INDEX

PRACTICAL NO.1

Preparing the IoT Hardware

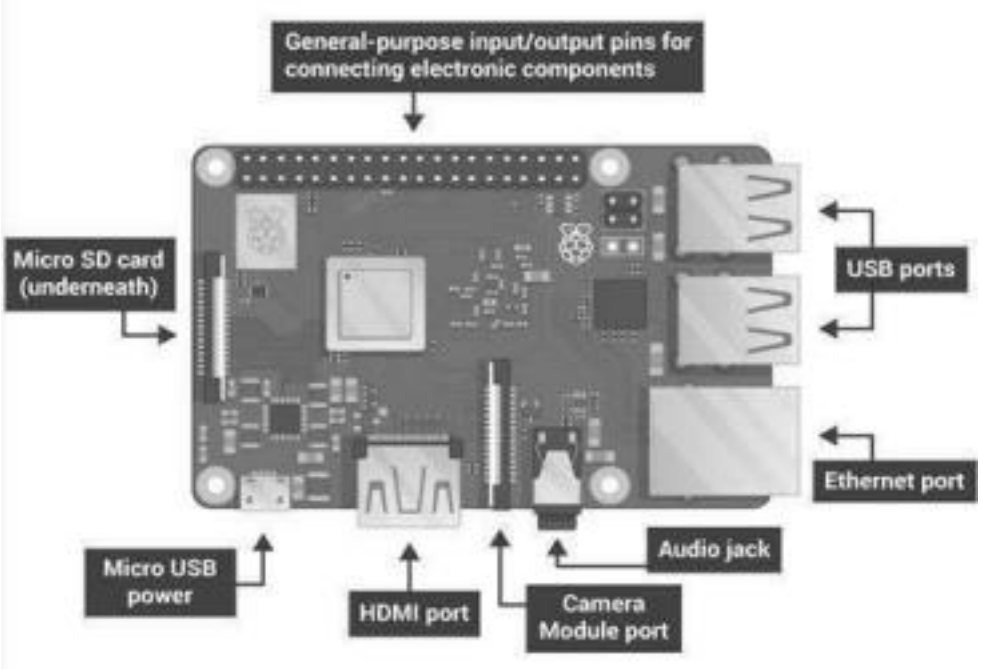
Set up Raspberry Pi OS / Arduino IDE
Configure GPIO settings and test basic connectivity
Demonstrate pin layout and onboard peripherals

Abstract:-
The Raspberry Pi is a mini computer specifically created to make tech learning easier. It has a lot of components for computer-based projects, like USB ports, an Ethernet port, an SD card slot, Wi Fi antenna ports, and more.

Required Components:-Hardware:

Particular	Qty
Raspberry Pi3	1
Power Supply 12V/2Amp	1
USB Keyboard	1
USB Mouse	1
Micro SD card	1
Micro SD USB card reader	1
A Monitor that supports HDMI	1
An HDMI cable	1
An Ethernet cable	1

Software: Raspbian, installed via NOOBS

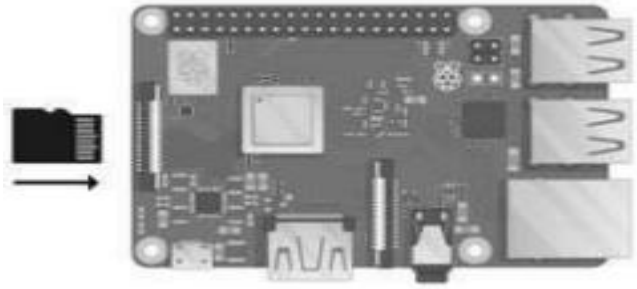


1. **USB ports** - these are used to connect a mouse and keyboard. You can also connect other components, such as a USB drive.
2. **SD card slot** - you can slot the SD card in here. This is where the operating system software and your files are stored.
3. **Ethernet port** - this is used to connect the Raspberry Pi to a network with a cable. The Raspberry Pi can also connect to a network via wireless LAN.
4. **Audio jack** - you can connect headphones or speakers here.
5. **HDMI port**- this is where you connect the monitor (or projector) that you are using to display the output from the Raspberry Pi. If your monitor has speakers, you can also use them to hear sound.
6. **Micro USB power connector** - this is where you connect a power supply. You should always do this last, after you have connected all your other components.
7. **GPIO ports** -these allow you to connect electronic components such as LEDs and buttons to the Raspberry Pi.

Procedure:-

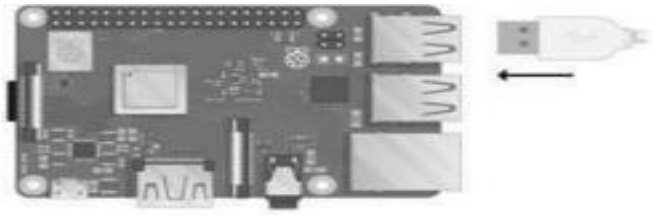
1. Insert the SD card into your SD card reader. Note the drive letter assigned to the SD card. You can see the drive letter in the right hand column of Windows Explorer.
2. Run the Win32DiskImager utility from desktop or menu.
3. Select the image file and click on Write Button.

4. Remove SD Card & Insert in Raspberry Pi

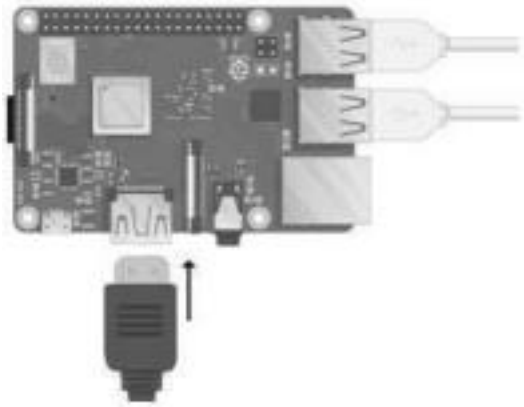


`

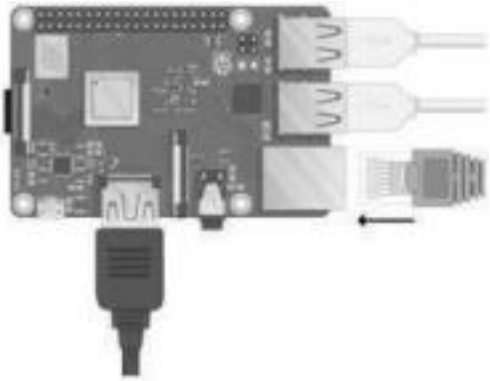
5. Connect the mouse & Keyboard to USB port of Raspberry Pi.



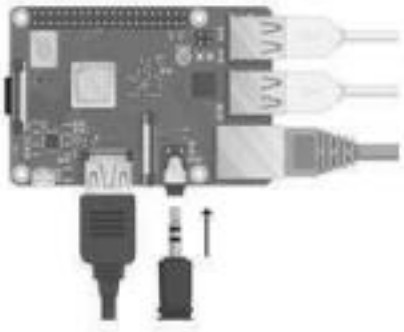
6. Connect Raspberry Pi to HDMI port Directly or use HDMI to VGAconnector.



7. Connect the Pi to the internet via Ethernet, use an Ethernet cable to connect the Raspberry Pi.

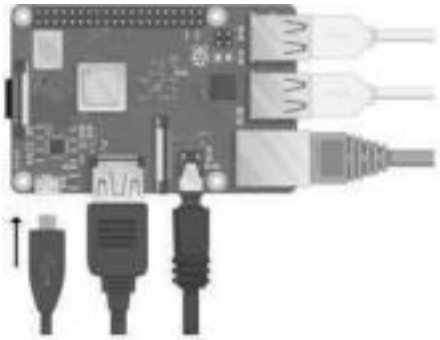


8. Sound will come if it has speakers or connect headphones or speakers to the Audio jack if necessary



1 .

9. Plug the power supply into a socket and connect it to the micro USB power port.



10. Red light will indicate Power On and Green Will Indicate the booting of the Raspberry Pi. The Pi will boot up into a graphical desktop.

11. Configuring your Pi

I. You can control most of your Raspberry Pi's settings, such as the password, through the Raspberry Pi Configuration application found in Preferences on the menu.

II. System

In this tab you can change basic system settings of your Pi.

Password - set the password of the pi user (it is a good idea to change the password from the factory default 'raspberry')
Boot - select to show the Desktop or CLI (command line interface) when your Raspberry Pi starts
Auto Login- enabling this option will make the Raspberry Pi automatically log in whenever it starts
Network at Boot-selecting this option will cause your Raspberry Pi to wait until a network connection is available before starting
Splash Screen -choose whether or not to show the splash (startup) screen when your Raspberry Pi boots
Resolution - you can set the screen resolution here
Underscan - choose whether your Pi should show black bars at the top and bottom of the screen when it can't match the screen resolution
II.I Interfaces
You can link devices and components to the Raspberry Pi using a lot of different types of connections. The Interfaces tab is where you turn these different connections on or off, so that the Pi recognizes that you've linked something to it via a particular type of connection.

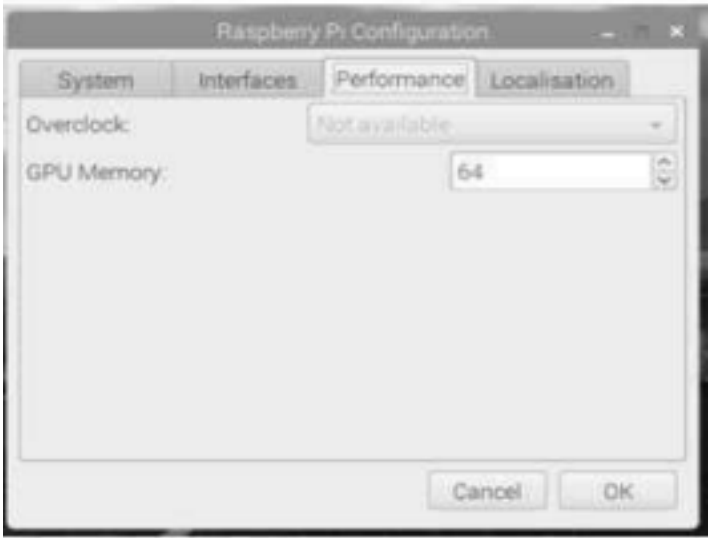
Camera - enable the Raspberry Pi Camera Module
SSH - allow remote access to your Raspberry Pi from another computer using SSH

VNC - allow remote access to the Raspberry Pi Desktop from another computer using VNC

SPI -enable the SPI GPIO pins

I2C - enable the I2C GPIO pins
Serial -enable the Serial (Rx, Tx) GPIO pins
1-Wire - enable the 1-Wire GPIO pin
Remote GPIO - allow access your Raspberry Pi's GPIO pins from another computer
IV. Performance
If you need to do so for a particular project you want to work on, you can change the

performance settings of your Pi in this tab.



Overclock - change the CPU speed and voltage to increase performance
GPU Memory - change the allocation of memory given to the GPU
V. Localisation

12. This tab allows you to change your Raspberry Pi settings to be specific to a country or location.

Locale -set the language, country, and character set used by your Raspberry Pi
Timezone- set the time zone
Keyboard - change your keyboard layout

Wi-Fi Country - set the Wi-Fi country code
13. After starting with Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop up and it will guide through the initial setup.
14. Click Next to start the setup.
15. Set respective Country, Language, and Time zone, then click Next again.
16. Enter a new password for Raspberry Pi and click next
17. Connect to your Wi-Fi network by selecting its name, entering the password, and clicking next
18. Click next let the wizard check for updates to Raspbian and install.

19. Click done or Reboot to finish the setup.
Simulation Environment: TinkerCAD (Free online simulator)
Part A: Basics of Arduino Circuits
Theory:

Arduino is an open-source electronics platform that has gained immense popularity for its ease of use and versatility. It was created in 2005 by a group of Italian engineers and is now maintained and developed by the Arduino community.
The heart of the Arduino platform is a microcontroller, which is a small, programmable computer on a single integrated circuit (IC) chip.
Arduino boards, which house these microcontrollers, provide a user-friendly environment for creating interactive electronic projects, prototypes, and various applications.
Key Components of Arduino:
1. Microcontroller: The core of an Arduino board is the microcontroller. The most commonly used microcontroller in Arduino is the ATmega series from Atmel (now a part of Microchip Technology). These microcontrollers come in different variations and are the brains behind your Arduino projects.
2. Input/output Pins: Arduino boards have a set of digital and analog pins that can be used to read data (inputs) or send data (outputs). Digital pins work with binary signals (0 or 1), while analog pins can read a range of values. The number and types of pins vary among different Arduino board models.
3. Power Supply: Arduino boards can be powered via USB, an external power supply, or a battery. Some boards have built-in voltage regulators, which make them compatible with a range of power sources.
4. USB Port: Arduino boards often feature a USB port for programming and power supply. This allows you to connect the board to your computer and upload code.
5. Reset Button: A reset button is provided to restart the Arduino, allowing you to upload new code or reset the program.
6. LED Indicator: Many Arduino boards include a built-in LED (Light Emitting Diode) on pin 13, which can be used for testing and basic visual feedback.

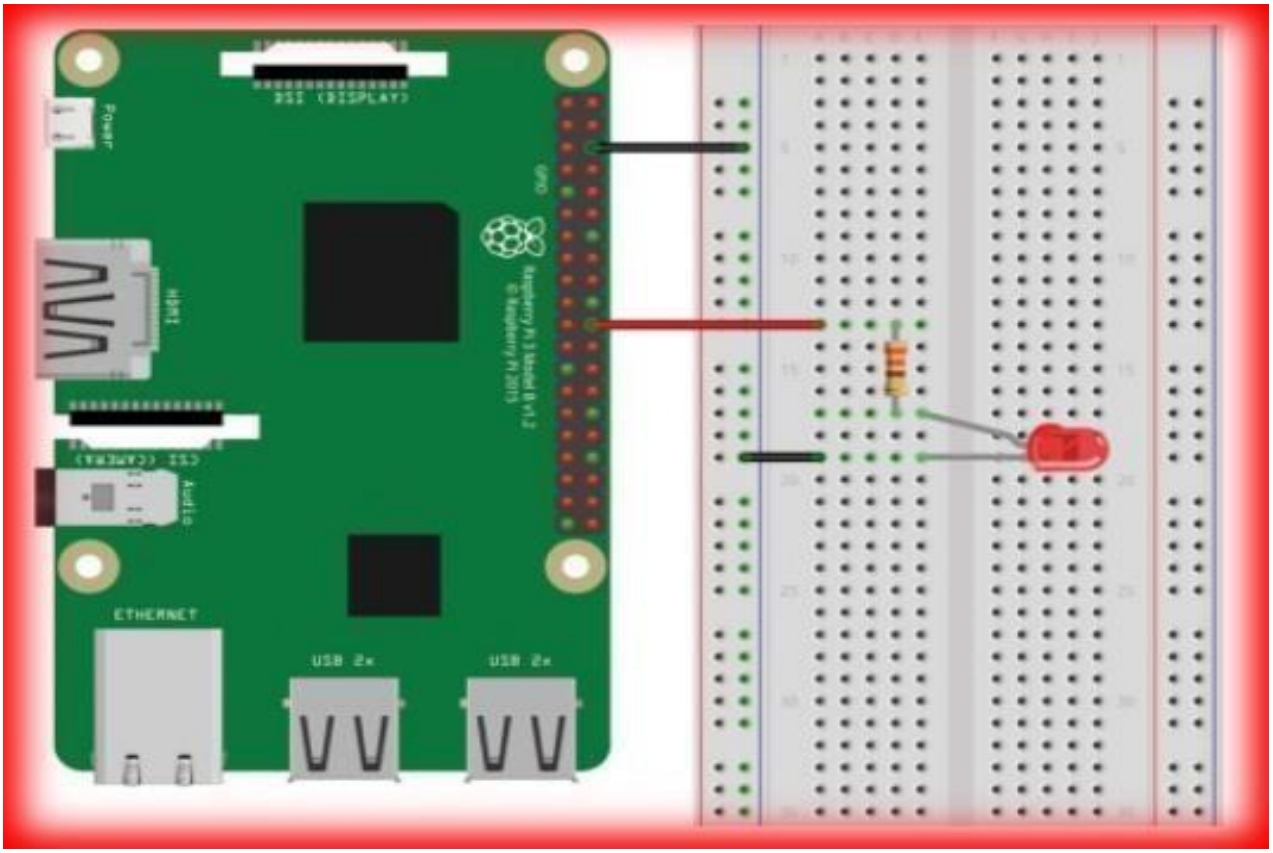
Arduino Software:
The Arduino platform comes with its integrated development environment (IDE). The Arduino IDE is a software tool that allows you to write, compile, and upload code to the Arduino board. Key features of the IDE include:
- Programming Language: Arduino uses a simplified version of the C/C++ programming language. It provides a set of libraries and functions tailored for easy interaction with the hardware.
- Code Library: Arduino has a vast library of pre-written code and functions that simplify common tasks, making it accessible to beginners.
- Serial Monitor: The IDE includes a serial monitor that allows you to communicate with the Arduino board and view debugging information. -
Community Support: The Arduino community is large and active, offering forums, tutorials, and extensive documentation to help users troubleshoot issues and learn.
Conclusion:-
Thus Raspbian OS was installed, Raspberry Pi Components and Interface were studied and implemented, Raspberry was also Connected to ethernet, Monitor, USB.

PRACTICAL NO.2

GPIO – Light the LED

Blink LED using Python (Raspberry Pi) or C++
Add a push button to toggle LED ON/OFF

```
import RPi.GPIO as GPIO ## Import GPIO library
import time
GPIO.setmode(GPIO.BOARD) ## Use board pin numbering
GPIO.setup(11, GPIO.OUT) ## Setup GPIO Pin 11 to OUT
while True:
GPIO.output(11,1) ## Turn on Led
time.sleep(1) ## Wait for one second
GPIO.output(11,0) ## Turn off Led
time.sleep(1) ## Wait for one second
```



USING PUSH BUTTON

```
int pbuttonPin = 2; // connect output to push button
int LEDpin = 13; // Connection to LED

int val = 0; // push value from pin 2
int lightON = 0; //light status
int pushed = 0; //push status

void setup() {
  // put your setup code here, to run once:

Serial.begin(115200);
pinMode(pbuttonPin, INPUT);
pinMode(LEDpin, OUTPUT);
digitalWrite(LEDpin, HIGH);// keep the load off at the beginning. if
on is better, change HIGH to LOW

}
```

Conclusion:-

Thus we have studied and displayed different LED patterns with Raspberry Pi.

PRACTICAL NO.3

SPI Interface – Camera Module Integration

Connect a Pi camera module (or SPI camera for Arduino)
Capture an image or short video
Store file or stream it locally

Software & Code (General Steps)
The specific code depends heavily on the exact camera module (e.g.OV7670, Arducam SPI camera) and the host platform's libraries.
Capture an Image

1. Initialize Communication: Start the SPI interface and communicate with the camera to verify connection and configure settings like resolution and color depth.
2. Trigger Capture: Send a command to the camera module to begin the image acquisition process.
3. Read Data (Transfer): The host microcontroller (Pi or Arduino) reads the image data, pixel by pixel, via the MISO line. This is often done in a buffer.
4. Process and Store: Reassemble the data into a usable image format (like a BMP or JPEG file) on an SD card or transfer it over a network.

Store or Stream Locally

- Store File: Use an SD card module interfaced via the SPI bus (or a separate channel) to write the captured data directly to a file system. Libraries such as the Arduino SD library or Python's built-in file handling are necessary for this.
- Stream Locally: To stream, you would continuously capture frames and send them over a local connection, such as Wi-Fi or Ethernet, using protocols like HTTP or UDP. This is often challenging with pure SPI cameras due to limited bandwidth and processing power.

```
libcamera-still -o ~/Desktop/image.jpg
libcamera-vid -t 10000 -o ~/Desktop/video.h264
```

Conclusion-Thus we have captured images and videos.

PRACTICAL NO.4

8x8 LED Grid Control (Matrix LED Programming)

Connect an 8×8 LED matrix module
Program animations or scrolling text patterns
Explore logical formulas for patterns

Connect 8x8 led matrix board to Connector No 6 (CN 6) of Discover Development Board.

Connect one end of FRC cable to Connector No 1 (CN 1) of Discover Development Board and the other end to GPIO pins of Raspberry Pi in such a way that the arrow must point towards the SD Card and Pin No 1 of Raspberry Pi.

Connect Power Supply to Connector No 8 (CN 8) of Discover Development Board. (Check positive and negative and don't switch ON power supply until all connections are connected properly)

Connect keyboard, mouse and HDMI cable

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import re
import time
import argparse
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT,
SINCLAIR_FONT, LCD_FONT
def demo(n, block_orientation, rotate, msg):
# create matrix device
serial = spi(port=0, device=0, gpio=noop())
```

device = max7219(serial, cascaded=n or 1, block_orientation=block_orientation,

rotate=rotate or 0)

show_message(device, msg, fill="white", font=proportional(LCD_FONT),

scroll_delay=0.1)

time.sleep(3)

pass

if name == " main ":

try:

text_display = raw_input("Enter message to be display on 8x8 matrix = ")

demo(1, 0, 0, text_display)

except KeyboardInterrupt:

pass

finally:

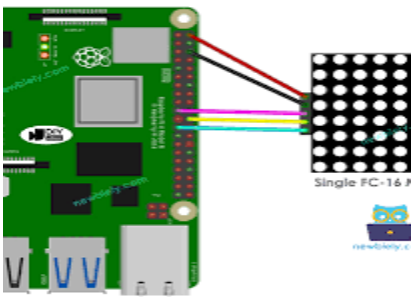
print "Program exit..."

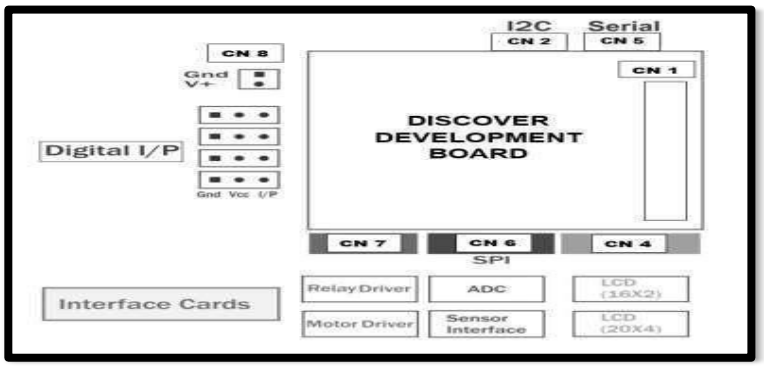
This project deals with displaying different LED patterns using module MAX7219 that uses SPI interface.

Required Components:-

Particular Qty

Raspberry Pi3 1,MAX7219 1,Discover Board 1,power Supply 1





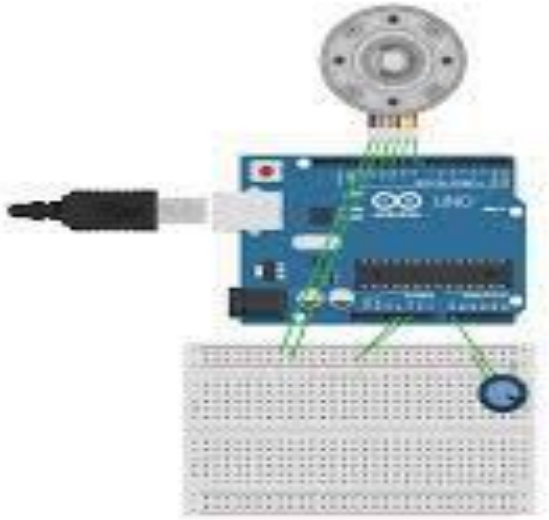
Conclusion- Thus we have studied and displayed different LED patterns with Raspberry Pi.

PRACTICAL NO.5

PWM – Stepper Motor Control

Interface a stepper motor using a motor driver
Control direction and vary speed using PWM signals
Observe effect of duty cycle changes on motor movement

```
#include <Stepper.h>
const int stepsPerRevolution = 200;
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);
int stepCount = 0;
void setup() {
}
void loop()
{
  int sensorReading = analogRead(A0);
  int motorSpeed = map(sensorReading, 0, 1023, 0, 100);
  if (motorSpeed > 0) {
    myStepper.setSpeed(motorSpeed);
    myStepper.step(stepsPerRevolution / 100);
  }
```



Conclusion-Thus we have studied stepper motor.

PRACTICAL NO.6

Node-RED for IoT Dashboard

Install and configure Node-RED on Raspberry Pi
Create a flow to turn LED ON/OFF via browser

Controlling an LED via a web browser using Node-RED involves installing Node-RED, adding the dashboard module, and creating a flow with UI elements and Raspberry Pi GPIO nodes.

Hardware Prerequisites

- Raspberry Pi (running Raspberry Pi OS)
- LED
- 220 Ohm resistor (to limit current)
- Breadboard and jumper wires
- A second computer or a browser on the Pi itself to access the Node-RED editor

Step 1: Install and Configure Node-RED on Raspberry Pi

While Node-RED might be pre-installed on some versions of Raspberry Pi OS, it's best to use the official script to ensure you have the latest version and the necessary Pi-specific nodes.

1. Open the terminal on your Raspberry Pi (either directly or via SSH).
2. Run the installation script:
3. bash

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

4. Follow the prompts, including the option to install Pi-specific nodes, by pressing `Y` and `Enter`.
5. Start the Node-RED service:
6. bash

node-red-start

7. (Optional: To have Node-RED start automatically on boot, run `sudo systemctl enable nodered.service`).
8. Access the editor: Open a web browser and navigate to `http://<your_pi_ip_address>:1880`. You can find your Pi's IP address by running `hostname -I` in the terminal.

Step 2: Install the Node-RED Dashboard Module

The dashboard nodes are required to create a web-based user interface.

1. In the Node-RED editor, click the menu icon (three horizontal lines) in the top-right corner.
2. Select Manage palette, then go to the Install tab.
3. Search for `node-red-dashboard` and click the install button next to the official package.
4. After installation, the new dashboard nodes will appear on the left-side palette under the `dashboard` section.

Step 3: Create a Flow to Turn an LED ON/OFF via Browser

This flow will use a dashboard switch to control a physical LED connected to a GPIO pin.

Hardware Connection:

Connect the long leg of the LED (anode) to a 220 Ohm resistor, and the other end of the resistor to a GPIO pin (e.g., GPIO17, physical pin 11). Connect the short leg of the LED (cathode) to a GND pin (e.g., physical pin 6).

Node-RED Flow Configuration:

1. Add a `ui_switch` node: Drag a Switch node from the `dashboard` section (not the `function` section) onto your flow.
    1. Double-click the node to configure it.
    2. In the Group dropdown, click the pencil icon to create a new UI group, then create a new Tab (e.g., "IoT Dashboard").
    3. Name the group (e.g., "LED Control") and name the switch "LED ON/OFF". Click

Update and then Done.

4. Ensure the "On Value" is set to `true` (boolean) and the "Off Value" is set to `false` (boolean).

2. Add a `rpi-gpio` output node: Scroll down the palette to the `Raspberry Pi` section and drag an rpi-gpio output node (the one with the raspberry icon on the right) onto the flow.
    1. Double-click the node to configure it.
    2. Change the Pin to GPIO17 (or whichever pin you used).
    3. Set the Type to "Digital output".
    4. Give it a name, e.g., "Physical LED", and click Done.
3. Wire the nodes: Connect the output port (grey dot on the right) of the Switch node to the input port (grey dot on the left) of the Physical LED node.
4. Deploy the flow: Click the red Deploy button in the top-right corner.
5. Access the dashboard: In your browser, open a new tab or window and go to `http://<your_pi_ip_address>:1880/ui`.

You will now see a switch on your dashboard. Toggling the switch will send `true` or `false` messages to the GPIO pin, turning the physical LED on and off.

Conclusion-Thus we have studied NODE-RED

PRACTICAL NO.7

Sensor Integration – Analog & Digital Sensors

Aim:

To study the working of Light sensor using Arduino

Simulation Environment: TinkerCAD (Free online simulator)

Components: Arduino UNO, LED, Photodiode and Resistors

Theory:

The goal of this practical is to create a system that can automatically control the brightness of an LED based on the light detected by a photodiode. This project leverages the principles of light sensing and feedback control. **Components:**

a) Photodiode: A photodiode is a light-sensitive semiconductor device that generates a current or voltage proportional to the incident light's intensity. It acts as the input sensor in this system.

b) LED: An LED (Light Emitting Diode) is used as the output device. It emits light and can be controlled to vary its brightness.

c) Arduino: The Arduino microcontroller is the brain of the project. It reads data from the photodiode, processes it, and controls the LED's brightness accordingly.

Working:

a) Photodiode Operation:

The photodiode is connected to one of the Arduino's analog input pins. When exposed to light, the photodiode generates a current or voltage that is directly proportional to the light intensity.

Arduino reads the analog voltage from the photodiode using one of its analog pins.

b) Control Algorithm:

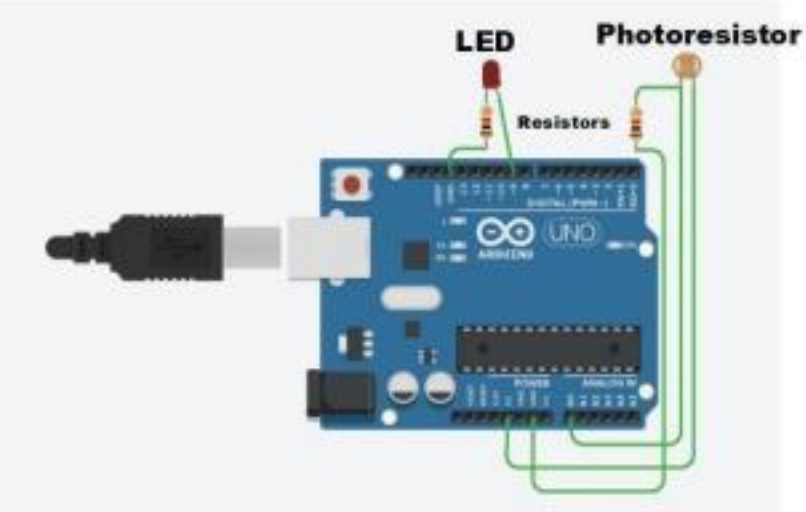
The Arduino is programmed with an algorithm that translates the analog reading from the photodiode into a control signal for the LED. The algorithm typically involves mapping the photodiode's output to the LED's brightness. For example, when the photodiode detects more light, the LED becomes brighter, and vice versa.

c) Feedback Loop:

The system operates in a feedback loop. As light conditions change, the photodiode detects the variations and sends this information to the Arduino. The Arduino processes the data and adjusts the LED's brightness in real time based on the input from the photodiode.

This closed-loop system ensures that the LED's brightness is always synchronized with the surrounding light levels.

Circuit Diagram:



Pin Connections:

Arduino	**Photoresistor**	**LED**
5V	Right pin	
GND (Power)	Left pin through a Series Resistor	
A0	Left pin	
Pin 9		Anode
GND (Digital)		Cathode through a Series Resistor

6

Code: The following C++ code is used in the given case

```
int lightSensorValue = 0;
void setup()
{
pinMode(A0, INPUT);
pinMode(9, OUTPUT);
Serial.begin(9600);
}
void loop()
{
lightSensorValue = analogRead(A0);
Serial.println(lightSensorValue);
int ledBrightness = map(lightSensorValue, 0, 1023, 0, 255);
analogWrite(9, ledBrightness);
delay(100);
}
```

Aim:

To study the working of Temperature sensors using Arduino

Simulation Environment: TinkerCAD (Free online simulator)

Components: Arduino UNO, Temperature Sensor TMP 36

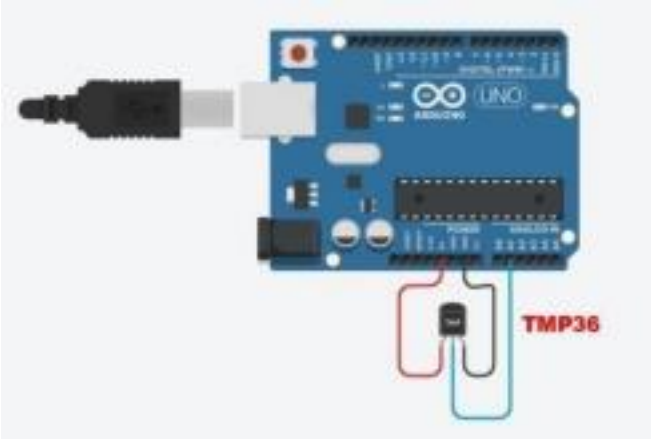
Theory:

The TMP36 is a low-cost analog temperature sensor that can be easily integrated with Arduino boards. It provides an analog voltage output that varies linearly with temperature. This practical aims to show how to measure and display real-time temperature data using a TMP36 temperature sensor and an Arduino. The temperature data will be displayed through suitable method.

The TMP36 temperature sensor is a precision analog sensor. It generates an output voltage that is linearly proportional to the Celsius temperature. It typically has three pins: VCC, GND, and OUT. The sensor's output voltage increases by 10 mV per degree Celsius. At 25°C, it outputs 750 mV.

The demonstration showcases the practical application of the TMP36 temperature sensor in conjunction with an Arduino board for real-time temperature monitoring. It highlights how to interface the sensor, read its analog output, and display the temperature information. This knowledge can be applied to various temperaturesensing applications, including weather stations, environmental monitoring, and more.

Circuit Diagram:




```
}
void loop()
{
int tmp = analogRead(sensor);
```

Arduino	TMP36 Sensor
5V	Left pin
GND	Right pin
A1	Center pin

```
char degree = 176;
const int sensor = A1;
void setup()
{
pinMode(sensor, INPUT);
Serial.begin(9600);
float voltage = (tmp*5.0)/1024; float
tmpCel = (voltage-0.5)* 100.0;
Serial.print("Celsius:");
Serial.print(tmpCel);
Serial.println(degree);
delay(1000);
}
```

Aim:

To study the working of Humidity sensors using Arduino

Simulation Environment: TinkerCAD (Free online simulator)

Components: Arduino UNO, Potentiometer (wiper)

Theory:

Potentiometer as a Sensor:

A potentiometer, often referred to as a "pot," is a variable resistor with three terminals. It consists of a resistive track and a wiper that moves along the track. By adjusting the wiper's position, you can vary the resistance.

In this demonstration, the potentiometer is used to simulate a variable sensor input.

Arduino:

Arduino is a versatile microcontroller platform commonly used for various electronic projects.

It can read analog voltage levels from sensors, including potentiometers, and convert them into digital values for processing.

TinkerCAD:

TinkerCAD is a web-based platform for simulating and designing electronic circuits and Arduino- based projects.

It's an excellent tool for testing and prototyping virtually, even when physical components are unavailable.

Demo Overview:

In this demo, we learn how to connect a potentiometer to an Arduino board in the TinkerCAD environment.

We understand the wiring and connections required to read variable resistance values from the potentiometer accurately.

Programming:

We see how to write the code to read and convert the analog voltage from the potentiometer into digital values and the humidity.

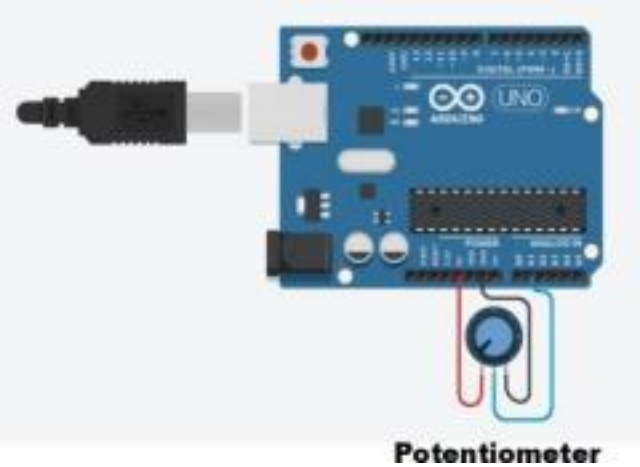
Practical Applications:

While the potentiometer doesn't directly measure humidity, we observe how variable sensor inputs are used in applications like volume control, dimmer switches, and other scenarios where adjustable values are required. The demo provides hands-on experience in interfacing a potentiometer with an Arduino, which can be a valuable skill for various electronic projects. Understanding how to read analog values and convert them into digital format is a fundamental aspect of working with sensors and input devices.

The demo serves as a practical example of using a potentiometer in an Arduino project and its potential applications in real-world scenarios.

While the potentiometer isn't a humidity sensor, this demonstration can still be educational and relevant, regarding interfacing variable sensors with Arduino.

Circuit Diagram:



Pin Connections:

	Potentiometer
5V	Left pin
GND	Right pin
A1	Center pin

Code:

```
/*
This code records the humidity using a simulated potentiometer. The Humidity is simulated by mapping the potentiometer output into percentages. */

const int analogIn = A1;
int humiditySensorOutput = 0;

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

void loop()
{
```

```
humiditySensorOutput = analogRead(analogIn);
int humidityPercentage = map(humiditySensorOutput, 0, 1023, 10,70);

Serial.print("Humidity: ");
Serial.print(humidityPercentage);
Serial.println("%");
delay(5000);
 }
```

Aim:

To detect smoke/fire using Gas sensor

Simulation Environment: TinkerCAD (Free online simulator)
Components: Arduino UNO, Gas sensor, LED, resistor and Breadboard

Theory:

Gas sensors are devices designed to detect and measure the concentration of gases in the surrounding environment. They are widely used in various applications, including industrial safety, environmental monitoring, medical diagnostics, and home automation. Gas sensors play a crucial role in ensuring the safety of individuals and detecting potential hazards.

The working principle of gas sensors can vary depending on the type of sensor and the specific gas it is designed to detect.
The basic principle used in a smoke detector, whether in a real-world device or a simulated one in Tinkercad, is the change in electrical conductivity or resistance in the presence of smoke particles.
The principle can be understood through the following steps:

1. Gas Sensing Element:
   - In a real smoke detector, a specialized gas sensing element is used. This element often consists of a material that interacts with smoke particles in the air.

2. Change in Conductivity or Resistance:
   - When smoke particles are present, they interfere with the normal operation of the gas sensing element. This interference leads to a change in the electrical conductivity or resistance of the sensing element.

3. Voltage Divider Circuit:
   - The gas sensor is typically part of a voltage divider circuit. In the case of a simulated circuit in TinkerCAD, a variable resistor is often used to represent the gas sensor.

4. Arduino Interface:
   - The output of the voltage divider circuit is connected to an analog pin on an Arduino. The Arduino

reads the analog value, which corresponds to the resistance or conductivity of the gas sensor.

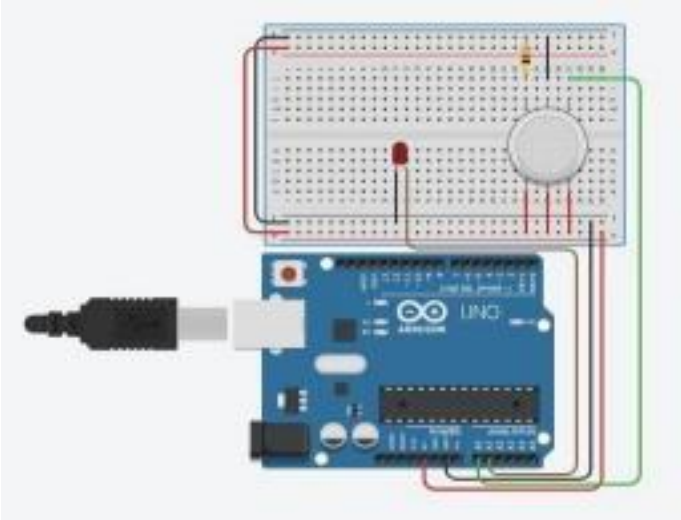
5. Threshold Detection:

- A threshold value is set in the Arduino code. If the analog value exceeds this threshold, it indicates that the resistance of the gas sensor has changed significantly, suggesting the presence of smoke.

6. Alarm Activation:

- When the threshold is surpassed, the Arduino activates an alarm signal. In the Tinkercad simulation, this is often represented by turning on an LED.

Circuit Diagram:



Arduino	**Gas Sensor**	**LED**	**Resistor**
5V	**B1**		
	B2		
	B3		
GND	**H1**	**Cathode**	**Other End**
	A1		**One End**
A0	**A2**		
A1		**Anode**	

Code:

```
int LED = A1;
const int gas= 0;
 int
MQ2pin =
A0;
void setup()
 {
Serial.begin(9600);
 }

void loop()
{
float
sensorValue,MQ2pin;
sensorValue = analogRead(MQ2pin);

if(sensorValue>= 470)
{
digitalWrite(LED,LOW);
Serial.print(sensorValue);
Serial.println(" |SMOKE DETECTED"); }
 else
  {
 digitalWrite(LED,HIGH);
Serial.println("Sensor     Value:     ");
Serial.println(sensorValue);
  }
 delay(1000);
}
float getsensorValue(int
          pin){ return
          (analogRead(pin));
}
```

Conclusion-Thus we have studied different analog and digital sensors.

PRACTICAL NO.8

Integration – Smart Monitoring System

```
#include <LiquidCrystal.h>
#include <Servo.h>
Servo motor1;
Servo motor2;
int pos = 0;
int pir = A4;
LiquidCrystal lcd(2,4,7,8,12,13);
const int trigpin = 10;
const int echopin = 9;
int gas_sensor = A3;
const int temp_sensor = A5;
const int photo_resistor = A2;

int bulb1 = 5;
int bulb2 = 6;
int fan1 = 11;
int fan2 = 3;
byte automatic[] = {
  B00000,
  B00000,
  B01110,
  B10001,
  B10101,
  B10001,
  B01110,
  B00000
};

int button1 = 3;
int button2 = 5;
int button3 = 6;
int button4 = 11;
double distancecm;
double duration;
double distanceinch;
```

```
float temp_reading;
float temp;
float temp_final;
int i;

void setup()
{
  pinMode(trigpin,OUTPUT);
  pinMode(echopin,INPUT);
    motor1.attach(A0);
    motor2.attach(A1);
    motor1.write(pos);
    motor2.write(pos);
  lcd.createChar(0,automatic);
  lcd.begin(16,2);
  lcd.setCursor(4,0);
  lcd.print("Complete");
  lcd.setCursor(3,1);
  lcd.print("Automation ");
  lcd.write(byte(0));
  delay(1000);
  lcd.clear();
  Serial.begin(9600);
}
void loop()
{
  digitalWrite(trigpin,LOW);
  delayMicroseconds(2);
  digitalWrite(trigpin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin,LOW);
  duration = pulseIn(echopin,HIGH);
   distancecm = (duration*0.034)/2;
  distanceinch = (duration*0.0133)/2;
 // Serial.print("Distance in cm =");
 // Serial.println(distancecm);

  int analog_sensor = analogRead(gas_sensor);
  temp_reading = analogRead(temp_sensor);
  temp = (temp_reading/1023)*5000;
```

```
 temp_final = (temp - 500)/10;
// Serial.print("Temperature : ");
// Serial.println(temp_final);
// Serial.print("Sensor : ");
 Serial.println(analog_sensor);

 int photo = analogRead(photo_resistor);
// Serial.println(photo);

// analogWrite(A4,255);
// analogWrite(fan2,64);

 if(photo <= 425 & distancecm <= 305 & analog_sensor < 180)
 {
   lcd.clear();
   analogWrite(bulb1,255);
   analogWrite(bulb2,255);
  lcd.setCursor(0,0);
 lcd.print("BL1");
 lcd.print(" BL2");
 lcd.print(" FN1 ");
 lcd.print("FN2");
 lcd.setCursor(0,1);
 lcd.print("100");
 lcd.setCursor(4,1);
 lcd.print("100");
   delay(400);
 }
 else if(photo > 425 & photo <=517 & distancecm <= 305 & analog_sensor < 180)
 {
   lcd.clear();
   analogWrite(bulb1,150);
   analogWrite(bulb2,150);
   lcd.setCursor(0,0);
 lcd.print("BL1");
 lcd.print(" BL2");
 lcd.print(" FN1 ");
 lcd.print("FN2");
 lcd.setCursor(0,1);
 lcd.print("75");
```

```
lcd.setCursor(4,1);
lcd.print("75");
  delay(400);
}
 else if(photo > 517 & photo <=574 & distancecm <= 305 & analog_sensor < 180)
{
  lcd.clear();
  analogWrite(bulb1,110);
  analogWrite(bulb2,110);
lcd.setCursor(0,0);
lcd.print("BL1");
lcd.print(" BL2");
lcd.print(" FN1 ");
lcd.print("FN2");
lcd.setCursor(0,1);
lcd.print("50");
lcd.setCursor(4,1);
lcd.print("50");
  delay(400);
}
 else if(photo > 574 & photo <=630 & distancecm <= 305 & analog_sensor < 180)
{
  lcd.clear();
  analogWrite(bulb1,64);
  analogWrite(bulb2,64);
lcd.setCursor(0,0);
lcd.print("BL1");
lcd.print(" BL2");
lcd.print(" FN1 ");
lcd.print("FN2");
lcd.setCursor(0,1);
lcd.print("25");
lcd.setCursor(4,1);
lcd.print("25");
  delay(400);
}
 else if(photo > 630 & distancecm <= 305 & analog_sensor < 180)
{
  lcd.clear();
  analogWrite(bulb1,0);
```

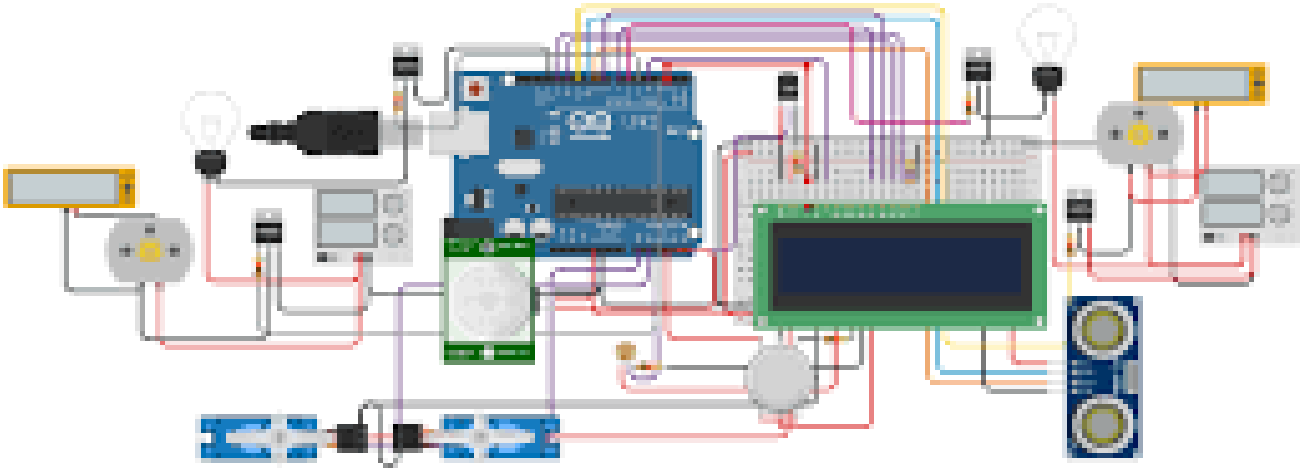
```
  analogWrite(bulb2,0);
 lcd.setCursor(0,0);
 lcd.print("BL1");
 lcd.print(" BL2");
 lcd.print(" FN1 ");
 lcd.print("FN2");
 lcd.setCursor(0,1);
 lcd.print("OFF");
 lcd.setCursor(4,1);
 lcd.print("OFF");
  delay(400);
 }
 else if(distancecm > 305  & analog_sensor < 180)
 {
  lcd.clear();
  analogWrite(bulb1,0);
  analogWrite(bulb2,0);
   analogWrite(fan1,0);
  analogWrite(fan2,0);
 lcd.setCursor(3,0);
 lcd.print("System off");

 lcd.setCursor(1,1);
 lcd.print("Nobody in Room");
// lcd.print("OFF ");
  delay(400);
 }

 else if(analog_sensor >= 180)
 {
  lcd.clear();
   analogWrite(bulb1,255);
  analogWrite(bulb2,255);
   analogWrite(fan1,255);
  analogWrite(fan2,255);
  motoropen();

   lcd.setCursor(3,0);
 lcd.print("DANGER");
```

```
 lcd.setCursor(0,1);
 lcd.print("EXIT FROM ROOM");
// lcd.print("OFF ");
  delay(400);
 }
 if(temp_final >= 20 & temp_final < 30 & distancecm <= 305 & analog_sensor < 180)
 {
  lcd.clear();
  analogWrite(fan1,64);
  analogWrite(fan2,64);
 lcd.setCursor(0,0);
 lcd.print("BL1");
 lcd.print(" BL2");
 lcd.print(" FN1 ");
 lcd.print("FN2");
 lcd.setCursor(8,1);
 lcd.print("25");
 lcd.setCursor(12,1);
 lcd.print("25");
  delay(400);
 }
 else if(temp_final >= 30 & temp_final < 45  & distancecm <= 305 & analog_sensor < 180)
 {
   lcd.clear();
  analogWrite(fan1,128);
  analogWrite(fan2,128);
 lcd.setCursor(0,0);
 lcd.print("BL1");
 lcd.print(" BL2");
 lcd.print(" FN1 ");
 lcd.print("FN2");
 lcd.setCursor(8,1);
 lcd.print("50");
 lcd.setCursor(12,1);
 lcd.print("50");
  delay(400);
 }
 else if(temp_final >= 45 & temp_final < 60  & distancecm <= 305 & analog_sensor < 180)
 {
   lcd.clear();
```

```
  analogWrite(fan1,150);
  analogWrite(fan2,150);
lcd.setCursor(0,0);
lcd.print("BL1");
lcd.print(" BL2");
lcd.print(" FN1 ");
lcd.print("FN2");
lcd.setCursor(8,1);
lcd.print("75");
lcd.setCursor(12,1);
lcd.print("75");
  delay(400);
}
 else if( temp_final < 0  & distancecm <= 305 & analog_sensor < 180)
{
   lcd.clear();
  analogWrite(fan1,0);
  analogWrite(fan2,0);
lcd.setCursor(0,0);
lcd.print("BL1");
lcd.print(" BL2");
lcd.print(" FN1 ");
lcd.print("FN2");
lcd.setCursor(8,1);
lcd.print("OFF");
lcd.setCursor(12,1);
lcd.print("OFF");
  delay(400);
}
 else if(temp_final >= 60 & distancecm <= 305 & analog_sensor < 180)
{
   lcd.clear();
  analogWrite(fan1,255);
  analogWrite(fan2,255);
lcd.setCursor(0,0);
lcd.print("BL1");
lcd.print(" BL2");
lcd.print(" FN1 ");
lcd.print("FN2");
lcd.setCursor(8,1);
```

```
 lcd.print("100");
 lcd.setCursor(12,1);
 lcd.print("100");
  delay(400);
 }


/* if(analog_sensor > 180)
 {
   lcd.clear();
  analogWrite(fan1,255);
  analogWrite(fan2,255);
  analogWrite(bulb1,255);
  analogWrite(bulb2,255);
  lcd.setCursor(0,0);
  lcd.print("Danger....");
  motoropen();
 // delay(400);
 }*/
  int pirvalue = digitalRead(pir);
 if(pirvalue == HIGH)
 {
   lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Somebody Enter");
  lcd.setCursor(0,1);
  lcd.print("Gates - OPENING");
  motoropen();
  delay(400);
 }
 else if(analog_sensor < 180)
 {
  motorclose();
 }



 }

void motoropen()
```

```
{
 for(;pos <= 90;pos+=10)
 {
  motor1.write(pos);
   motor2.write(pos);
   delay(80);
 }
}
void motorclose()
{
 for(;pos>=0;pos-=10)
 {
  motor1.write(pos);
   motor2.write(pos);
   delay(80);
 }
}
```

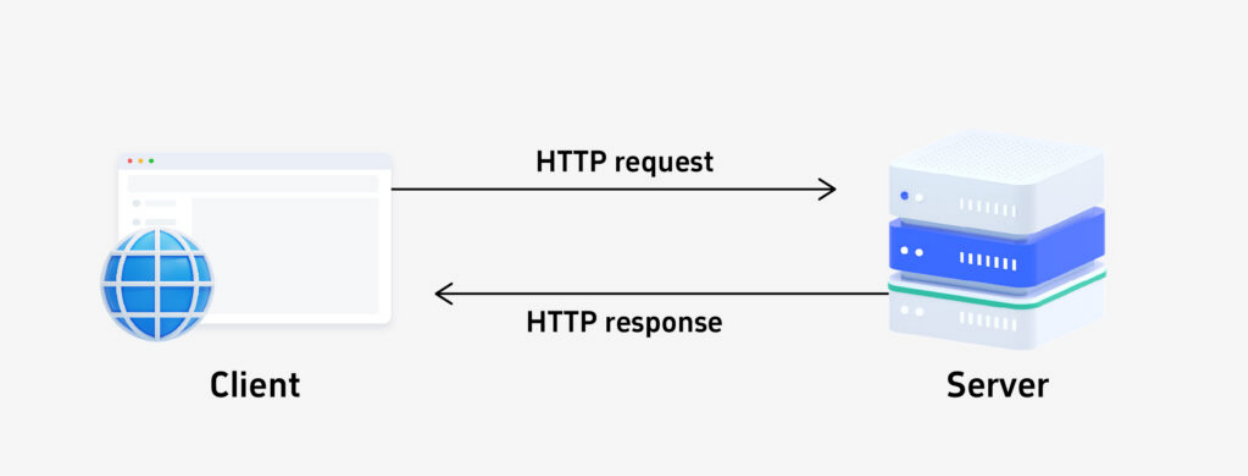


Conclusion-Thus we have studied smart monitoring system(home automation)

PRACTICAL NO.9

MQTT & HTTP PROTOCOL

HTTP PROTOCOL

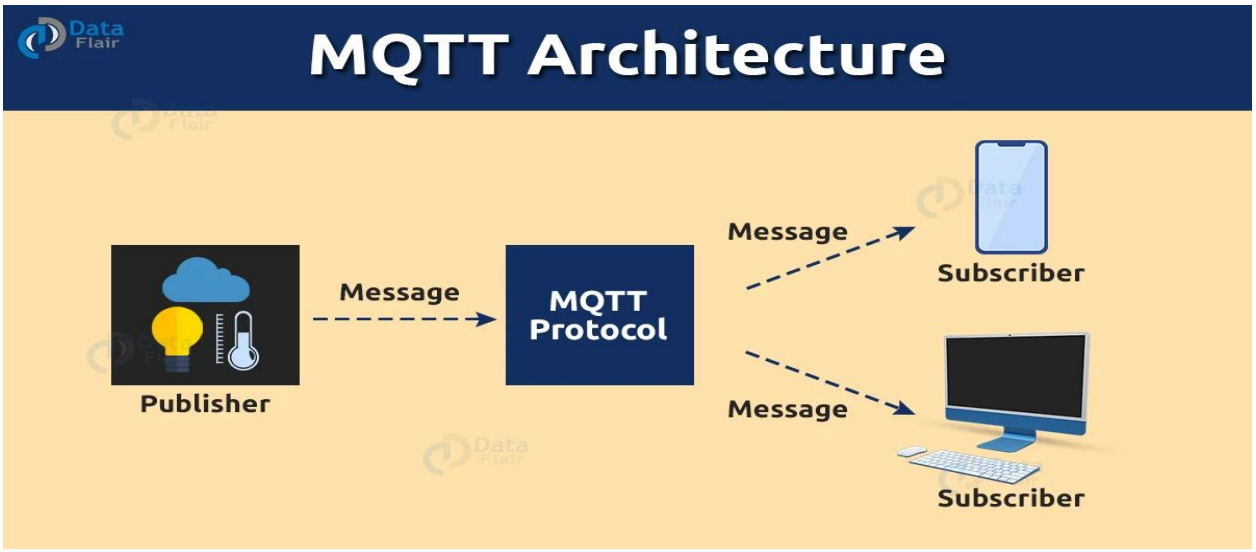


In IoT, the HTTP protocol allows devices to communicate with servers using a standard client-server model, where devices act as clients sending requests and servers responding with data

How it works

- Client-Server Architecture: An IoT device sends an HTTP request to a server, which processes the request and sends back an HTTP response.
- Requests: Devices send requests to perform actions, such as fetching sensor data (GET) or sending a sensor reading to a server (POST).
- Responses: The server responds to the device with a status code (like 200 OK) and the requested data, which can be text, images, or other content.
- RESTful APIs: HTTP is used to communicate with RESTful APIs, which are a common way to build IoT applications.
- Implementation: For devices like Arduino, a web connection is established by defining the server address and port (typically 80 for HTTP) and using libraries to handle the internet connection and data transfer.
- Security: For secure communication, the HTTPS protocol is used, which encrypts the data

MQTT PROTOCOL IN IOT



MQTT is a lightweight, publish-subscribe messaging protocol crucial for IoT because it's designed for reliable communication between devices in low-bandwidth, high-latency networks

How MQTT works

- Broker: A central server that receives all messages from clients and routes them to the appropriate subscribers.
- Publisher: A device that sends messages to the broker.
- Subscriber: A device that registers its interest in specific topics with the broker and receives messages on those topics.
- Topic: A hierarchical "address" or label for a message. For example, home/livingroom/temperature could be a topic.
- Payload: The actual data or message being sent.