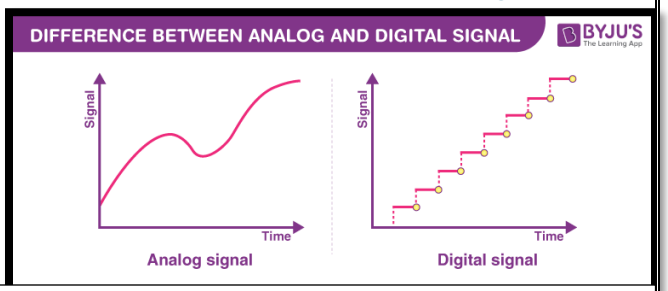
Q1:

1. Explain the difference between analog and digital signals and give two examples of each in real-life applications.

Analog and digital signals are the types of signals carrying information. The major difference between both signals is that the analog signals have continuous electrical signals, while digital signals have non-continuous electrical signals. The difference between analog and digital signals can be observed with examples of different types of waves.



Examples of Analog Signals

Some common examples of analog signals include:

- Human voice
- Analog Radio and TV Broadcast
- Audio signals transferred via cables
- Radio signals
- Analog timepieces

Examples of Digital Signals

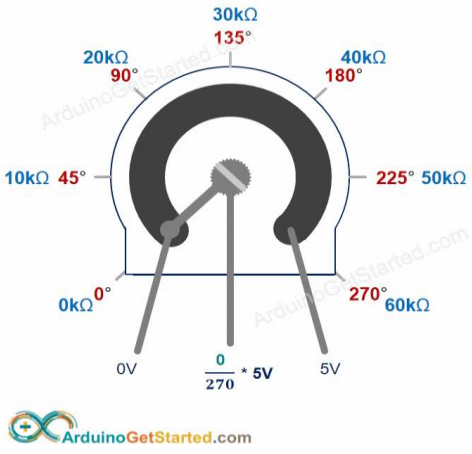
Some common examples include:

- Digital Audio
- Digital Video
- Binary Data
- Digital Clocks
- Smartphones

Difference between Analog and Digital Signal	
Analog Signals	**Digital Signals**
Continuous signals	Discrete signals
Represented by sine waves	Represented by square waves
Human voice, natural sound, analog electronic devices are a few examples	Computers, optical drives, and other electronic devices
Continuous range of values	Discontinuous values
Records sound waves as they are	Converts into a binary waveform
Only used in analog devices	Suited for digital electronics like computers, mobiles and more

Q2:

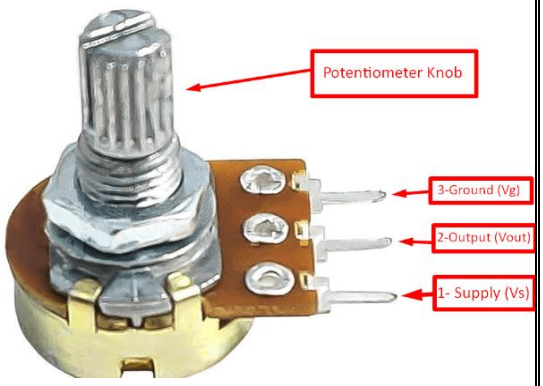
2. Describe how a potentiometer works. How is it used with ESP32 to provide analog input? Explain how its analog voltage is read and used in programming.



A potentiometer is a simple mechanical device that comes in many different forms. It provides a variable amount of resistance that changes as you manipulate it.

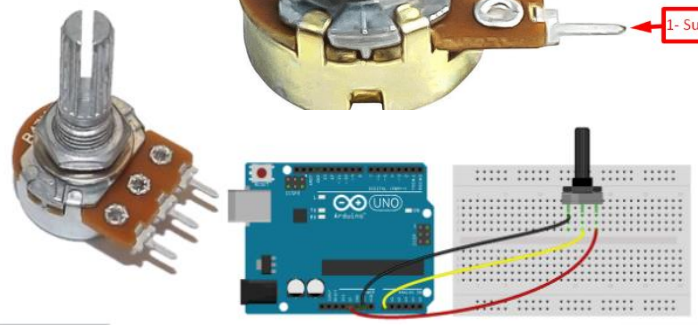
Potentiometer Parts:

1- Outer Pin (connected to GND
2- Outer Pin (connected to 3.3V)
3- Middle Pin – called the Wiper
4- Knob (the part you turn with your fingers)



How a Potentiometer Works

Inside the potentiometer, there is a resistive track connecting the two outer pins. The wiper (middle pin) moves across this track when you rotate the knob. This movement changes the resistance ratio between the two halves of the resistive track, effectively changing the voltage at the wiper.



- When the knob is turned fully toward GND, the wiper reads 0V.
- When turning fully toward 3.3V, the wiper reads 3.3V.
- Anywhere in between, the voltage varies linearly between 0V and 3.3V.

How It Is Used with ESP32 to Provide Analog Input:

The ESP32 has several ADC (Analog-to-Digital Converter) pins capable of reading varying voltages between 0 and 3.3V. When a potentiometer is connected to one of these analog pins, the ESP32 can read the voltage from the wiper and convert it into a digital value between 0 and 4095 (since the ESP32's ADC is 12-bit by default).

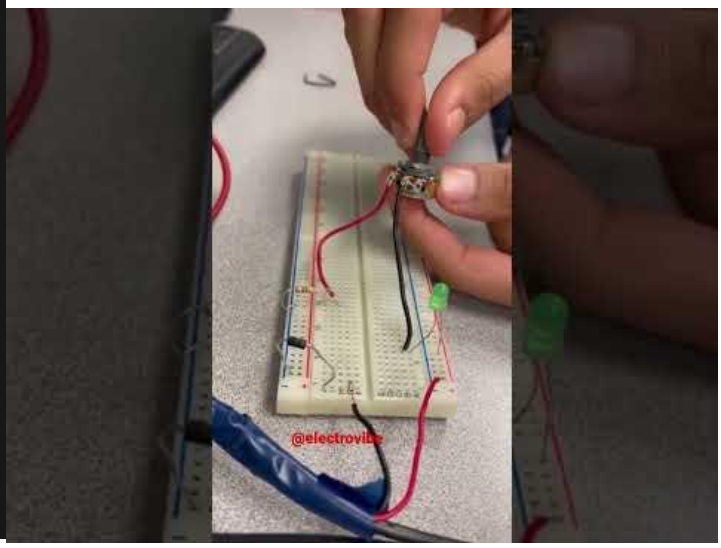
How You Read It in Code

Using the **Arduino IDE** (or similar), you can use the built-in function analog Read(pinNumber) to get the analog value.

```
int potPin = 34;
int potValue = 0;

void setup() {
  Serial.begin(115200);
}

void loop() {
  potValue = analogRead(potPin);
  Serial.println(potValue);
  delay(500);
}
```



Q3:

What is Pulse Width Modulation (PWM)?

○ **Describe its waveform.** Pulse Width Modulation (PWM) is a technique used in electronics to control the average power delivered to a load by varying the width of the pulses in a periodic waveform. In PWM, a fixed frequency signal (the carrier wave) is turned on and off at varying intervals, creating pulses with different widths. PWM is considered a smart and efficient way to simulate analog signals using only digital outputs. By adjusting the duty cycle — the proportion of time the signal stays high in each cycle — PWM can mimic different voltage levels, making it ideal for controlling devices like LEDs, motors, and more, without needing a dedicated digital-to-analog converter (DAC).

waveform is defined by two key parameters:

1. **Frequency**:
   The number of times the PWM cycle repeats per second (measured in Hertz, Hz).
   It determines how fast the signal turns on and off.

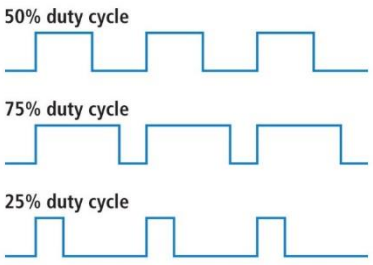
2. **Duty Cycle**:
   The percentage of time the signal stays HIGH during one full cycle.
   It controls how much power is delivered to the device.

PWM Duty Cycle:

The duty cycle is a parameter used in the context of pulse width modulation (PWM). It represents the ratio of the duration when a signal is in the "on" state (high) to the total period of the signal (which includes both the "on" and "off" states). The duty cycle is usually expressed as a percentage.

A higher duty cycle means that the signal is on for a greater portion of the total period, delivering more power to the load, while a lower duty cycle means less power is delivered. The duty cycle is a key parameter in PWM, as it determines the average voltage or current supplied to a device, influencing its behavior or output.

$$\textbf{Duty Cycle}\ (\%) = \left(\frac{T_{ON}}{T_{ON} + T_{Off}}\right) \times 100$$



Pulse Width Modulation (PWM) Applications:

- Changing the Screen's Brightness By Utilizing PWM
- Switching Power Supplies
- Heating Systems
- Audio Amplifiers
- Battery Charging
- Servo Control
- Inverters and Frequency Control
- Using Arduino to Implement PWM
- Motor Control

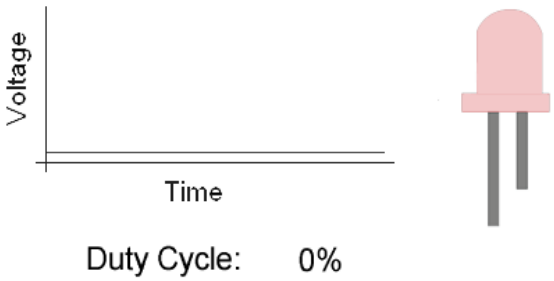
```
#include <Arduino.h>

const int led25 = 5;
const int led50 = 6;
const int led75 = 9;

void setup() {
  pinMode(led25, OUTPUT);
  pinMode(led50, OUTPUT);
  pinMode(led75, OUTPUT);
}

void loop() {

  analogWrite(led25, 64);
  analogWrite(led50, 128);
  analogWrite(led75, 192);
}
```



○ Explain how ESP32 uses PWM to control devices like LEDs or motors.

The ESP32 uses PWM to control the power delivered to electronic devices like **LEDs** or **motors** by rapidly switching a digital pin **ON** and **OFF** at a certain frequency.

How it works:

- The ESP32 has up to **16 PWM channels**.
- You can assign any GPIO pin to any PWM channel.
- Each channel allows you to:
- Set the **frequency** (how fast it switches ON/OFF).
- Set the **duty cycle** (how long it stays ON in each cycle).

Why PWM is useful on ESP32:

- It avoids using DACs or analog components.
- It provides **efficient**, **digital control** of analog-like behavior.
- It supports **multiple devices** with different settings at the same time

○ Include examples of duty cycle usage.

LED Brightness Control using PWM (Duty Cycle Examples):

10% Duty Cycle
Very Dim Light
The LED is ON only 10% of the time.
Ideal for:

- **Night light** mode
- Power saving
- Subtle visual indicators

2- 50% Duty Cycle
Medium Brightness
The LED is ON for half the cycle.
Commonly used in:

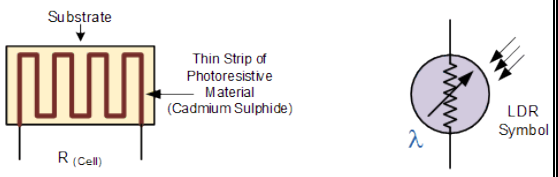
- Dashboard lights
- Room ambient lighting
- Notification indicators

3- 90% Duty Cycle
Full Brightness
The LED is ON almost all the time.
Suitable for:

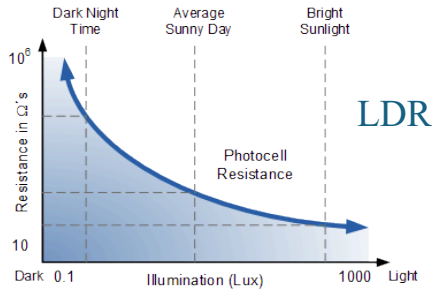
- Flashlight mode
- Task lighting
- High-visibility alerts

Q4:

Explain how an LDR (Light Dependent Resistor) works and how its resistance changes with light. How can you use ESP32 to detect changes in light intensity using an LDR?



LDR (Light Dependent Resistor) as the name states is a special type of resistor that works on the photoconductivity principle means that resistance changes according to the intensity of light. Its resistance decreases with an increase in the intensity of light.
It is often used as a light sensor, light meter, Automatic street light, and in areas where we need to have light sensitivity.
is also known as a Light Sensor. LDR are usually available in 5mm, 8mm, 12mm, and 25mm dimensions.



LDR

How does it work?

- When light increases, the resistance of the LDR decreases.
- When light decreases (darkness), the resistance increases.

How to Use an LDR with ESP32

We connect the LDR to an analog input pin of the ESP32 (like GPIO 34, 35, etc.) and read the voltage that changes based on light intensity.

- Connect one leg of the LDR to 3.3V.
- Connect the other leg to an analog pin (e.g., GPIO 34) and a 10kΩ resistor going to GND.

Example Usage:

Application	LDR Role
Automatic streetlights	Turn on lights when dark
Light meters	Measure brightness levels
Smart blinds	Adjust blinds based on light

Q5:

Discuss the behavior of push buttons as a digital input.

A push button is a simple switch that allows current to flow when pressed. It's either:

High (1): when the button is not pressed (due to a pull-up resistor).

Low (0): when the button is pressed and connected to ground.

○ What issues can occur when reading a push button?

Bouncing: When a button is pressed or released, the contact points inside the button physically "bounce" for a few milliseconds before settling. This can cause multiple rapid transitions (high/low) instead of a single clean one.

○ Explain debouncing and how it can be implemented in code.

Debouncing:

Debouncing is the process of eliminating noise and bouncing effects from the button's signal.

Implementation in Code:
There are two common methods:

1. Software Debouncing (Delay or Timer-based):

Wait a small delay (e.g., 20–50ms) after detecting a change before acting.

Check if the signal remains stable during that delay.

```
#include <Arduino.h>

const int buttonPin = 14;
const int ledPin = 2;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (digitalRead(buttonPin) == LOW) {
    delay(50);
    if (digitalRead(buttonPin) == LOW) {
      digitalWrite(ledPin, HIGH);
    }
  } else {
    digitalWrite(ledPin, LOW);
}
}
```

2. Using a State Machine with millis():

Record the last time the state changed.

Only accept a change if a set time (e.g., 50ms) has passed.

```
#include <Arduino.h>
const int buttonPin = 14;
const int ledPin = 2;
int buttonState = HIGH;
int lastButtonState = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;
void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
  Serial.begin(115200);
}
void loop() {
  int reading = digitalRead(buttonPin);

  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;
      if (buttonState == LOW) {
        digitalWrite(ledPin, !digitalRead(ledPin));
      }
    }
  }
  lastButtonState = reading;
}
```

Q6:

Compare analog and PWM signals.

Feature	Analog signal	PWM signal
Signal type	Continuous voltage signal	Digital signal (rapid switching between HIGH and LOW)
Value Range	Infinite	Binary: only 0V (LOW) or full voltage (HIGH)
Generated by	DAC (Digital to Analog converter	Digital pin with PWM capability
Control method	Directly set voltage	Control duty cycle to simulate analog behavior
Power efficiency	LOW (linear voltage control wastes power)	High (switching reduces power loss)
Precisions	Very high (True analog levels)	Depends on resolution of PWM (e.g., 8-bit, 10-bit, etc.)
Used for	Audio, outputs, sensors	LED brightness control, motor speed control
Noise resistance	Sensitive to noise	More immune to electrical noise

- **Can PWM simulate analog output on a digital pin?**

Yes, PWM can simulate analog output on a digital pin.
PWM achieves this by rapidly turning the pin ON and OFF, and the duty cycle (percentage of time ON) determines the average voltage delivered to the device.

For example:

50% duty cycle → ~1.65V average (on a 3.3V system)

100% duty cycle → 3.3V (always ON)

0% duty cycle → 0V (always OFF)

○ What are the benefits and limitations?

Benefits of PWM:

1) **Power Efficiency:**
   PWM turns devices fully ON or OFF instead of partially powering them like analog signals. This reduces power loss and increases energy efficiency, especially in battery-powered systems.
2) **Easy to Implement:**
   Most microcontrollers (like ESP32 or Arduino) have built-in support for PWM on digital pins, so you don't need extra components to use it.
3) **Precise Control:**
   By adjusting the duty cycle, you can control the brightness of LEDs, speed of motors, or even generate audio signals with high precision.
4) **Digital Compatibility:**
   Since PWM uses digital signals (HIGH or LOW), it's easier to process and is more noise-resistant than analog signals.

Limitations of PWM:

1) **Not a True Analog Signal:**
   PWM is just a digital signal switching rapidly. Some devices that require smooth voltage changes may not respond well unless filtered.
2) **Requires Filtering for Analog Use:**
   If you need a steady analog voltage (like 2.5V), you need a low-pass filter (resistor + capacitor) to smooth out the PWM wave.
3) **Can Cause Electromagnetic Interference (EMI):**
   Rapid switching in PWM can generate electrical noise that may interfere with nearby circuits, especially in sensitive audio applications.
4) **Timing Sensitive:**
   For smooth operation, PWM signals need consistent timing. If the microcontroller is too busy or interrupted, PWM accuracy might be affected.

Q7:

Explain the working principle of LEDs and how they are controlled by ESP32.

An LED is a semiconductor device that emits light when electric current passes through it in the correct direction (forward bias). The energy from the moving electrons is released in the form of light.

LED Working Principle:

Quantum Theory:

LEDs work based on the principle that when electrons move from a higher energy level to a lower one, they release energy in the form of light

Forward Bias:

When a diode (including an LED) is forward-biased, it allows current to flow through it.

Semiconductor Material:

ESP32: is a microcontroller that can turn LEDs on or off using its digital output pins.
Controlling the LED:

- Connect the anode (long leg) of the LED to an ESP32 GPIO pin through a resistor.
- Connect the cathode (short leg) to GND.
- In code, setting the pin HIGH (3.3V) turns the LED ON, and setting it LOW (0V) turns it OFF.

- **What is the significance of a current-limiting resistor?**

A current-limiting resistor is crucial for protecting electronic components from damage by reducing the flow of excessive current. It acts as a safeguard, preventing components from burning out or malfunctioning due to high current levels

LEDs require **very little current**, usually around **10–20

○ **What happens if it's omitted?**

1. Too Much Current Flows Through the LED

- LEDs are not like regular resistors. If you connect them directly to a power source (like from the ESP32) without resistance, excessive current will flow.
- This can burn the LED instantly or degrade it over time.

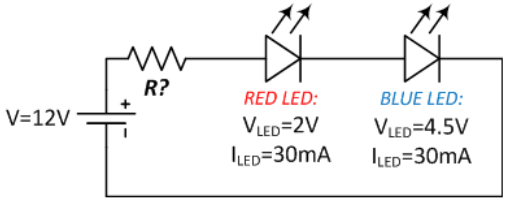
2. ESP32 Pin May Get Damaged

- If too much current is drawn through a digital output pin (typically more than 12mA–20mA), you risk damaging the microcontroller pin or the whole chip.

3. LED May Overheat or Flicker

- Without control, the LED may get hot or behave unpredictably (like flickering or glowing too brightly before burning out).

Example of multiple LEDs in series



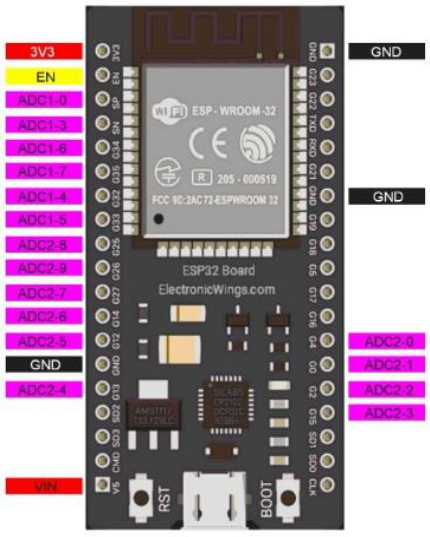
$$R = \frac{V - V_{LED1} - V_{LED2}}{I} = \frac{12 - 2 - 4.5}{0.03} = 183.3\Omega$$

Q8:

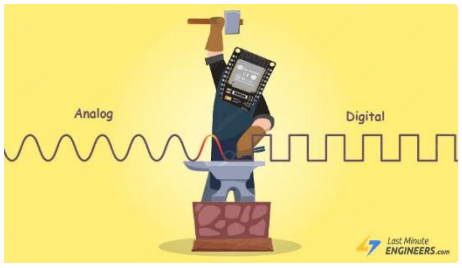
How do internal ADCs (Analog to Digital Converters) in ESP32 work?

An ADC (Analog to Digital Converter) converts an analog voltage (a continuous signal like from a sensor or LDR) into a digital value that the ESP32 can understand and process. When you use analogRead(pin) in your code, the ESP32 samples the voltage on that pin and converts it into a digital number

ADC Pins of ESP32



ESP32 ADC Pin



○ **How many bits of resolution does ESP32 ADC provide?**

ESP32 provides a 12-bit ADC resolution by default it will give digital values in the range of 0 – 4096 (2^12). This is called resolution which indicates the number of discrete values it can produce over the range of analog values.

○ **What is the voltage range it can read?**

0V to 3.3V

If you give more than 3.3V, you may damage the pin

Q9:

List and describe any two real-world IoT applications that use:

○ **a potentiometer for user input**

○ **an LDR for environmental sensing**

A. IoT Applications using a Potentiometer for User Input

Smart Light Dimmer Switch:

- Use Case: In smart homes, users can manually adjust light brightness.
- How: A potentiometer acts as a physical dimming control knob. which adjusts the
- LED brightness accordingly.
- The current dimmer level can be monitored or controlled via Wi-Fi using a mobile app

Smart Thermostat Control Knob:

- Use Case: Users adjust the desired room temperature using a dial.
- How: A potentiometer sets the desired temperature, and the system compares it to
- sensor readings to control HVAC.
- The system sends temperature data to the cloud and allows remote control via an app.

B. IoT Applications an LDR for environmental sensing

Automatic Street Lighting System:

- Use Case: Turns ON lights automatically at night and OFF during the day.
- How: An LDR senses ambient light. Low light (high resistance) triggers the system to
- turn on streetlights.
- Light status and faults can be reported to a central dashboard over Wi-Fi or LoRa.

Smart Window Blinds:

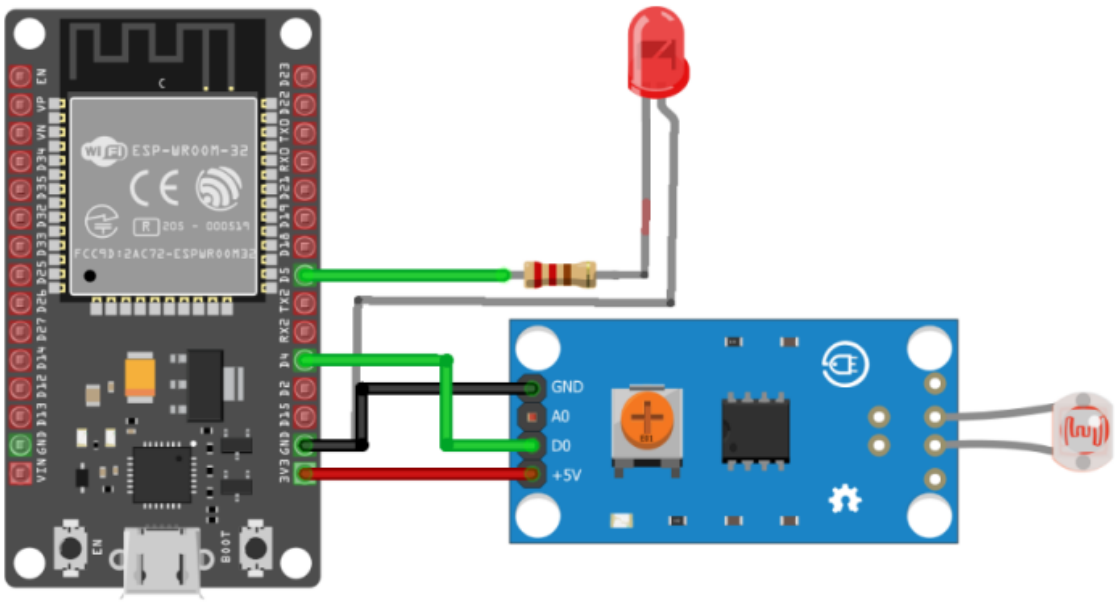
- Use Case: Adjust blinds based on room brightness.
- How: An LDR detects incoming sunlight. When it gets too bright, blinds close
- automatically using a motor.
- Brightness data is sent to the cloud, and users can override blind settings via an app.

Q10:

Design a simple circuit using ESP32, an LDR, and an LED that turns the LED on when it gets dark.

○ Draw the circuit

○ Explain how the analog value from LDR is processed to control the LED.



The ESP32 reads light levels using an LDR connected as a voltage divider. When it gets dark,

the analog value drops below a set threshold. The code checks this value, and if it's low, it turns

the LED ON. If it's bright, the LED stays OFF.

Function Name: analogRead(pin)

Value Range on ESP32: 0 (dark) to 4095 (bright)

```
#include <Arduino.h>

#define LDR_PIN 15
#define LED_PIN 4

const int threshold = 2000;

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

void loop() {
  int lightLevel = analogRead(LDR_PIN);
  Serial.println(lightLevel);

  if(lightLevel < threshold) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }

  delay(500);
}
```

References:

1- https://study.com/academy/lesson/what-are-digital-and-analog-signals-definition-lesson-quiz.html

1) https://byjus.com/physics/difference-between-analog-and-digital/

2) https://testbook.com/physics/difference-between-analog-and-digital-signal

3) https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/

4) https://esp32io.com/tutorials/esp32-potentiometer

5) https://www.allelcoelec.com/blog/what-is-the-function-of-a-potentiometer.html?srsltid=AfmBOoruc0o7COb_F0lm-5M7RLW-8J8C7Cqct5OTm1z1Fg8xhYgErb4r

6) https://www.circuits-diy.com/how-to-use-a-potentiometer-arduino-tutorial/

7) https://docs.arduino.cc/learn/electronics/potentiometer-basics/

8) https://learn.sparkfun.com/tutorials/pulse-width-modulation/duty-cycle

9) https://www.geeksforgeeks.org/electronics-engineering/applications-of-pwm/

10) https://maplesystems.com/tutorial/what-is-pulse-width-modulation/?srsltid=AfmBOorNeguAQiEA58c47ERxDFcVD5c-ynaVn7Me4Z66xDmTtVWSx-8F

11) https://deepbluembedded.com/esp32-pwm-tutorial-examples-analogwrite-arduino/

12) https://www.elprocus.com/ldr-light-dependent-resistor-circuit-and-working/

13) https://www.electronicsforu.com/technology-trends/learn-electronics/ldr-light-dependent-resistors-basics

14) https://www.techtarget.com/whatis/definition/debouncing

15) https://resources.pcb.cadence.com/blog/2021-how-and-why-to-convert-analog-signals-to-pwm-signals

16) https://www.elprocus.com/light-emitting-diode-led-working-application/

17) https://eepower.com/resistor-guide/resistor-applications/resistor-for-led/#

18) https://www.electronicwings.com/esp32/adc-of-esp32

19) https://www.electronicwings.com/esp32/adc-of-esp32

20) https://www.youtube.com/watch?v=U4NSoW90THk

21) https://youtu.be/cUca-srjPcs?si=UyGd6dDXMpKXspQ9

22) https://www.youtube.com/watch?v=iYcXjmJk7dQ&feature=youtu.be

23) https://www.youtube.com/watch?v=IJPHdkSgV4E

24) https://www.youtube.com/watch?v=nbIa0sbGhBk

25) https://www.simplilearn.com/internet-of-things-iot-projects-article