Sensors step wise interfacing with Arduino using MATLAB

1. Interfacing LM35 with Arduino using MATLAB:-

1. Pin Connection of LM35 with Arduino

LM35	arduino
Vcc	5 volt
Ground	Ground
Aout	A0

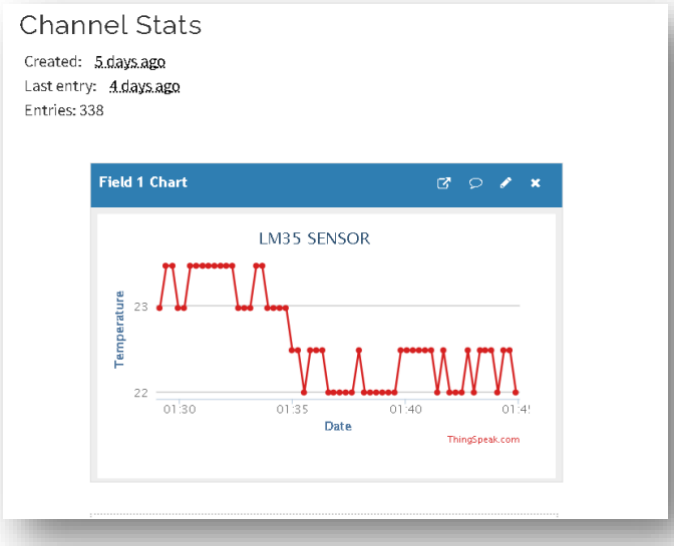
Buzzer	Arduino
Positive Node	D12
Negative Node	Ground

2. Code for interfacing LM35 with Arduino and displaying data on Thingspeak:

```
clear;
a = arduino()     %Define the Arduino

ChannelID =  2423758;
writeAPIKey = 'I4ZMNIS1KEZRJF8B';    %Channel Settings
readAPIKey = 'MJIX26NSU3OHHITE';

while (1)
   voltage = readVoltage(a, 'A0')
   temp = voltage*100


   writeData = thingSpeakWrite(ChannelID, temp, 'WriteKey',
writeAPIKey);

   readData = thingSpeakRead(ChannelID, 'ReadKey', readAPIKey);
%%Displaying data on
   pause(15);
%%Thingspeak

   if readData >= 50
      writeDigitalPin(a, 'D12', 1);    %%Interfacing buzzer
   else
      writeDigitalPin(a, 'D12', 0);
```

```
    end

end
```

3. Graph of Temperature vs date graph on thingspeak:-



2. Interfacing IR sensor, LDR sensor, with Arduino and displaying data on Thingspeak:-

Implementing a smart street light system with an LDR (Light Dependent Resistor) sensor, LED, and Arduino Uno involves several steps. Below is a basic outline of how you can set up the system:

Components Needed:

Arduino Uno board

LDR sensor

LED

Resistor (220 ohms)

Breadboard and jumper wires

Circuit Setup:

Connect the LDR sensor:

Connect one leg of the LDR to the 5V pin of the Arduino.

Connect the other leg of the LDR to one end of a resistor (220 ohms).

Connect the other end of the resistor to analog pin A0 of the Arduino.

Connect the same leg of the LDR to GND pin of the Arduino.

Connect the LED:

Connect the anode (long leg) of the LED to digital pin 13 of the Arduino through a current-limiting resistor (220 ohms).

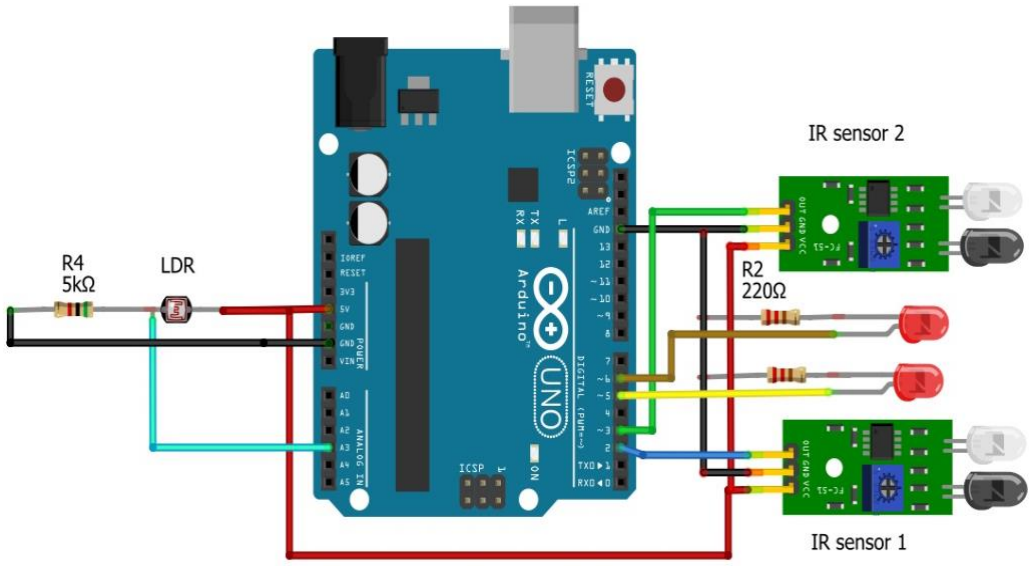
Connect the cathode (short leg) of the LED to the GND pin of the Arduino.

Make sure all connections are secure and correctly placed on the breadboard.

Arduino Code:

```
const int ldrPin = A0;   // LDR sensor pin
const int ledPin = 13;   // LED pin
int threshold = 500;     // Adjust this value as needed

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

void loop() {
```

```
  int sensorValue = analogRead(ldrPin);
  Serial.println(sensorValue);

  if (sensorValue < threshold) {
    digitalWrite(ledPin, HIGH);   // Turn on the LED if it's dark
  } else {
    digitalWrite(ledPin, LOW);    // Turn off the LED if it's bright
  }

  delay(1000);  // Delay for stability
}
```



Explanation:

The LDR sensor measures the intensity of light falling on it. A lower analog reading indicates more light, while a higher reading indicates less light.

The threshold variable is used to set a value above which the LED will turn off (considered as bright) and below which the LED will turn on (considered as dark).

In the loop, the Arduino continuously reads the analog value from the LDR sensor. If the reading falls below the threshold, it turns on the LED; otherwise, it turns it off.

Testing:

Upload the code to the Arduino Uno board.

Open the Serial Monitor to observe the analog readings from the LDR sensor.

Cover the LDR with your hand to simulate darkness and observe the LED turning on. Remove your hand to simulate light and observe the LED turning off.

This setup forms a basic smart street light system that turns on the LED when it gets dark and turns it off when it's bright. Further enhancements like adding a real-time clock module for scheduling or integrating wireless communication for remote control can be done depending on your requirements.

1. Pin connections

2. IR sensor	Arduino
Vcc	5 V
GND	GND
OUT	D2
LDR sensor	Arduino
One leg	5V
Other Leg	10kohm Resistor and then their junction to A0 of Arduino

10kohm other leg	Ground of Arduino

2. Code:-

The following part is of IR sensor :

```
a=arduino();
ChannelID=2456439;
writeAPIkey='LB0IJ3ZCZFAF5G23';
readAPIkey='7U436XWS4D68S9ER';
 irPin = 'D2';
irValue = readDigitalPin(a,'D2');
writeData = thingSpeakWrite(ChannelID,irValue,'WriteKey',writeAPIkey);
  readData = thingSpeakRead(ChannelID,'ReadKey',readAPIkey);
pause(15);
```

This part includes the command used to interface LDR sensor with Arduino :

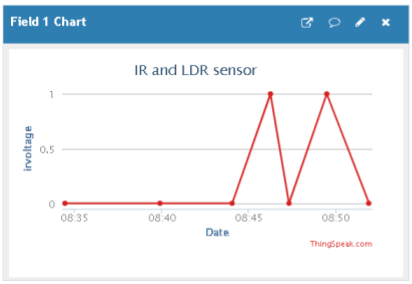
```
% Initialize Arduino
a = arduino();



% Define pin connection for LDR sensor
ldrPin = 'A0';

% Define ThingSpeak parameters
channelID = 123456; % Replace with your channel ID
writeAPIKey = 'WriteAPIkey'; % Replace with your Write API Key
readAPIKey = ‘readAPIKey’

% Main loop
```

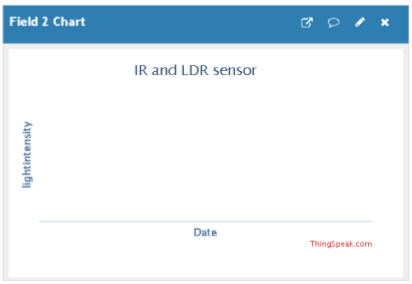
```
While(1)
    % Read LDR sensor value
    ldrValue = readVoltage(a, ldrPin);


    % Send data to ThingSpeak
    thingSpeakWrite(channelID, ldrValue, 'WriteKey', writeAPIKey);

    % Pause for a few seconds before reading again (adjust as needed)
    pause(15); % Send data every 15 seconds
end
```

3.Graph of Voltage and light intensity vs date:-

Channel Stats

Created: about 18 hours ago
Last entry: about 17 hours ago
Entries: 7





3. Interfacing Flame Sensor with Arduino and displaying the voltage on Thingspeak:

1. Pin Connections

Flame Sensor	Arduino
Vcc	5 V
GND	GND

A0	A0

2. Code:

```
clear;
a = arduino();

ChannelID = 2457997;
writeAPIKey = '5XKMOND3TFW7AL1W';
readAPIKey = '5XLM9HS9ZU23FBLH';

while(1)
  pause(15);
voltage = readVoltage(a,'A0');

 flame=(5-voltage)*10;
    writeData = thingSpeakWrite(ChannelID,flame,'WriteKey',
writeAPIKey);
    readData = thingSpeakRead(ChannelID,'ReadKey', readAPIKey);
pause(15);

    if(readData>=20)

        writeDigitalPin(a,'D9',1);

    else
        writeDigitalPin(a,'D9',0);
```

4. Interfacing Flame sensor, buzzer Arduino using MATLAB and sending the data to Thingspeak:

Implementing a fire detection system using a flame sensor, buzzer, and Arduino Uno is relatively straightforward. Here's a step-by-step guide to setting it up:

Components Needed:

Arduino Uno board

Flame sensor module

Buzzer

Breadboard and jumper wires

Circuit Setup:

Connect the flame sensor:

The flame sensor typically has three pins: VCC, GND, and OUT.

Connect the VCC pin to the 5V pin of the Arduino.

Connect the GND pin to the GND pin of the Arduino.

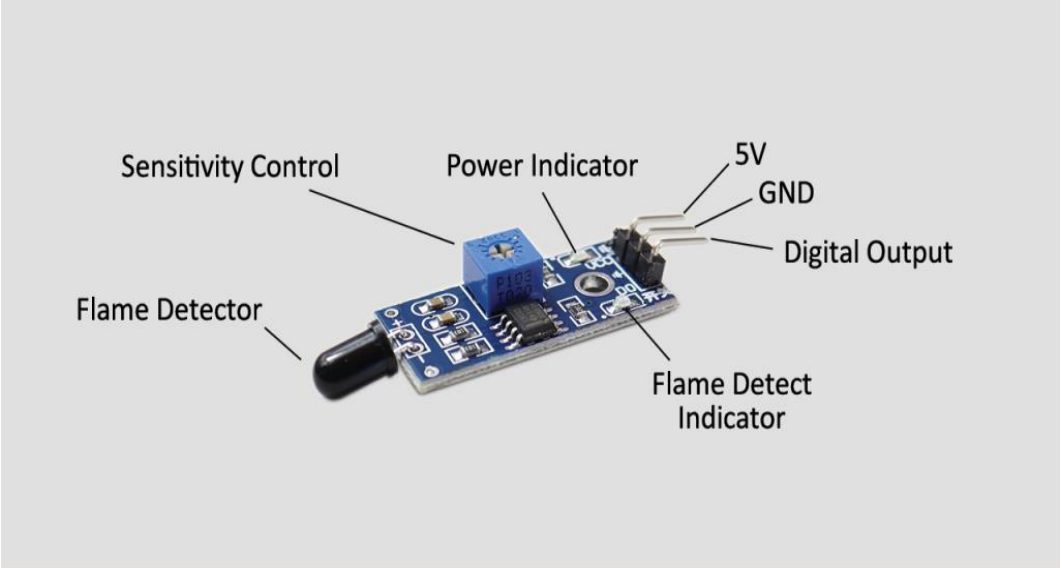
Connect the OUT pin to digital pin 2 of the Arduino.

Connect the buzzer:

Connect the positive (red) terminal of the buzzer to digital pin 3 of the Arduino.

Connect the negative (black) terminal of the buzzer to the GND pin of the Arduino.

Make sure all connections are secure and correctly placed on the breadboard.



Arduino Code:

```
#define FLAME_SENSOR_PIN 2
#define BUZZER_PIN 3

void setup() {
  pinMode(FLAME_SENSOR_PIN, INPUT);  // Flame sensor pin as input
  pinMode(BUZZER_PIN, OUTPUT);       // Buzzer pin as output
  Serial.begin(9600);                // Initialize serial communication
}

void loop() {
  int flameDetected = digitalRead(FLAME_SENSOR_PIN); // Read flame sensor state

  if (flameDetected == HIGH) {
    Serial.println("Fire detected!");      // Print message to serial monitor
```

```
    digitalWrite(BUZZER_PIN, HIGH);       // Activate the buzzer
    delay(1000);                          // Buzzer on for 1 second
    digitalWrite(BUZZER_PIN, LOW);        // Turn off the buzzer
    delay(1000);                          // Wait for 1 second
  }
}
```

Explanation:

The flame sensor outputs a digital signal. When it detects a flame, it sends a HIGH signal to the Arduino; otherwise, it sends a LOW signal.

In the loop, the Arduino continuously checks the state of the flame sensor.

If the flame sensor detects a flame (HIGH state), the buzzer is activated for a short duration to alert about the fire.

Testing:

Upload the code to the Arduino Uno board.

Open the Serial Monitor to observe the messages printed when a flame is detected.

Test the system by exposing the flame sensor to a source of heat or fire. When the flame is detected, the buzzer should activate briefly.

5. Interfacing ultrasonic sensor and a buzzer with Arduino using MATLAB and sending the data to Thingspeak:

1. Pin connection:-

Ultrasonic sensor	Arduino
Vcc	5V

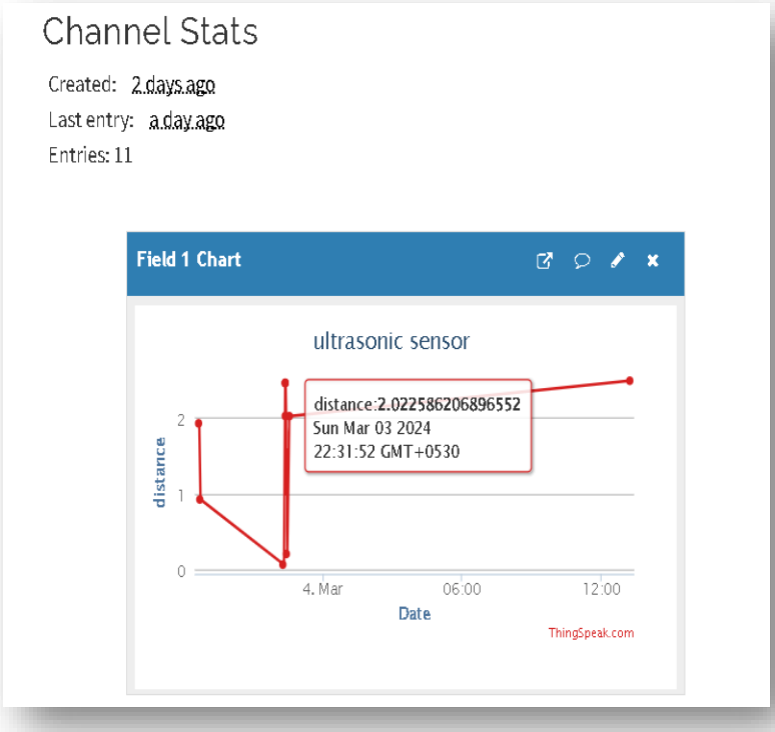
GND	GND
Echo	D11
Trigger	D12

Buzzer	Arduino
Positive terminal	D9
Negative terminal	GND

2. Code:

```
clear
a= arduino
ChannelID =  2454257;
writeAPIkey = 'NP6QWGVKYOK1UPDI';
readAPIkey = 'L5MTWQ20ODNFH2RG';

ultrasonicObj = ultrasonic(a,'D12','D11');  %%Defining the ultrasonic
object using ultrasonic
 distance = readDistance(ultrasonicObj);   %%command
writeData = thingSpeakWrite(ChannelID,
distance,'WriteKey',writeAPIkey);
  readData = thingSpeakRead(ChannelID,'ReadKey',readAPIkey);
pause(15);
if readData >= 0.5
      writeDigitalPin(a, 'D9', 1);
      pause(5);
      writeDigitalPin(a,'D9',0);
   else
      writeDigitalPin(a, 'D9', 0);
   end
```

3. Graph of distance on y-axis with date on x-axis :



PART 3

HARDWARE PROJECT DESCRIPTION:-

Q1. Basic idea of your project?

Ans:-The basic idea of a person counter IoT project is to create a system that can automatically count the number of individuals passing through a certain area or entering/exiting a specific location.

Q2. Hardware required in your project?

Ans:- 1. Esp8266

2. IR Sensor

3. LED

4. Resistor(220 ohms)

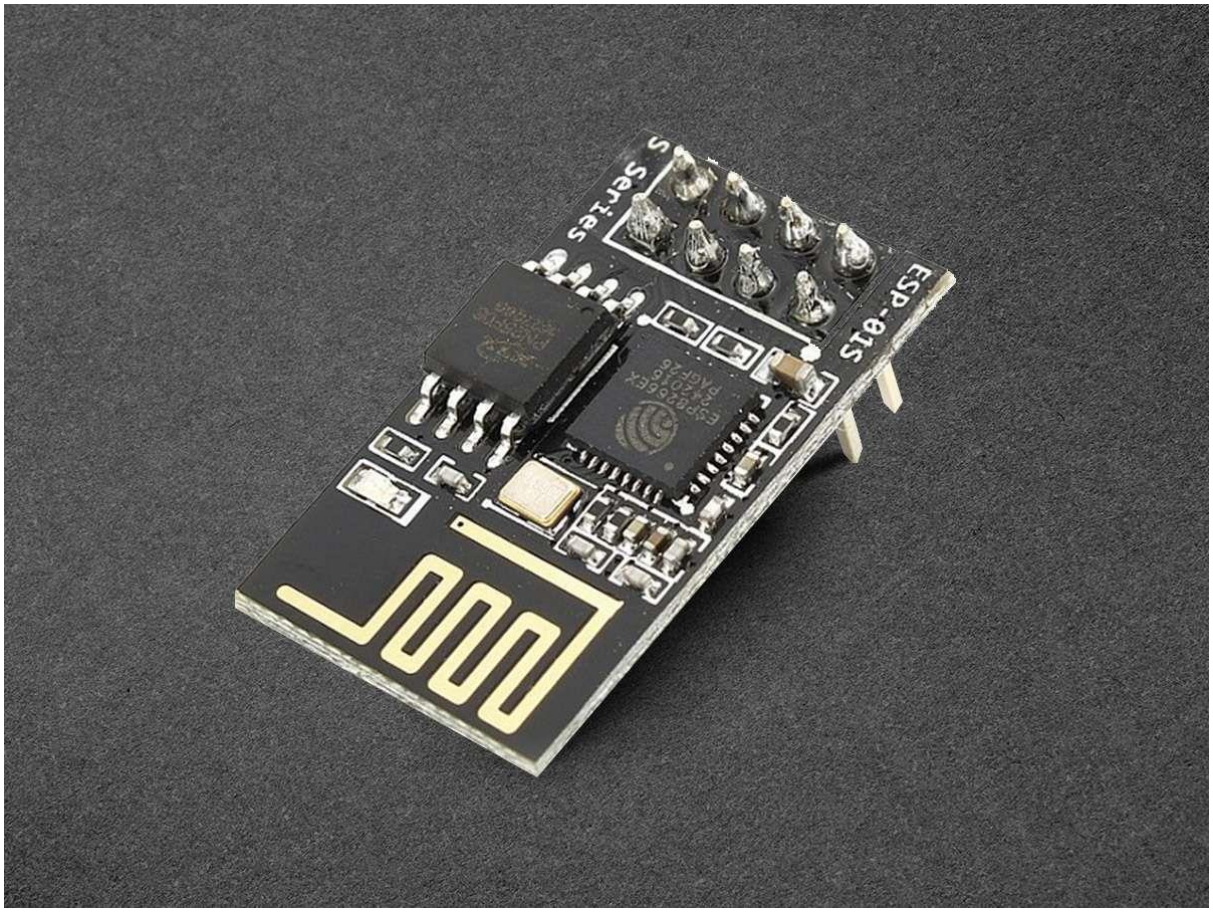
Q3. Brief Description of the hardware required?

Ans:- 1) Esp8266:- The ESP8266 sensor is a versatile and widely used component in the realm of Internet of Things (IoT) and embedded systems.

The ESP8266 itself is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capability. It's commonly used as the main processing unit in IoT projects due to its affordability, ease of use, and extensive community support.

When coupled with various sensors such as temperature, humidity, motion, or light sensors, the ESP8266 becomes a powerful tool for collecting real-time data and transmitting it over Wi-Fi networks. This capability makes it ideal for applications like home automation, environmental monitoring, and smart agriculture.

By leveraging the ESP8266's Wi-Fi connectivity and sensor integration, developers can create efficient and scalable IoT solutions. Its small form factor and low power consumption further enhance its suitability for a wide range of projects.



2) IR Sensor:- An IoT (Internet of Things) project utilizing an IR (Infrared) sensor offers numerous applications and advantages. IR sensors detect infrared radiation emitted by objects, enabling them to measure distance, detect motion, or sense temperature changes without direct contact. Here's a brief overview of how IR sensors are used in IoT projects:-

1)Motion Detection:- IR sensors can detect movement by measuring changes in the infrared radiation emitted by objects in their field of view. This capability is widely used in IoT projects for security systems, automated lighting, and occupancy detection in smart buildings.

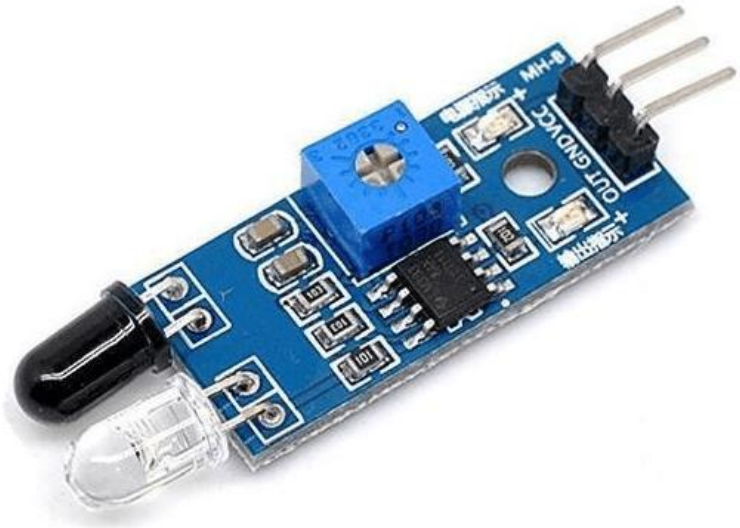
2)Distance Measurement:- IR sensors can also measure the distance to objects based on the time it takes for infrared light to reflect back to the sensor. This

feature is valuable in IoT applications such as robotics, smart agriculture (for measuring crop height), and industrial automation.

3)Object Detection:- IR sensors can detect the presence or absence of objects by measuring the reflection or absorption of infrared light. This functionality is employed in IoT projects for inventory management, smart retail (detecting when items are removed from shelves), and obstacle detection in autonomous vehicles and drones.

4)Temperature Sensing:- Some IR sensors are designed to measure temperature variations by detecting infrared radiation emitted by objects. In IoT projects, these sensors are used for environmental monitoring, HVAC systems, and health monitoring devices (e.g., fever detection).

5)Energy Efficiency:- Utilizing IR sensors in IoT projects can lead to energy savings by enabling more efficient use of resources. For example, smart thermostats equipped with IR sensors can detect occupancy and adjust heating or cooling accordingly, leading to reduced energy consumption.



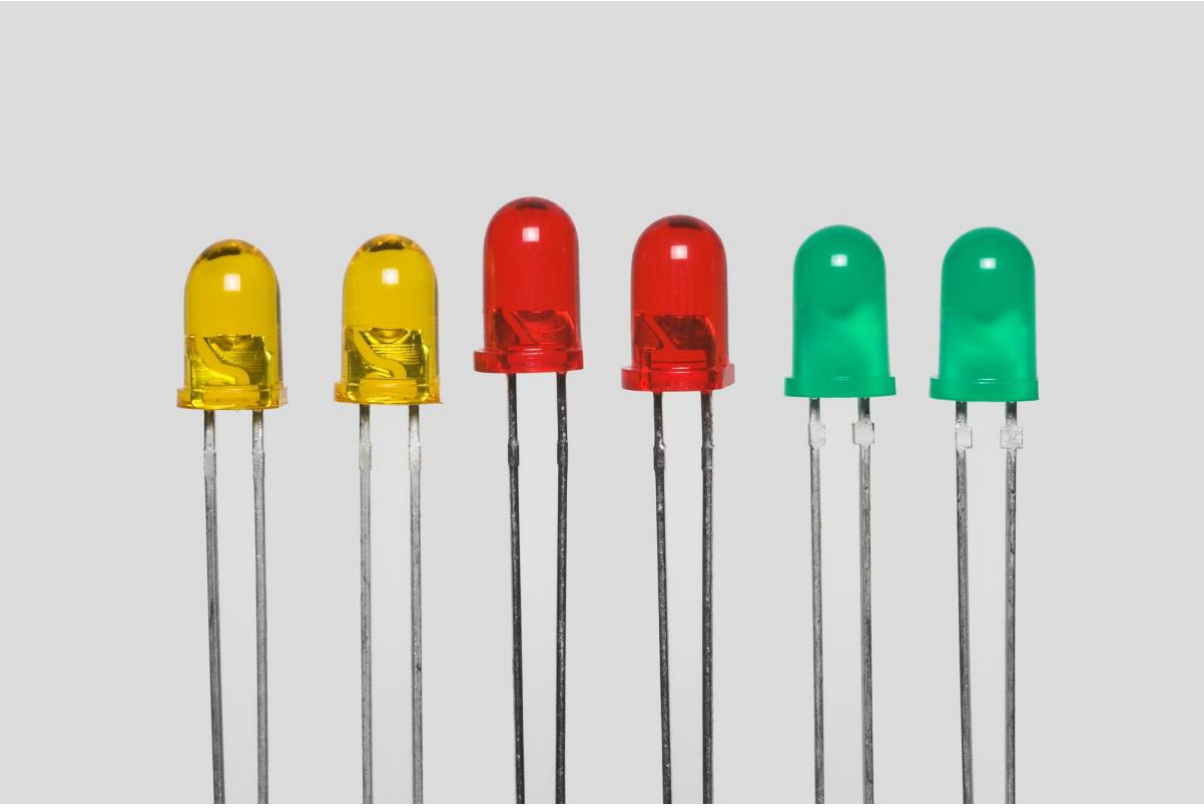
3) LED:- LEDs are ubiquitous in IoT projects, serving as indicators, notifications, and status lights. They provide visual feedback to users, communicate device status, and enhance user interaction in various IoT applications.

1) Status Indication:- LEDs are commonly used to indicate the operational status of IoT devices. For example, a green LED may indicate that a device is connected to a network and functioning normally, while a red LED could signify an error or malfunction. This allows users to quickly assess the status of their IoT devices at a glance.

2) Notifications and Alerts:- LEDs can be employed to deliver notifications and alerts to users. For instance, a flashing LED could indicate the receipt of a new message or notification on a smart home device, a wearable device might use LEDs to alert the wearer of incoming calls or messages, and IoT sensors could use LEDs to signal the detection of an event or anomaly.

3)User Interaction:- LEDs facilitate user interaction in IoT projects by providing feedback in response to user inputs or commands. For example, pressing a button on a smart home controller might cause an LED to illuminate, confirming that the command has been received and executed. In this way, LEDs enhance the user experience and make IoT devices more intuitive to use.

4)Energy Efficiency:- LEDs are highly energy-efficient, making them ideal for battery-powered IoT devices where energy consumption is a concern. Their low power consumption allows them to be used for extended periods without draining the device's battery excessively.

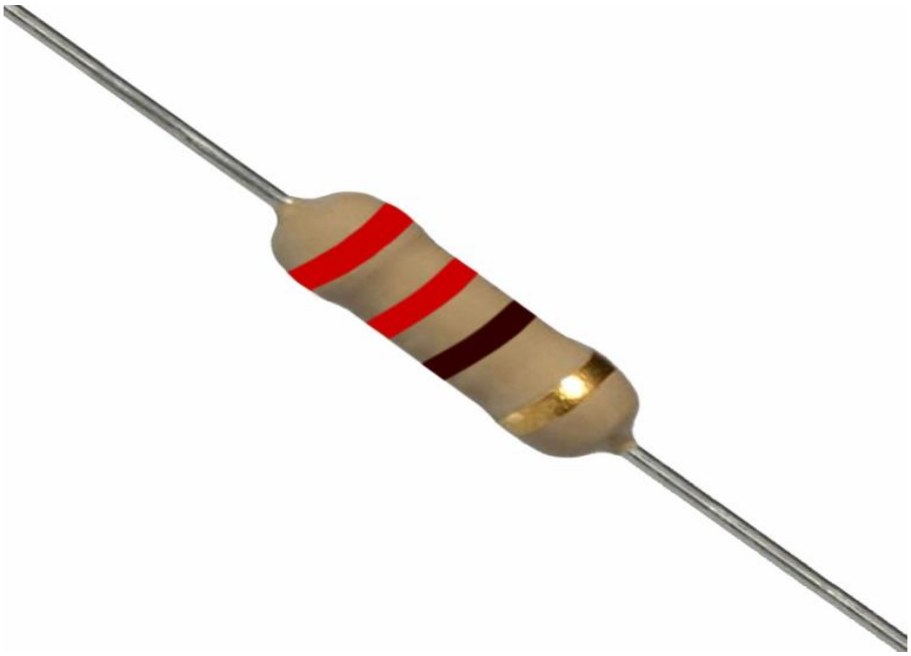


4) Resistor(220 ohms):- LED Current Limiting: One of the most common uses of a 220 ohm resistor in an IoT project is to limit the current flowing through an LED (Light Emitting Diode). When an LED is connected to a voltage source (e.g., a microcontroller pin) without a current-limiting resistor, it can draw excessive current and potentially burn out. By placing a 220 ohm resistor in series with the LED, the current flowing through the LED is restricted to a safe level, preventing damage to the LED and ensuring its longevity.

1)Pull-Up or Pull-Down Resistor:- In digital circuits, pull-up and pull-down resistors are often used to ensure that a signal remains at a known state when no active input is present. A 220 ohm resistor can be used as a pull-up or pull-down resistor in IoT projects to bias a digital input pin of a microcontroller to a specific voltage level (typically either Vcc or GND). This is particularly useful for interfacing with sensors, switches, or other external devices where a stable signal is required.

2)Voltage Divider:- A 220 ohm resistor can be part of a voltage divider circuit used to scale down a voltage level to a range that is suitable for the input of a microcontroller or other integrated circuit. By combining the 220 ohm resistor with another resistor of a different value, specific voltage levels can be obtained, allowing for accurate voltage measurements or sensor readings in IoT applications.

3)Current Sensing:- In some IoT projects, it may be necessary to measure the current flowing through a circuit or component for monitoring or control purposes. A 220 ohm resistor can be used in conjunction with the voltage drop across it to calculate the current flowing through the circuit using Ohm's Law (I = V/R), where I is the current, V is the voltage across the resistor, and R is the resistance (220 ohms in this case).



Q7. Viva based questions with answers based on your project?

1)What is the purpose of the Person Counter project?

Ans:- The purpose of the Person Counter project is to develop a system that can accurately count the number of people entering or exiting a certain area, such as a building, room, or public space.

2) How does the Person Counter system detect and track individuals?

Ans:- The system typically uses computer vision techniques to detect and track individuals. This involves processing video or image data from sensors to identify human shapes or features, and then tracking these shapes or features over time to count the number of people entering or exiting the area of interest.

3) What are some challenges faced in developing a Person Counter system?

Ans:- Some challenges include:-

i)Ensuring accuracy in counting, especially in crowded or complex environments.

ii)Dealing with variations in lighting conditions.

iii)Handling occlusions where individuals may partially block each other from view.

iv)Addressing privacy concerns related to the use of surveillance technology.

4) How can the accuracy of the Person Counter system be evaluated?

Ans:- Accuracy can be evaluated by comparing the counts generated by the system with ground truth counts obtained through manual counting or other reliable methods. Metrics such as precision, recall, and F1 score can be used to assess the system's performance.

5) What are some potential applications of the Person Counter system?

Ans:- Some potential applications include:

i)Monitoring foot traffic in retail stores or public spaces.

ii)Managing crowd control at events or gatherings.

iii)Optimizing staffing levels in buildings or facilities based on occupancy.

iv) Enhancing security by detecting unauthorized access or loitering.

6) How does the Person Counter system handle situations where multiple people enter or exit simultaneously?

Ans:- The system should be designed to handle such situations by accurately detecting and counting each individual person. This may involve using advanced algorithms to separate and track individuals even when they are close together or overlapping in the sensor data.

7) How does the Person Counter system handle scenarios where individuals may enter or exit in groups?

Ans:- The system may employ algorithms to detect groups of individuals as a single entity and count them accordingly. This could involve analyzing the size and shape of detected objects to differentiate between individuals and groups.