SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2628001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE CODE: 22CS702

INTERNET OF EVERYTHING LABORATORY

Continuous Assessments Record

Submitted by

NAME	
REGISTER NO	
YEAR & SEMESTER	
DEPARTMENT	**Computer Science and Engineering**
SECTION	
ACADEMIC YEAR	**2025-2026 (ODD Semester)**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2628001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

Department of Artificial Intelligence and Data Science

22CS702 – INTERNET OF EVERYTHING LABORATORY

Submitted by

Register No.:

Name :

Degree : **Branch :**

BONAFIDE CERTIFICATE

This is to certify that this is a Bonafide record work by Mr./Ms……Register No:…… during the academic year 2025 – 2026.

Faculty In-charge

Submitted for the Autonomous Practical Examination held on

…….....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2628001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

Department of Artificial Intelligence and Data Science

VISION OF THE INSTITUTE

To Produce Globally Competitive Engineers with High Ethical Values and Social Responsibilities.

MISSION OF THE INSTITUTE

- To impart highest quality state-of-the-art technical education by providing impetus to innovation, Research and Development and empowering students with Entrepreneurship skills.
- To instill ethical values, imbibe a sense of social responsibility and strive for societal wellbeing.
- To identify needs of society and offer sustainable solutions through outreach programs.

VISION OF THE DEPARTMENT

To produce globally competitive professionals in Artificial Intelligence and Data Science by imparting cognitive learning and encouraging industry collaboration towards serving the greater cause of society.

MISSION OF THE DEPARTMENT

- Impart knowledge in cutting edge Artificial Intelligence and Data Science technologies in par with industrial standards.
- Inculcate research and lifelong learning that benefit society at large.
- Promote ethical values and entrepreneurial skills

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

The following Programme Educational Objectives are designed based on the department mission

PEO 1	To build a successful career in IT/relevant industry or carryout research in advance areas of Artificial Intelligence, Data Science and address various issues in the society.
PEO 2	To develop problem solving skills and ability to provide solution for real time problems.
PEO 3	To develop the ability and attitude of adapting themselves to emerging technological Challenges.
PEO 4	To excel with excellent communication skills, leadership qualities and social responsibilities.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

On successful completion of Bachelor of Engineering in Electronics and Communication Engineering Programme from Sri Krishna College of Engineering and Technology, the graduate will demonstrate:

PSO 1: Understand, analyze and develop innovative solutions for real world problems in industry and research establishments related to Artificial Intelligence and Data Science.

PSO 2: Ability to choose or develop the right tool for Data analysis and develop high end intelligent systems.

PSO 3: Apply programming principles and practices for developing software solutions to meet future business and society needs.

PROGRAMME OUTCOMES (POs)

At the time of their graduation students of Electronics and Communication Engineering Programme should be in possession of the following Programme Outcomes

PO**1**.**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO**2. Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO**3. Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO**4. Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO**5. Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO**6. The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO**7**. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO**8. Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO**9**. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO**10**. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO**11**. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SYLLABUS

Ex. No	Description	Course Outcome	Level of Bloom's Taxonomy
1	Introduction to Arduino platform and programming.	C702.1	U
2	Interfacing sensors and actuators to Arduino	C702.1	U
3	Introduction to Raspberry PI platform and python programming.	C702.2	AP
4	Interfacing sensors and actuators to Raspberry	C702.2	AP
5	Setup a cloud platform to log the data	C702.3	AP
6	Log Data using Raspberry PI and upload to the cloud platform	C702.3	AP
7	Design an IOT based system.	C702.4	AP
8	Develop the Vehicle Parking System in Smart Cities using loE	C702.5	A
9	Develop an loE-Enabled Healthcare System for Remote and Near Patient Monitoring.	C702.5	A
10	Develop an IOE application to handle data acquisition with integration of the output devices	C702.5	A

Text Books:

1. Peter Waher, "Mastering Internet of Things: Design and create your own loT applications using Raspberry Pi 3", First Edition, Packt Publishing, 2020.
2. Marco Schwartz, "Internet of Things with Arduino Cookbook", Packt Publishing, 2019.
3. Adrian McEwen, "Designing the Internet of Things", Wiley Publishers, 2021.

Reference Books:

1. Vandana Roy, Piyush Kumar, Anoop Kumar Chaturvedi, Priti Maheswary, "Internet of Everything for Biomedical Applications", CRC Press, 2021.
2. Gaston C. Hillar "Internet of Things with Python", Packt Publishing, 2016.

Web References:

1. www.ptc.com > Internet of Things (IoT)
2. http://wwwusers.di.uniroma1.it/~spenza/files/labloT2015/Lab-loT-1.pdf
3. https://onlinecourses.nptel.ac.in/noc22_cs53/preview

Online Resources:

1. http://www.iotlab.eu/
2 http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

Expected outcome of the course:

Upon successful completion of this course, the student will be able to:

C702.1	Attain the fundamental knowledge on the loT platform.	U
C702.2	Interface sensors and actuators with Raspberry Pi and Arduino and test its functionalities.	AP
C702.3	Create cloud connectivity with loT devices.	AP
C702.4	Implement an loT based real-time system.	AP
C702.5	Analysis and report the sensitive application using loE.	A

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
An Autonomous Institution | Approved by AICTE | Affiliated to Anna University | Accredited by NAAC with A++ Grade
Kuniamuthur, Coimbatore – 641008
Phone : (0422)-2628001 (7 Lines) | Email : info@skcet.ac.in | Website : www.skcet.ac.in

Department of Artificial Intelligence and Data Science

22CS702 – INTERNET OF EVERYTHING LABORATORY (EVEN SEMESTER: 2024-2025)

Components	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8	EX9	EX10
DATE										
Objective & Components Required (20)										
Program / Connection Diagram (30)										
Simulation & Execution (30)										
Result (10)										
Documentation & Viva (10)										
Consolidated Mark (100)										
Faculty Signature										
Average (100)					Signature of the Faculty					

TABLE OF CONTENTS				
Exp. No.	**Date**	**EXPERIMENT LIST**	**PAGE NO.**	**Sign**
1		Introduction to Arduino platform and programming		
2		Interfacing sensors and actuators to Arduino		
3		Introduction to Raspberry PI platform and python programming		
4	_	Interfacing sensors and actuators to Raspberry		
5		Setup a cloud platform to log the data		
6		Log Data using Raspberry PI and upload to the cloud platform		
7		Design an IOT based system		
8		Develop the Vehicle Parking System in Smart Cities using loE		
9		Develop an loE-Enabled Healthcare System for Remote and Near Patient Monitoring		
10		Develop an IOE application to handle data acquisition with integration of the output devices		
	Average (100)			

Faculty Signature

CONTINUOUS EVALUATION RUBRICS SHEET

22CS702 INTERNET OF EVERYTHING LABORTORY

Items	Excellent	Good	Satisfactory	Needs Improvement
Objective & Components Required (20)	**10 Marks**	**(9-7) Marks**	**(6-4) Marks**	**(3-1)Marks**
	The aim and purpose of the experiment are clearly defined. Preliminary questions are to be answered. The equipment's required are to be clearly listed with specifications.	Able to define the aim and purpose of the experiment but not clearly. Preliminary questions are answered but not in detail. The equipment's required are clearly listed but with inappropriate specifications.	The aim and purpose of the experiment are defined but not clear. Few Preliminary questions are unanswered. The equipment's required are clearly listed but without specifications.	Unable to explain the aim and purpose of the experiment. Preliminary questions are unanswered. The equipment's required are not clearly listed.
Program / Connection Diagram (30)	**20Marks**	**(19-17) Marks**	**(16-14) Marks**	**(13-11) Marks**
	Able to draw Connection diagram neatly with no errors. Formulas required are written and used fully in computation. Model graphs are provided with necessary information.	Connection diagram of the experiment is provided with minor mistakes. Formulas required are written. Model graphs are provided with few missing data.	Connection diagram not drawn legibly and with fewer specifications of the components involved Some mistakes in formula. Model graph is not clear.	Connection diagram of the experimental setup are given wrongly. No model graph provided.
Simulation & Execution (30)	**40Marks**	**(39-37) Marks**	**(36-34) Marks**	**(33-31) Marks**
	The components are identified and connections done without error. Experimentation is as required. Readings taken are justified and tabulated.	The components are identified and connections done without error. The method adopted is relevant but the measurements made are partial. Readings are tabulated.	The components are identified but the connection done with few mistakes. The measured values are deviated.	Poorly experimentation and the method adopted is not relevant to the stated objective.

Result (10)	**20Marks**	**(19-17) Marks**	**(16-14) Marks**	**(13-11) Marks**
	Readings/measurements are utilized to draw necessary charts/graphs. The results are interpreted and compared with desired values successfully.	Almost all of the results have been correctly recorded and summarized; only minor improvements are needed in post lab discussion.	Some mistakes in tables and graph. Conclusions drawn from the results are not clear.	Experimental measurements are incorrect and wrongly interpreted. Not able to take a measurements and proceed further.
Documentation & Viva (10)	**10 Marks**	**(9-7)Marks**	**(6-4)Marks**	**(3-1)Marks**
	The aim, procedure, circuit diagram, experiment details are well documented.	Able to present the aim, procedure, circuit diagram and experiment details to some extent.	Not able to draw and tabulate the content and organize properly.	Incomplete work and Poor writing presentation.

Signature of Evaluator

Ex No: 1

Date:

INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING

AIM:

To study about the installation procedure of "Arduino Uno" and verifying the successful installation.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	Off board LED	2
5	Jumper Wire	4
6	PC with Windows OS -32/64 bit	1
7	Resistor (220Ω Each)	2
8	Bread Board	1

CONFIGURATION OF ARDUINO UNO & DIY-JRM TECH'S ARDUINO UNO:

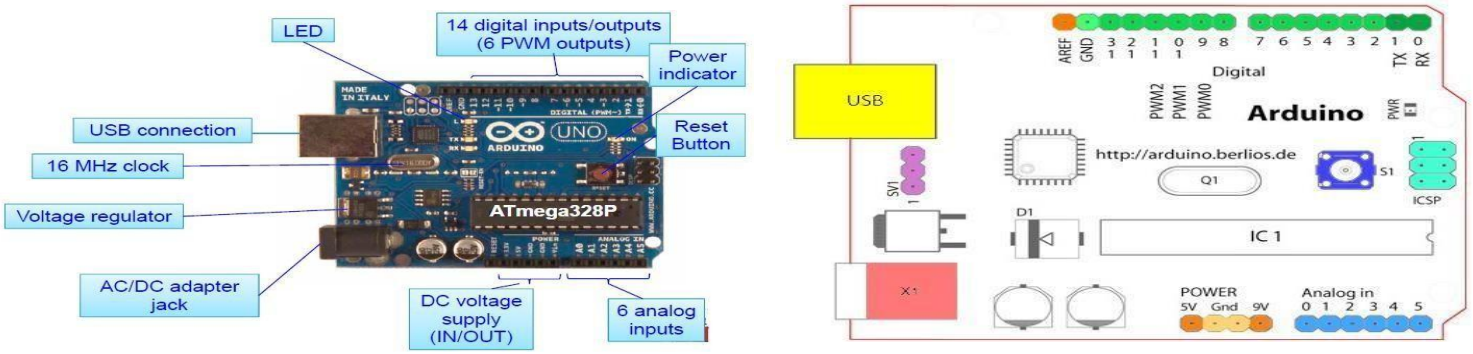


Fig.1: Standard Arduino UNO

- Digital I/O Pins 0 –13;
    - 0 & 1 (Dark Green) can be Serial In/Out – Tx/Rx; These pins cannot be used for digital I/O *(digitalRead and digitalWrite)* if we are also using serial communication (e.g. Serial.begin)
- Analog In Pins 0-5 (light blue)
- ISP– InSystemProgrammingalso called In-circuit serialprogramming(ICSP)
- External Power Supply In (9-12VDC) – X1 (pink)
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board)

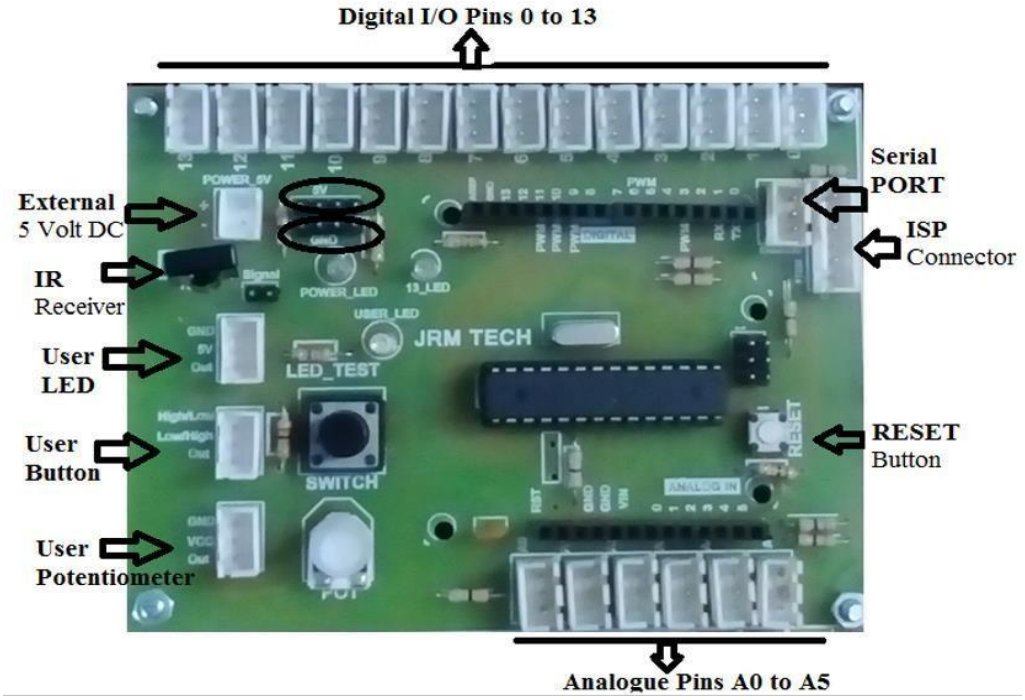


Fig.2: DIY – JRM Arduino Uno

- One Onboard LED 13th Pin
- One User LED
- One User Button-Switch
- One Potentiometer
- IR Receiver

INSTALLATION PROCEDURE:

1. Download & install the Arduino environment (IDE)
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Open the blink example (for checking)
8. Upload the program

Task #1

Blink the onboard LED in Uno

Program:

```
void setup()
{
    pinMode(13, OUTPUT);
// initialize digital pin 13 as an output.
}

void loop()
{
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

OUTPUT:

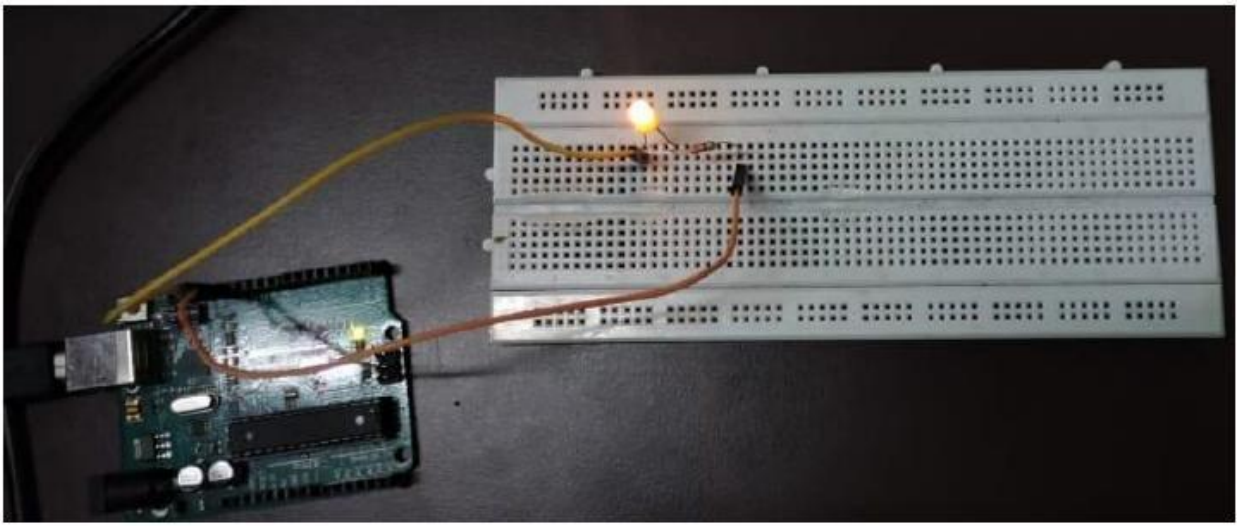
Task #2:

Blink the user LED in Uno using the port D8

Program:

```
void  setup()
{
pinMode(8, OUTPUT);
}
void loop()
{
digitalWrite(8,HIGH);
delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(8,LOW);
delay(1000); // Wait for 1000 millisecond(s)
}
```

OUTPUT:



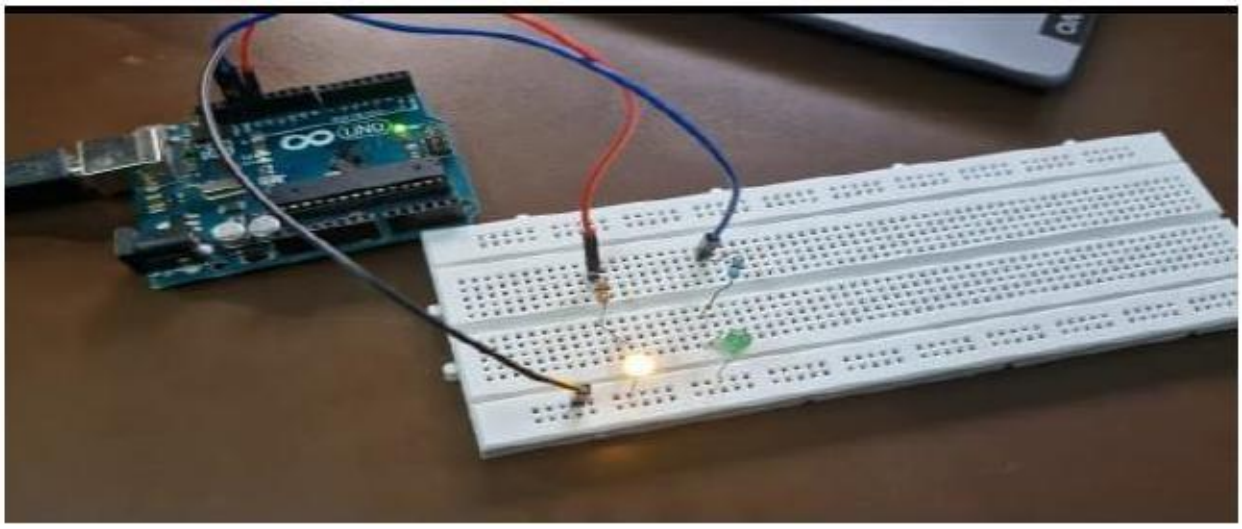
Task #3:

Blink the Two user LED in Uno using the port D8,D9

Program:

```
void setup() {
  pinMode(8, OUTPUT);
  pinMode(9,  OUTPUT);
}
void loop() {
  digitalWrite(8, HIGH);
  digitalWrite(9, LOW);
  delay(1000);
  digitalWrite(8, LOW);
  digitalWrite(9, HIGH);
  delay(1000);
}
```

OUTPUT:



RESULT:

Thus, the installation of Arduino Uno is done and working process is verified with the help of “blinking LED programs”.

Ex No: 02

Date:

INTERFACING SENSORS AND ACTUATORS ARDUINO

AIM:

To interface sensors and actuators with Arduino to acquire data from the environment and control devices, enabling the implementation of automated systems.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Arduino Uno	1
2	USB 2.0 – Mini B Type	1
3	Power Adaptor – 12 V	1
4	Temperature Sensor (LM35)	1
5	DC Motor/Servo Motor	1
6	Buzzer	1
7	PC with Windows OS -32/64 bit	1
8	Resistor (220Ω Each)	2
9	Jumper Wires	As Required
10	Bread Board	1

THEORY:

SENSOR:

In simple words, a sensor is a device that "listens" to the physical environment and tells you what happens. Each sensor is able to listen to a specific input, that can be light, heat, motion, moisture, pressure, or any one of a great variety of other environmental phenomena.

ACTUATOR:

In the strict meaning, actuator is a device that converts energy in movement. It could be a valve or a motor. I consider an actuator anything that can convert electric energy in an output, for example a display, a led, a loudspeaker.

DIGITAL AND ANALOG:

- Information may be represented and transmitted both in analogic or digital format. The difference between them is that in analogic technology information is represented by amplitude variations, while in digital it's represented in binary format.
- Arduino can do analog and digital inputs, and only digital outputs. The sensors themselves usually are analog, but most of times they have a small circuit that translates values into digital format. In some cases, you'll find both connectors on the same device.
- So, analog pins on Arduino are always input. Digital pins can be input or output, your code will tell which pin should be choose.

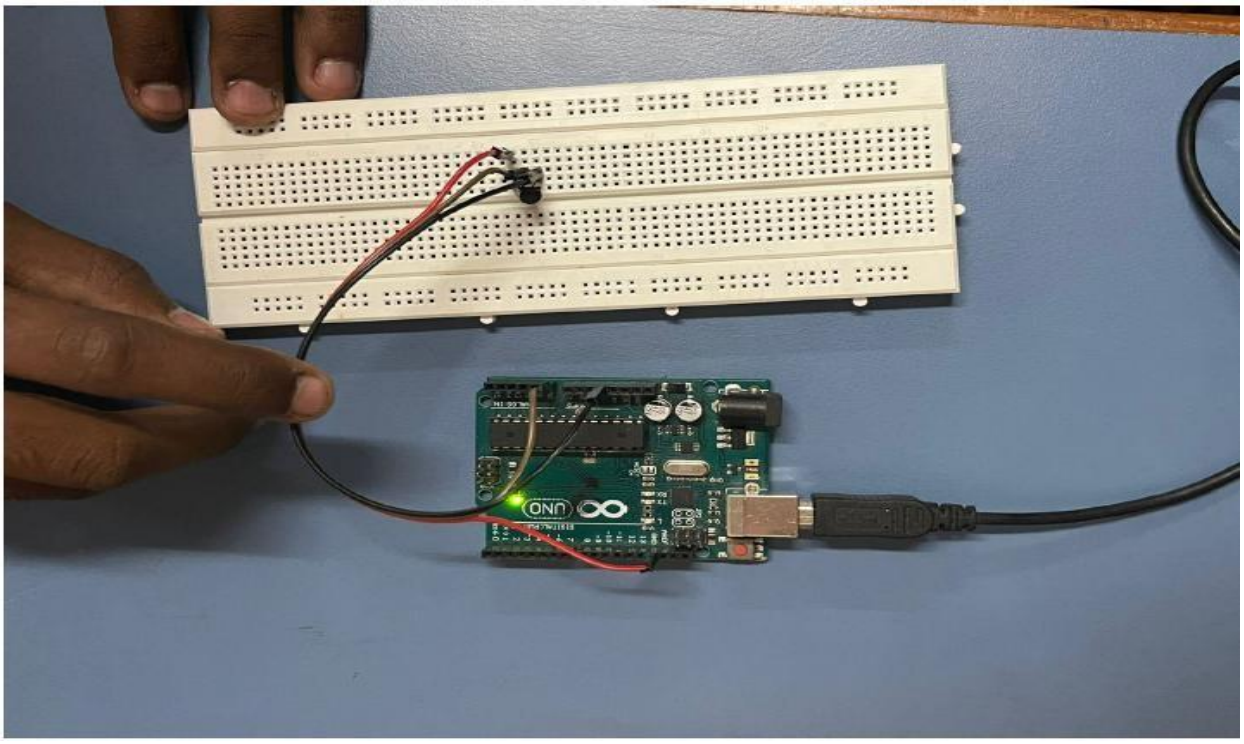
Task #1

To interface a temperature sensor (e.g., LM35) with Arduino

Program:

```
const int sensorPin = A0;
void setup() {
Serial.begin(9600);
  Serial.println("Temperature Sensor Interface Starting...");
}
void loop() {
  int sensorValue = analogRead(sensorPin);
  float voltage = sensorValue * (5.0 / 1023.0);
  float temperature = voltage * 100.0;
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  delay(1000);
}
```

OUTPUT:



Task #2

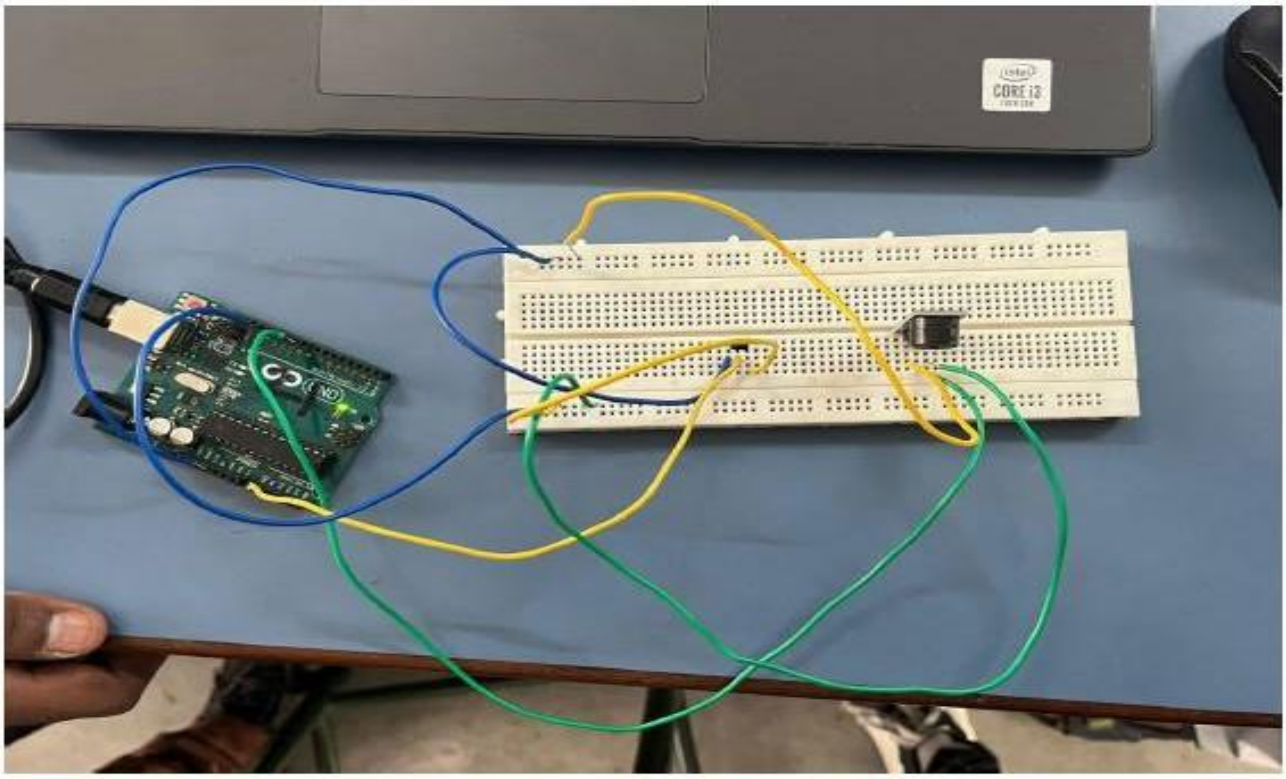
Interfacing LM35 Temperature Sensor and Buzzer with Arduino to Implement a Threshold-based Alert System

Program:

```
const int tempSensorPin = A0;
const int buzzerPin = 9;
const float thresholdTemp = 40.0;
void setup() {
pinMode(tempSensorPin, INPUT);
pinMode(buzzerPin, OUTPUT);
Serial.begin(9600);
}
void loop() {
  int analogValue = analogRead(tempSensorPin);
  float voltage = (analogValue / 1023.0) * 5.0;
  float temperature = voltage * 100.0;
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
   if (temperature > thresholdTemp) {
    digitalWrite(buzzerPin, HIGH);
  } else {
    digitalWrite(buzzerPin,  LOW);
}
```

```
  delay(1000);
}
```

OUTPUT:



Task #3

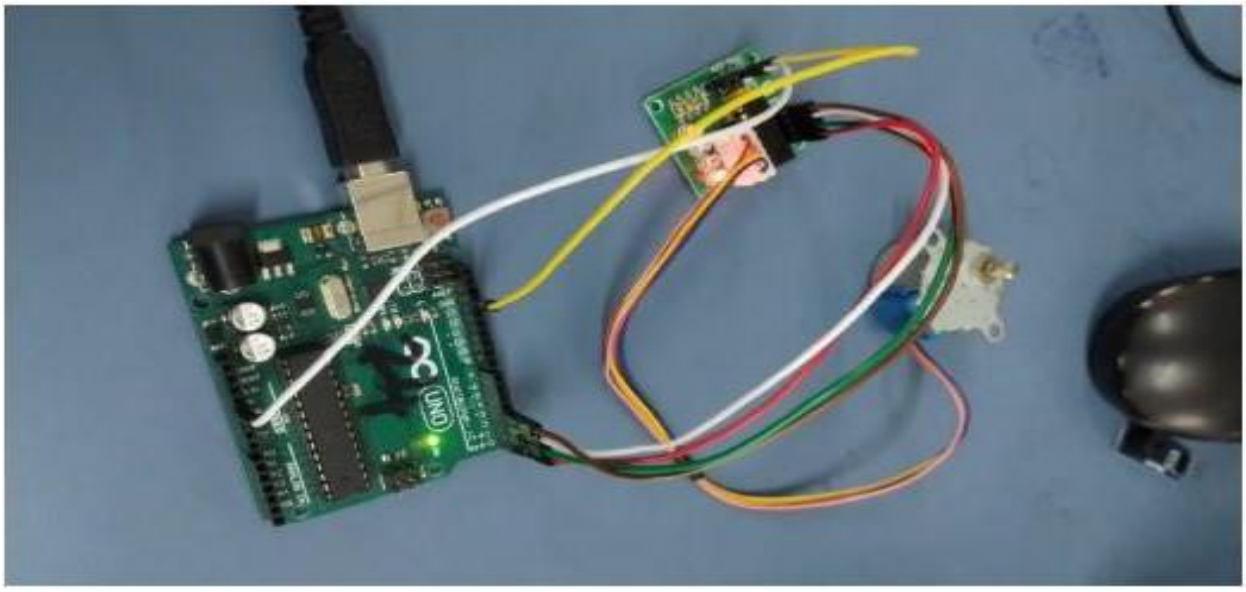
Interfacing Stepper Moror / Servo Motor with Arduino

Program:

```
void setup() {
  pinMode (8,OUTPUT);
  pinMode (9,OUTPUT);
  pinMode (10,OUTPUT);
  pinMode (11,OUTPUT);
}
void loop() {
  digitalWrite(8,HIGH);
  delay(5);
  digitalWrite(8,LOW);
  delay(5);
  digitalWrite(9,HIGH);
  delay(5);
  digitalWrite(9,LOW);
  delay(5);
  digitalWrite(10,HIGH);
  delay(5);
  digitalWrite(10,LOW);
```

```
  delay(5);
  digitalWrite(11,HIGH);
  delay(5);
  digitalWrite(11,LOW);
  delay(5);
}
```

OUTPUT:



RESULT:

Thus, the interface of sensors and actuators with Arduino to acquire data from the environment and control devices, enabling the implementation of automated systems have been verified.

Ex No: 03

Date:

INTRODUCTION TO RASBERRY PI PLATFORM AND PYTHON PROGRAMMING

AIM:

To study the installation procedure of Raspberry Pi OS, set up the Wokwi simulation, and verify its successful setup**.**

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Wokwi Simulation Platform	1
2	Raspberry Pi Pico (Virtual)	1
3	LED	2
4	Resistor (220Ω each)	2
5	Jumper Wires	4
6	Breadboard (Virtual)	1

THEORY:

CONFIGURATION OF RASPBERRY PI ON WOKWI

Wokwi

An online simulator for embedded systems development.

Raspberry Pi Pico (Virtual):

Simulated microcontroller for testing Python code.

GPIO Pins (General Purpose Input/Output)

Used to connect virtual components.

Code Editor

Built-in Python editor to run MicroPython code.

INSTALLATION PROCEDURE:

Access Wokwi Simulation :

- Open Wokwi in a web browser.

- Select 'New Project' and choose 'Raspberry Pi Pico'.

Adding Components:

- Drag and drop an LED, resistors, and jumper wires onto the simulation canvas.

- Connect the components as per the circuit diagram.

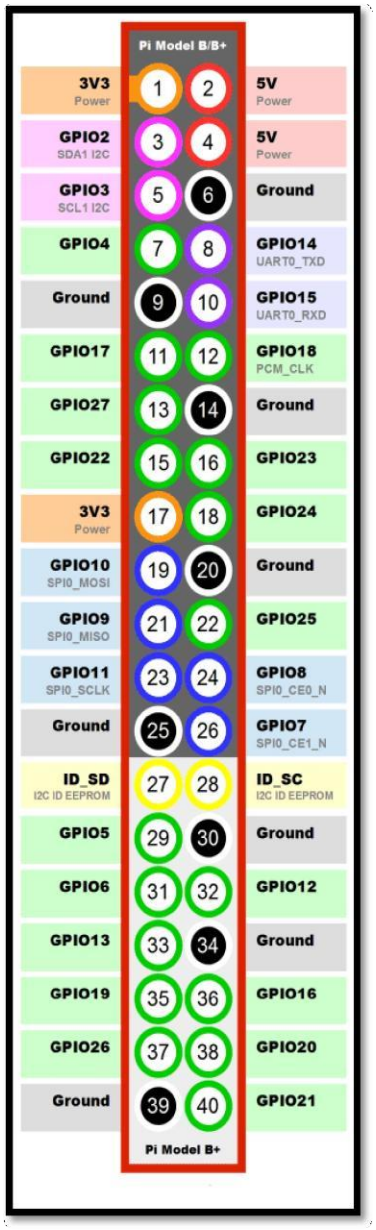
Writing and Running the Code:

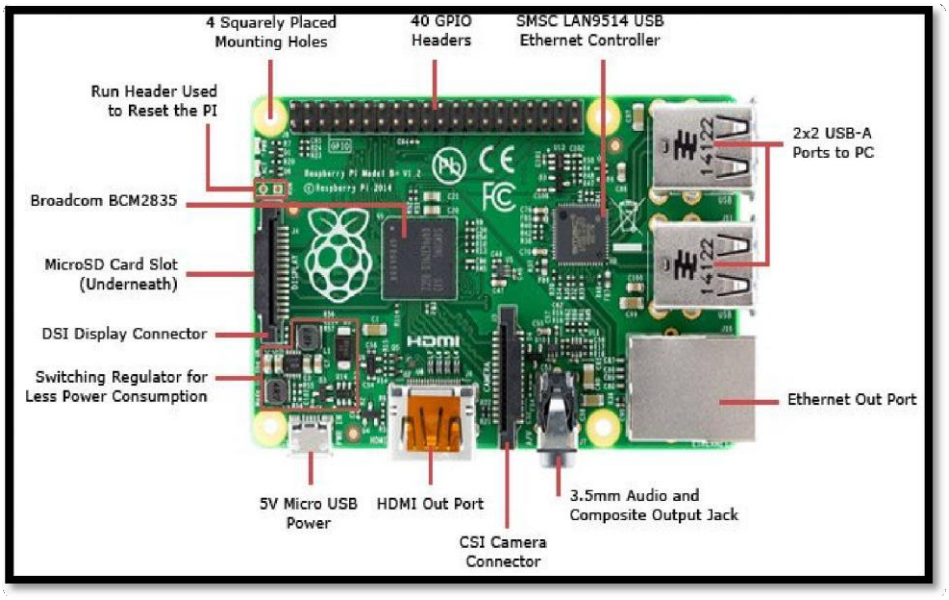
- Open the code editor in Wokwi.

- Write the Python program for blinking LEDs.

- Click the 'Run' button to execute the program

PIN CONFIGURATION





Asshown in above figure, there are 40output pins for the PI. But all 40 pin out cannot be programmed to our use. Only 26 GPIO pins can be programmed. These pins go from **GPIO2 to GPIO27**. These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions. With special GPIO put aside, we have **17 GPIO** remaining (Light green Cirl).

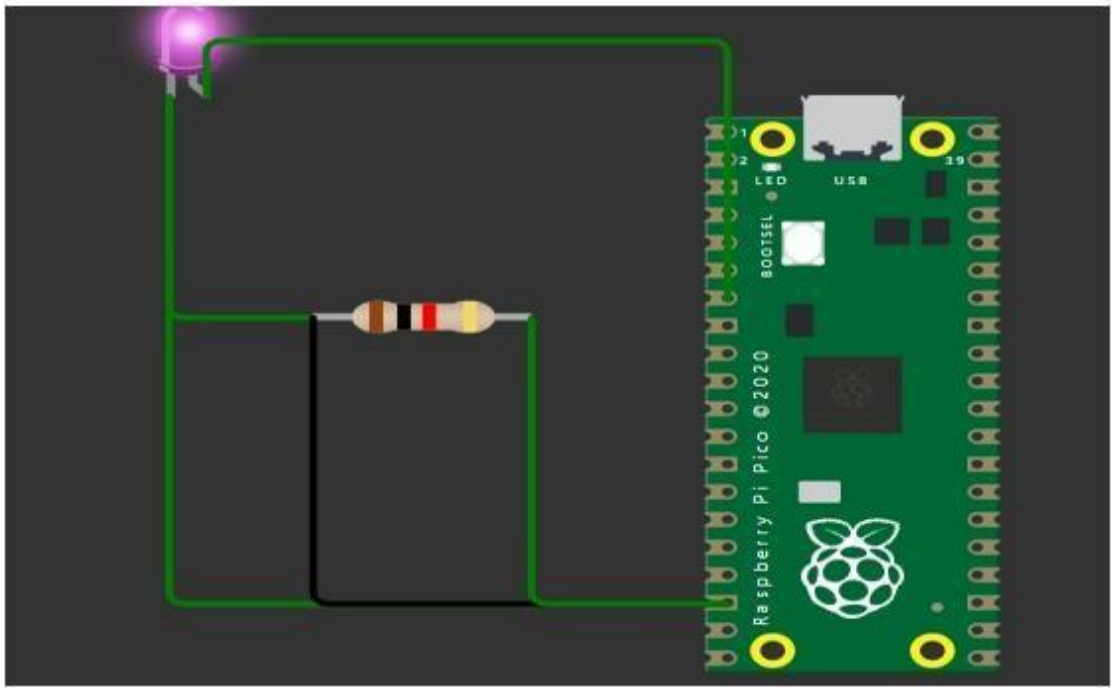
Task #1

Blink an LED Using Wokwi Simulation

Program:

```
import machine
import utime

LED_PIN = machine.Pin(17, machine.Pin.OUT)

while True:
    LED_PIN.value(1)  # Turn LED ON
    utime.sleep(1)  # Wait for 1 second
    LED_PIN.value(0)  # Turn LED OFF
    utime.sleep(1)  # Wait for 1 second
```

OUTPUT:



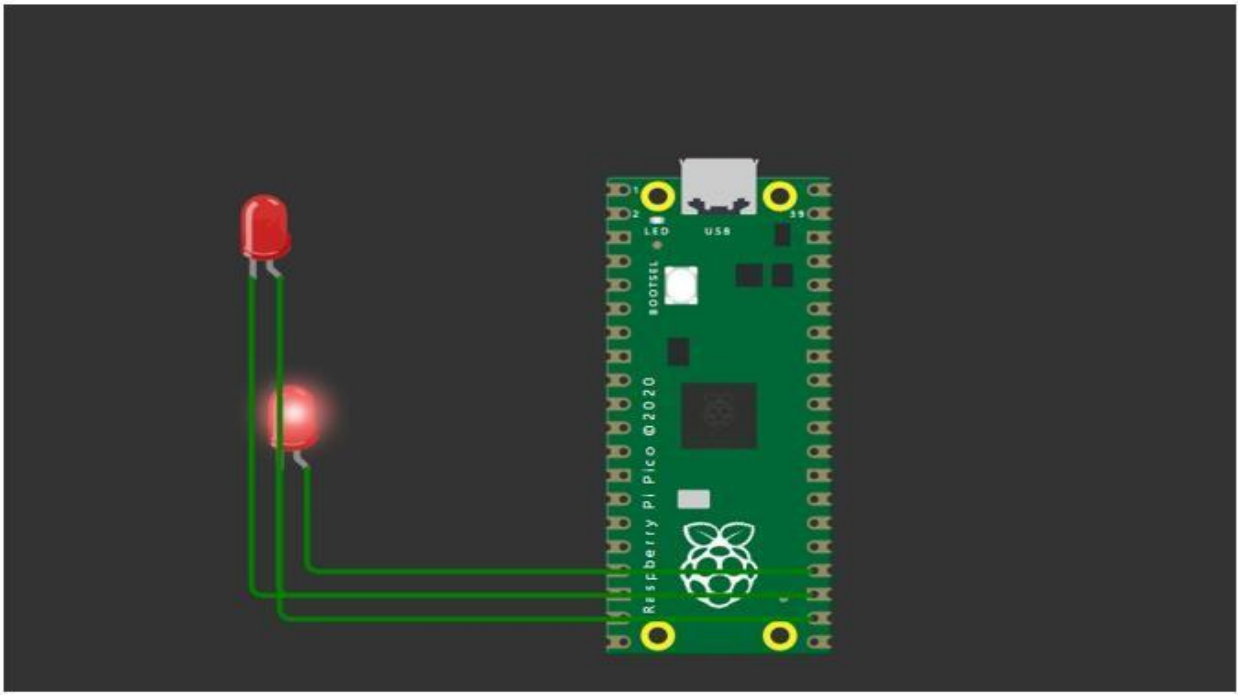
Task #2

Blink Two LEDs Alternately Using Wokwi

Program:

```
import machine
import utime

LED1 = machine.Pin(17, machine.Pin.OUT)
LED2 = machine.Pin(18, machine.Pin.OUT)

while True:
    LED1.value(1)
    LED2.value(0)
    utime.sleep(1)
    LED1.value(0)
    LED2.value(1)
    utime.sleep(1)
```

OUTPUT:



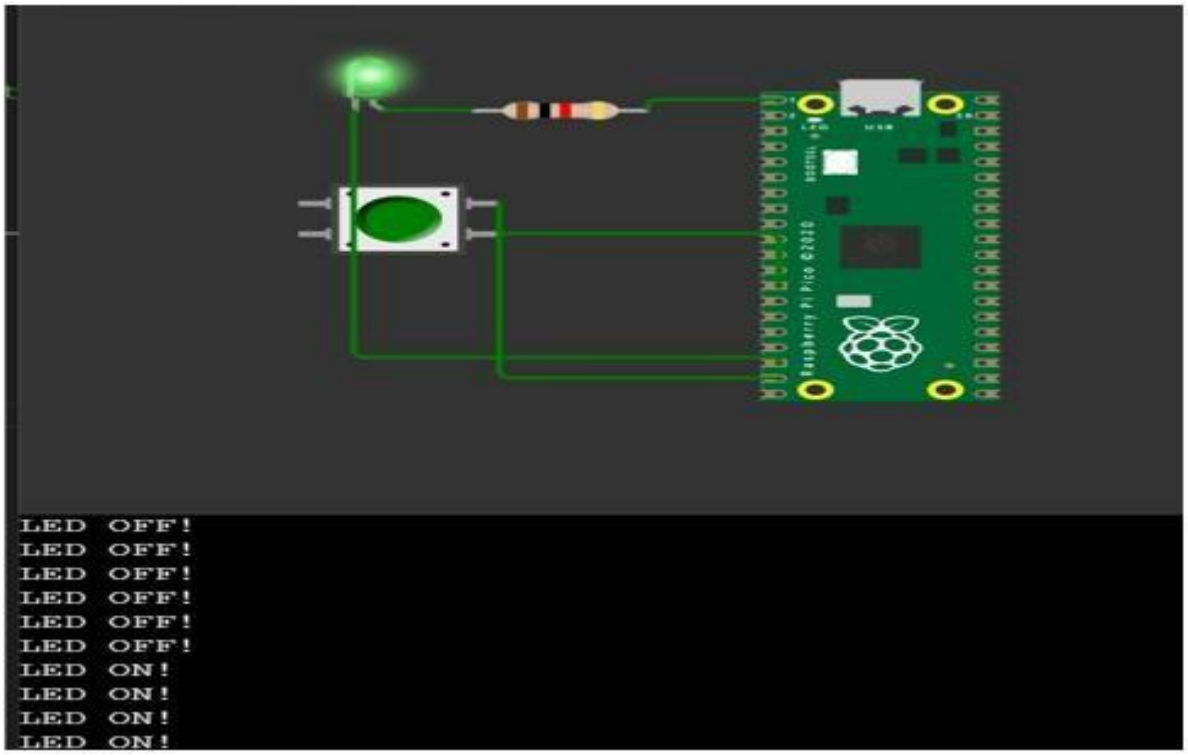
Task #3

Control an LED using a push button on the Wokwi simulator using a Raspberry Pi Pico

Program:

```
from machine import Pin
import time
# Define the LED and Button pins
led = Pin(15, Pin.OUT)    # LED connected to GPIO15
button = Pin(14, Pin.IN, Pin.PULL_UP)  # Button connected to GPIO14 with internal pull-up resistor


while True:
    if button.value() == 0:  # Button is pressed (active LOW)
        led.value(1)  # Turn LED ON
    else:
        led.value(0)  # Turn LED OFF
    time.sleep(0.1)  # Small delay for button debounce
```

OUTPUT:



RESULT:

Thus, the Raspberry Pi platform was successfully simulated in Wokwi, and the working process was verified using MicroPython programs to blink an LED.

Ex No: 04

Date:

INTERFACING SENSORS AND ACTUATORS TO RASPBERRY PI

AIM:

To interface various sensors and actuators with Raspberry Pi using the Wokwi simulation platform and verify their functionality using Python programming**.**

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Wokwi Simulation Platform	1
2	Raspberry Pi Pico (Virtual)	1
3	LED	2
4	Resistor (220Ω each)	2
5	Jumper Wires	4
6	Breadboard (Virtual)	1
7	Ultrasonic Sensor (HC-SR04)	1
8	Servo Motor	1
9	DHT11 Temperature Sensor	1

THEORY:

CONFIGURATION OF RASPBERRY PI ON WOKWI

Wokwi is an online simulator for embedded systems development that allows testing Micro Python code with virtual components.

GPIO Pins (General Purpose Input/Output)

Used to interface sensors and actuators like LEDs, buttons, servo motors, and ultrasonic sensors.

INSTALLATION PROCEDURE:

Access Wokwi Simulation :

- Open Wokwi in a web browser.

- Select 'New Project' and choose 'Raspberry Pi Pico'.

Adding Components:

- Drag and drop the required sensors, actuators, and necessary circuit elements onto the simulation canvas.

- Connect the components as per the circuit diagram.

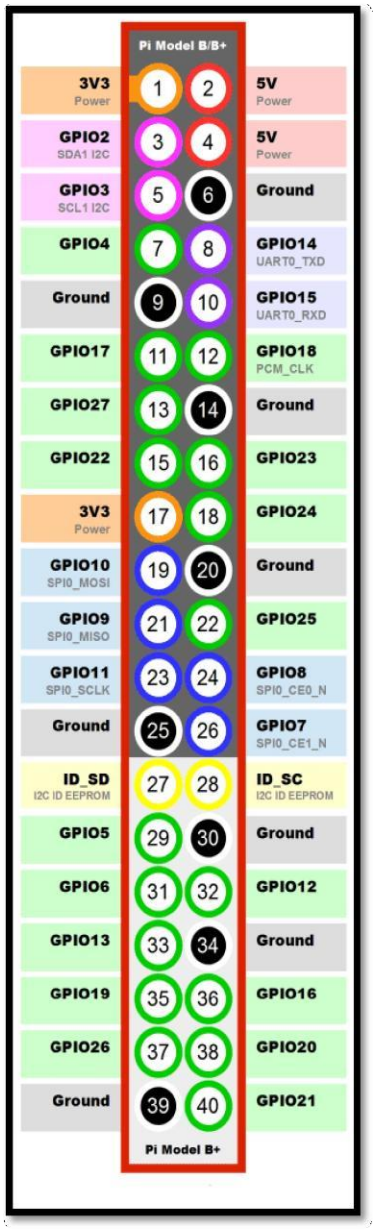
Writing and Running the Code:

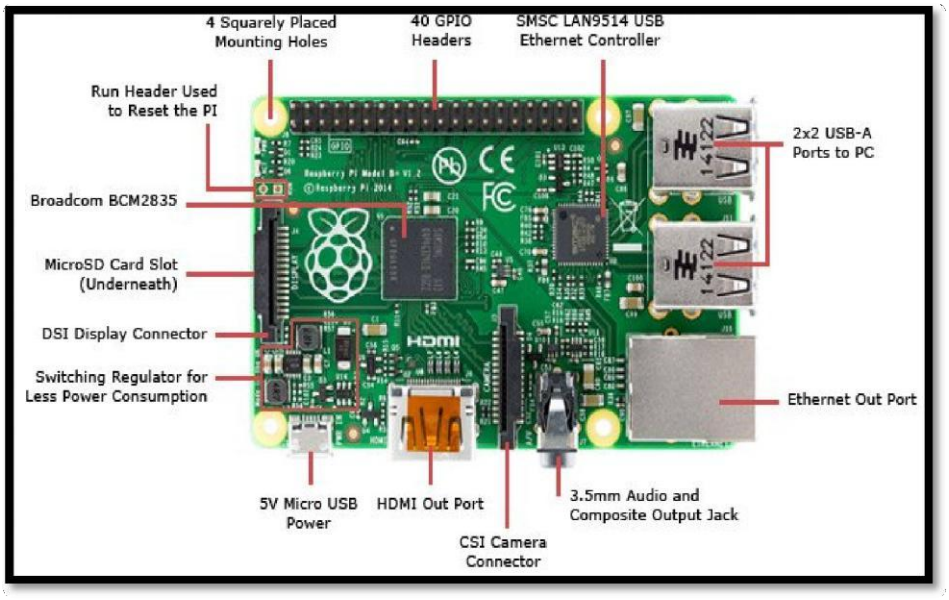
- Open the code editor in Wokwi.

- Write the Python program to interface the selected sensor or actuator.

- Click the 'Run' button to execute the program

PIN CONFIGURATION





Asshown in above figure, there are 40output pins for the PI. But all 40 pin out cannot be programmed to our use. Only 26 GPIO pins can be programmed. These pins go from **GPIO2 to GPIO27**. These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions. With special GPIO put aside, we have **17 GPIO** remaining (Light green Cirl).

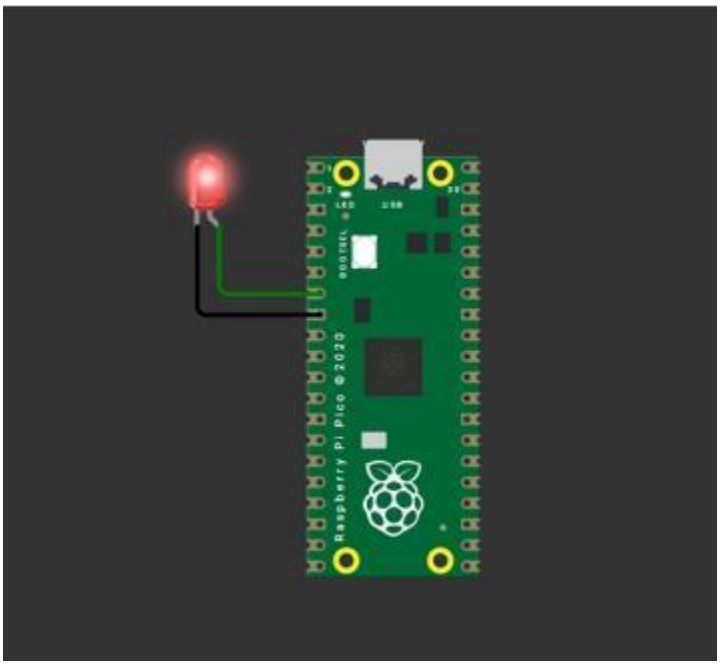
Task #1

Blink an LED Using Wokwi Simulation

Program:

```
import machine

import utime

LED_PIN = machine.Pin(17, machine.Pin.OUT)

while True:

    LED_PIN.value(1)  # Turn LED ON

    utime.sleep(1)  # Wait for 1 second

    LED_PIN.value(0)  # Turn LED OFF

    utime.sleep(1)  # Wait for 1 second
```

OUTPUT:



Task #2

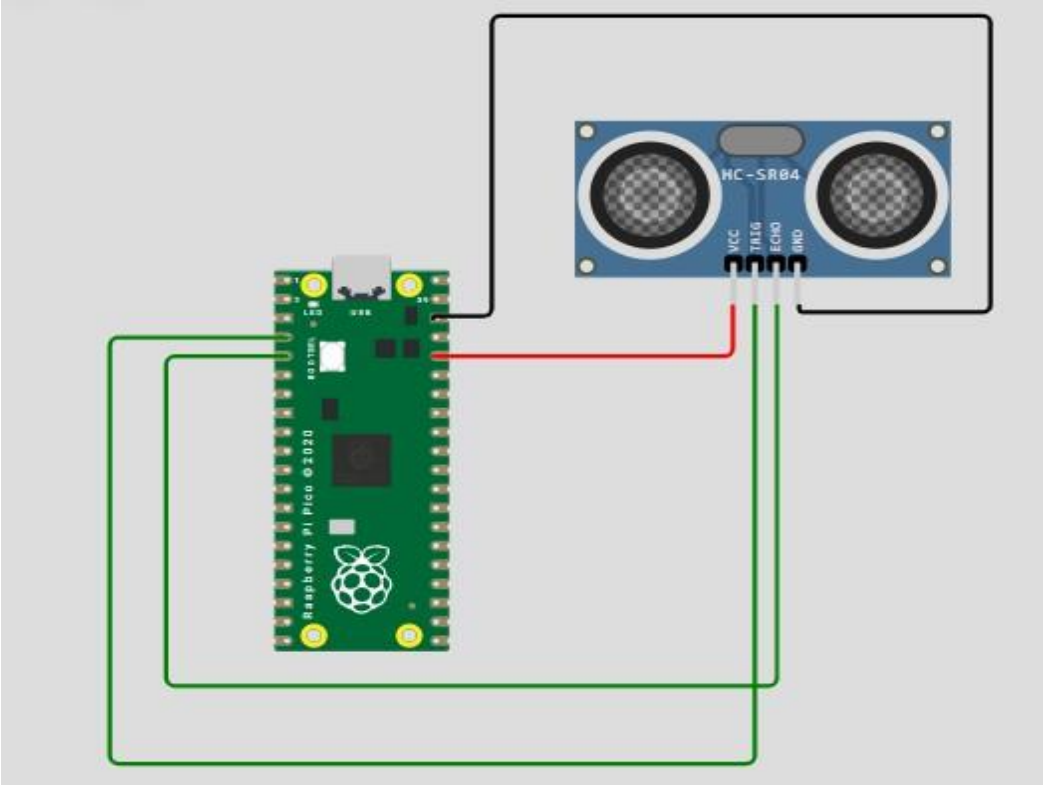
Interface an Ultrasonic Sensor (HC-SR04) to Measure Distance

Program:

```
import machine
import utime
trigger = machine.Pin(2, machine.Pin.OUT)
echo = machine.Pin(3, machine.Pin.IN)
def get_distance():
    trigger.value(1)
    utime.sleep_us(10)
    trigger.value(0)
    while echo.value() == 0:
        signaloff = utime.ticks_us()
    while echo.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    return distance
while True:
    print("Distance:", get_distance(), "cm")
    utime.sleep(1)
```

OUTPUT:

```
Distance: 235 cm
Distance: 235 cm
Distance: 236 cm
Distance: 3 cm
Distance: 3 cm
Distance: 3 cm
Distance: 9 cm
Distance: 12 cm
Distance: 13 cm
Distance: 11 cm
```



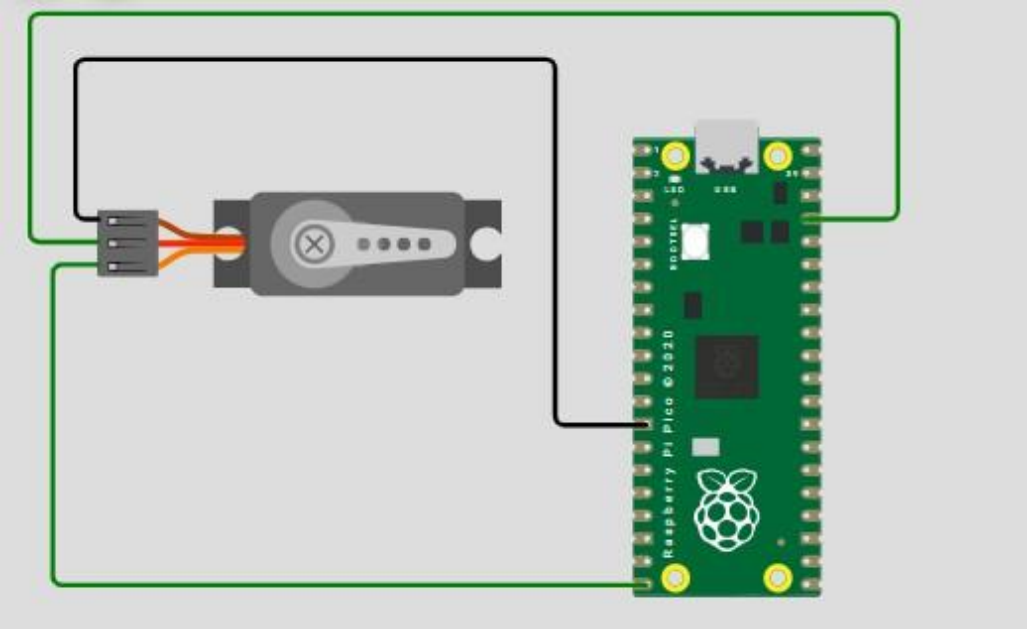
Task #3

Control a Servo Motor Using Raspberry Pi Pico

Program:

```
import machine
import utime
servo = machine.PWM(machine.Pin(15))
servo.freq(50)
def set_angle(angle):
    duty = int((angle / 180) * 1023)
    servo.duty_u16(duty)
    utime.sleep(1)
while True:
    set_angle(0)
    utime.sleep(1)
    set_angle(90)
    utime.sleep(1)
    set_angle(180)
    utime.sleep(1)
```

OUTPUT:



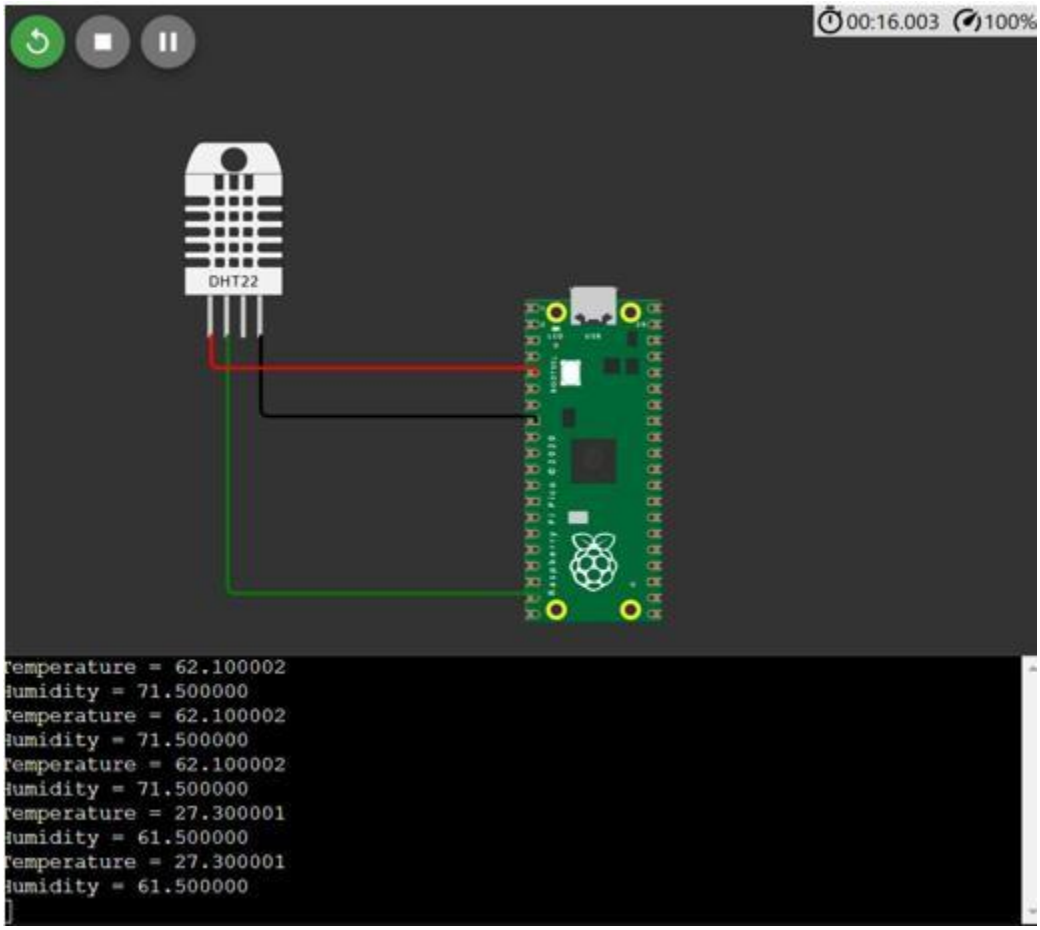
Task # 4

Read Temperature and Humidity Using DHT11 Sensor

Program

```
import machine
import dht
import utime
dht_sensor = dht.DHT11(machine.Pin(16))
while True:
    dht_sensor.measure()
    temp = dht_sensor.temperature()
    humidity = dht_sensor.humidity()
    print("Temperature:", temp, "C")
    print("Humidity:", humidity, "%")
    utime.sleep(2)
```

OUTPUT:



RESULT:

Thus, various sensors and actuators were successfully interfaced with the Raspberry Pi Pico using the Wokwi simulator, and their operation was verified using Python programs.

Ex No: 05

Date:

SETUP A CLOUD PLATFORM TO LOG THE DATA

AIM:

To set up a cloud platform for logging data collected from sensors interfaced with Raspberry Pi using the Wokwi simulation platform and verify the data storage and retrieval process**.**

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Wokwi Simulation Platform	1
2	Raspberry Pi Pico (Virtual)	1
3	DHT11 Temperature Sensor	1
4	Wi-Fi Module (ESP8266 – Virtual)	1
5	Cloud Platform (Thing speak)	1
6	Jumper Wires	4

THEORY:

Cloud platforms allow remote data logging and monitoring of sensor readings from IoT devices. In this experiment, we use Thingspeak (or Firebase) to log data from a temperature and humidity sensor connected to a Raspberry Pi Pico via Wokwi simulation. The Wi-Fi module (ESP8266) transmits the data to the clou.

INSTALLATION PROCEDURE:

Access Wokwi Simulation :

- Open Wokwi in a web browser.
- Select 'New Project' and choose 'Raspberry Pi Pico'.

Adding Components:

- Drag and drop the DHT11 sensor, ESP8266 Wi-Fi module, and necessary connections onto the simulation canvas.
- Connect the components as per the circuit diagram

Setting Up Cloud Logging (Thingspeak/Firebase):

- Create an account on Thingspeak (https://thingspeak.com/) or Firebase (https://firebase.google.com/).
- Set up a new data stream to receive sensor values.
- Obtain API keys for authentication.

Writing and Running the Code:

- Open the code editor in Wokwi.
- Write the Python program to interface the DHT11 sensor, connect to Wi-Fi, and send data to the cloud.
- Click the 'Run' button to execute the program

PIN CONFIGURATION

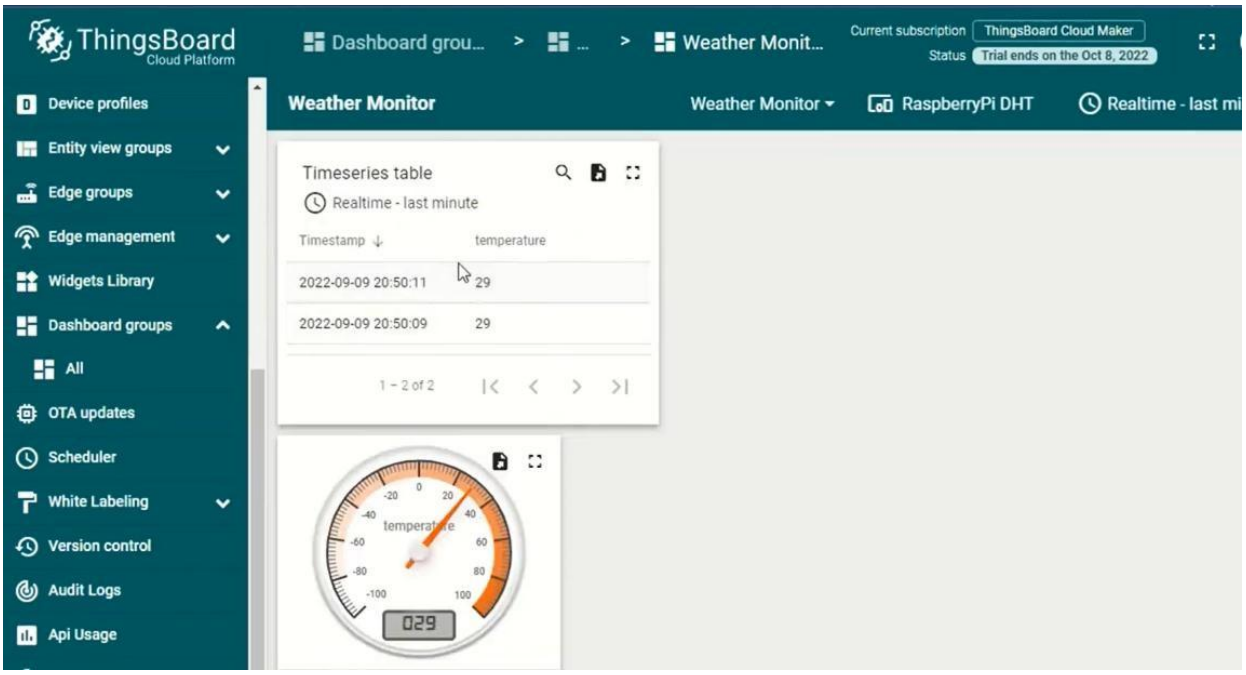
- DHT11 Sensor → GPIO16 (Data)
- ESP8266 Wi-Fi Module → UART (GPIO0 & GPIO1 for TX/RX)

Program:

```
import network

import urequests

import utime

import machine

import dht

# Wi-Fi Credentials

SSID = "Your_WiFi_SSID"

PASSWORD = "Your_WiFi_Password"

API_KEY = "Your_Thingspeak_API_Key"

# Setup Sensor

dht_sensor = dht.DHT11(machine.Pin(16))

# Connect to Wi-Fi

def connect_wifi():

    wlan = network.WLAN(network.STA_IF)

    wlan.active(True)

    wlan.connect(SSID, PASSWORD)

    while not wlan.isconnected():
```

```
        pass
    print("Connected to Wi-Fi")
# Send Data to Thingspeak
def send_data():
    dht_sensor.measure()
    temp = dht_sensor.temperature()
    humidity = dht_sensor.humidity()
    url =  f"https://api.thingspeak.com/update?api_key={API_KEY}&
    field1={temp}&field2={humidity}"
    response = urequests.get(url)
    response.close()
    print(f"Data Sent: Temp={temp}C, Humidity={humidity}%")
# Main Loop
connect_wifi()
while True:
    send_data()
    utime.sleep(15)  # Send data every 15 seconds
```

SCREENSHORT:



RESULT:

Thus, a cloud platform was successfully set up for logging sensor data using the Wokwi simulator, and data storage and retrieval were verified using Python programming.

Ex No: 06

Date:

LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

AIM:

To log real-time sensor data using Raspberry Pi and upload it to a cloud platform for remote monitoring and analysis**.**

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Raspberry Pi	1
2	DHT11 Temperature Sensor	1
3	Wi-Fi Module (Built-in or USB)	1
4	Cloud Platform (ThingSpeak/Firebase)	1
5	Breadboard	1
6	Jumper Wires	4

THEORY:

Cloud platforms enable IoT devices to send sensor data for real-time storage and remote access. This experiment demonstrates how to log temperature and humidity data using a DHT11 sensor connected to a Raspberry Pi and upload it to ThingSpeak for visualization.

STEPS FOR BUILDING RASPBERRY PI DATA LOGGER ON CLOUD

STEP 1: SIGNUP FOR THINGSPEAK

For creating your channel on ThingSpeak you first need to sign up on ThingSpeak. In case if you already have account on ThingSpeak just sign in using your id and password.

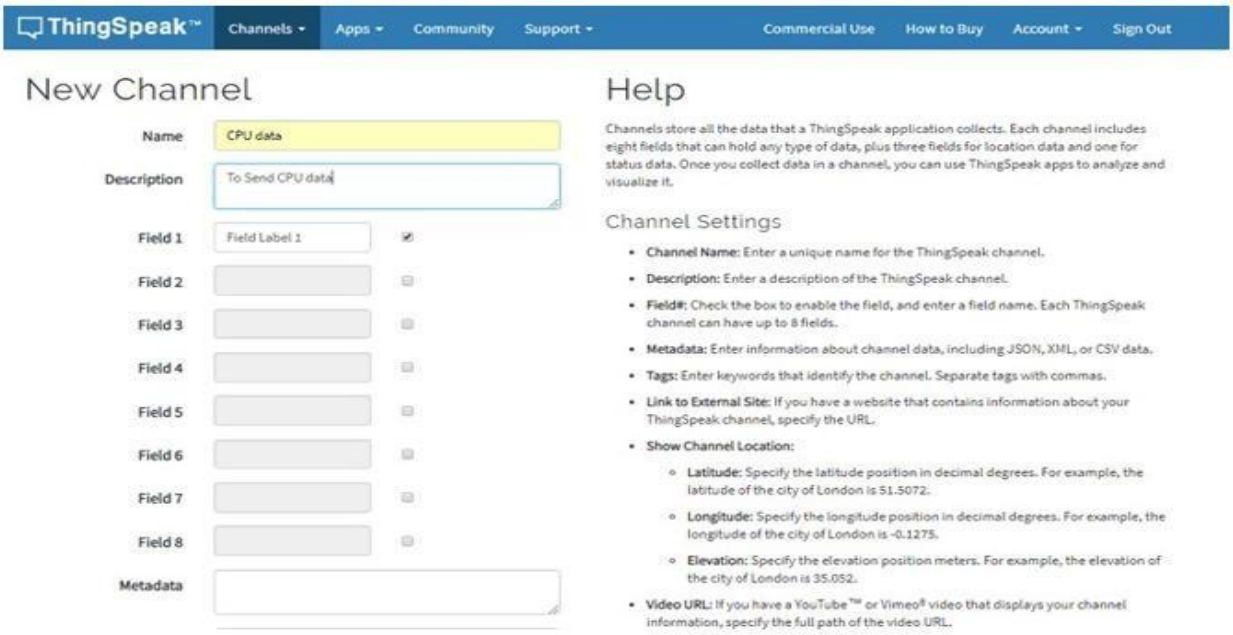
Click on signup if you don't have account and if you already have account click on sign in.

After clicking on signup fill your details.

After this verify your E-mail id and click on continue.

STEP 2: CREATE A CHANNEL FOR YOUR DATA

Once you Sign in after your account verification, create a new channel by clicking "New Channel" button.



After clicking on “New Channel”, enter the Name and Description of the data you want to upload on this channel.

For example, I am sending my CPU data (temperature), so I named it as CPU data.

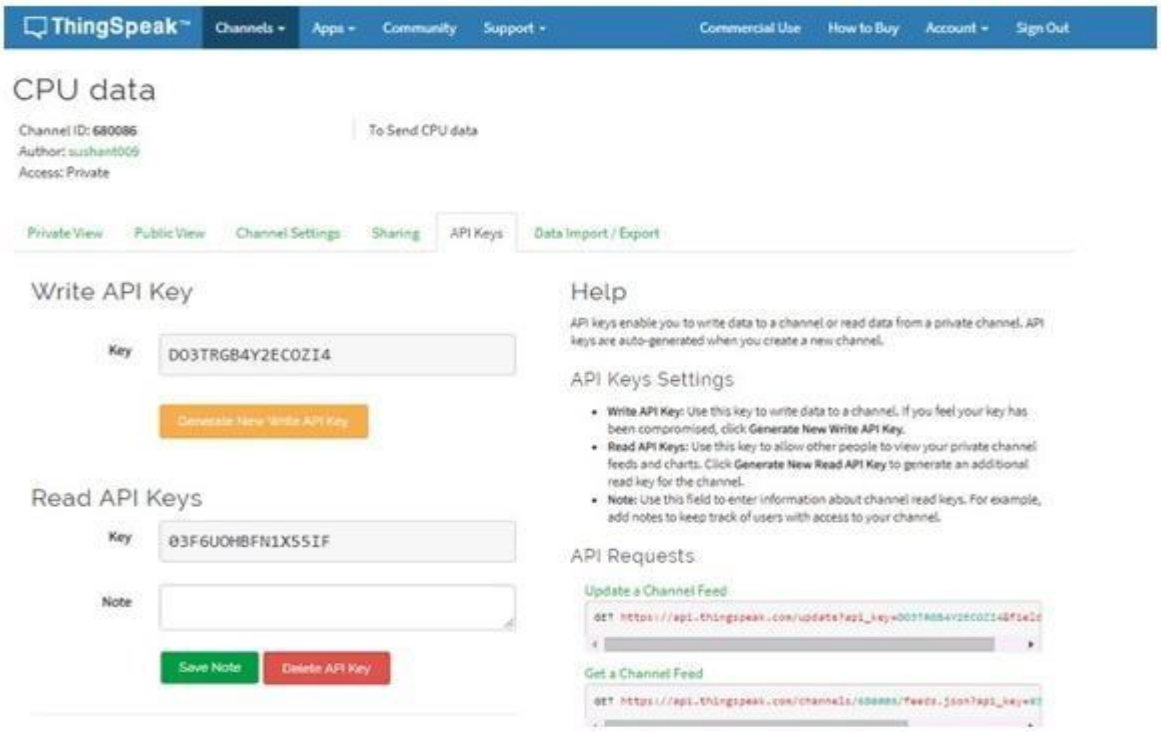
Now enter the name of your data (like Temperature or pressure) in Field1. If you want to use more than one Field you can check the box next to Field option and enter the name and description of your data.

After this click on save channel button to save your details.

STEP 3: GETTING API KEY IN THINGSPEAK

To send data to ThingSpeak, we need a unique API key, which we will use later in our python code to upload our CPU data to ThingSpeak Website.

Click on “API Keys” button to get your unique API key for uploading your CPU data.



Now copy your “Write API Key”. We will use this API key in our code.

STEP 4: PYTHON CODE FOR RASPBERRY PI

Complete code is given at the end of this tutorial, just make a file with any name and .py extension and copy-paste the code and save the file. Don't forget to replace the API key with yours. You can run the python file any time using below command:

```
python /path/filename.py
```

Assuming you already installed python in Raspberry pi using this command

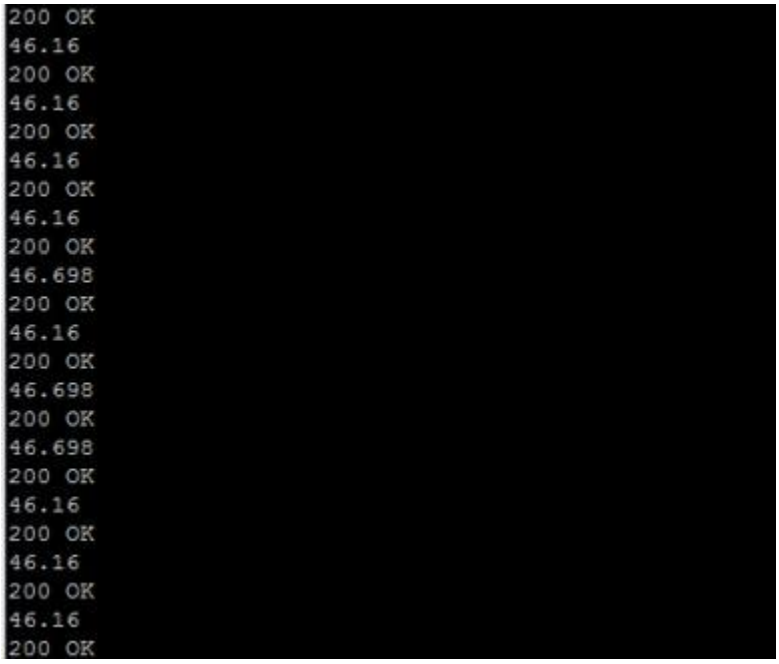
```
sudo apt-get install python
```

CASE 1:

If you are using monitor screen then just use the given code. Now install all libraries

```
sudo

apt-get
```

- After installing libraries run your python code (*python /path/filename.py*)
- If the code runs properly, you will see some CPU temperature values as shown below image
- If there are any errors uploading the data, you will receive "connection failed" message



CASE 2:

If you are using “Putty” then you should follow these commands

First update your pi using

```
sudo apt-get update
```

After this make a file cpu.py using

```
nano cpu.py
```

After creating this file copy your code to this file and save it using CTRL + X and then ‘y’ and enter.

After this install all libraries using

```
sudo apt-get install httplib

sudo apt-get install urllib
```

After installing libraries run your python code using:

```
python cpu.py
```

If the code runs properly you will see some CPU temperature values as shown in above image.

STEP 6: CHECK THINGSPEAK SITE FOR DATA LOGGING

After completing these steps open your channel and you will see the CPU temperature data is updating into ThingSpeak website.

- Like this you can send any sensor data connected with Raspberry pi to the ThingSpeak Cloud. In next article we will connect LM35

temperature sensor with Raspberry Pi and send the temperature data to ThingSpeak, which can be monitored from anywhere.

- Complete Python code for this Raspberry Pi Cloud Server is given below. Code is easy and self-explanatory to understand.

CODE:

```
import time

import board

import adafruit_dht

import requests

# Setup DHT Sensor

dht_sensor = adafruit_dht.DHT11(board.D4)

THINGSPEAK_API_KEY = "YOUR_API_KEY"

THINGSPEAK_URL = "https://api.thingspeak.com/update"

while True:

    try:

        temp = dht_sensor.temperature

        humidity = dht_sensor.humidity

        print(f"Temperature: {temp}C, Humidity: {humidity}%")

        # Send Data to ThingSpeak

        params = {"api_key": THINGSPEAK_API_KEY, "field1": temp, "field2": humidity}

        requests.get(THINGSPEAK_URL, params=params)
```

```
except RuntimeError as e:
    print("Error reading sensor: ", e)

time.sleep(15)  # Log every 15 seconds
```

RESULT:

Thus, a cloud platform was successfully set up for logging sensor data using the Wokwi simulator, and data storage and retrieval were verified using Python programming.

Ex No: 07 A)

Date:

DESIGN AN IOT BASED SYSTEM - HC - 05

AIM:

To design and implement an IoT-based system using Raspberry Pi and an HC-05 Bluetooth module to control an LED remotely.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Raspberry Pi	1
2	HC-05 Bluetooth Module	1
3	LED	1
4	Resistor (220Ω)	1
5	Breadboard	1
6	Jumper Wires	4

THEORY:

The HC-05 Bluetooth module enables wireless communication between Raspberry Pi and external devices like smartphones or microcontrollers. In this experiment, the Raspberry Pi receives commands via Bluetooth and controls an LED accordingly.

INSTALLATION PROCEDURE:

STEP 1: SET UP THE RASPBERRY PI

1. Install Raspberry Pi OS and ensure Bluetooth is enabled.
2. Update the system:

```
sudo apt update && sudo apt upgrade -y
```

3. Install Bluetooth libraries:

sudo apt-get install bluetooth bluez python3-bluetooth

STEP 2: CONNECT HC-05 BLUETOOTH MODULE AND LED

HC-05 WIRING:

VCC → 5V

GND → GND

TX → RX (GPIO15)

RX → TX (GPIO14)

LED WIRING:

Anode (+) → GPIO17 (through a 220Ω resistor)

Cathode (-) → GND

STEP 3: CONFIGURE HC-05 ON RASPBERRY PI

1. Enable the serial interface on Raspberry Pi:

   sudo raspi-config

   Go to Interfacing Options → Serial

   Disable login shell access and enable the serial port.

2. Restart the Raspberry Pi.

STEP 4: WRITE PYTHON CODE TO CONTROL LED VIA BLUETOOTH

1. Open a new Python script:

   nano bluetooth_led.py

   CODE:

```
import RPi.GPIO as GPIO
import serial
# Setup LED Pin
LED_PIN = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED_PIN, GPIO.OUT)
# Setup Bluetooth Communication
bt_serial = serial.Serial("/dev/serial0", baudrate=9600, timeout=1)
print("Waiting for Bluetooth command...")
while True:
    data = bt_serial.readline().decode('utf-8').strip()
    if data:
        print(f"Received: {data}")
        if data == "ON":
            GPIO.output(LED_PIN, GPIO.HIGH)
            print("LED Turned ON")
```

```
        elif data == "OFF":
            GPIO.output(LED_PIN, GPIO.LOW)
            print("LED Turned OFF")
```

2. Save and exit

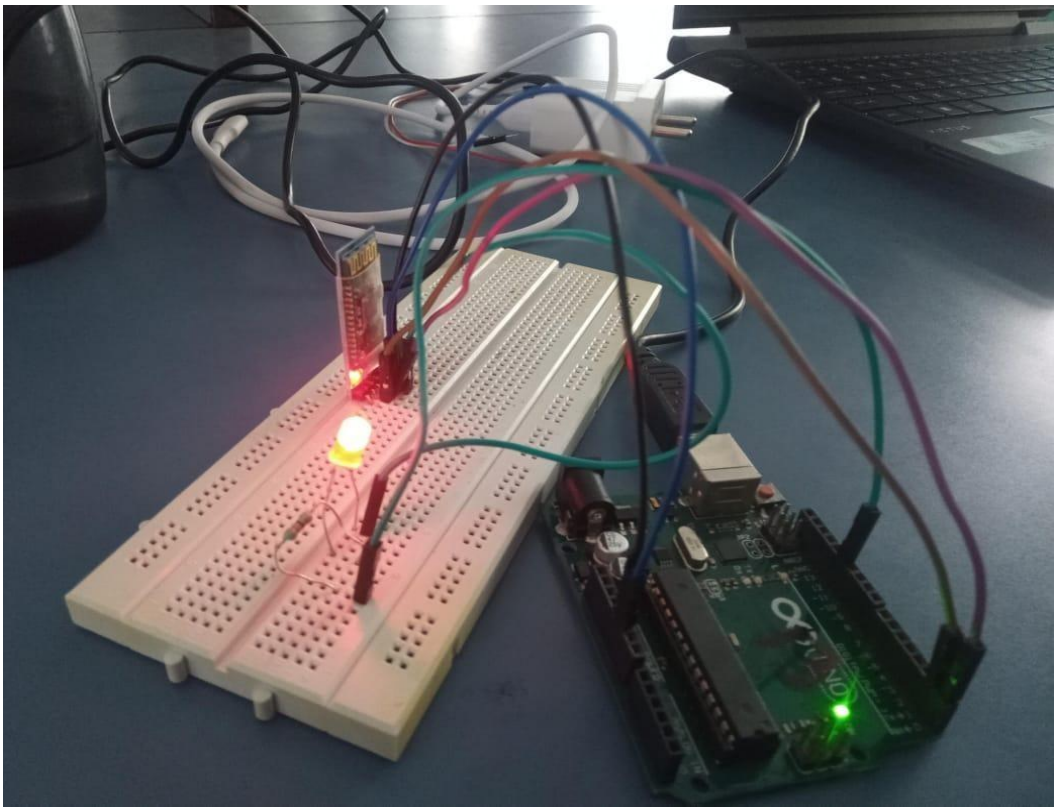
STEP 5: RUN THE SCRIPT

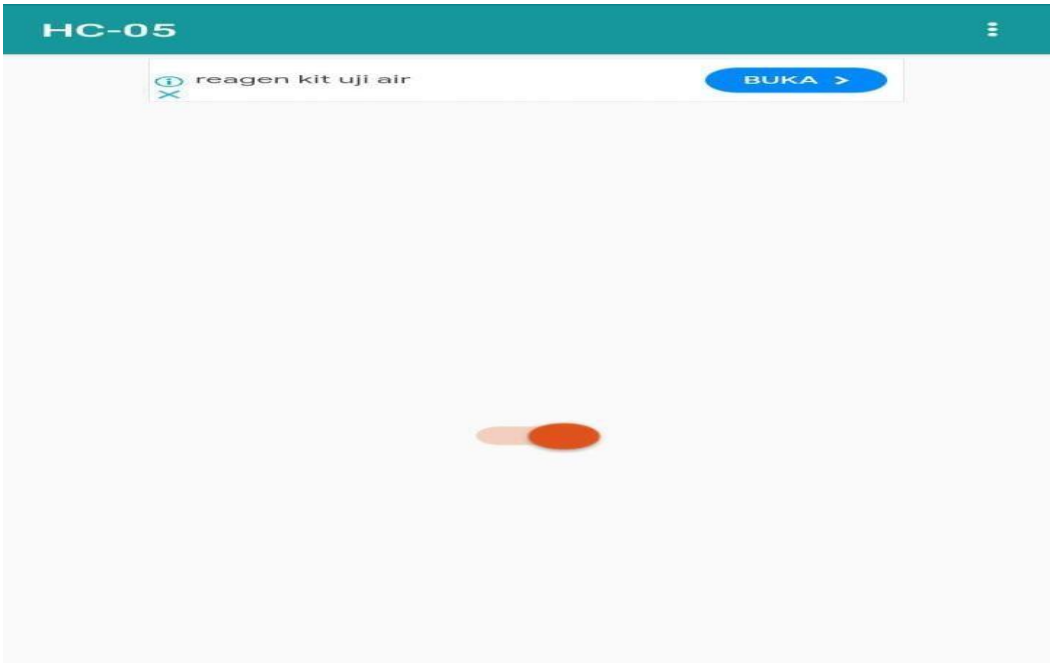
```
python3 bluetooth_led.py
```

STEP 6: SEND COMMANDS FROM A BLUETOOTH TERMINAL APP

1. Pair the HC-05 module with a smartphone using a Bluetooth terminal app.
2. Send "ON" to turn the LED ON.
3. Send "OFF" to turn the LED OFF.

SCREENSHOTS:





RESULT:

Thus, an IoT-based system using an HC-05 Bluetooth module was successfully designed to control an LED remotely.

B) DESIGN AN IoT-BASED SYSTEM USING SOIL MOISTURE SENSOR

AIM:

To design and implement an IoT-based system using an Arduino and Soil Moisture Sensor to monitor soil moisture levels and control an LED indicator remotely.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Arduino Uno	1
2	Soil Moisture Sensor	1
3	LED	1
4	Resistor (220Ω)	1
5	Breadboard	1
6	Jumper Wires	5

THEORY:

Soil moisture sensors measure the volumetric water content of soil, providing data for automated irrigation systems. In this experiment, the Arduino Uno reads sensor data, processes it, and turns ON/OFF an LED based on the soil moisture level.

INSTALLATION PROCEDURE:

STEP 1: CONNECT SOIL MOISTURE SENSOR AND LED

SOIL MOISTURE SENSOR WIRING:

VCC → 5V

GND → GND

A0 → A0 (Analog pin for moisture reading)

LED WIRING:

Anode (+) → Pin 8 (through a 220Ω resistor)

Cathode (-) → GND

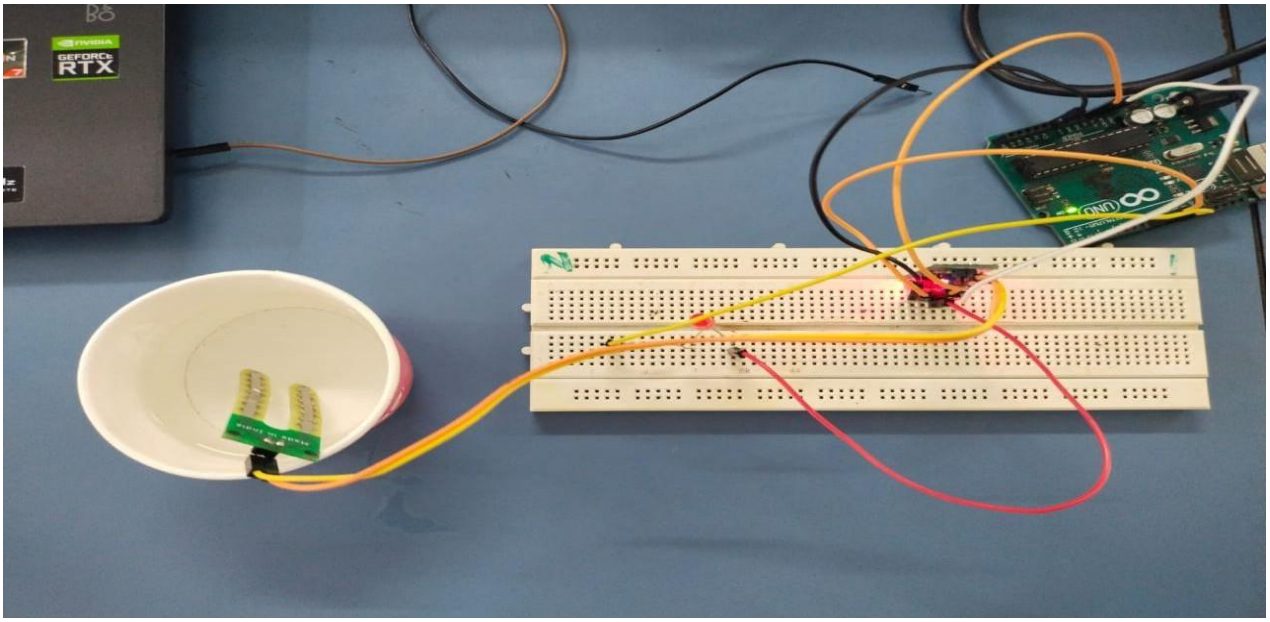
STEP 2: WRITE ARDUINO CODE TO MONITOR SOIL MOISTURE

1. Open the **Arduino IDE**.
2. Create a new sketch and copy the following code:

```
#define sensorPin A0
#define sensorPower 7
#define LED_PIN 8
void setup() {
   pinMode(sensorPower, OUTPUT);
   pinMode(LED_PIN, OUTPUT);
   Serial.begin(9600);
}
void loop() {
   Serial.print("Analog output : ");
   int val = readSensor();
   Serial.println(val);
   if (val > 1000) {
      Serial.println("Low moisture detected! Water the plants");
      digitalWrite(LED_PIN, HIGH);
```

```
    } else {
        digitalWrite(LED_PIN,  LOW);
    }
    delay(1000);
}
int readSensor() {
    digitalWrite(sensorPower, HIGH);
    delay(10);
    int val = analogRead(sensorPin);
    digitalWrite(sensorPower, LOW);
    return val;
}
```

SCREENSHOTS:



RESULT

Thus, an IoT-based system using an Arduino and Soil Moisture Sensor was successfully designed to monitor soil conditions and indicate the need for irrigation.

Ex No: 08

Date:

DEVELOP THE VEHICLE PARKING SYSTEM IN SMART CITIES USING IOE

AIM:

To design and implement an IoT-based Vehicle Parking System using an Ultrasonic Sensor and Arduino to detect the presence of vehicles and indicate availability using an LED.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Arduino Uno	1
2	Ultrasonic Sensor (HC-SR04)	1
3	LED	1
4	Resistor (220Ω)	1
5	Breadboard	1
6	Jumper Wires	5

THEORY:

An IoT-based Vehicle Parking System uses an Ultrasonic Sensor to detect whether a parking space is occupied or vacant. The distance measured by the sensor helps determine the presence of a vehicle, and an LED is used as an indicator for parking availability.

INSTALLATION PROCEDURE:

STEP 1: CONNECT THE ULTRASONIC SENSOR AND LED

ULTRASONIC SENSOR WIRING:

Ultrasonic Sensor Pin	Arduino Pin
VCC	5V
GND	GND
TRIG	9
ECHO	10

LED WIRING:

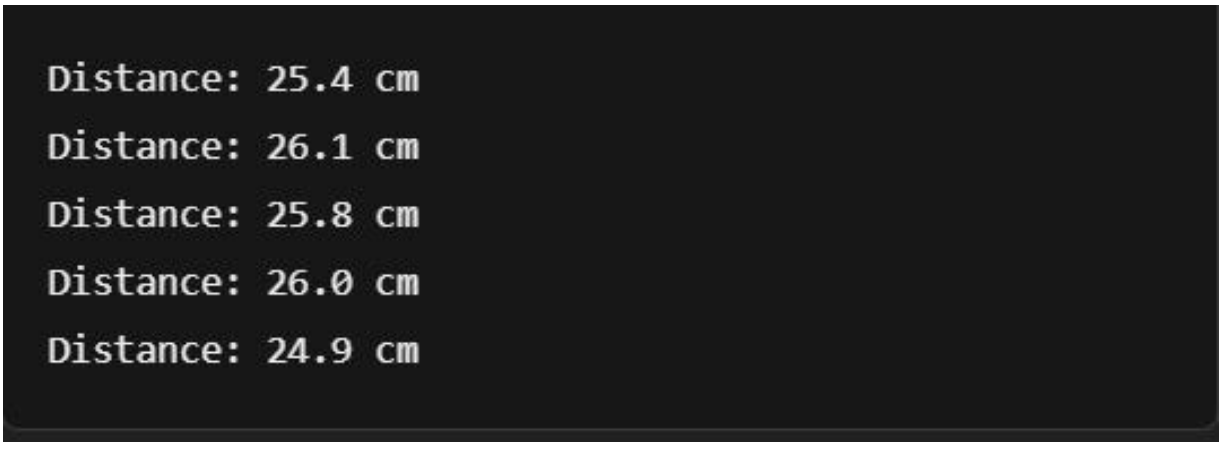
LED Pin	Arduino Pin
Anode (+)	6 (through a 220Ω resistor)
Cathode (-)	GND

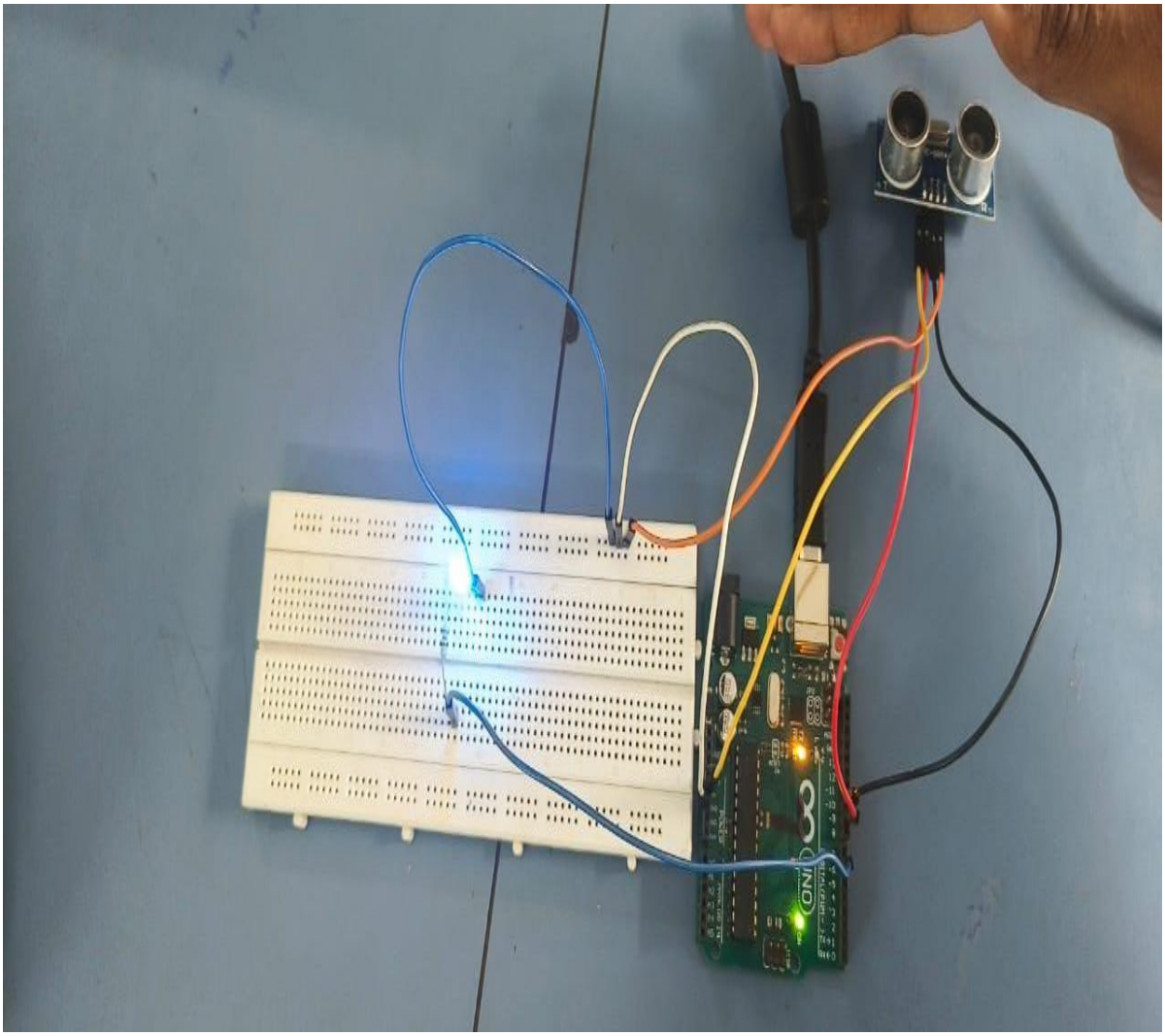
STEP 2: WRITE ARDUINO CODE TO MEASURE DISTANCE

```
#define TRIG_PIN 9
#define ECHO_PIN 10
#define LED_PIN 6
void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  long duration;
  int distance;
  // Send 10us pulse to trigger pin
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
```

```
    // Read echo pin and calculate distance
    duration = pulseIn(ECHO_PIN, HIGH);
    distance = duration * 0.034 / 2;  // Convert to cm
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    // If object is closer than 10 cm, turn on LED
    if (distance > 0 && distance < 10) {
      digitalWrite(LED_PIN,  HIGH);
    } else {
      digitalWrite(LED_PIN,  LOW);
    }
    delay(500);  // Wait before next measurement
  }
```

SCREENSHOTS:





RESULT:

Thus, an IoT-based system using an ultrasonic sensor and an LED was successfully designed and implemented to detect proximity and trigger an LED accordingly.

Ex No: 09

Date:

DEVELOP AN LOE-ENABLED HEALTHCARE SYSTEM FOR REMOTE AND NEAR PATIENT MONITORING

AIM:

To develop an Internet of Everything (IoE)-enabled healthcare system using Wokwi to monitor heart rate, temperature, and humidity remotely using a Pulse Rate Sensor and DHT22.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	ESP32	1
2	DHT22 Sensor	1
3	Pulse Rate Sensor	1
4	Resistor (220Ω)	1
5	Breadboard	1
6	Jumper Wires	6

THEORY:

IoE in healthcare enables real-time remote monitoring of vital parameters, improving patient care and response time. The ESP32 microcontroller reads data from a DHT22 temperature-humidity sensor and a pulse rate sensor, then transmits the values to an MQTT broker for real-time monitoring.

INSTALLATION PROCEDURE:

STEP 1: SET UP WOKWI SIMULATION ENVIRONMENT

1. Open Wokwi in a web browser.
2. Select "New Project" and choose "ESP32" as the microcontroller.
3. Add the required components: DHT22 sensor and Pulse Rate Sensor.
4. Connect the components as per the wiring table.

STEP 2: CONNECT THE SENSORS

DHT22 SENSOR WIRING:

DHT22 Pin	ESP32 Pin
VCC	3.3V
GND	GND
Data	12

PULSE RATE SENSOR WIRING:

Pulse Sensor Pin	ESP32 Pin
VCC	3.3V
GND	GND
Signal	35

STEP 3: WRITE ESP32 CODE FOR DATA ACQUISITION AND MQTT PUBLISHING

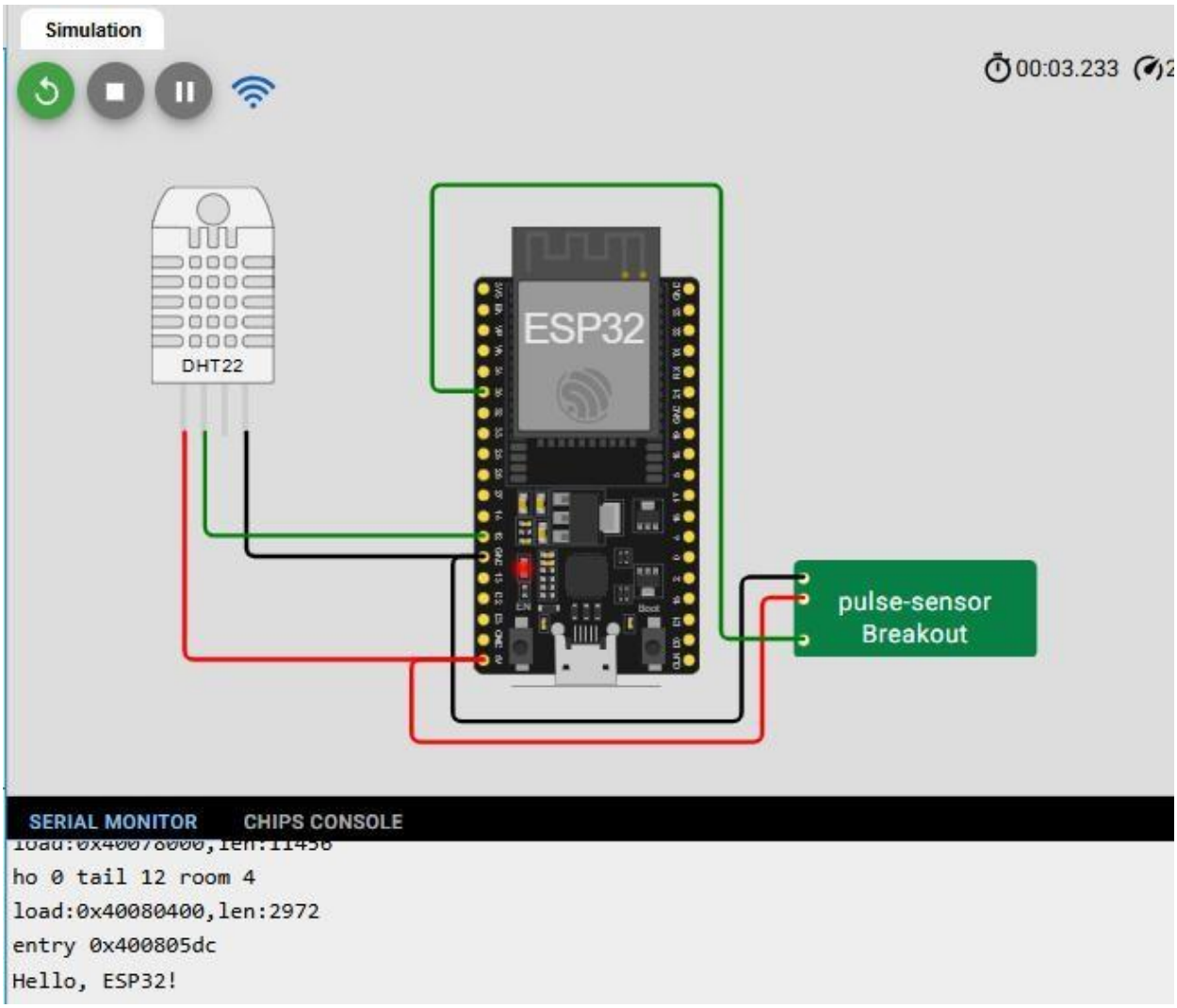
```
#include <Wire.h>

#include <WiFi.h>

#include <DHT.h>

#include <PubSubClient.h>

#include <Arduino.h>

#define DHT_PIN 12

// Define PULSE_PIN

#define PULSE_PIN 35

#define MQTT_SERVER "broker.emqx.io"

#define MQTT_PORT 1883

const char *ssid = "Wokwi-GUEST";

const char *password = "";

#define MQTT_TOPIC_HR "/heartRate"

#define MQTT_TOPIC_TEMP "/tempValue"

#define MQTT_TOPIC_HUM "/humValue"

WiFiClient espClient;

PubSubClient client(espClient);

DHT dht(DHT_PIN, DHT22);

void setup() {

  Wire.begin(23, 22);
```

```
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
  connectToWiFi();
  client.setServer(MQTT_SERVER, MQTT_PORT);
  dht.begin();
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  // Read pulseValue from PULSE_PIN
  int16_t pulseValue = analogRead(PULSE_PIN);
  // Convert pulseValue to voltage
  float voltage = pulseValue * (5 / 4095.0);
  // calculate heartRate from volatge
  int heartRate = (voltage / 3.3) * 675;
  // Print HeartRate
   Serial.print("Heart Rate");
   Serial.print(heartRate);
```

```
  Serial.print(" Temp: ");
  Serial.print(temperature);
  Serial.print(" Humidity: ");
  Serial.println(humidity)
  delay(100);
  client.publish(MQTT_TOPIC_HR, String(heartRate).c_str());
  client.publish(MQTT_TOPIC_TEMP, String(temperature).c_str());
  client.publish(MQTT_TOPIC_HUM, String(humidity).c_str());
  Serial.print("Sent mtqq data");
  delay(10);
}
void connectToWiFi() {
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to WiFi");
}
void reconnect() {
 while (!client.connected()) {
```

```
    if (client.connect("esp32_neopixel_controller")) {
      Serial.println("Connected to MQTT");
      // subscribeToCommands();
    } else {
      Serial.print("Failed, rc=");
      Serial.print(client.state());
      Serial.println(" Retrying in 5 seconds...");
      delay(5000);
    }
  }
}
```

SCREENSHOTS:



RESULT:

Thus, an IoE-enabled healthcare system was successfully developed using Wokwi to monitor heart rate, temperature, and humidity remotely.

Ex No: 10

Date:

DEVELOP AN IOE APPLICATION TO HANDLE DATA ACQUISITION WITH INTEGRATION OF THE OUTPUT DEVICES

AIM:

To develop an Internet of Everything (IoE) application using Wokwi to monitor temperature, humidity, and gas concentration, and activate an alert system using an LED and buzzer.

COMPONENTS REQUIRED:

SI.No.	Components Name	Quantity
1	Arduino Uno	1
2	DHT11 Sensor	1
3	MQ-2 Gas Sensor	1
4	Buzzer	1
5	LED	1
6	Breadboard	1
7	Jumper Wires	6

THEORY:

IoE applications integrate data acquisition with output devices for monitoring and automation. This project utilizes an Arduino Uno to read temperature and humidity data from a DHT11 sensor and detect gas concentration using an MQ-2 gas sensor. If the temperature exceeds 40°C or the gas level surpasses a threshold, a buzzer and LED alert the user.

INSTALLATION PROCEDURE:

STEP 1: SET UP WOKWI SIMULATION ENVIRONMENT

1. Open Wokwi in a web browser.
2. Select "New Project" and choose "Arduino Uno" as the microcontroller.
3. Add the required components: DHT11 sensor, MQ-2 Gas Sensor, Buzzer, and LED.
4. Connect the components as per the wiring table.

STEP 2: CONNECT THE SENSORS AND OUTPUT DEVICES

DHT11 SENSOR WIRING:

DHT11 Pin	Arduino Pin
VCC	5V
GND	GND
Data	2

MQ-2 GAS SENSOR WIRING:

MQ-2 Pin	Arduino Pin
VCC	5V
GND	GND
A0	A0

BUZZER AND LED WIRING:

Device	Pin
Buzzer	**5**
LED	**6**

STEP 3: WRITE ARDUINO CODE FOR DATA ACQUISITION AND ALERT SYSTEM

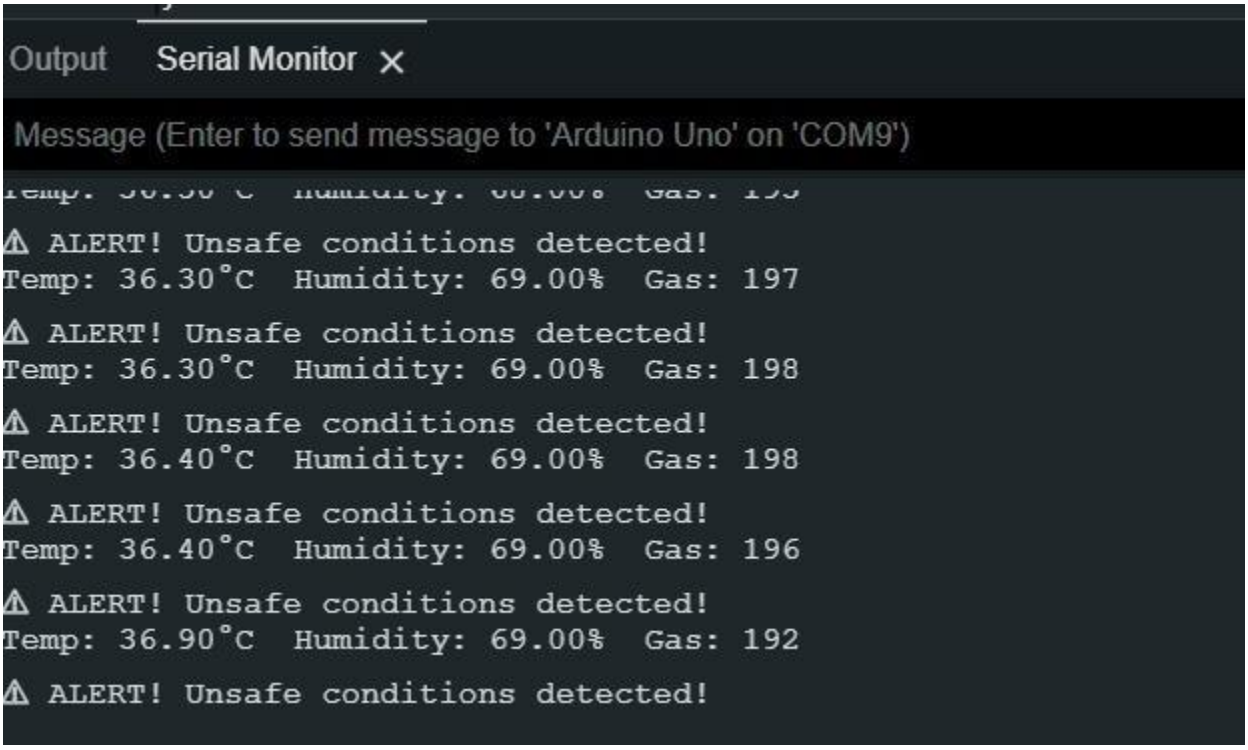
```
#include <DHT.h>

#define DHTPIN 2        // DHT sensor connected to pin 2

#define DHTTYPE DHT11  // Change to DHT22 if using DHT22

DHT dht(DHTPIN, DHTTYPE);

#define GAS_SENSOR A0  // MQ-2 Gas Sensor connected to A0

#define BUZZER 5        // Buzzer on pin 5

#define LED 6          // LED on pin 6

void setup() {

   Serial.begin(9600);

   dht.begin();

   pinMode(GAS_SENSOR, INPUT);

   pinMode(BUZZER, OUTPUT);

   pinMode(LED, OUTPUT);

}
```

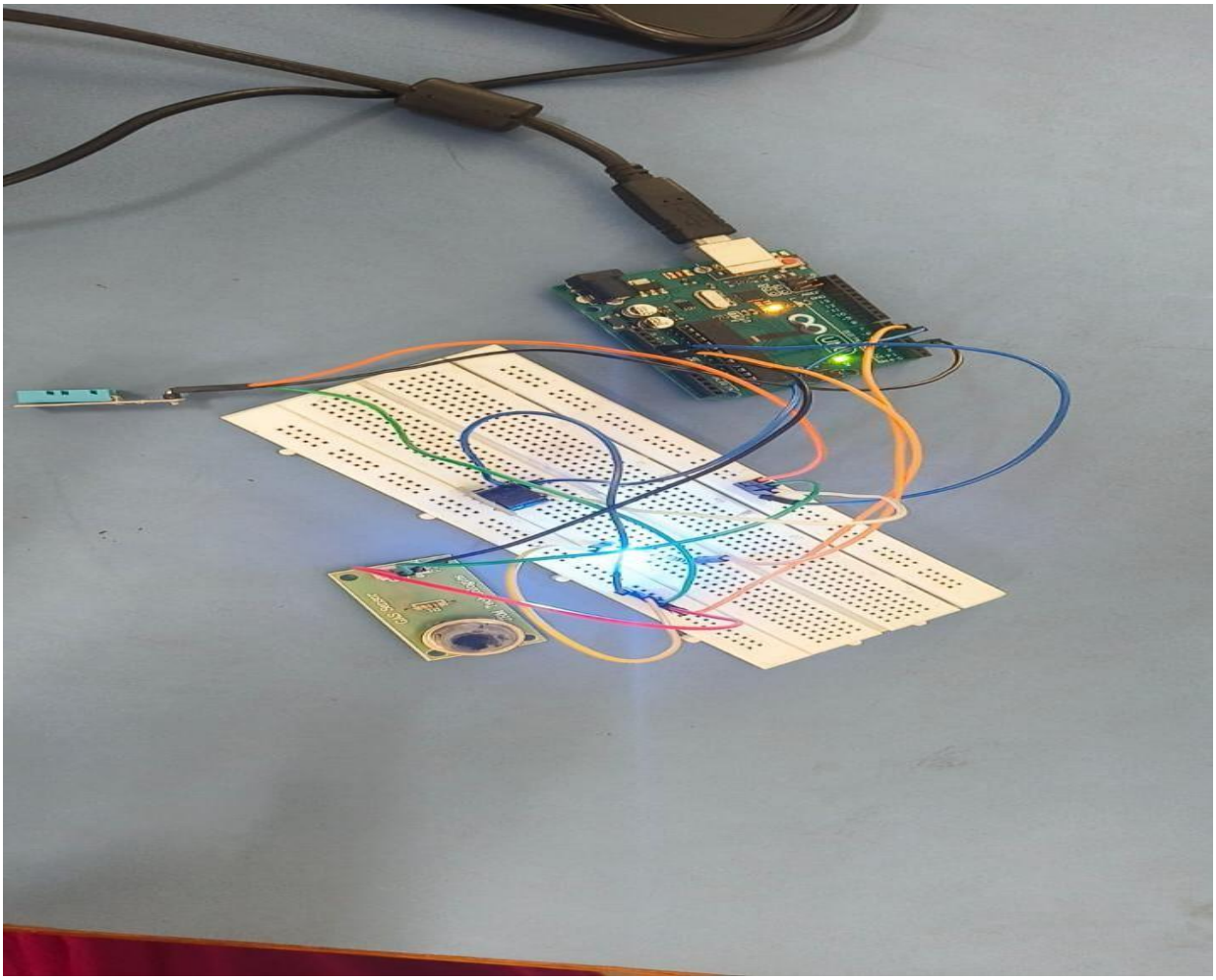
```
void loop() {
  // Read Temperature and Humidity
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  // Read Gas Sensor Value
  int gasValue = analogRead(GAS_SENSOR);
  // Display Sensor Data in Serial Monitor
  Serial.print("Temp: "); Serial.print(temperature); Serial.print("°C ");
  Serial.print("Humidity: "); Serial.print(humidity); Serial.print("% ");
  Serial.print("Gas: "); Serial.println(gasValue);
  // Condition: Temperature > 40°C or Gas Level > 300
  if (temperature > 40 || gasValue > 300) {
    digitalWrite(BUZZER, HIGH);
    digitalWrite(LED, HIGH);
    Serial.println("⚠️ ALERT! Unsafe conditions detected!");
  } else {
    digitalWrite(BUZZER, LOW);
    digitalWrite(LED, LOW);
  }
  delay(2000); // Wait for next reading
}
```

STEP 4: RUN THE SIMULATION

1. Click the "Start Simulation" button in Wokwi.
2. Open the Serial Monitor to observe temperature, humidity, and gas readings.
3. Verify that the LED and buzzer activate when conditions exceed safe limits.

SCREENSHOTS:





RESULT:

Thus, an IoE application was successfully developed using Wokwi to monitor temperature, humidity, and gas concentration, with an integrated alert system using an LED and buzzer.