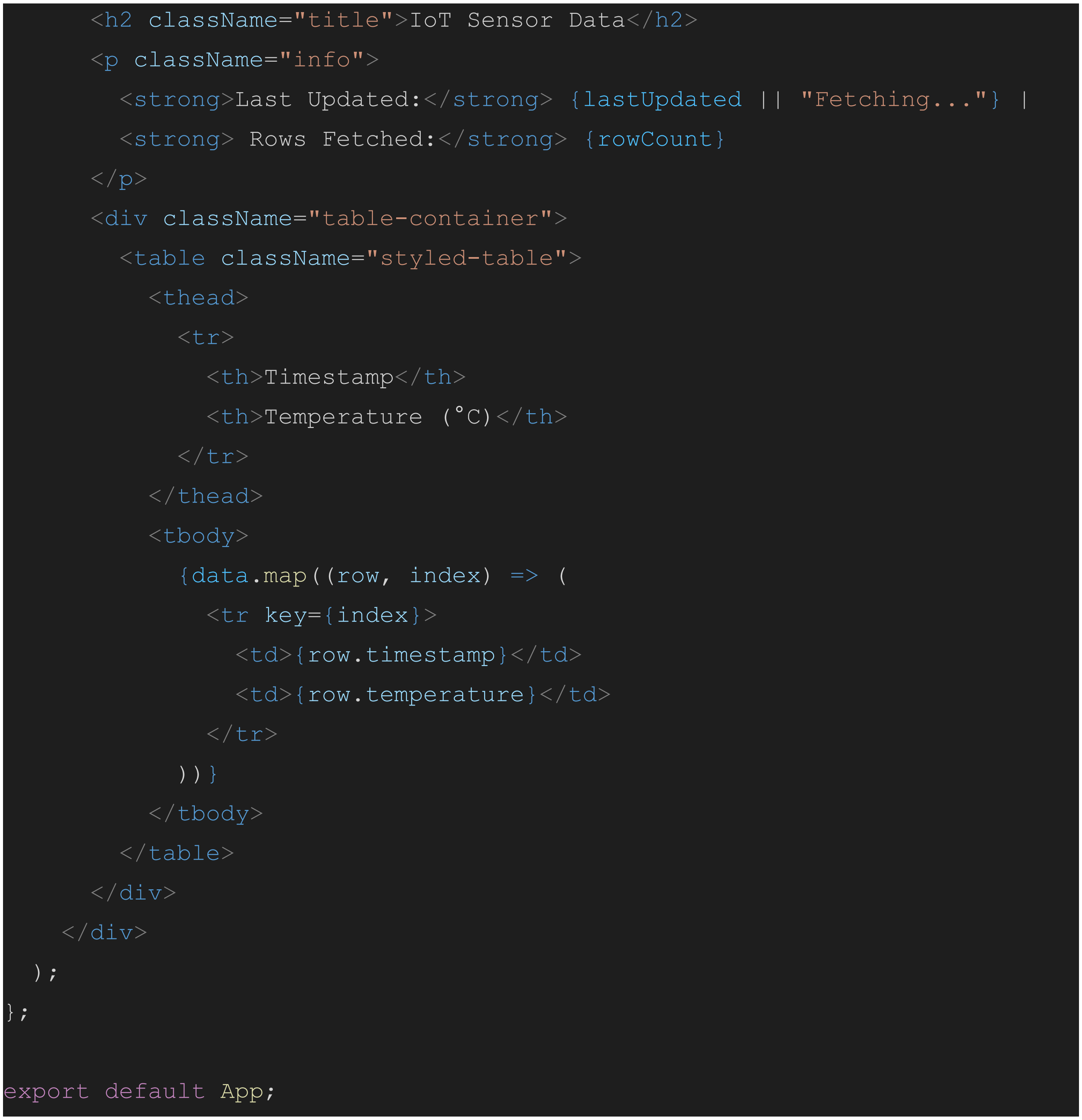
2. Start the server. You will observe that the data will get stored in the CSV file continuously.

3. Create React dashboard by creating the react app and installing the necessary dependencies.

4. Fetch the data from the CSV file on the server and fetch the data using the GET api by integrating the react frontend with node backend.

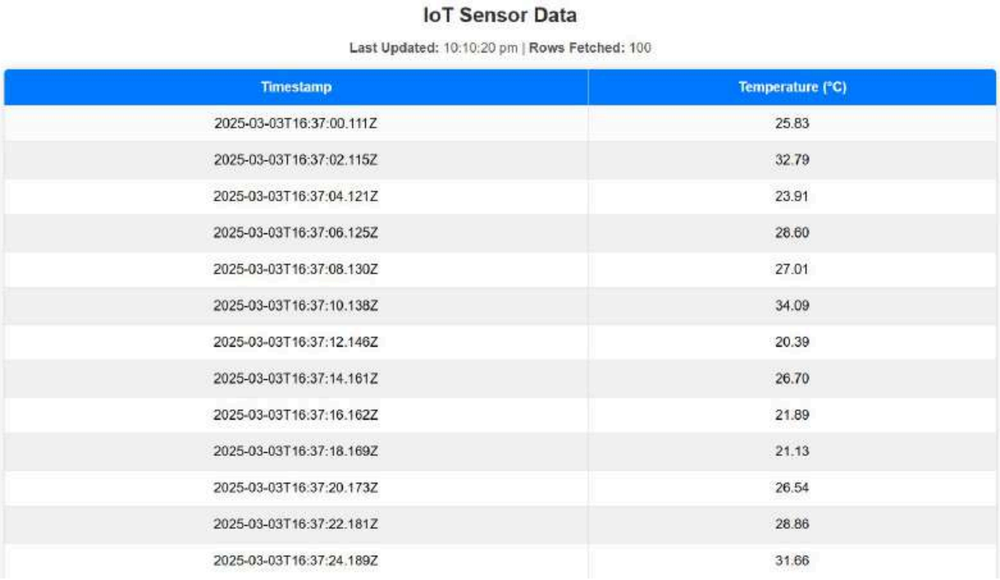
Code: App.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import "./App.css"; // Import external CSS

const API_URL = "http://localhost:5000/data";

const App = () => {
  const [data, setData] = useState([]);
  const [lastUpdated, setLastUpdated] = useState(null);
  const [rowCount, setRowCount] = useState(0);

  useEffect(() => {
    const fetchData = () => {
      axios.get(API_URL)
        .then((response) => {
          setData(response.data);
          setLastUpdated(new Date().toLocaleTimeString()); // Set last
update time
          setRowCount(response.data.length); // Set row count
        })
        .catch((error) => console.error("Error fetching data:", error));
    };

    fetchData(); // Initial fetch
    const interval = setInterval(fetchData, 5000); // Fetch every 5
seconds

    return () => clearInterval(interval); // Cleanup on unmount
  }, []);

  return (
    <div className="container">
```



```
      <h2 className="title">IoT Sensor Data</h2>
      <p className="info">
        <strong>Last Updated:</strong> {lastUpdated || "Fetching..."} |
        <strong> Rows Fetched:</strong> {rowCount}
      </p>
      <div className="table-container">
        <table className="styled-table">
          <thead>
            <tr>
              <th>Timestamp</th>
              <th>Temperature (°C)</th>
            </tr>
          </thead>
          <tbody>
            {data.map((row, index) => (
              <tr key={index}>
                <td>{row.timestamp}</td>
                <td>{row.temperature}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
};

export default App;
```

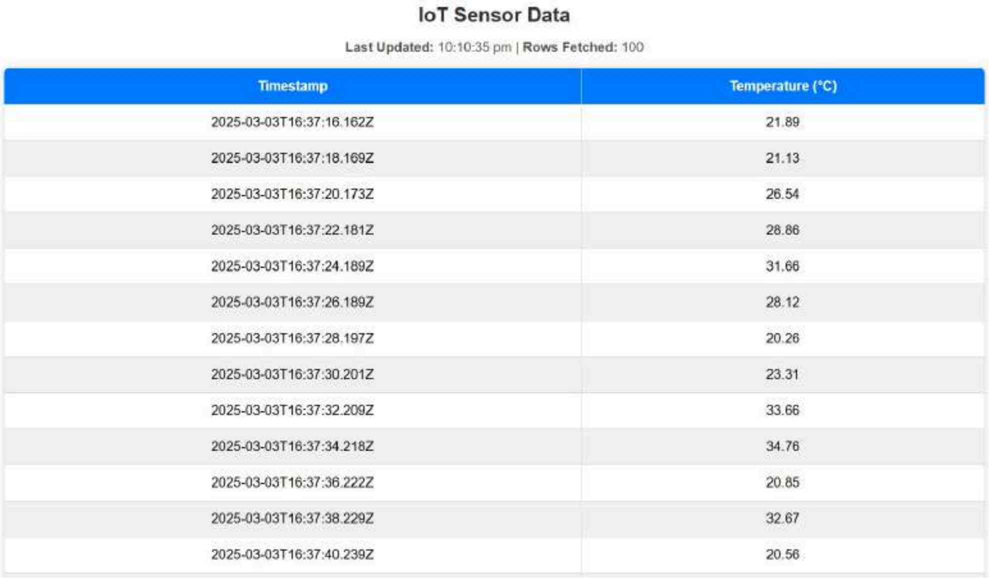
5. Open the localhost and view the updated data on the dashboard.



IoT Sensor Data

Last Updated: 10:10:20 pm | **Rows Fetched:** 100

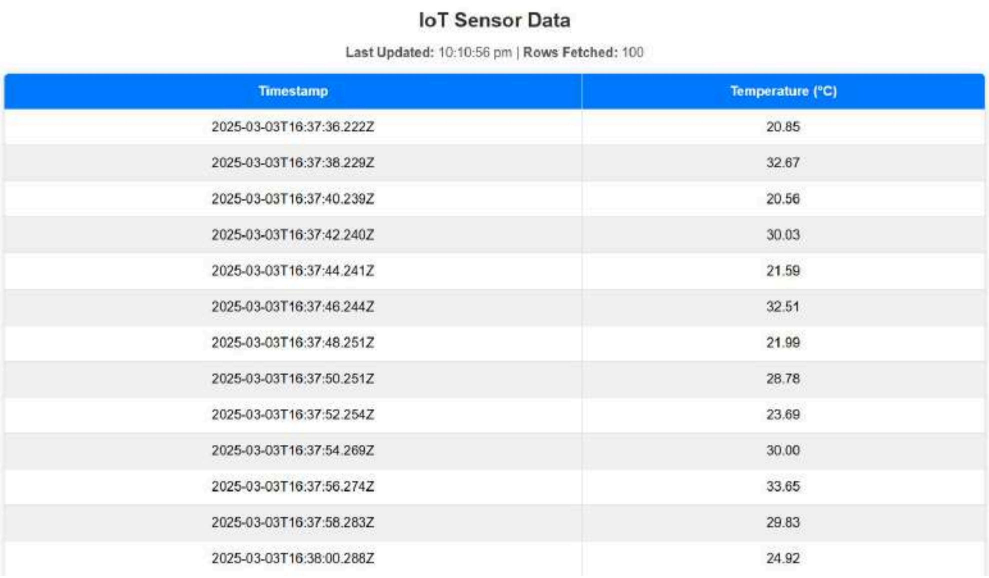
Timestamp	Temperature (°C)
2025-03-03T16:37:00.111Z	25.83
2025-03-03T16:37:02.115Z	32.79
2025-03-03T16:37:04.121Z	23.91
2025-03-03T16:37:06.125Z	28.60
2025-03-03T16:37:08.130Z	27.01
2025-03-03T16:37:10.138Z	34.09
2025-03-03T16:37:12.146Z	20.39
2025-03-03T16:37:14.161Z	26.70
2025-03-03T16:37:16.162Z	21.89
2025-03-03T16:37:18.169Z	21.13
2025-03-03T16:37:20.173Z	26.54
2025-03-03T16:37:22.181Z	28.86
2025-03-03T16:37:24.189Z	31.66



IoT Sensor Data

Last Updated: 10:10:35 pm | **Rows Fetched:** 100

Timestamp	Temperature (°C)
2025-03-03T16:37:16.162Z	21.89
2025-03-03T16:37:18.169Z	21.13
2025-03-03T16:37:20.173Z	26.54
2025-03-03T16:37:22.181Z	28.86
2025-03-03T16:37:24.189Z	31.66
2025-03-03T16:37:26.189Z	28.12
2025-03-03T16:37:28.197Z	20.26
2025-03-03T16:37:30.201Z	23.31
2025-03-03T16:37:32.209Z	33.66
2025-03-03T16:37:34.218Z	34.76
2025-03-03T16:37:36.222Z	20.85
2025-03-03T16:37:38.229Z	32.67
2025-03-03T16:37:40.239Z	20.56

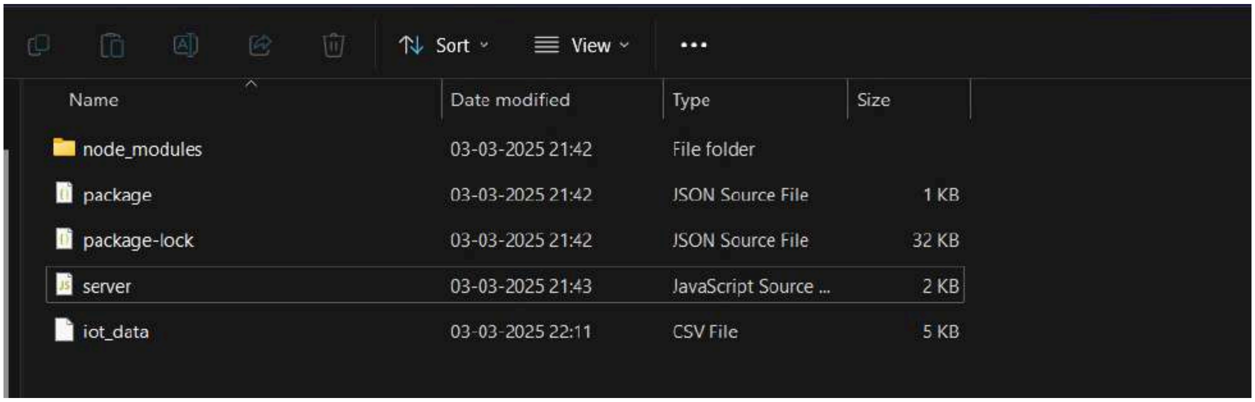


IoT Sensor Data

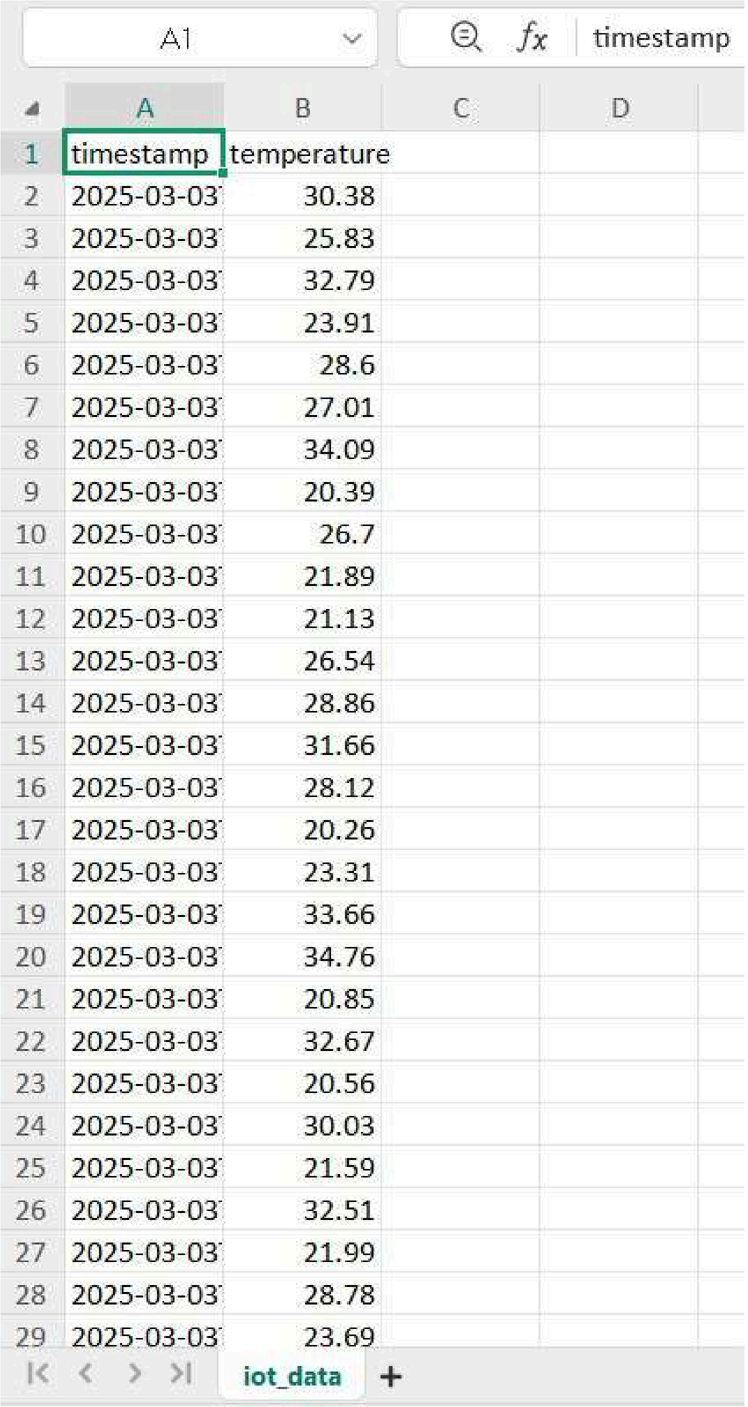
Last Updated: 10:10:56 pm | **Rows Fetched:** 100

Timestamp	Temperature (°C)
2025-03-03T16:37:36.222Z	20.85
2025-03-03T16:37:38.229Z	32.67
2025-03-03T16:37:40.239Z	20.56
2025-03-03T16:37:42.240Z	30.03
2025-03-03T16:37:44.241Z	21.59
2025-03-03T16:37:46.244Z	32.51
2025-03-03T16:37:48.251Z	21.99
2025-03-03T16:37:50.251Z	28.78
2025-03-03T16:37:52.254Z	23.69
2025-03-03T16:37:54.269Z	30.00
2025-03-03T16:37:56.274Z	33.65
2025-03-03T16:37:58.283Z	29.83
2025-03-03T16:38:00.288Z	24.92

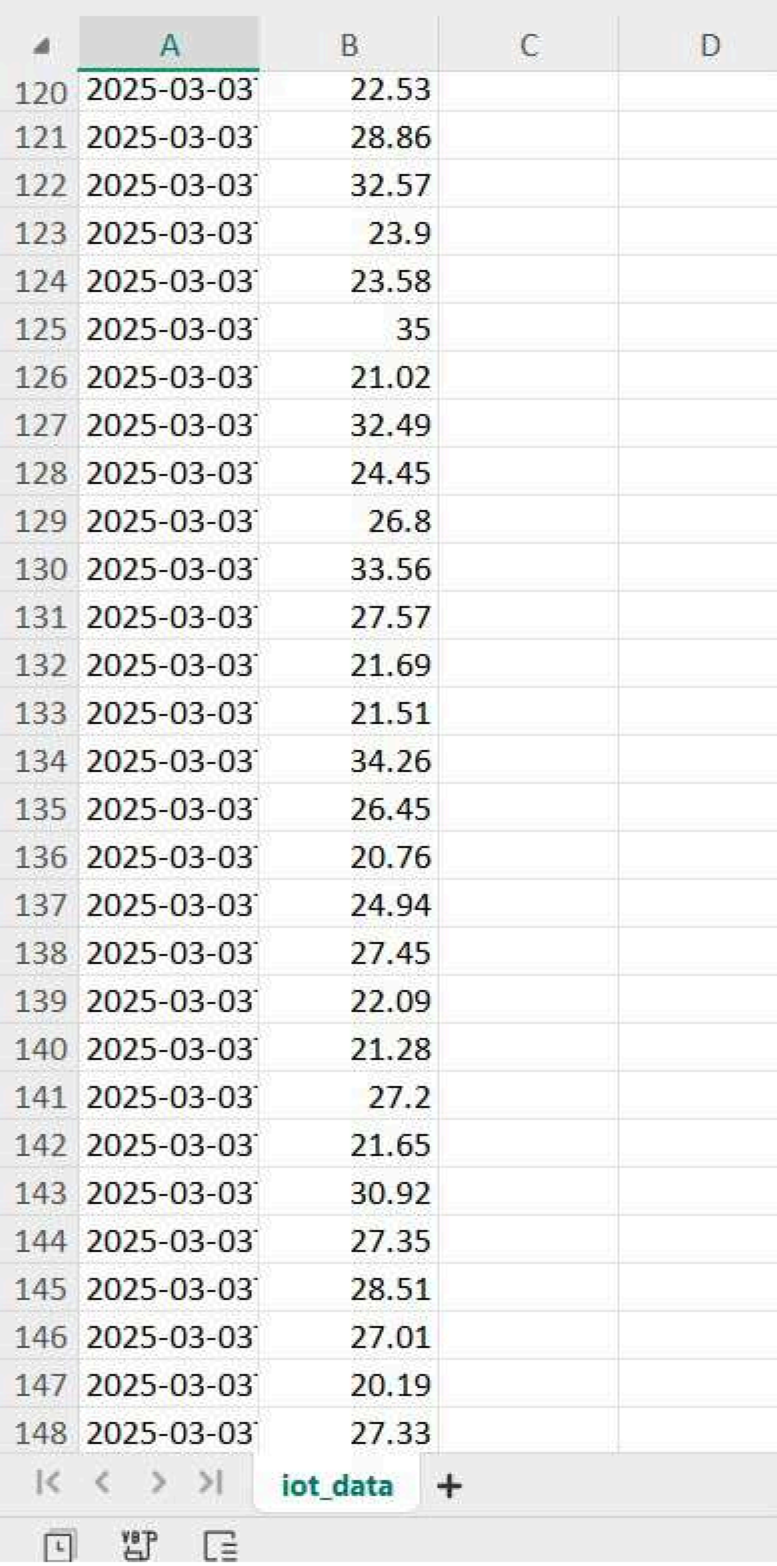
6. View and verify the Data getting stored in CSV file.



Name	Date modified	Type	Size
node_modules	03-03-2025 21:42	File folder	
package	03-03-2025 21:42	JSON Source File	1 KB
package-lock	03-03-2025 21:42	JSON Source File	32 KB
server	03-03-2025 21:43	JavaScript Source ...	2 KB
iot_data	03-03-2025 22:11	CSV File	5 KB



	A	B	C	D
1	timestamp	temperature		
2	2025-03-03	30.38		
3	2025-03-03	25.83		
4	2025-03-03	32.79		
5	2025-03-03	23.91		
6	2025-03-03	28.6		
7	2025-03-03	27.01		
8	2025-03-03	34.09		
9	2025-03-03	20.39		
10	2025-03-03	26.7		
11	2025-03-03	21.89		
12	2025-03-03	21.13		
13	2025-03-03	26.54		
14	2025-03-03	28.86		
15	2025-03-03	31.66		
16	2025-03-03	28.12		
17	2025-03-03	20.26		
18	2025-03-03	23.31		
19	2025-03-03	33.66		
20	2025-03-03	34.76		
21	2025-03-03	20.85		
22	2025-03-03	32.67		
23	2025-03-03	20.56		
24	2025-03-03	30.03		
25	2025-03-03	21.59		
26	2025-03-03	32.51		
27	2025-03-03	21.99		
28	2025-03-03	28.78		
29	2025-03-03	23.69		

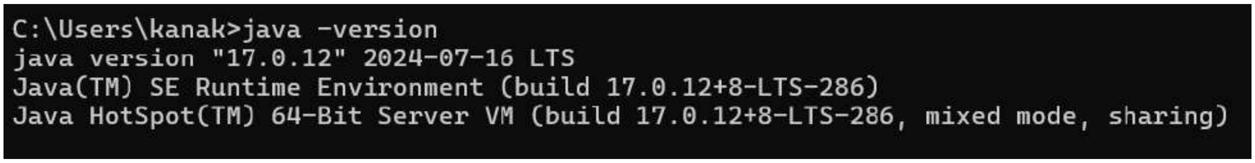


	A	B	C	D
120	2025-03-03	22.53		
121	2025-03-03	28.86		
122	2025-03-03	32.57		
123	2025-03-03	23.9		
124	2025-03-03	23.58		
125	2025-03-03	35		
126	2025-03-03	21.02		
127	2025-03-03	32.49		
128	2025-03-03	24.45		
129	2025-03-03	26.8		
130	2025-03-03	33.56		
131	2025-03-03	27.57		
132	2025-03-03	21.69		
133	2025-03-03	21.51		
134	2025-03-03	34.26		
135	2025-03-03	26.45		
136	2025-03-03	20.76		
137	2025-03-03	24.94		
138	2025-03-03	27.45		
139	2025-03-03	22.09		
140	2025-03-03	21.28		
141	2025-03-03	27.2		
142	2025-03-03	21.65		
143	2025-03-03	30.92		
144	2025-03-03	27.35		
145	2025-03-03	28.51		
146	2025-03-03	27.01		
147	2025-03-03	20.19		
148	2025-03-03	27.33		

Task 2

Algorithm:

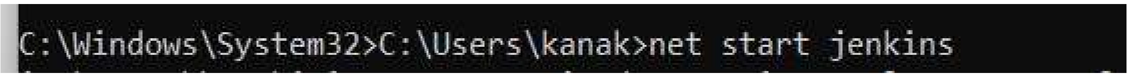
1. Install Java version 17 or 21. (For our case we will proceed with Java version 17)



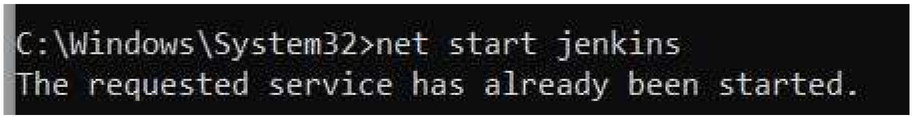
```
C:\Users\kanak>java -version
java version "17.0.12" 2024-07-16 LTS
Java(TM) SE Runtime Environment (build 17.0.12+8-LTS-286)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.12+8-LTS-286, mixed mode, sharing)
```

2. Install Jenkins.

3. Start jenkins using CMD (as administrator)



```
C:\Windows\System32>C:\Users\kanak>net start jenkins
```



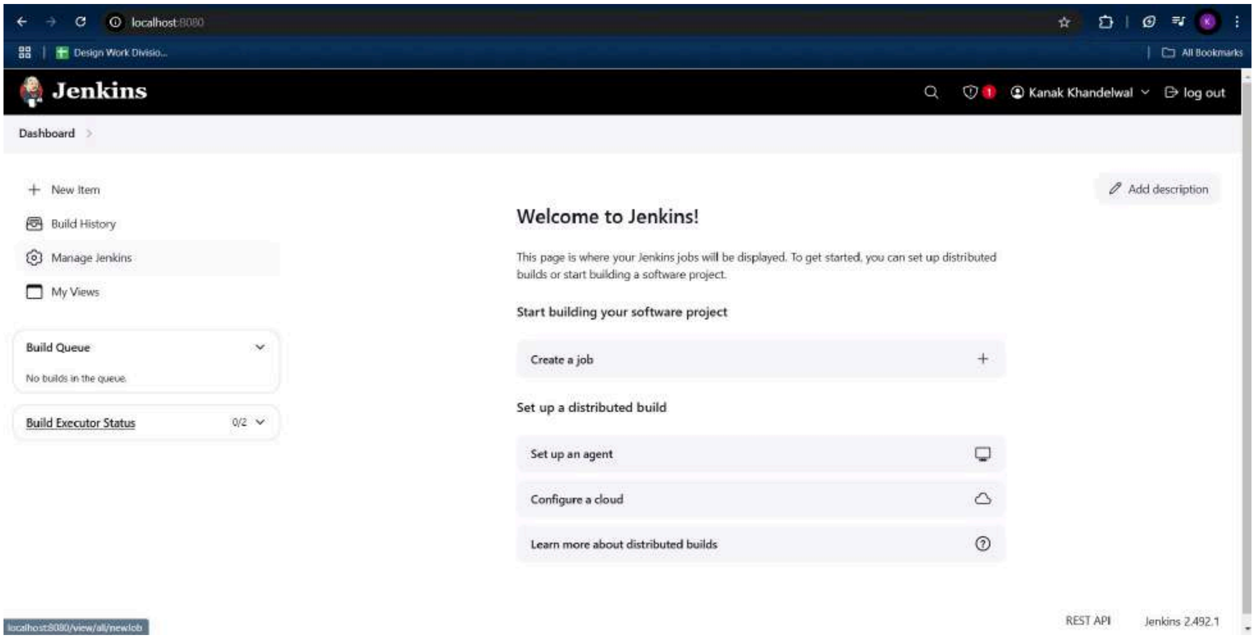
```
C:\Windows\System32>net start jenkins
The requested service has already been started.
```

4. Start your localhost (port 8080 which was added during installation).

http://localhost:8080

5. Install necessary plugins and setup admin.

Login to jenkins. Go to dashboard



Welcome to Jenkins!

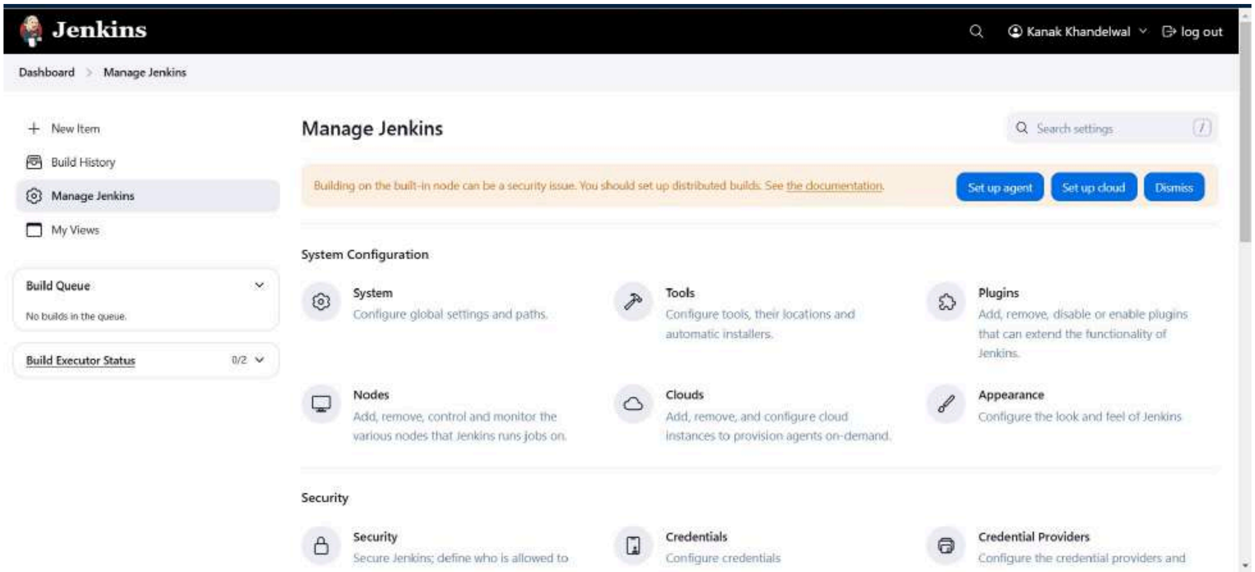
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

No builds in the queue.

7. Go to manage jenkins from the left panel.



Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

No builds in the queue.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

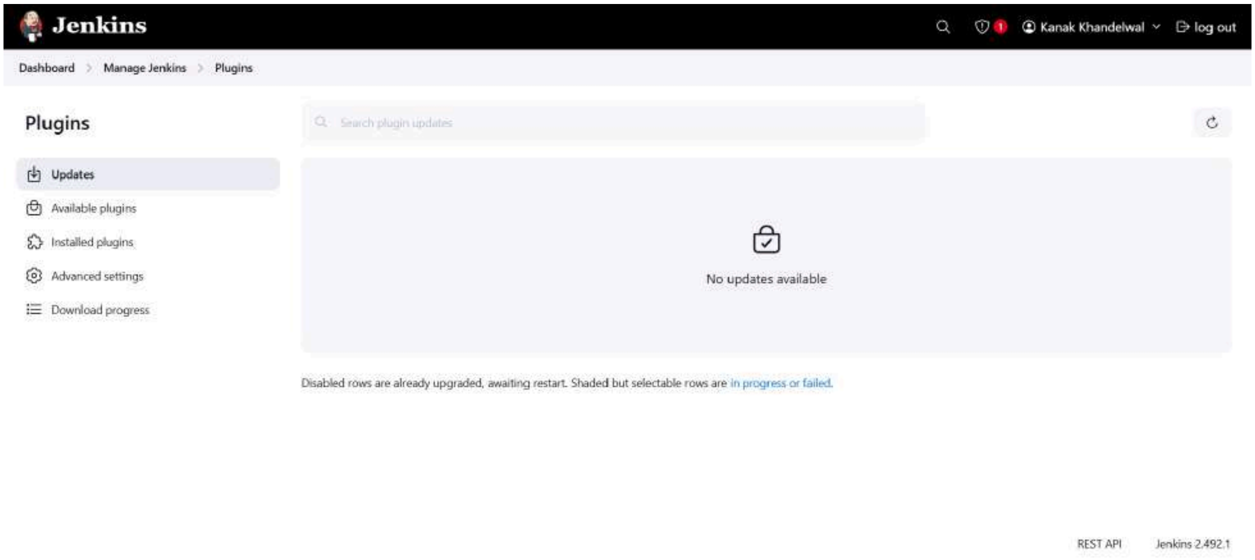
Security

Security
Secure Jenkins; define who is allowed to

Credentials
Configure credentials

Credential Providers
Configure the credential providers and

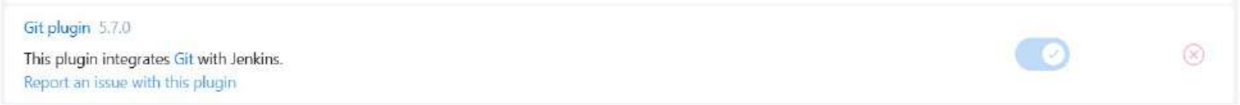
8. Go to plugins



No updates available

Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed.

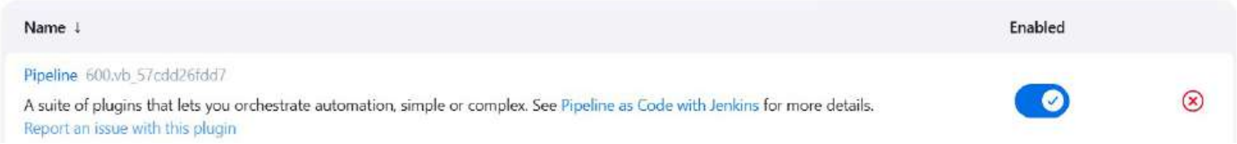
9. Install and enable Git Plugin and Pipeline Plugin.



Git plugin 5.7.0

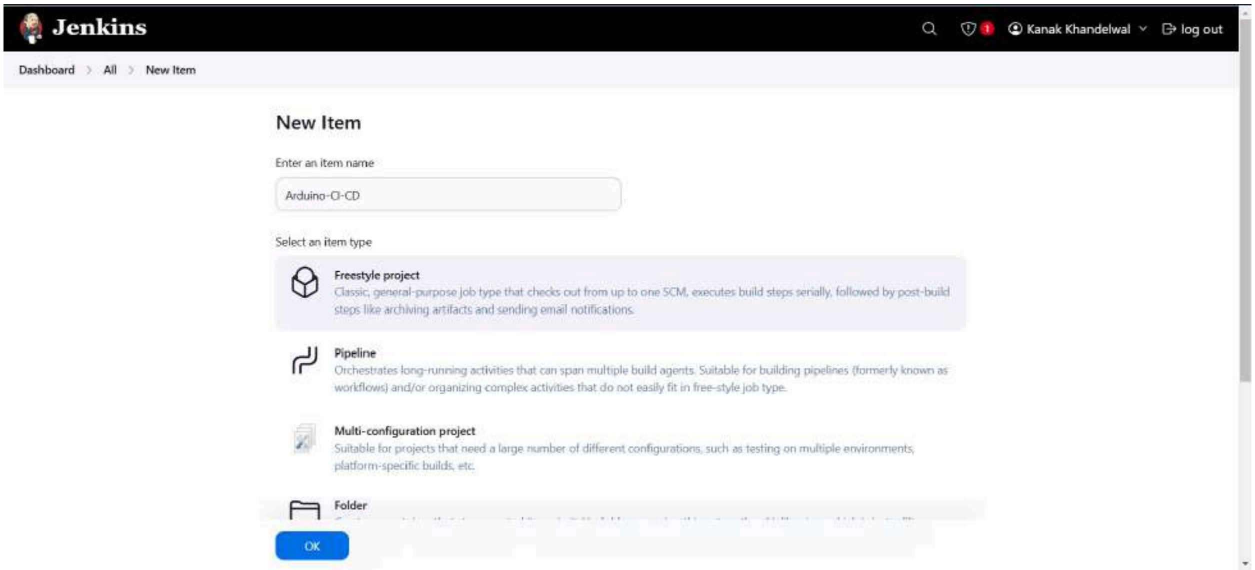
This plugin integrates Git with Jenkins.

Report an issue with this plugin



Name ↓	Enabled
Pipeline 600.vb_57cdd26fdd7 A suite of plugins that lets you orchestrate automation, simple or complex. See Pipeline as Code with Jenkins for more details. Report an issue with this plugin	✓

10. Create a new jenkins job, enter desired name and select pipeline.



New Item

Enter an item name

Arduino-CI-CD

Select an item type

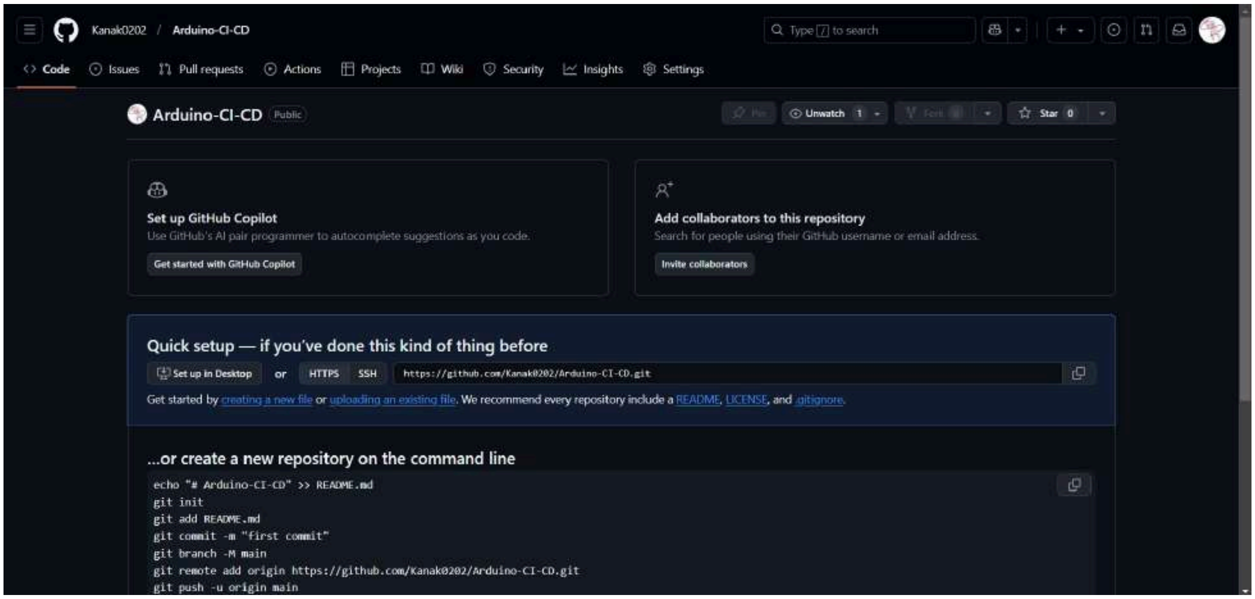
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

11. Create git repository for arduino.



Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Add collaborators to this repository
Search for people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

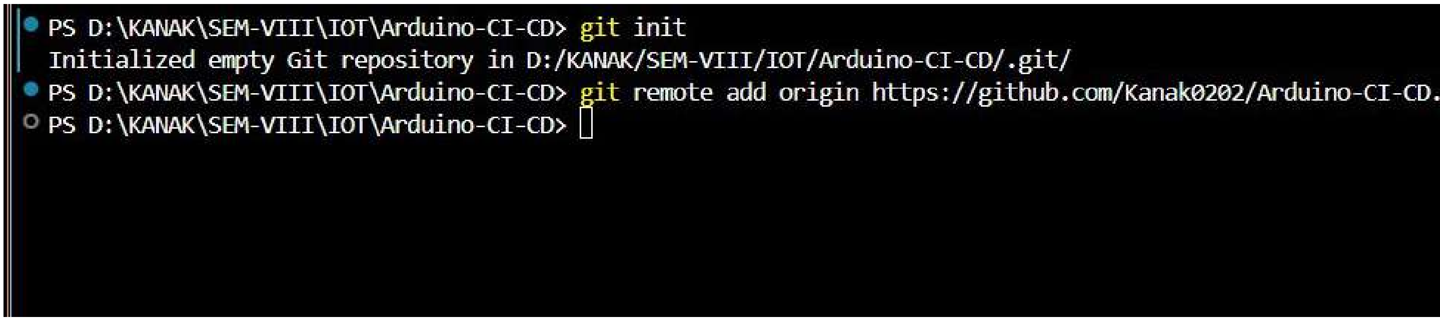
https://github.com/Kanak0202/Arduino-CI-CD.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

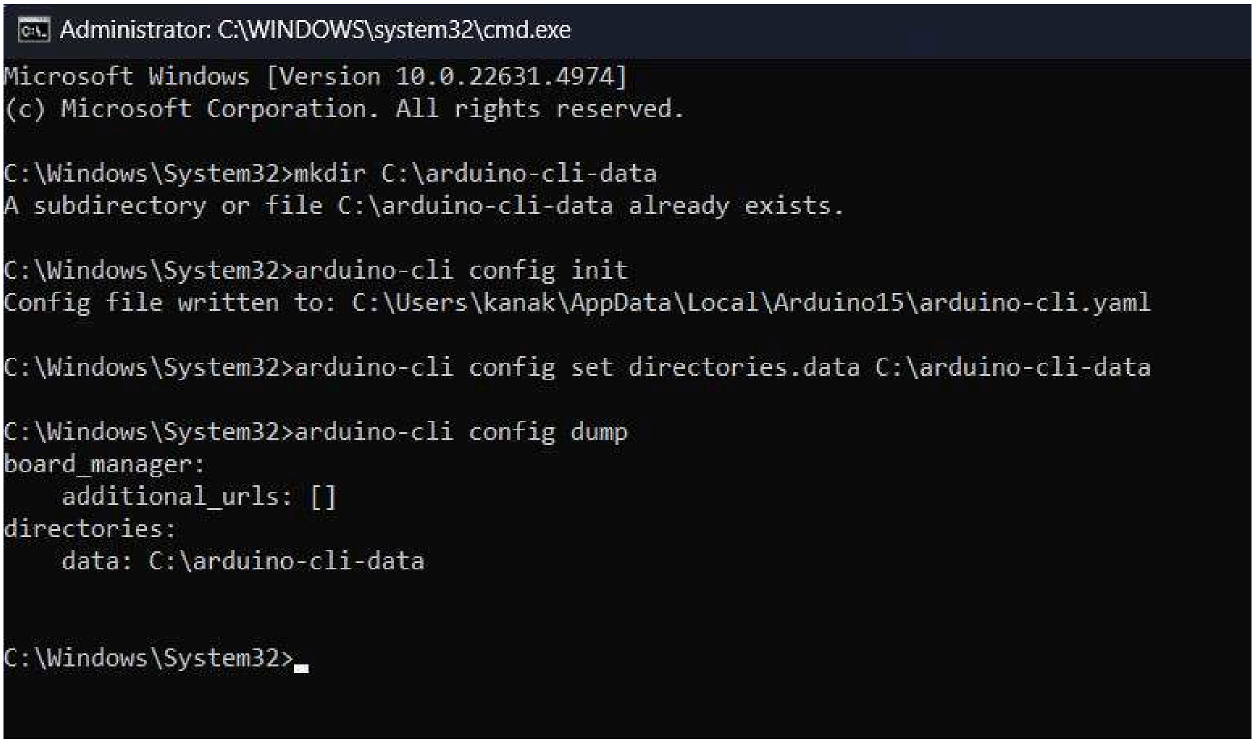
...or create a new repository on the command line

```
echo "# Arduino-CI-CD" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Kanak0202/Arduino-CI-CD.git
git push -u origin main
```

12. Setup project on local.



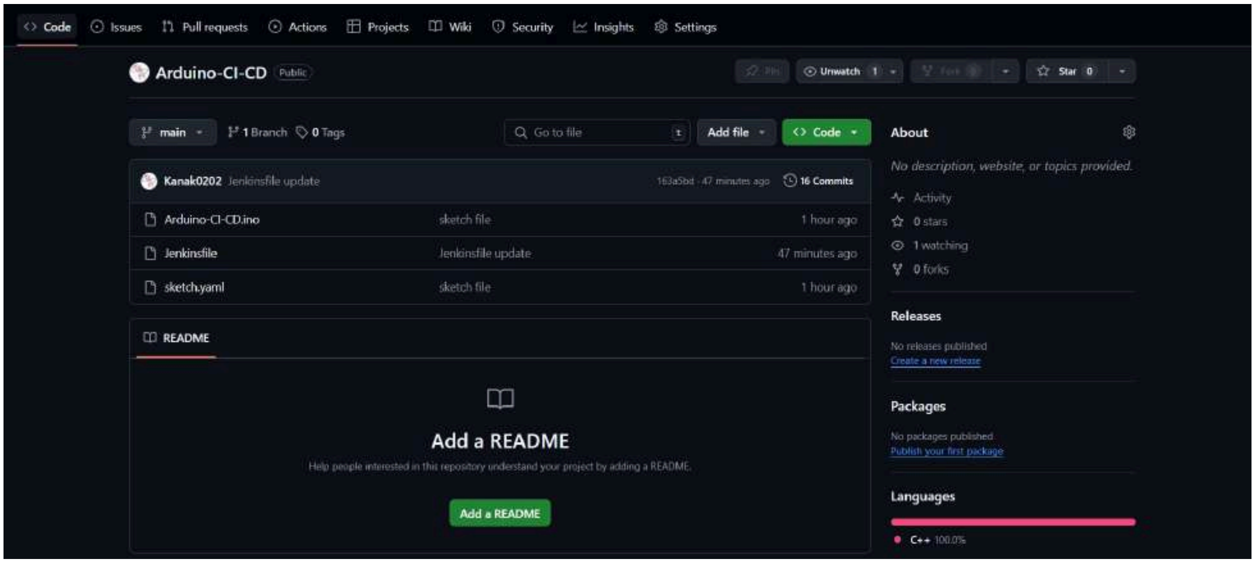
```
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD> git init
Initialized empty Git repository in D:/KANAK/SEM-VIII/IOT/Arduino-CI-CD/.git/
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD> git remote add origin https://github.com/Kanak0202/Arduino-CI-CD.
PS D:\KANAK\SEM-VIII\IOT\Arduino-CI-CD> 
```



Administrator: C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.22631.4974]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>mkdir C:\arduino-cli-data
A subdirectory or file C:\arduino-cli-data already exists.

C:\Windows\System32>arduino-cli config init
Config file written to: C:\Users\kanak\AppData\Local\Arduino15\arduino-cli.yaml

C:\Windows\System32>arduino-cli config set directories.data C:\arduino-cli-data

C:\Windows\System32>arduino-cli config dump
board_manager:
    additional_urls: []
directories:
    data: C:\arduino-cli-data


C:\Windows\System32>_
```

Push the code to github onto your repository.



Kanak0202 Jenkinsfile update · 163a5bd · 47 minutes ago · 16 Commits

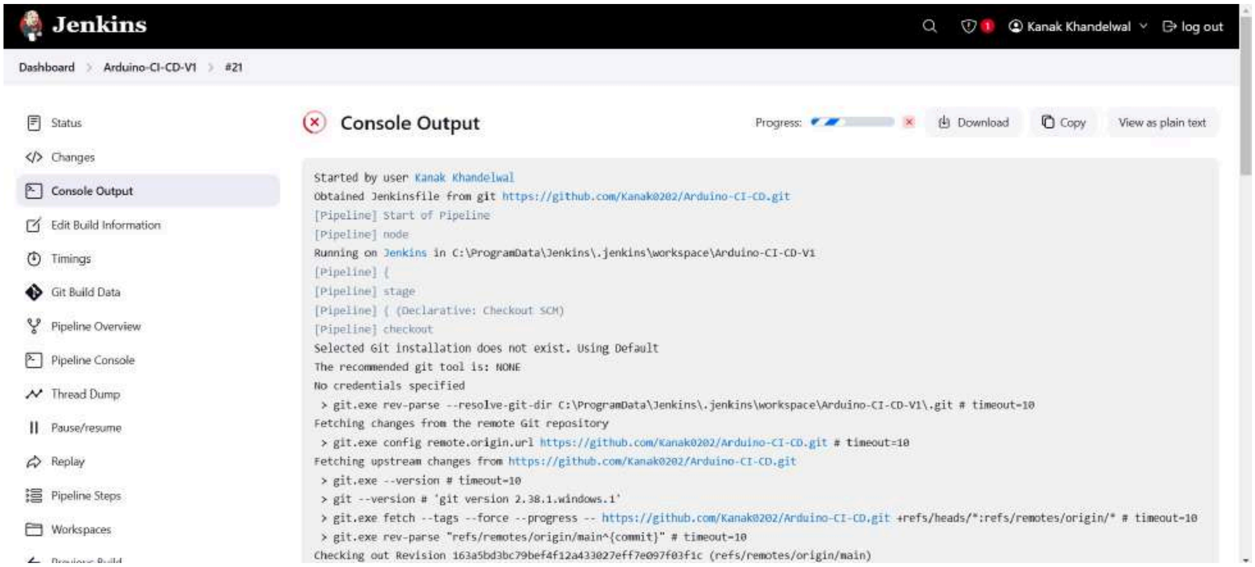
Arduino-CI-CD.ino	sketch file	1 hour ago
Jenkinsfile	Jenkinsfile update	47 minutes ago
sketch.yaml	sketch file	1 hour ago

README

Add a README

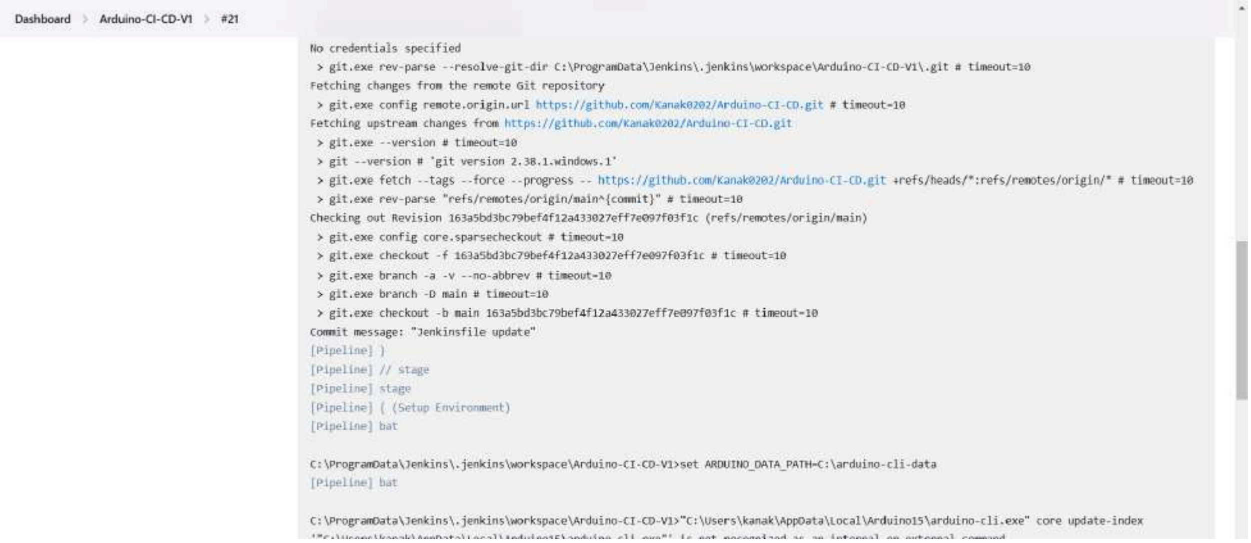
Help people interested in this repository understand your project by adding a README.

Run your build.

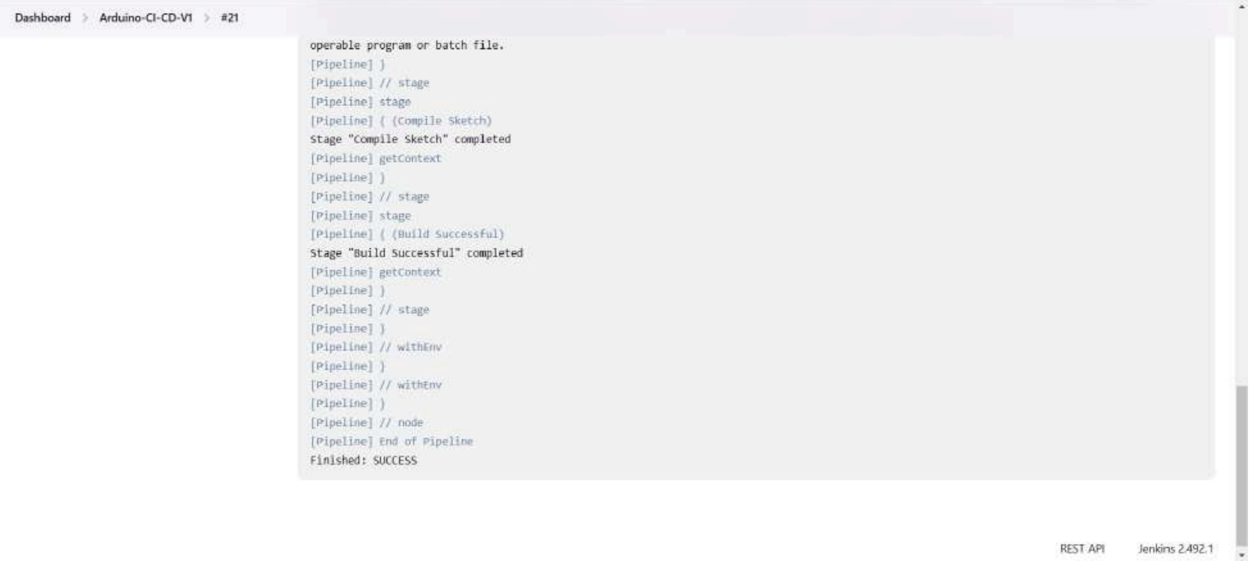


Console Output

```
Started by user Kanak Khandelwal
Obtained Jenkinsfile from git https://github.com/Kanak0202/Arduino-CI-CD.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\Arduino-CI-CD-V1
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\Arduino-CI-CD-V1\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Kanak0202/Arduino-CI-CD.git # timeout=10
Fetching upstream changes from https://github.com/Kanak0202/Arduino-CI-CD.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.38.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/Kanak0202/Arduino-CI-CD.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 163a5bd3bc79bef4f12a433027eff7e097f03f1c (refs/remotes/origin/main)
```



```
No credentials specified
 > git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\Arduino-CI-CD-V1\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Kanak0202/Arduino-CI-CD.git # timeout=10
Fetching upstream changes from https://github.com/Kanak0202/Arduino-CI-CD.git
 > git.exe --version # timeout=10
 > git --version # 'git version 2.38.1.windows.1'
 > git.exe fetch --tags --force --progress -- https://github.com/Kanak0202/Arduino-CI-CD.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 163a5bd3bc79bef4f12a433027eff7e097f03f1c (refs/remotes/origin/main)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 163a5bd3bc79bef4f12a433027eff7e097f03f1c # timeout=10
 > git.exe branch -a -v --no-abbrev # timeout=10
 > git.exe branch -D main # timeout=10
 > git.exe checkout -b main 163a5bd3bc79bef4f12a433027eff7e097f03f1c # timeout=10
Commit message: "Jenkinsfile update"
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Setup Environment)
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\Arduino-CI-CD-V1>set ARDUINO_DATA_PATH=C:\arduino-cli-data
[Pipeline] bat

C:\ProgramData\Jenkins\.jenkins\workspace\Arduino-CI-CD-V1>"C:\Users\kanak\AppData\Local\Arduino15\arduino-cli.exe" core update-index
'"C:\Users\kanak\AppData\Local\Arduino15\arduino-cli.exe"' is not recognized as an internal or external command,
```



```
operable program or batch file.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Compile Sketch)
Stage "Compile Sketch" completed
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build Successful)
Stage "Build Successful" completed
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Code

Arduino-CI-CD.ino

```
// Simple Blink Sketch

#define LED_PIN 13
void setup() {
    pinMode(LED_PIN, OUTPUT);
}
```

```
void loop() {
    digitalWrite(LED_PIN, HIGH); // Turn LED on
    delay(1000);
    digitalWrite(LED_PIN, LOW); // Turn LED off
    delay(1000);
}
```

Jenkinsfile

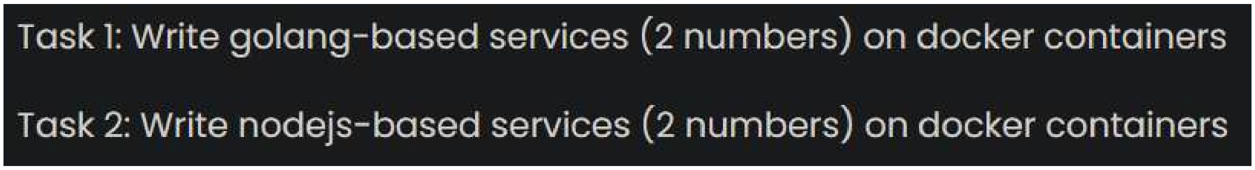
```
pipeline {
    agent any

    environment {
        ARDUINO_CLI =
"C:\\Users\\kanak\\AppData\\Local\\Arduino15\\arduino-cli.exe"
        ARDUINO_DATA_PATH = "C:\\arduino-cli-data"
    }

    stages {
        stage('Checkout Code') {
        steps {
            git branch: 'main', url:
'https://github.com/Kanak0202/Arduino-CI-CD.git'
        }
    }

        stage('Setup Environment') {
            steps {
                bat 'set ARDUINO_DATA_PATH=C:\\arduino-cli-data'
                bat '"%ARDUINO_CLI%" core update-index'
                bat '"%ARDUINO_CLI%" core install arduino:avr'
            }
        }

        stage('Compile Sketch') {
            steps {
                bat '"%ARDUINO_CLI%" compile --fqbn arduino:avr:uno .'
            }
        }

        stage('Build Successful') {
            steps {
                echo 'Build completed successfully!'
            }
        }
    }
}
```

Lab 9

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



Task 1: Write golang-based services (2 numbers) on docker containers

Task 2: Write nodejs-based services (2 numbers) on docker containers

Task 1

1. Introduction

This project involves developing two microservices using Golang, containerizing them using Docker, and orchestrating them with Docker Compose. The microservices perform basic arithmetic operations:

- Service A: Addition (/add)
- Service B: Multiplication (/multiply)

Each service runs independently in a Docker container, ensuring scalability and ease of deployment.

---

2. Methodology

The following steps were followed:

1. Developed two Golang microservices using net/http.
2. Containerized each service using Docker.
3. Created a Docker Compose file to orchestrate the services.
4. Built and deployed the containers.
5. Tested the endpoints using curl.

3. Implementation

3.1. Service A: Addition Microservice

1. Create serviceA/go.mod

module serviceA

go 1.24

2. Create serviceA/main.go

```go
package main

import (
    "encoding/json"
    "fmt"
    "net/http"
)
type Request struct {
    A float64 `json:"a"`
    B float64 `json:"b"`
}

type Response struct {
    Result float64 `json:"result"`
}

func addHandler(w http.ResponseWriter, r *http.Request) {
    var req Request
    err := json.NewDecoder(r.Body).Decode(&req)
    if err != nil {
        http.Error(w, "Invalid request", http.StatusBadRequest)
        return
    }

    result := req.A + req.B
    resp := Response{Result: result}

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(resp)
}

func main() {
    http.HandleFunc("/add", addHandler)
    fmt.Println("Adder Service running on port 8081...")
    http.ListenAndServe(":8081", nil)
}
```

3. Create serviceA/Dockerfile

```dockerfile
FROM golang:1.24

WORKDIR /app

COPY . .

RUN go mod tidy && go build -o main .

CMD ["/app/main"]
```

3.2. Service B: Multiplication Microservice

1. Create serviceB/go.mod

module serviceB

go 1.24

2. Create serviceB/main.go

```go
package main

import (
    "encoding/json"
    "fmt"
    "net/http"
)

type Request struct {
    A float64 `json:"a"`
    B float64 `json:"b"`
}

type Response struct {
    Result float64 `json:"result"`
}

func multiplyHandler(w http.ResponseWriter, r *http.Request) {
    var req Request
    err := json.NewDecoder(r.Body).Decode(&req)
    if err != nil {
        http.Error(w, "Invalid request", http.StatusBadRequest)
        return
    }

    result := req.A * req.B
    resp := Response{Result: result}

    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(resp)
}

func main() {
    http.HandleFunc("/multiply", multiplyHandler)
    fmt.Println("Multiplier Service running on port 8082...")
    http.ListenAndServe(":8082", nil)
}
```

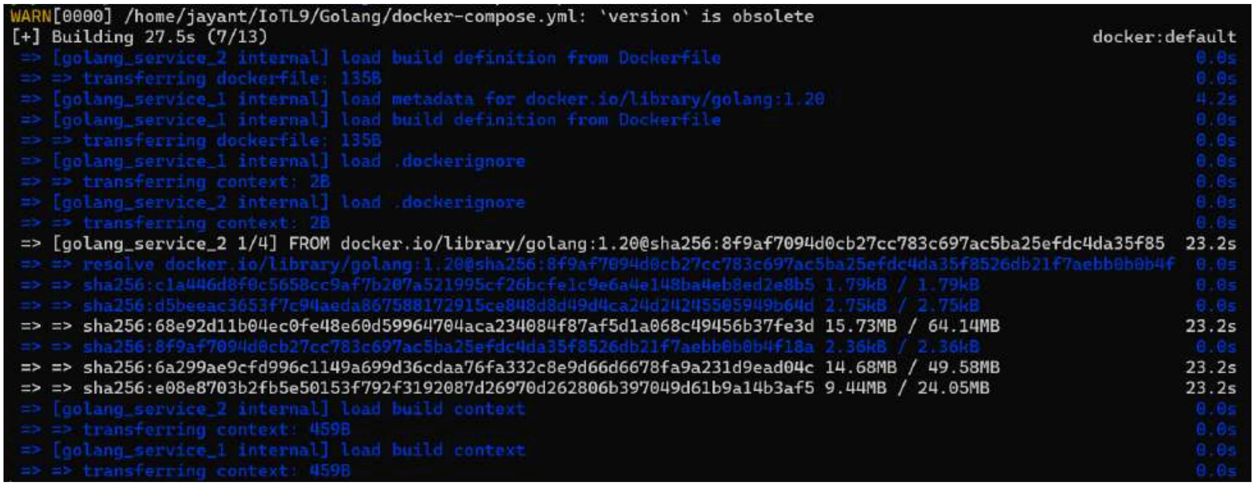
3. Create serviceB/Dockerfile

```
FROM golang:1.24

WORKDIR /app

COPY . .

RUN go mod tidy && go build -o main .

CMD ["/app/main"]
```

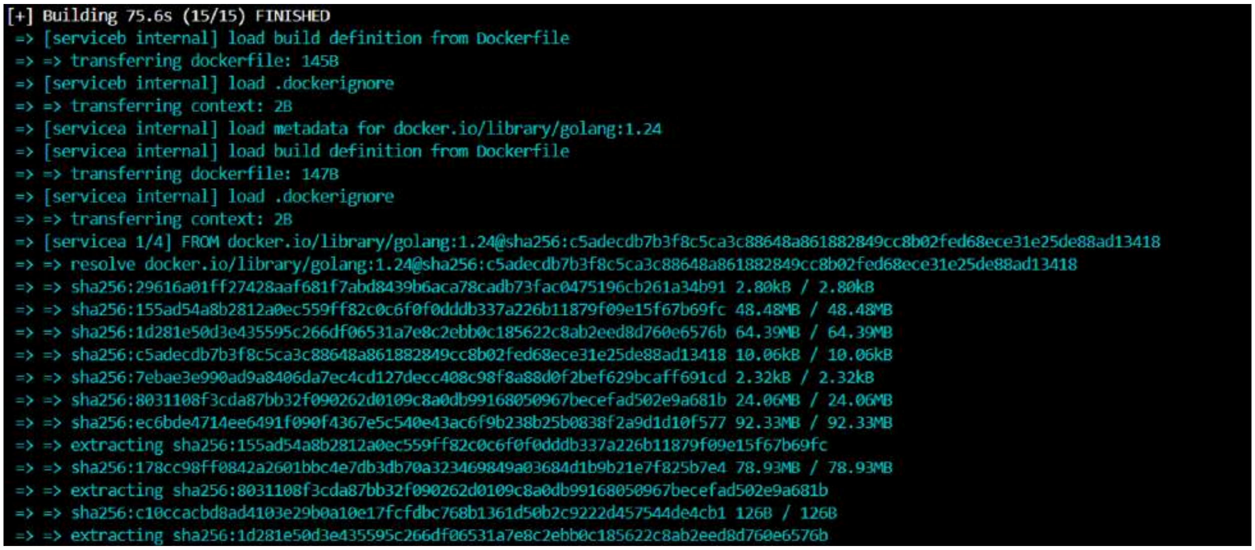
4. Deployment Steps

Step 1: Build and Start Containers

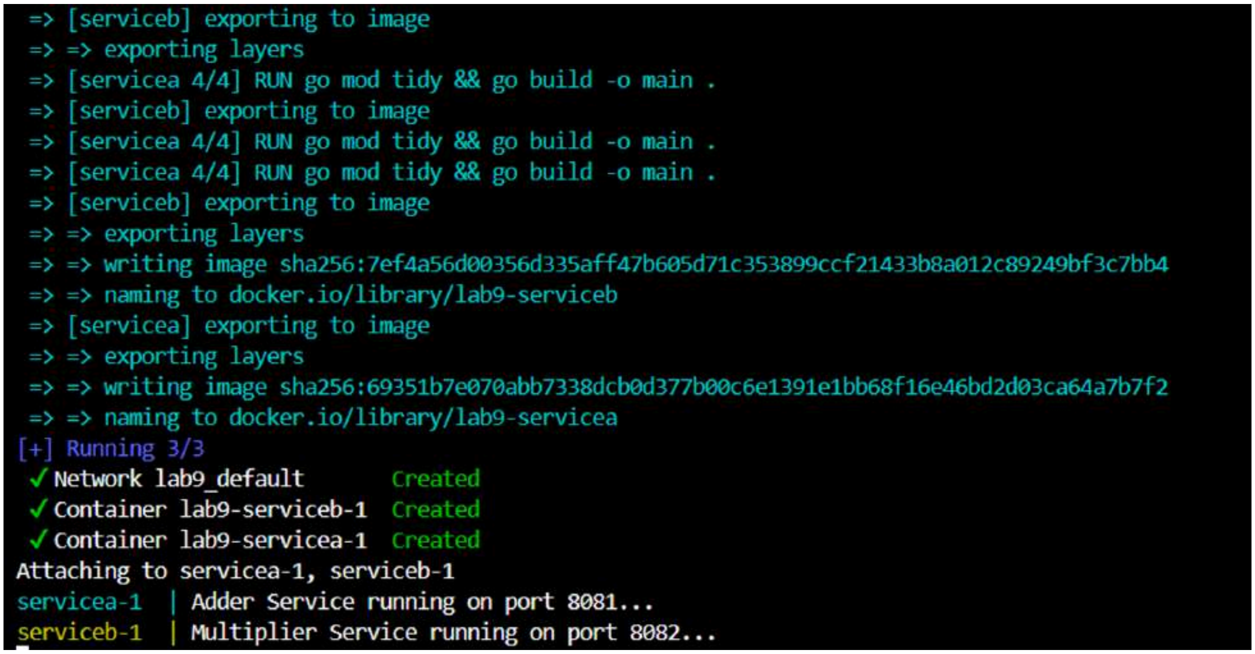
docker-compose up –build



```
WARN[0000] /home/jayant/IoTL9/Golang/docker-compose.yml: `version` is obsolete
[+] Building 27.5s (7/13)                                                                         docker:default
 => [golang_service_2 internal] load build definition from Dockerfile                                       0.0s
 => => transferring dockerfile: 135B                                                                        0.0s
 => [golang_service_1 internal] load metadata for docker.io/library/golang:1.20                             4.2s
 => [golang_service_1 internal] load build definition from Dockerfile                                       0.0s
 => => transferring dockerfile: 135B                                                                        0.0s
 => [golang_service_1 internal] load .dockerignore                                                          0.0s
 => => transferring context: 2B                                                                             0.0s
 => [golang_service_2 internal] load .dockerignore                                                          0.0s
 => => transferring context: 2B                                                                             0.0s
 => [golang_service_2 1/4] FROM docker.io/library/golang:1.20@sha256:8f9af7094d0cb27cc783c697ac5ba25efdc4da35f85  23.2s
 => => resolve docker.io/library/golang:1.20@sha256:8f9af7094d0cb27cc783c697ac5ba25efdc4da35f8526db21f7aebb0b0b4f  0.0s
 => => sha256:c1a446d8f0c5658cc9af7b207a521995cf26bcfe1c9e6a4e148ba4eb8ed2e8b5 1.79kB / 1.79kB              0.0s
 => => sha256:d5beeac3653f7c94aeda867588172915ce848d8d49d4ca24d24245505949b64d 2.75kB / 2.75kB              0.0s
 => => sha256:68e92d11b04ec0fe48e60d59964704aca234084f87af5d1a068c49456b37fe3d 15.73MB / 64.14MB           23.2s
 => => sha256:8f9af7094d0cb27cc783c697ac5ba25efdc4da35f8526db21f7aebb0b0b4f18a 2.36kB / 2.36kB              0.0s
 => => sha256:6a299ae9cfd996c1149a699d36cdaa76fa332c8e9d66d6678fa9a231d9ead04c 14.68MB / 49.58MB           23.2s
 => => sha256:e08e8703b2fb5e50153f792f3192087d26970d262806b397049d61b9a14b3af5 9.44MB / 24.05MB            23.2s
 => [golang_service_2 internal] load build context                                                          0.0s
 => => transferring context: 459B                                                                           0.0s
 => [golang_service_1 internal] load build context                                                          0.0s
 => => transferring context: 459B                                                                           0.0s
```



```
[+] Building 75.6s (15/15) FINISHED
 => [serviceb internal] load build definition from Dockerfile
 => => transferring dockerfile: 145B
 => [serviceb internal] load .dockerignore
 => => transferring context: 2B
 => [servicea internal] load metadata for docker.io/library/golang:1.24
 => [servicea internal] load build definition from Dockerfile
 => => transferring dockerfile: 147B
 => [servicea internal] load .dockerignore
 => => transferring context: 2B
 => [servicea 1/4] FROM docker.io/library/golang:1.24@sha256:c5adecdb7b3f8c5ca3c88648a861882849cc8b02fed68ece31e25de88ad13418
 => => resolve docker.io/library/golang:1.24@sha256:c5adecdb7b3f8c5ca3c88648a861882849cc8b02fed68ece31e25de88ad13418
 => => sha256:29616a01ff27428aaf681f7abd8439b6aca78cadb73fac0475196cb261a34b91 2.80kB / 2.80kB
 => => sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879f09e15f67b69fc 48.48MB / 48.48MB
 => => sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2eed8d760e6576b 64.39MB / 64.39MB
 => => sha256:c5adecdb7b3f8c5ca3c88648a861882849cc8b02fed68ece31e25de88ad13418 10.06kB / 10.06kB
 => => sha256:7ebae3e990ad9a8406da7ec4cd127decc408c98f8a88d0f2bef629bcaff691cd 2.32kB / 2.32kB
 => => sha256:8031108f3cda87bb32f090262d0109c8a0db99168050967becefad502e9a681b 24.06MB / 24.06MB
 => => sha256:ec6bde4714ee6491f090f4367e5c540e43ac6f9b238b25b0838f2a9d1d10f577 92.33MB / 92.33MB
 => => extracting sha256:155ad54a8b2812a0ec559ff82c0c6f0f0dddb337a226b11879f09e15f67b69fc
 => => sha256:178cc98ff0842a2601bbc4e7db3db70a323469849a03684d1b9b21e7f825b7e4 78.93MB / 78.93MB
 => => extracting sha256:8031108f3cda87bb32f090262d0109c8a0db99168050967becefad502e9a681b
 => => sha256:c10ccacbd8ad4103e29b0a10e17fcfdbc768b1361d50b2c9222d457544de4cb1 126B / 126B
 => => extracting sha256:1d281e50d3e435595c266df06531a7e8c2ebb0c185622c8ab2eed8d760e6576b
```



```
 => [serviceb] exporting to image
 => => exporting layers
 => [servicea 4/4] RUN go mod tidy && go build -o main .
 => [serviceb] exporting to image
 => [servicea 4/4] RUN go mod tidy && go build -o main .
 => [servicea 4/4] RUN go mod tidy && go build -o main .
 => [serviceb] exporting to image
 => => exporting layers
 => => writing image sha256:7ef4a56d00356d335aff47b605d71c353899ccf21433b8a012c89249bf3c7bb4
 => => naming to docker.io/library/lab9-serviceb
 => [servicea] exporting to image
 => => exporting layers
 => => writing image sha256:69351b7e070abb7338dcb0d377b00c6e1391e1bb68f16e46bd2d03ca64a7b7f2
 => => naming to docker.io/library/lab9-servicea
[+] Running 3/3
 ✓ Network lab9_default       Created
 ✓ Container lab9-serviceb-1  Created
 ✓ Container lab9-servicea-1  Created
Attaching to servicea-1, serviceb-1
servicea-1  | Adder Service running on port 8081...
serviceb-1  | Multiplier Service running on port 8082...
```

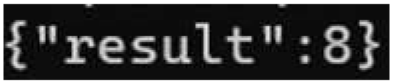
5. Testing the Services

5.1. Test Service A (Addition)

curl -X POST http://localhost:8081/add -H "Content-Type: application/json" -d "{\"a\":5, \"b\":3}"

Expected Response:

{"result":8}



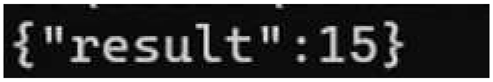
```
{"result":8}
```

5.2. Test Service B (Multiplication)

curl -X POST http://localhost:8082/multiply -H "Content-Type: application/json" -d "{\"a\":5, \"b\":3}"

Expected Response:

{"result":15}



```
{"result":15}
```

Task 2

1. Introduction

This project involves developing two microservices using Node.js, containerizing them using Docker, and orchestrating them with Docker Compose. The microservices perform basic arithmetic operations:

Service A: Addition (/add)

Service B: Multiplication (/multiply)

Each service runs independently in a Docker container, ensuring scalability and ease of deployment.

2. Methodology

The following steps were followed:

1. Developed two Node.js microservices using express.

2. Containerized each service using Docker.

3. Created a Docker Compose file to orchestrate the services.

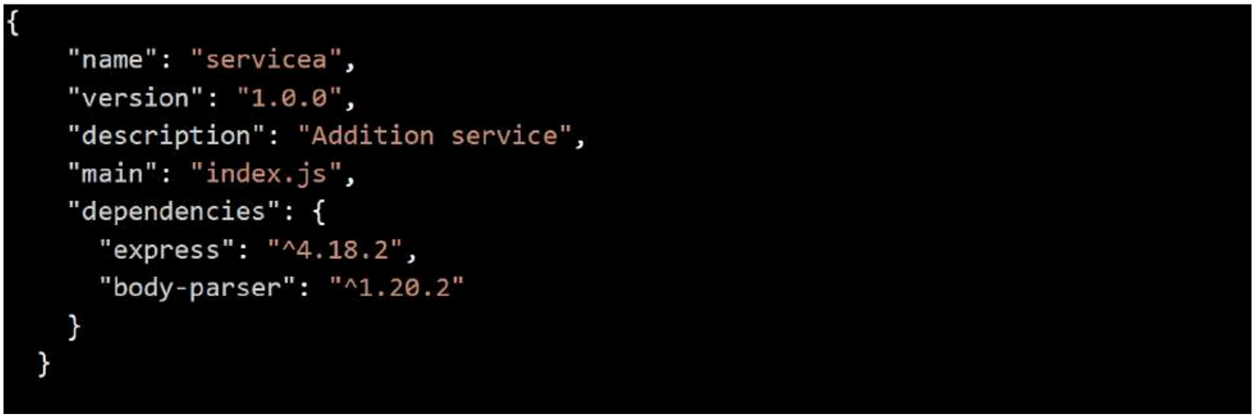
4. Built and deployed the containers.

5. Tested the endpoints using curl.

3. Implementation

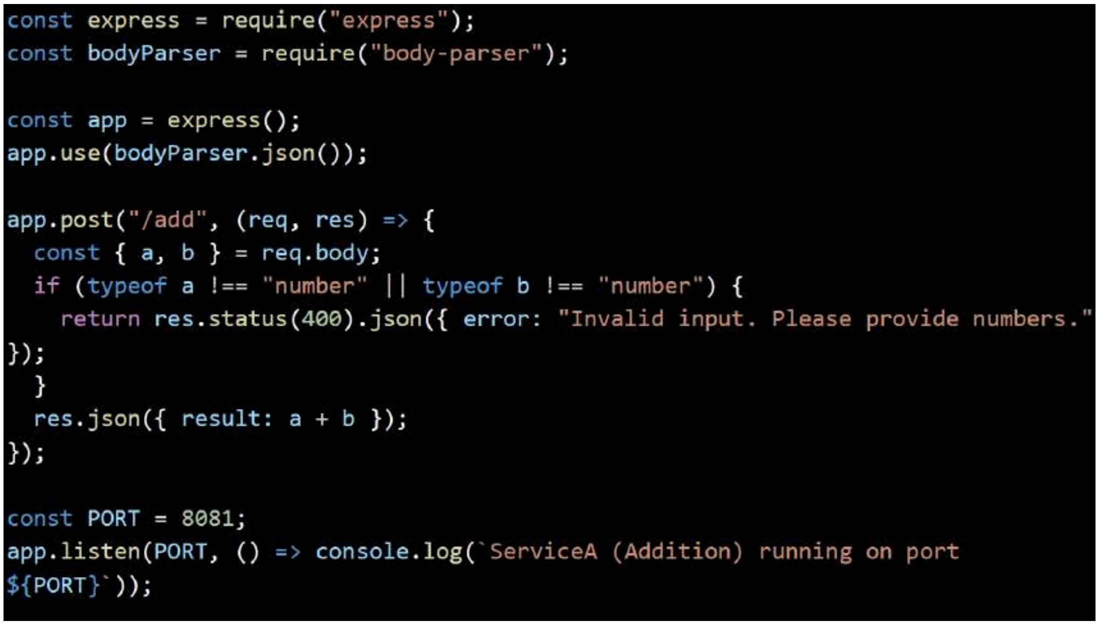
3.1. Service A: Addition Microservice

1. Create serviceA/package.json



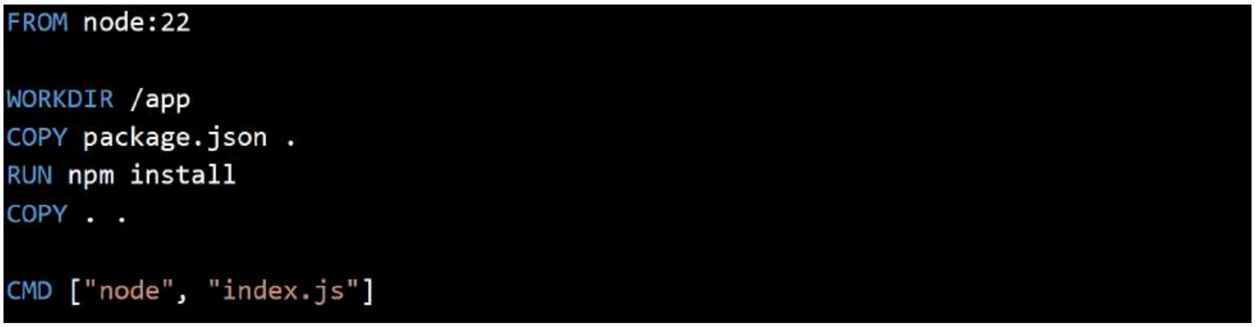
```
{
    "name": "servicea",
    "version": "1.0.0",
    "description": "Addition service",
    "main": "index.js",
    "dependencies": {
      "express": "^4.18.2",
      "body-parser": "^1.20.2"
    }
  }
```

2. Create serviceA/index.js const



```
const express = require("express");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.json());

app.post("/add", (req, res) => {
  const { a, b } = req.body;
  if (typeof a !== "number" || typeof b !== "number") {
    return res.status(400).json({ error: "Invalid input. Please provide numbers."
});
  }
  res.json({ result: a + b });
});

const PORT = 8081;
app.listen(PORT, () => console.log(`ServiceA (Addition) running on port
${PORT}`));
```

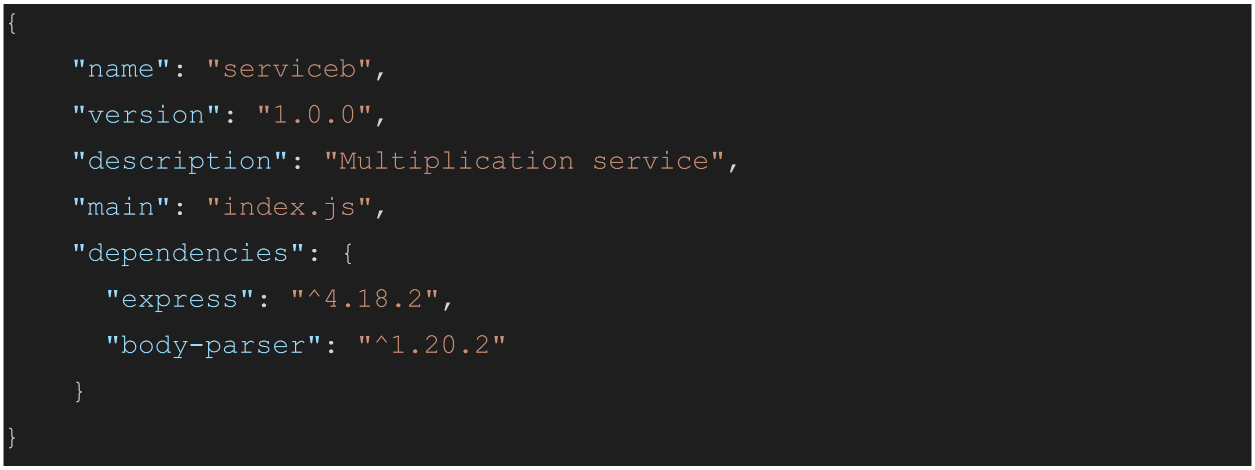
3. Create serviceA/Dockerfile



```
FROM node:22

WORKDIR /app
COPY package.json .
RUN npm install
COPY . .

CMD ["node", "index.js"]
```

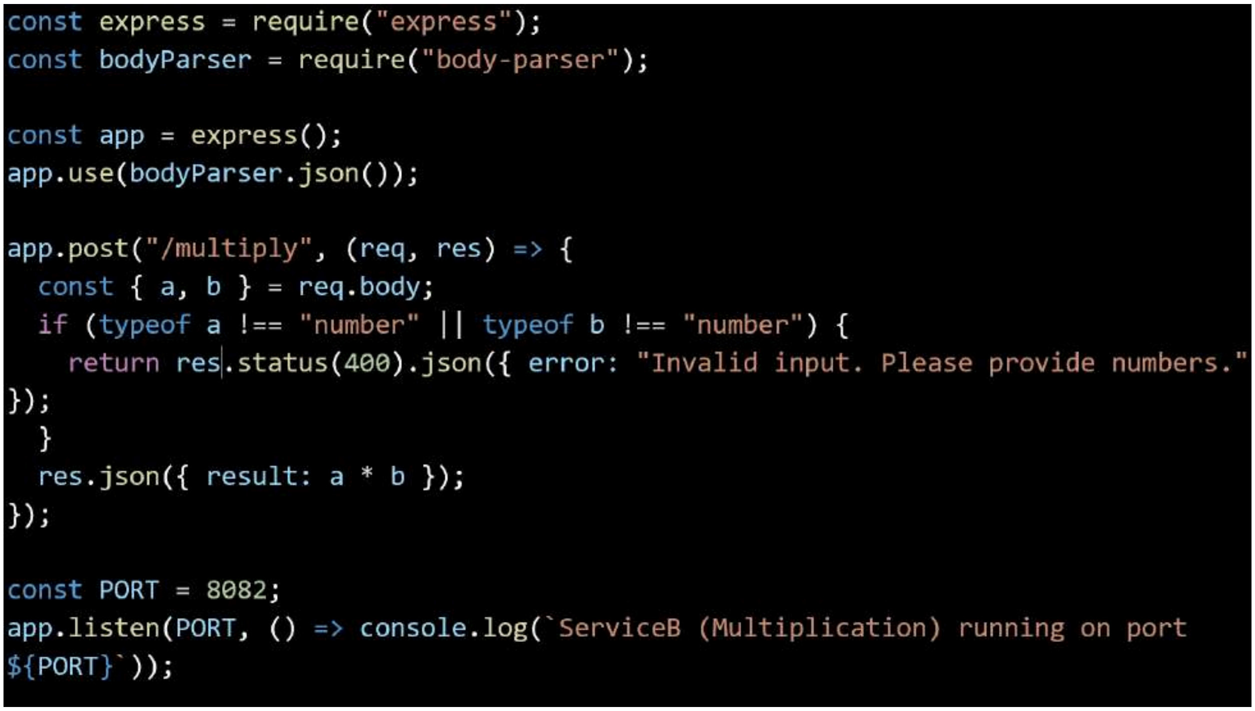
3.2. Service B: Multiplication Microservice

1. Create serviceB/package.json



```
{
    "name": "serviceb",
    "version": "1.0.0",
    "description": "Multiplication service",
    "main": "index.js",
    "dependencies": {
      "express": "^4.18.2",
      "body-parser": "^1.20.2"
    }
}
```

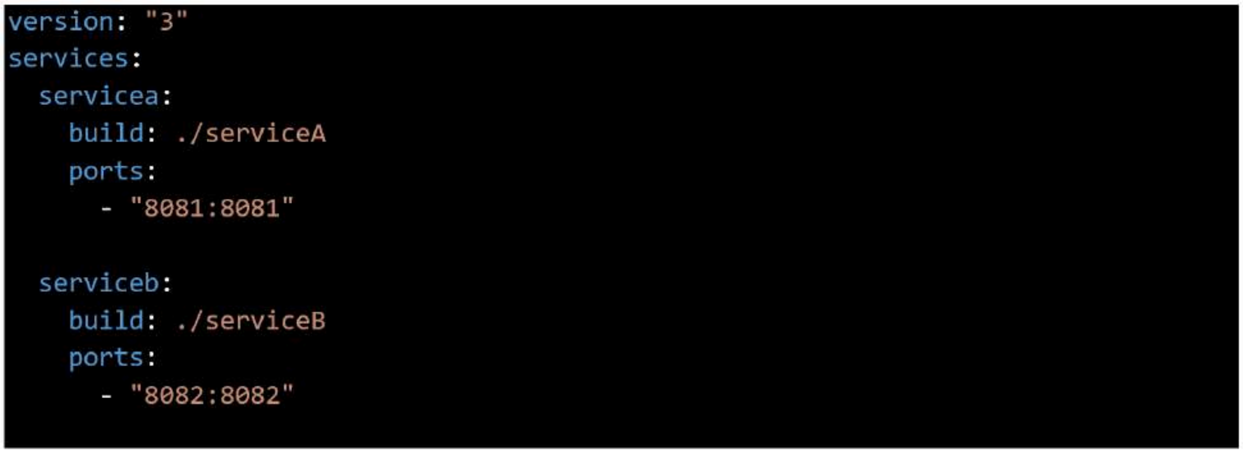
2. Create serviceB/index.js



```javascript
const express = require("express");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.json());

app.post("/multiply", (req, res) => {
  const { a, b } = req.body;
  if (typeof a !== "number" || typeof b !== "number") {
    return res.status(400).json({ error: "Invalid input. Please provide numbers."
});
  }
  res.json({ result: a * b });
});

const PORT = 8082;
app.listen(PORT, () => console.log(`ServiceB (Multiplication) running on port
${PORT}`));
```

3.3. Docker Compose Configuration

Create docker-compose.yml

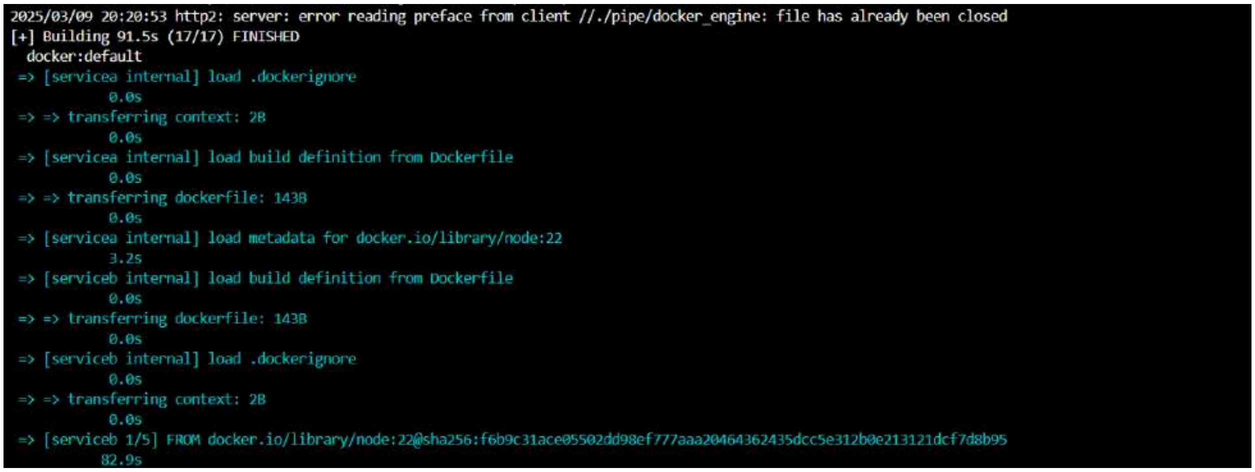


```yaml
version: "3"
services:
  servicea:
    build: ./serviceA
    ports:
      - "8081:8081"

  serviceb:
    build: ./serviceB
    ports:
      - "8082:8082"
```

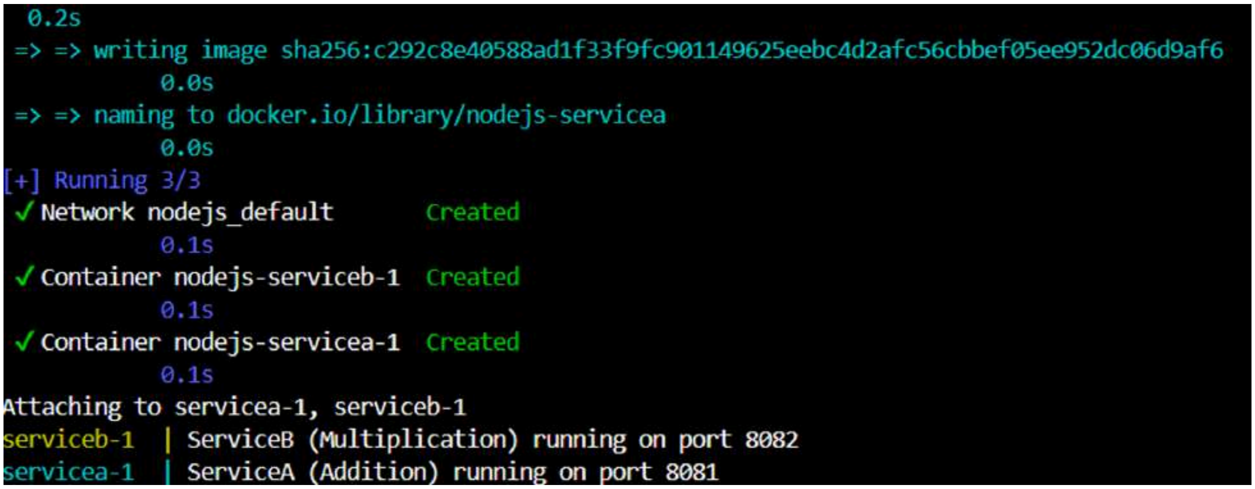
4. Deployment Steps

Step 1: Build and Start Containers

docker-compose up –build



```
2025/03/09 20:20:53 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 91.5s (17/17) FINISHED
  docker:default
 => [servicea internal] load .dockerignore
           0.0s
 => => transferring context: 2B
           0.0s
 => [servicea internal] load build definition from Dockerfile
           0.0s
 => => transferring dockerfile: 143B
           0.0s
 => [servicea internal] load metadata for docker.io/library/node:22
           3.2s
 => [serviceb internal] load build definition from Dockerfile
           0.0s
 => => transferring dockerfile: 143B
           0.0s
 => [serviceb internal] load .dockerignore
           0.0s
 => => transferring context: 2B
           0.0s
 => [serviceb 1/5] FROM docker.io/library/node:22@sha256:f6b9c31ace05502dd98ef777aaa20464362435dcc5e312b0e213121dcf7d8b95
          82.9s
```



```
  0.2s
 => => writing image sha256:c292c8e40588ad1f33f9fc901149625eebc4d2afc56cbbef05ee952dc06d9af6
           0.0s
 => => naming to docker.io/library/nodejs-servicea
           0.0s
[+] Running 3/3
 ✓ Network nodejs_default       Created
           0.1s
 ✓ Container nodejs-serviceb-1  Created
           0.1s
 ✓ Container nodejs-servicea-1  Created
           0.1s
Attaching to servicea-1, serviceb-1
serviceb-1  | ServiceB (Multiplication) running on port 8082
servicea-1  | ServiceA (Addition) running on port 8081
```

5. Testing the Services

5.1. Test Service A (Addition)

curl -X POST http://localhost:8081/add -H "Content-Type: application/json" -d "{\"a\":5, \"b\":3}"

Expected Response:

{"result":8}

```
{"result":8}
```

5.2. Test Service B (Multiplication)

curl -X POST http://localhost:8082/multiply -H "Content-Type: application/json" -d "{\"a\":5, \"b\":3}"

Expected Response:

{"result":15}

```
{"result":15}
```

Conclusion

This project successfully implemented two microservices using Node.js, deployed them in Docker containers, and managed them using Docker Compose. The services were tested using curl commands to verify their functionality.

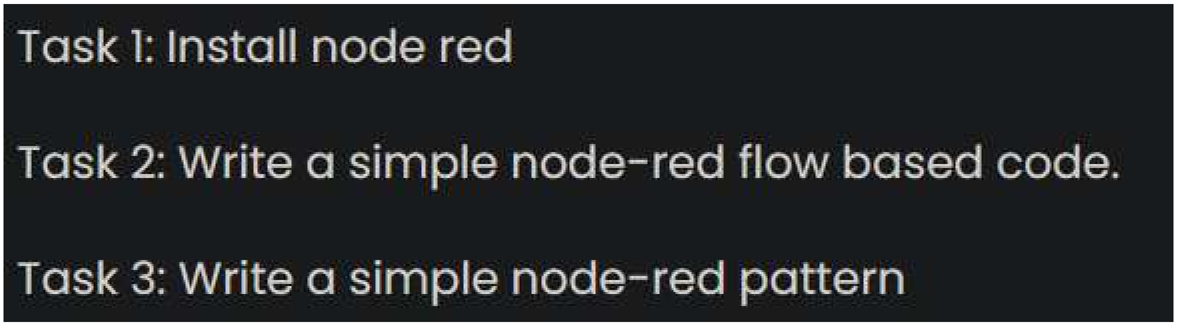
Through this implementation, we demonstrated how to create, containerize, and orchestrate microservices efficiently using Docker. This approach can be extended to build scalable and distributed applications.

Lab 10

ICS423 - Internet of Things

Jayant Kolapkar - 2021BCS0132

Question



Task 1: Install node red

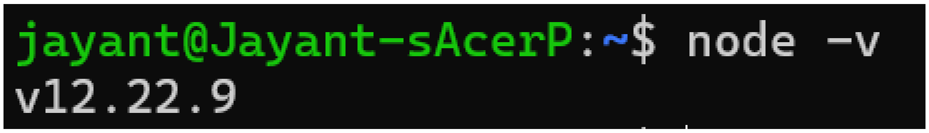
Task 2: Write a simple node-red flow based code.

Task 3: Write a simple node-red pattern

Task 1

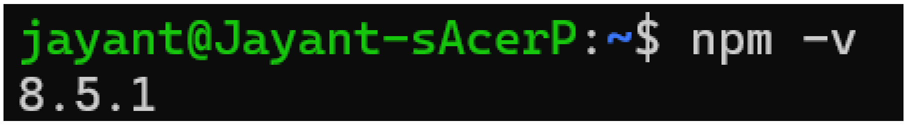
Algorithm

1. Install node.js



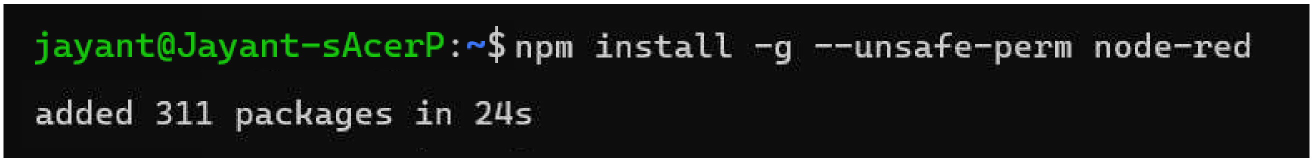
```
jayant@Jayant-sAcerP:~$ node -v
v12.22.9
```

2. Install npm



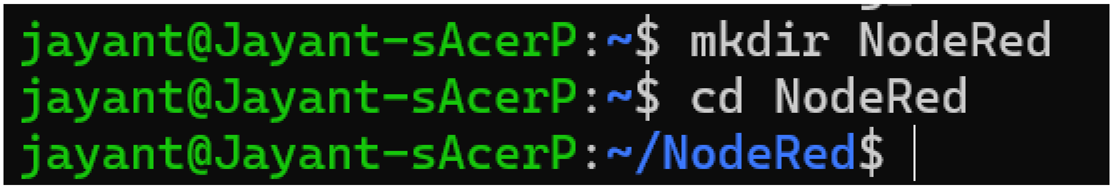
```
jayant@Jayant-sAcerP:~$ npm -v
8.5.1
```

3. Install node-red dependency



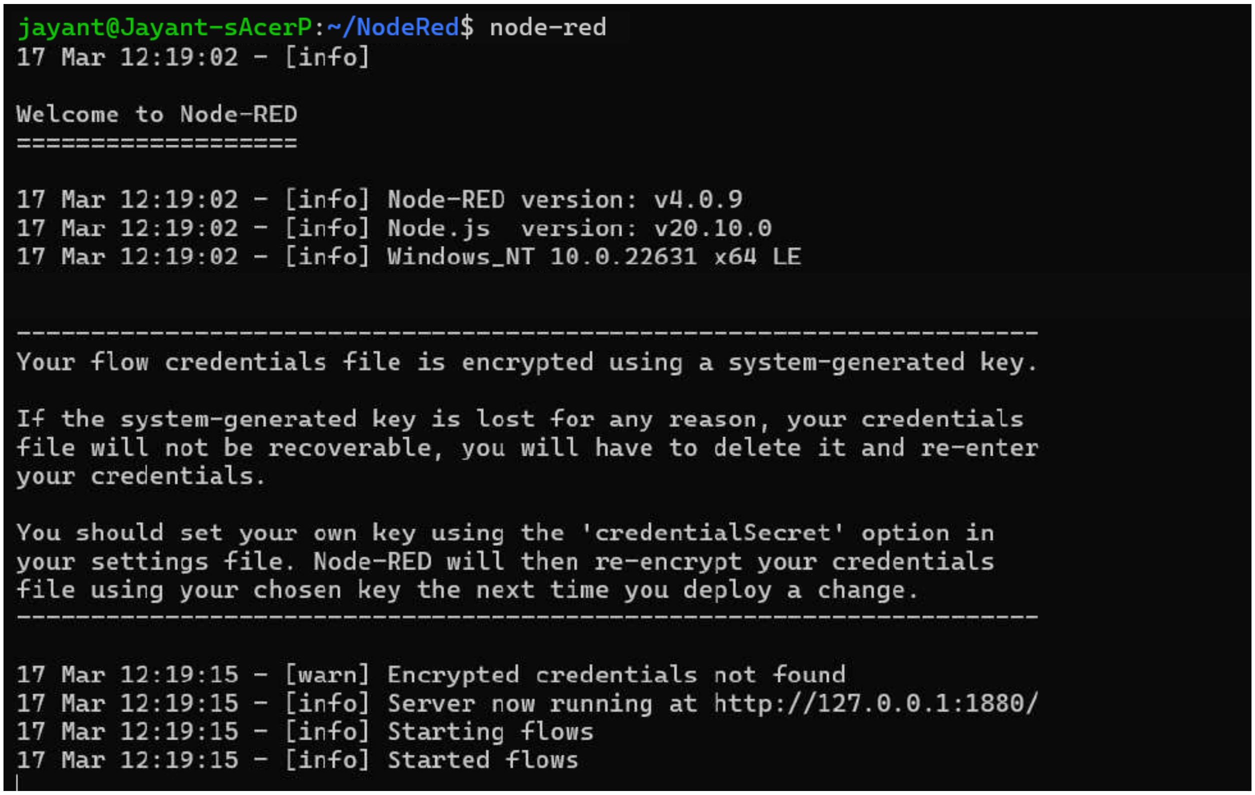
```
jayant@Jayant-sAcerP:~$ npm install -g --unsafe-perm node-red

added 311 packages in 24s
```

4. Create a folder for NodeRed



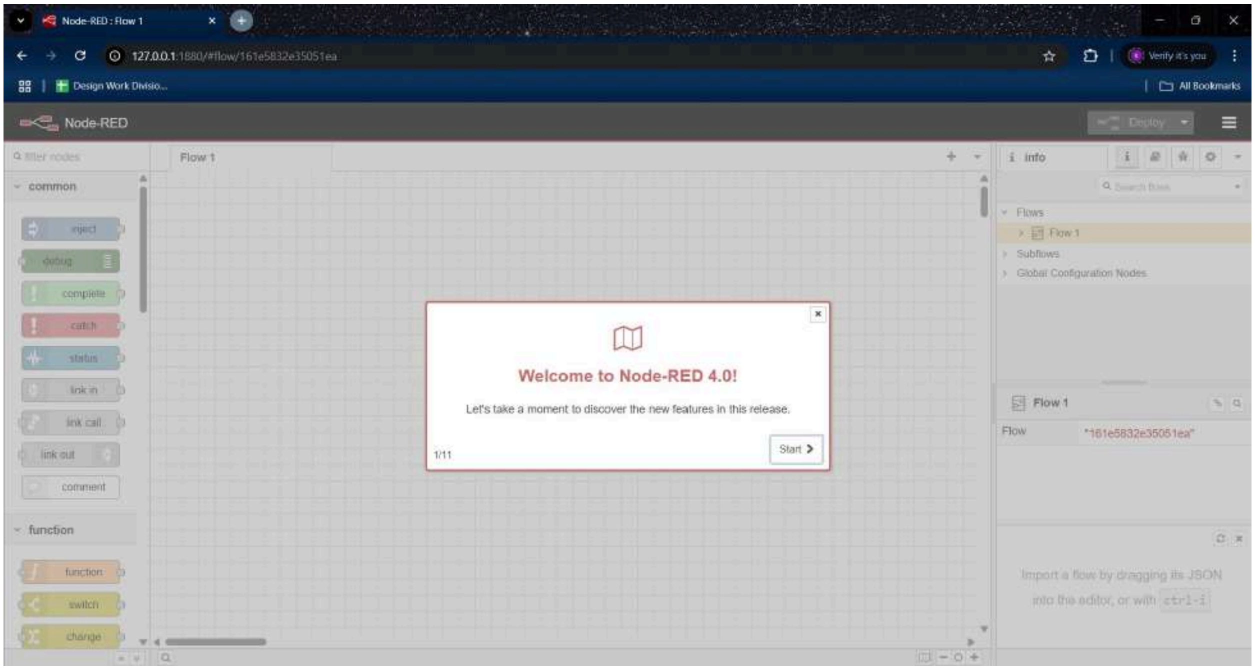
```
jayant@Jayant-sAcerP:~$ mkdir NodeRed
jayant@Jayant-sAcerP:~$ cd NodeRed
jayant@Jayant-sAcerP:~/NodeRed$
```

5. Start node-red using the node-red command. This will create a new flow.



```
jayant@Jayant-sAcerP:~/NodeRed$ node-red
17 Mar 12:19:02 - [info]

Welcome to Node-RED
===================

17 Mar 12:19:02 - [info] Node-RED version: v4.0.9
17 Mar 12:19:02 - [info] Node.js  version: v20.10.0
17 Mar 12:19:02 - [info] Windows_NT 10.0.22631 x64 LE


---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

17 Mar 12:19:15 - [warn] Encrypted credentials not found
17 Mar 12:19:15 - [info] Server now running at http://127.0.0.1:1880/
17 Mar 12:19:15 - [info] Starting flows
17 Mar 12:19:15 - [info] Started flows
```

6. Now open your browser and visit http://127.0.0.1:1880/. Your node-red will be running here.



Welcome to Node-RED 4.0!

Let's take a moment to discover the new features in this release.

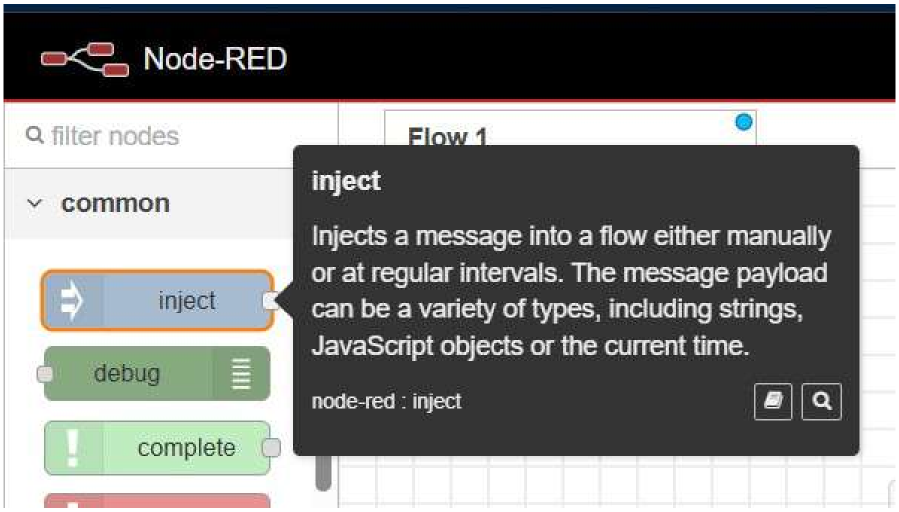
Import a flow by dragging its JSON into the editor, or with ctrl-i

Task 2

Algorithm

1. Open the node-red editor at http://127.0.0.1:1880/.

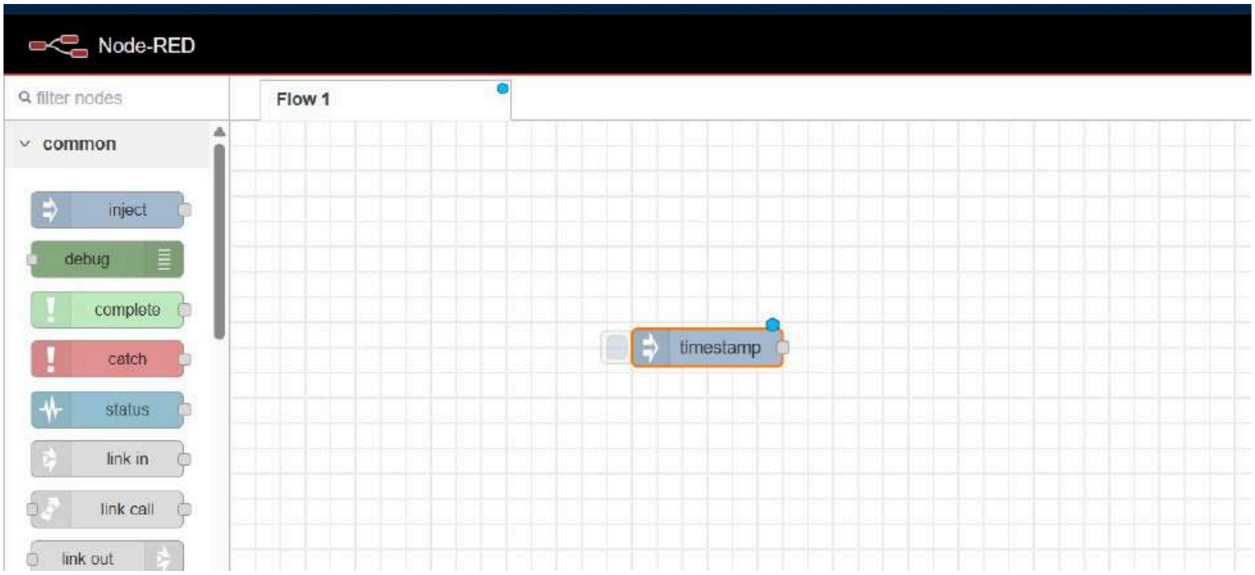
2. Drag inject node to trigger the flow manually.



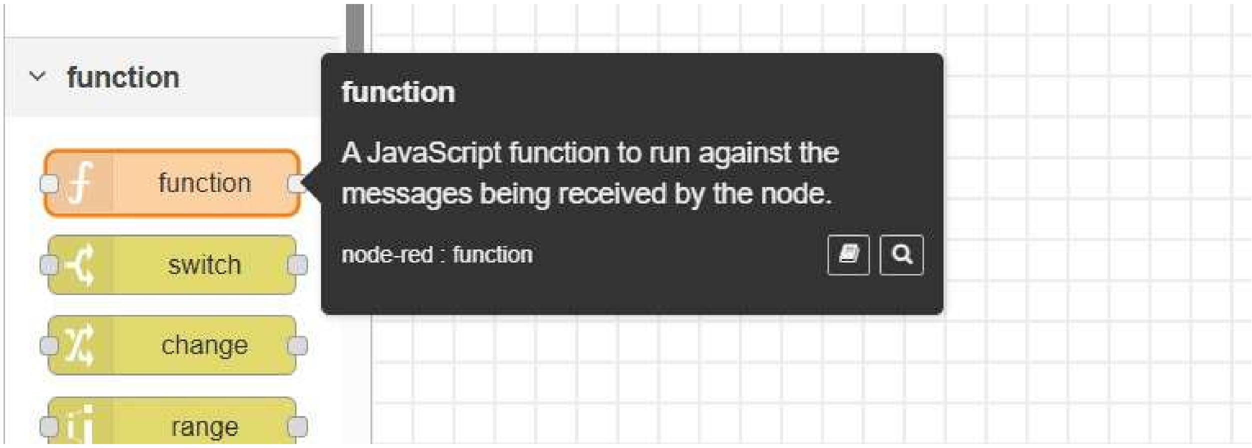
inject

Injects a message into a flow either manually or at regular intervals. The message payload can be a variety of types, including strings, JavaScript objects or the current time.

node-red : inject



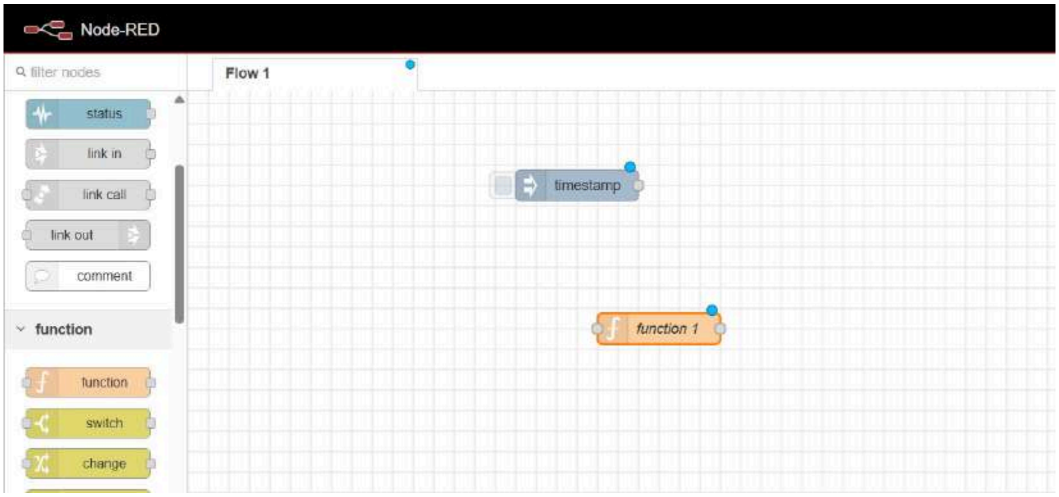
3. Drag a function node to process the data.



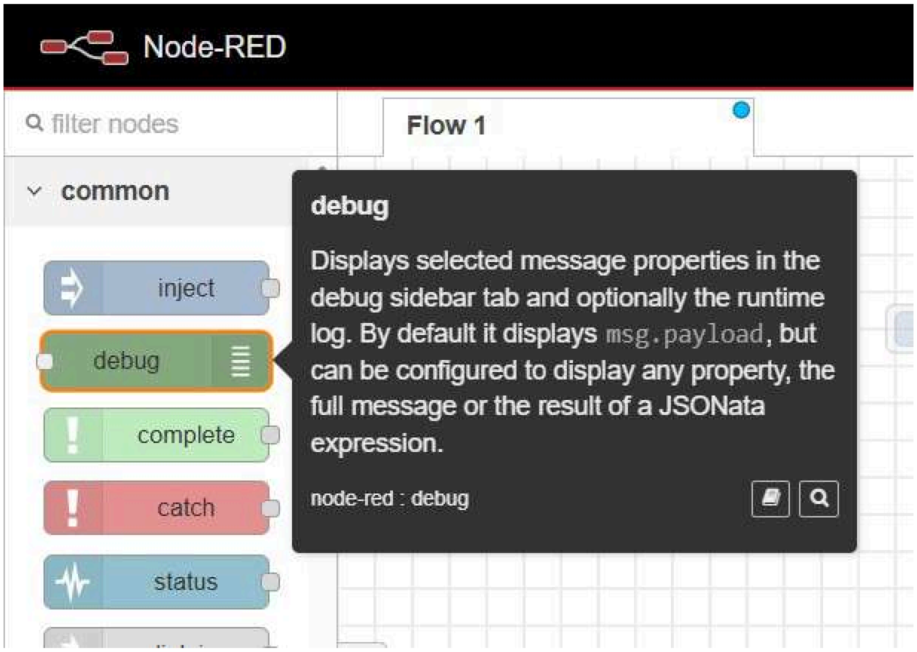
function

A JavaScript function to run against the messages being received by the node.

node-red : function



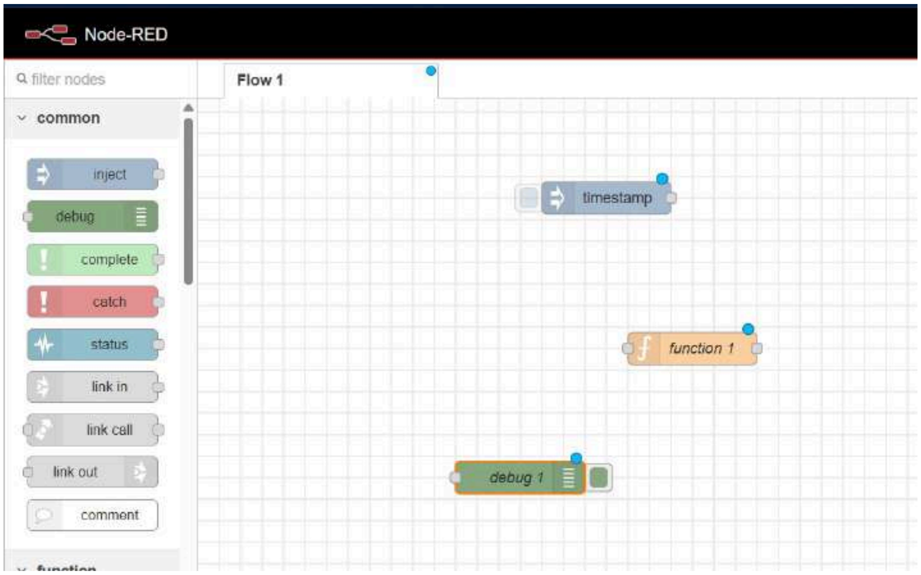
4. Drag a Debug node to log the output.



debug

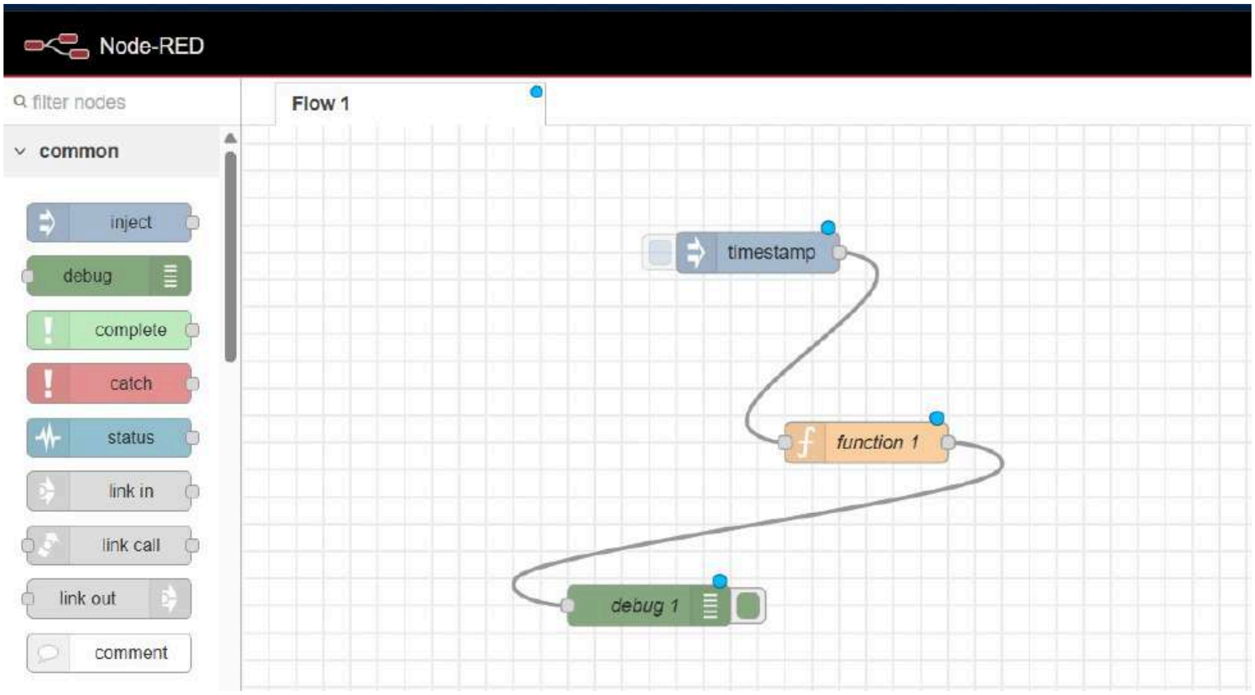
Displays selected message properties in the debug sidebar tab and optionally the runtime log. By default it displays `msg.payload`, but can be configured to display any property, the full message or the result of a JSONata expression.

node-red : debug



5. Connect the nodes in the following manner.

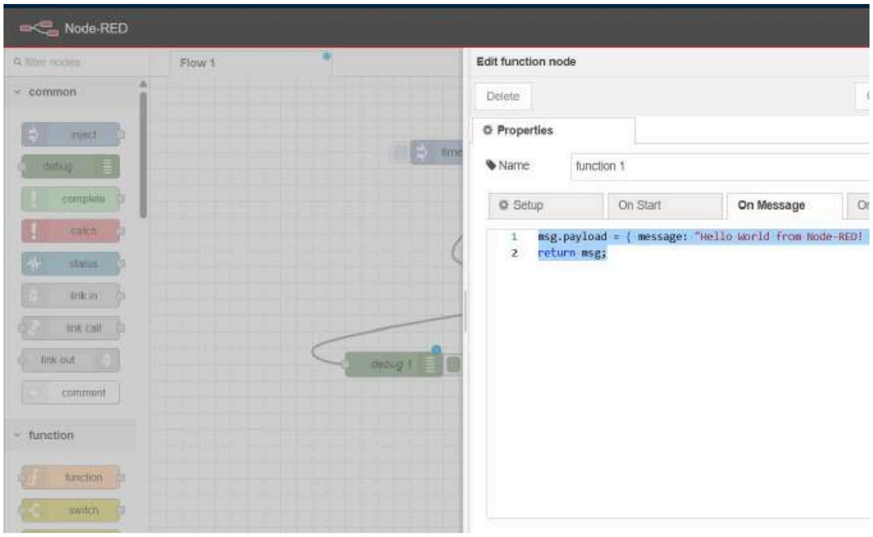
Inject -> Function -> Debug



6. Configure the function with the following js code.

Code

```
msg.payload = { message: "Hello World from Node-RED! ft.Jayant" };

return msg;
```

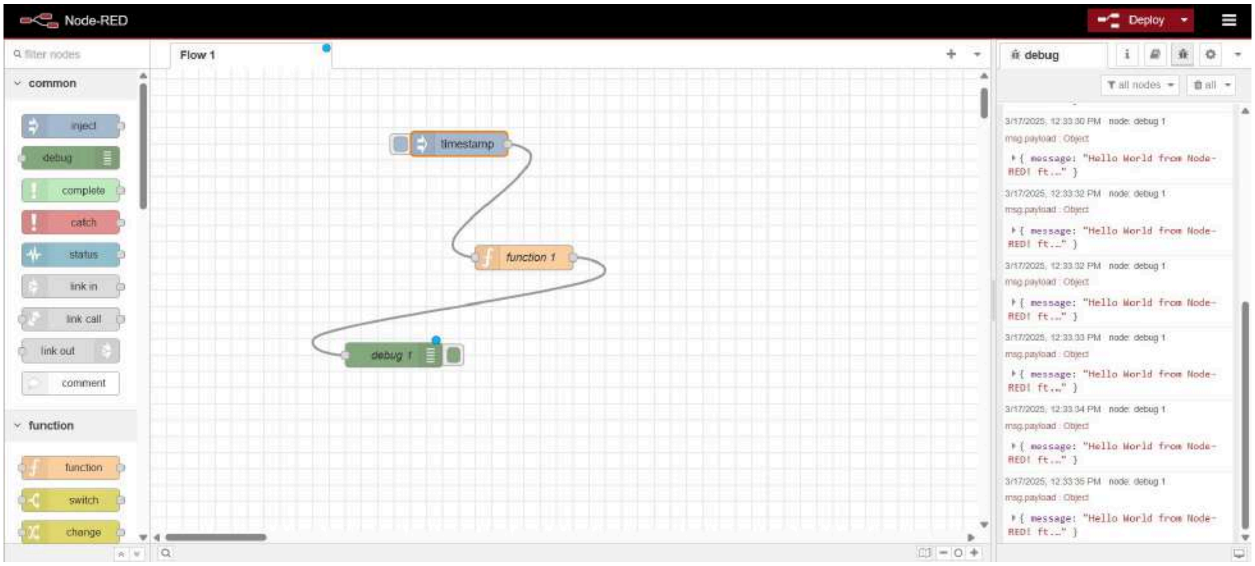


Edit function node

Name: function 1

```
msg.payload = { message: "Hello World from Node-RED!
return msg;
```

7. Click on the inject node to trigger the flow. Output can be viewed on the debug panel.



```
3/17/2025, 12:33:30 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
3/17/2025, 12:33:32 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
3/17/2025, 12:33:32 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
3/17/2025, 12:33:33 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
3/17/2025, 12:33:34 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
3/17/2025, 12:33:35 PM  node: debug 1
msg.payload : Object
▸{ message: "Hello World from Node-RED! ft..." }
```

Task 3

Algorithm

1. We will start with creating a flow for the pattern which processes data from a sensor.

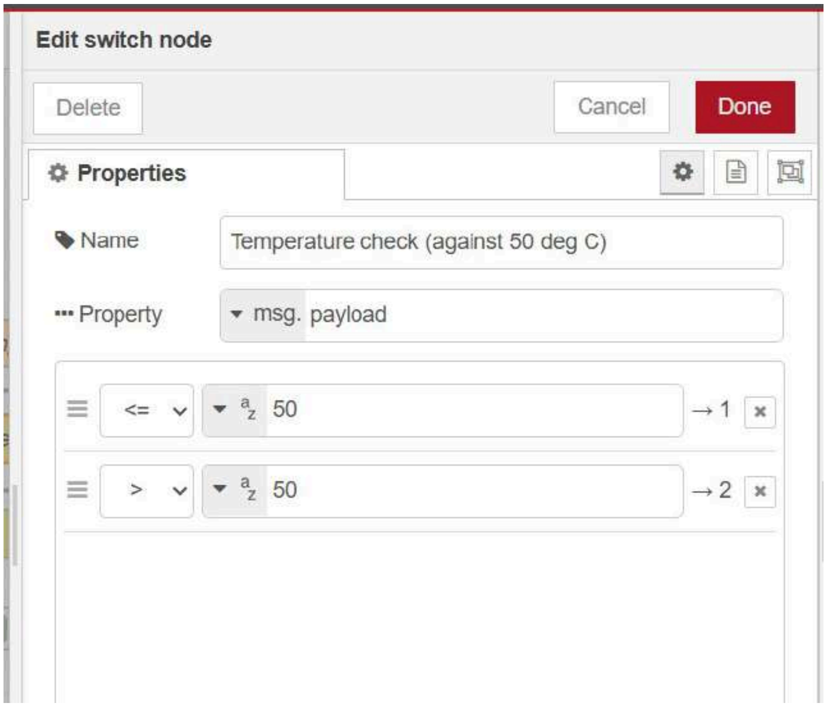
2. Flow will be:

- Inject node to simulate sensor data
- Random node to generate random temperature values
- Switch node to check if temp > 50°C
- Change node to format alert message
- Debug node to log the alert

3. The nodes will do the following job.

Random Temp Generator: Generates random temperature values between 10 and 80 deg C.

Switch node: Will check the temperature values and log based on the set threshold values (here 50 deg C).



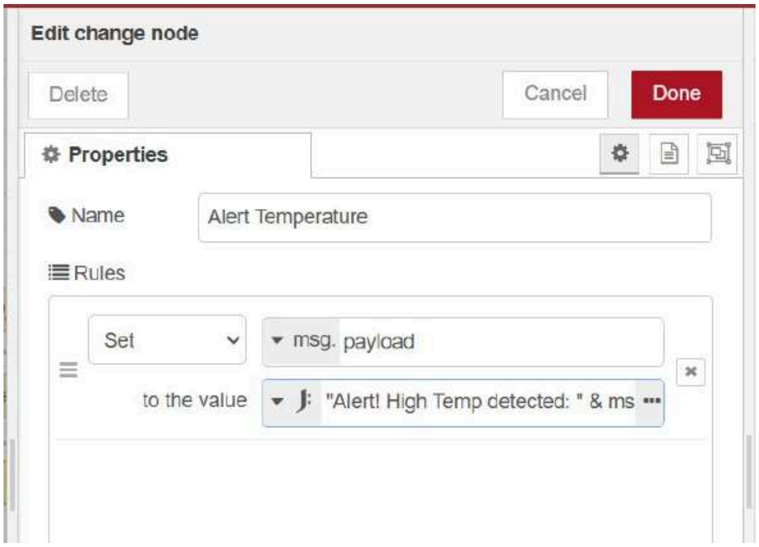
Name: Temperature check (against 50 deg C)

Property: msg. payload

<= 50 → 1

> 50 → 2

Change node: Sets msg.payload = "Alert! High Temp detected."



Name: Alert Temperature

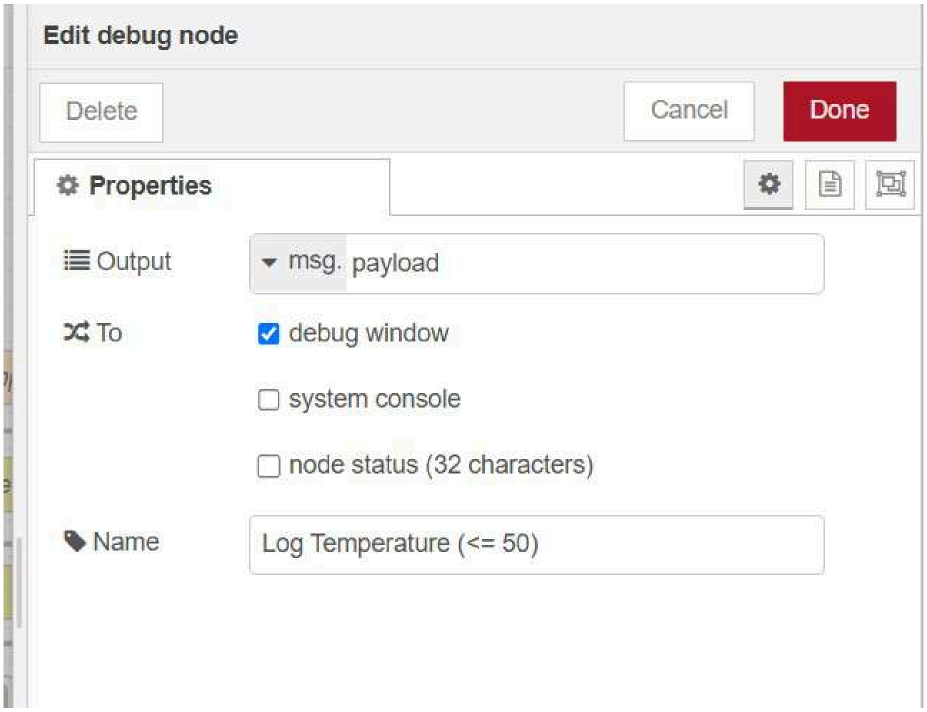
Rules

Set msg. payload

to the value "Alert! High Temp detected: " & ms

Debug Node:

Log Temperature: If temperature is <= 50 deg C.



Output: msg. payload

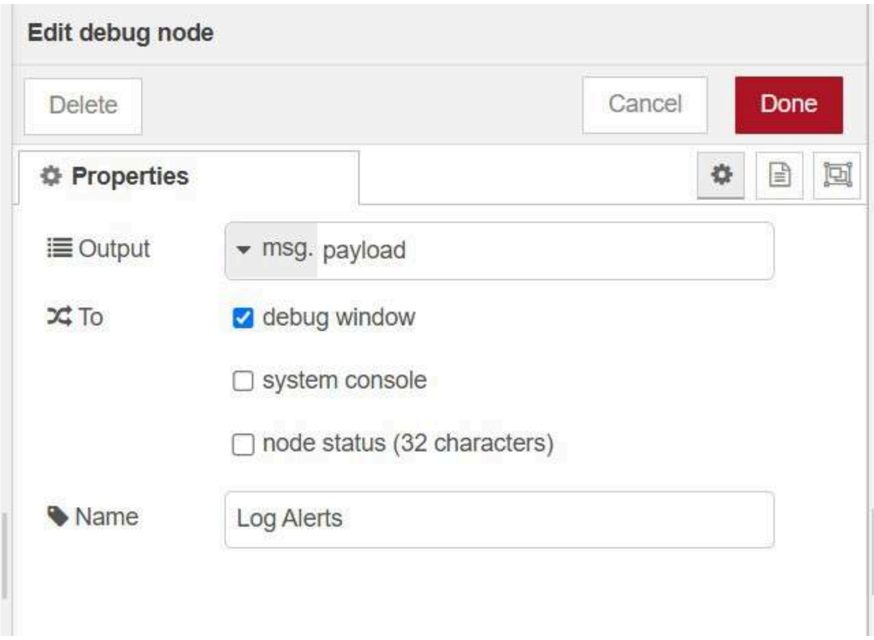
To: ☑ debug window

☐ system console

☐ node status (32 characters)

Name: Log Temperature (<= 50)

Alert: If Temperature is >50 degC.



Output: msg. payload

To: ☑ debug window

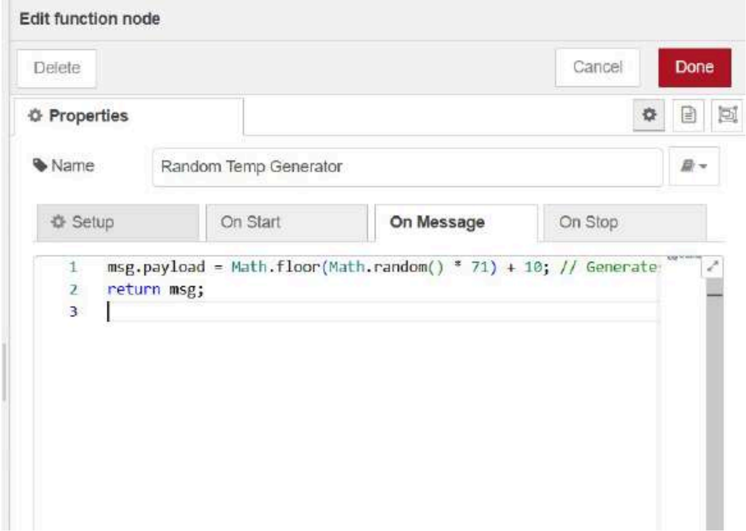
☐ system console

☐ node status (32 characters)

Name: Log Alerts

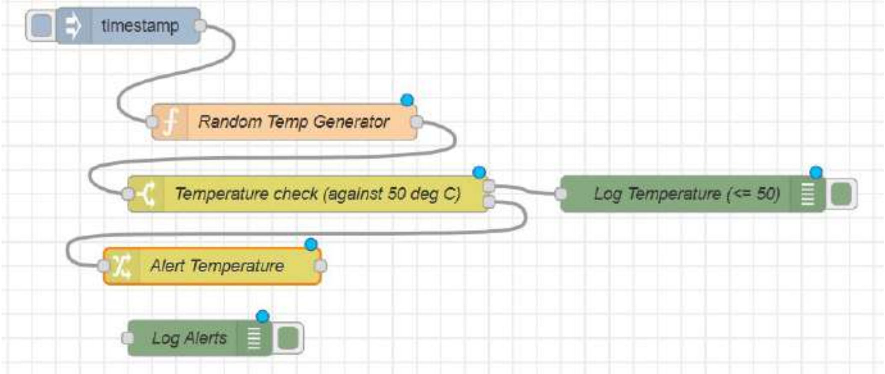
Code for Random Temp Generator function

```
msg.payload = Math.floor(Math.random() * 71) + 10; // Generates temp between 10°C - 80°C

return msg;
```



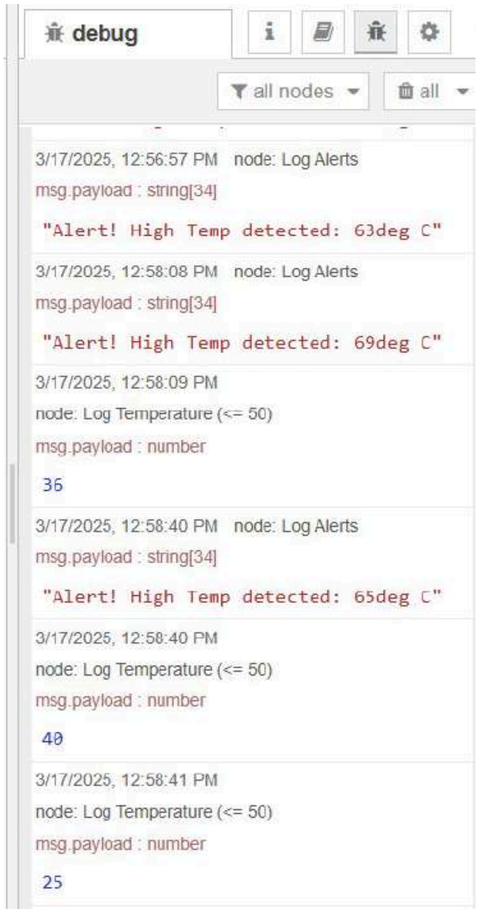
Name: Random Temp Generator

```
msg.payload = Math.floor(Math.random() * 71) + 10; // Generate
return msg;
```



4. Deploy and inject the timestamp node. Output will be visible in the debug mode.

Output



```
3/17/2025, 12:56:57 PM   node: Log Alerts
msg.payload : string[34]
"Alert! High Temp detected: 63deg C"
3/17/2025, 12:58:08 PM   node: Log Alerts
msg.payload : string[34]
"Alert! High Temp detected: 69deg C"
3/17/2025, 12:58:09 PM
node: Log Temperature (<= 50)
msg.payload : number
36
3/17/2025, 12:58:40 PM   node: Log Alerts
msg.payload : string[34]
"Alert! High Temp detected: 65deg C"
3/17/2025, 12:58:40 PM
node: Log Temperature (<= 50)
msg.payload : number
40
3/17/2025, 12:58:41 PM
node: Log Temperature (<= 50)
msg.payload : number
25
```