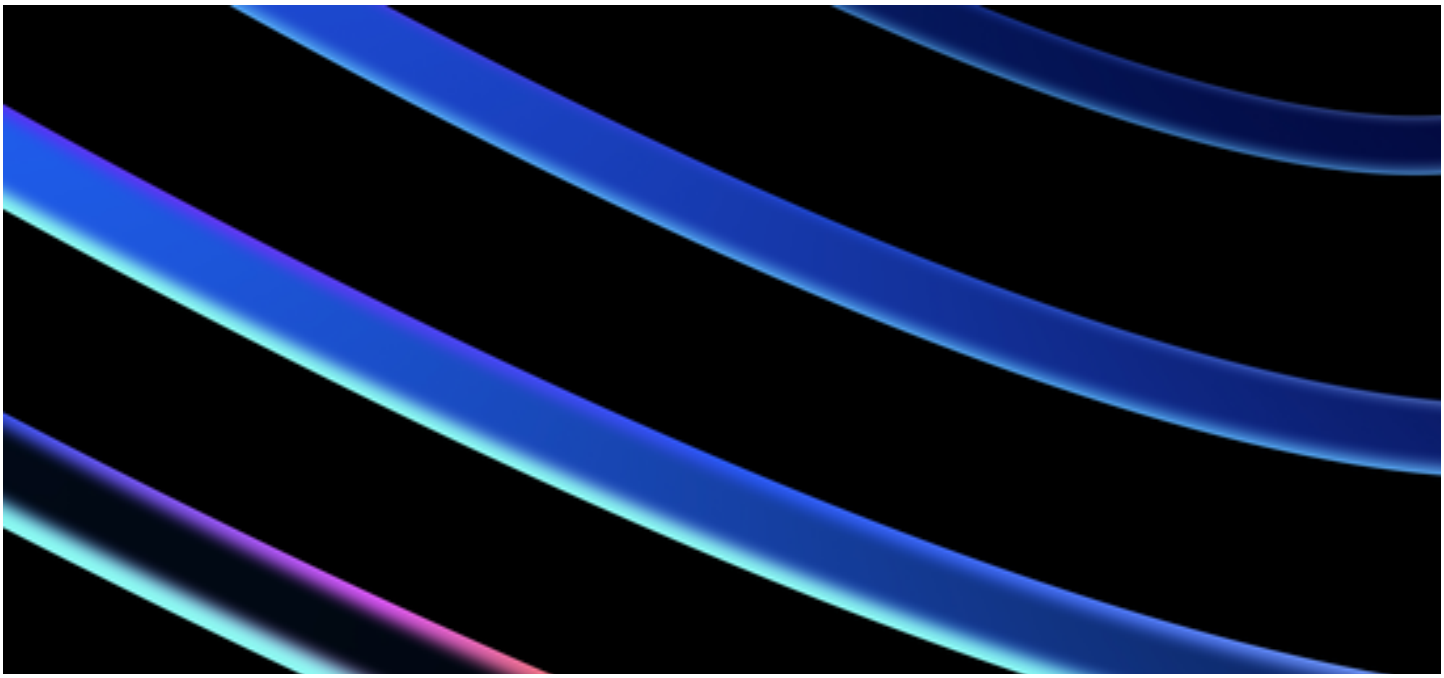


IOT Team 20 Assignment 2 report

Assignment background

This assignment is part of the **Summer 2025 Field Training** for the *IoT Software & Hardware Engineering* course. It aims to reinforce students' understanding of key concepts in embedded systems and electronics through both theoretical questions and practical projects using the ESP32 microcontroller.

The assignment challenges students to explore the differences between analog and digital signals, understand the operation of common components like potentiometers, LEDs, LDRs, and push buttons, and apply these components in real-world IoT circuits.

By completing this assignment, students will gain hands-on experience in:

- Reading analog inputs and using PWM for output control
- Designing circuits and simulating them using Wokwi
- Writing functional code for ESP32 that interacts with sensors and actuators
- Implementing basic control logic and debouncing techniques
- Creating multi-input systems and understanding how different inputs can control outputs

In addition, the bonus and analog-only projects provide an opportunity to deepen understanding of circuit behavior both with and without a microcontroller, enhancing the students’ flexibility in IoT system design.

This integrated approach is designed to prepare students for real-world IoT applications by blending theoretical knowledge with practical skills.

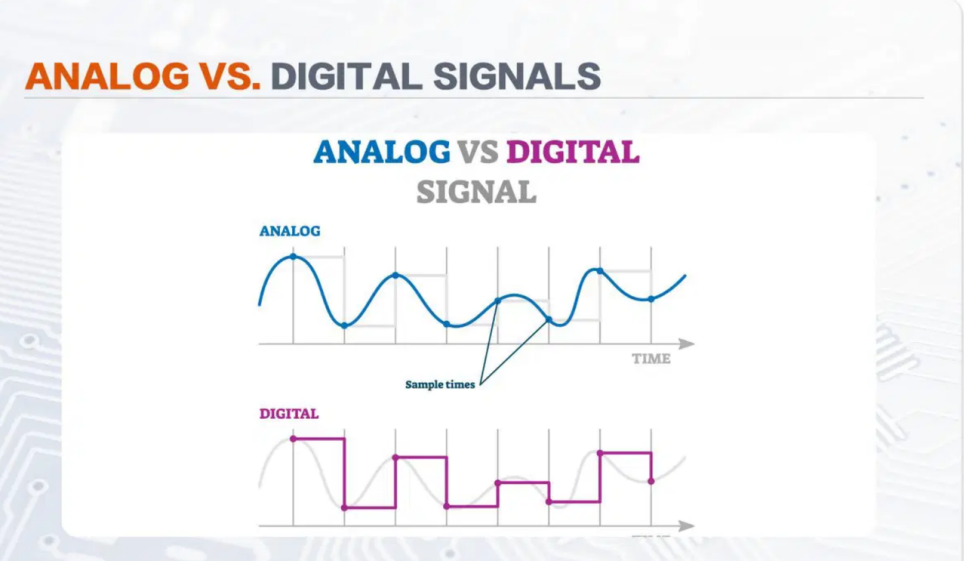
Team Members

Team member	Id
Mostafa Mohamed Morshedy (Team leader)	23011156
Yousef Khaled Shawkyy	23011635
Yousef Othman Mohamed	23011645
Badr Ahmed Mohamed	23011235
Yehia Mostafa Mohamed	23011620

Problem 1

1-

Attribute	Digital Input	Analog Input
Signal	Analog inputs require lower bandwidth when compared to digital inputs. It's one of the few advantages analog inputs have over digital inputs.	Analog inputs use a continuous signal.
Physical Representation	We usually represent digital inputs as a square wave.	Analog inputs usually come in the form of sine waves. However, on rare occasions, they may also present themselves as triangle waves.
Bandwidth Requirements	Digital input signals are highly resistant to electronic noise, making them more reliable and accurate than analog input signals.	Analog signals are highly susceptible to electronic noise. PCBs and circuits can experience sudden changes (such as spikes or dips). This causes noise. In such circumstances, analog signals' current can be disrupted, rendering them inaccurate.
Suitable Applications	Ideal for communications between computational machines and digital electronics.	Analog input signals are ideal for media applications and sensor information. This includes audio, video and various forms of measurement data.
Susceptibility to Electronic Noise	In computer peripherals such as keyboards. They send digital signals to and from your computer in a stream of binary numbers to and from your computer.	In computer peripherals such as keyboards. They send digital signals to and from your computer in a stream of binary numbers.
Power Consumption	Because digital inputs do not transmit in a single continuous stream, they're more power efficient than analog input signals.	Analog signals transmit in a single continuous stream. Thus, they have relatively high power consumption.
Circuit Components	In computer peripherals such as keyboards. They send digital signals to and from your computer in a stream of binary computer and from your computer.	In computer peripherals such as keyboards. They send digital signals to and from your computer in a stream of binary Computerand from your computer.
Use Cases	In computer peripherals, keyboards. They senrom your computer in a stream of binary numbers to and from your computer.	Electric fans, sensors and volume controls for radios are some of the most widespread uses for analog signals.



Source : https://www.wellpcb.com/blog/pcb-basics/digital-input-vs-analog-input/

2-

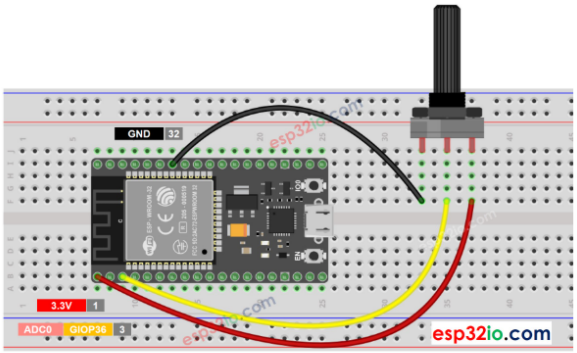
How Potentiometer Works

The potentiometer's shaft can be rotated from 0° (closest to GND pin) to an angle (closest to VCC pin), called `ANGLE_MAX`.

The voltage in the output pin is in direct proportion to the angle position of the shaft, varing from 0 to **VCC**. The below table shows the relation between the angle and the voltage on the output pin:

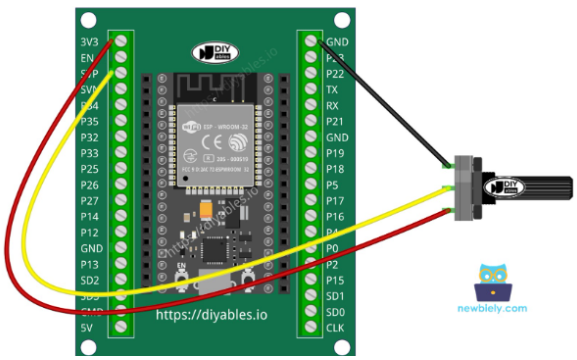
Angle	Voltage
0°	0v
`ANGLE_MAX`°	VCC
angle°	angle° × **VCC** / `ANGLE_MAX`°

> ※ ***NOTE THAT:***
>
> The `ANGLE_MAX` value is depended on manufacturers.



This image is created using Fritzing. Click to enlarge image

How to connect ESP32 and potentiometer using screw terminal block breakout board



Source : https://esp32io.com/tutorials/esp32-potentiometer

3-

Waveform Description

A PWM signal is a rectangular (square-like) waveform that alternates between “on” (high voltage) and “off” (zero or low voltage). Its key parameters include:

- **Frequency (f)**: How many ON/OFF cycles occur per second
- **Period (T)**: The inverse of frequency (T = 1/f)
- **Duty Cycle (D)**: The ratio of ON time to total period, expressed as a percentage = PW/T×100%

. ESP32 PWM – LEDC Peripheral

The ESP32 includes a dedicated PWM controller called **LEDC (LED Controller)** used for generating configurable PWM signals. Here's how it works:

- **Channels**: Multiple independent PWM outputs (commonly 6–16)
- **Configurable frequency**: Typical settings are 1 kHz–5 kHz or higher
- **Resolution**: Up to 13-bit, allowing 8192 discrete duty steps (≈ 0.012% granularity) at lower frequencies

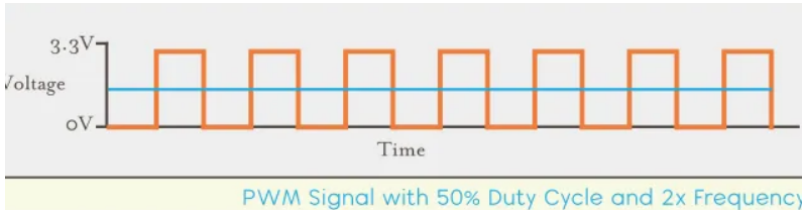
How PWM Controls LEDs or Motors

- **LEDs**: Varying duty cycle changes the average voltage, altering brightness
- **Motors**: Higher duty = more average current = faster rotation
- **Smooth control**: The ON/OFF switching is so fast it appears analog to the load WikipediaThomson Linear

A Reddit user explains it well:

> “The duty-cycle is how long the signal is ‘high’ ⋯ what you see is an ‘average’ ⋯ a small duty-cycle gives the LED power for a very short time, you see that as a dimmed light.”

Duty Cycle	Effect
25%	LED/motor gets 25% of max power → dim or slow
50%	Standard square wave → medium brightness/ speed
75%	Brighter/faster
0% or **100%**	Always OFF or always ON (rarely useful in PWM) (Soldered Electronics, Cadence PCB Resources)



PWM Signal with 50% Duty Cycle and 2x Frequency

Source : https://en.wikipedia.org/wiki/Pulse-width_modulation ,https://www.reddit.com/

4-

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. LDR is also known as a Light Sensor. LDR are usually available in 5mm, 8mm, 12mm, and 25mm dimensions.

The Light-dependent resistors made with photosensitive semiconductor materials like Cadmium Sulphides (CdS), lead sulfide, lead selenide, indium antimonide, or cadmium selenide

and they are placed in a Zig-Zag shape as you can see in the pic below.

Two metal contacts are placed on both ends of the Zig-Zag shape these metal contacts help in creating a connection with the LDRs.

Now, a transparent coating is applied on the top so that the zig-zag-shaped photosensitive material gets protected and as the coating is transparent the LDR will be able to capture light from the outer environment for its working.

LDR Working Principle

It works on the principle of photoconductivity whenever the light falls on its photoconductive material, it absorbs its energy and the electrons of that photoconductive material in the valence band get excited and go to the conduction band and thus increasing the conductivity as per the increase in light intensity.

Also, the energy in incident light should be greater than the bandgap gap energy so that the electrons from the valence band got excited and go to the conduction band.

We have a detailed article on working, circuit, and construction of LDR.

The LDR has the highest resistance in dark around 1012 Ohm and this resistance decreases with the increase in Light.

Light Dependent Resistor

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

---

5-

1. **Push button behavior as a digital input**

A push button is a simple mechanical switch that connects or disconnects a circuit when pressed or released. Connected to a digital GPIO pin (often with a pull-up or pull-down resistor), it reports either **HIGH** or **LOW**.
However, due to mechanical “bouncing,” the signal doesn't switch cleanly once—it often flickers rapidly before settling. This can lead the microcontroller to register multiple false

presses.

2. **Common issues when reading a push button**

- **Signal bouncing**: A single press can generate ~5–10 rapid transitions within 1–20 ms
- **False triggers**: The MCU may detect multiple presses/releases when only one occurred.
- **Unreliable state changes**: Bouncing can cause erratic behavior, like double toggles or skipped states

3. **Debouncing explained**

Debouncing filters out bounce-induced noise so that each physical press or release is registered once.

Two main techniques:

a) Hardware debouncing

- Uses simple RC filters (resistor + capacitor) or Schmitt-trigger circuits
- Smooths out bouncing signal before it reaches the MCU

b) Software debouncing

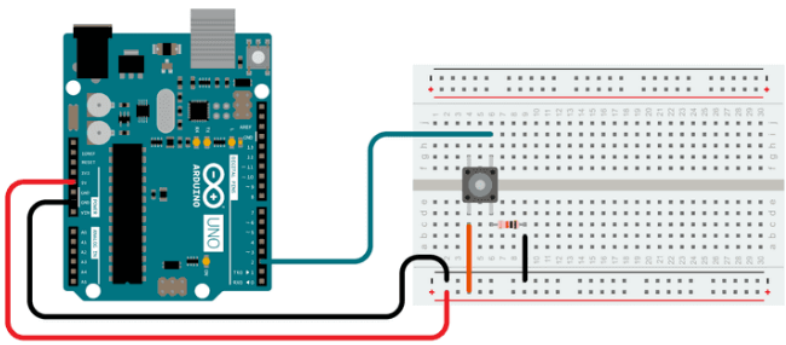
- MCU code ignores rapid toggles after detecting a press
- Common strategy: after detecting a state change, wait a defined interval (e.g., 50 ms) before accepting another change

```
const int BUTTON_PIN = 21;
const unsigned long DEBOUNCE_DELAY = 50;
int lastState = HIGH;
int stableState = HIGH;
unsigned long lastChangeTime = 0;

void setup() {
 pinMode(BUTTON_PIN, INPUT_PULLUP);
 Serial.begin(9600);
}

void loop() {
 int reading = digitalRead(BUTTON_PIN);
 if (reading != lastState) {
```

```
    lastChangeTime = millis();
  }
  if ((millis() - lastChangeTime) > DEBOUNCE_DELAY) {
    if (reading != stableState) {
      stableState = reading;
      if (stableState == LOW) {
        Serial.println("Button pressed");
      } else {
        Serial.println("Button released");
      }
    }
  }
  lastState = reading;
}
```



Sources : https://circuitdigest.com/electronic-circuits/what-is-switch-bouncing-and-how-to-prevent-it-using-debounce-circuit , https://esp32.com/viewtopic.php?t=15089&utm_source=chatgpt.com , https://docs.arduino.cc/built-in-examples/digital/Debounce/ , https://esp32io.com/tutorials/esp32-button-debounce

---

6-

PWM vs. Analog — What’ s the difference?

- **True analog signal**: Outputs a continuous range of voltages (e.g., 0–3.3 V or 0–5 V), directly set by a DAC or voltage source.

- **Pulse Width Modulation (PWM)**: A digital square wave (only HIGH or LOW) where the **duty cycle** (the proportion of ON vs. OFF time) controls the **average output voltage**

Can PWM simulate analog output?

Yes, PWM can simulate analog output effectively **if filtered or if the load averages the signal**:

- With an **RC low-pass filter**, PWM is transformed into a smooth voltage.
- Many devices — **LEDs, motors, heaters, inductive loads** — naturally smooth PWM pulses internally

---

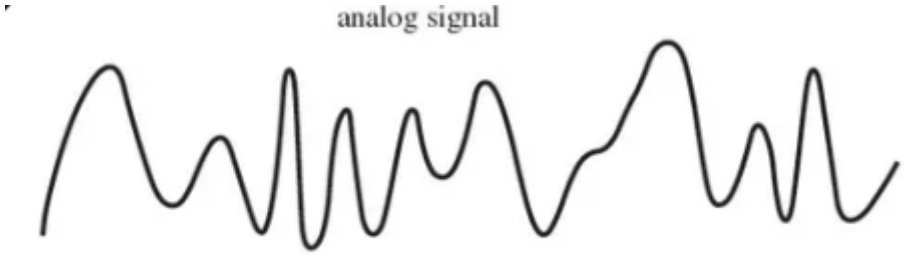
Advantages of PWM

1. **High efficiency** – MOSFETs switch fully on/off, minimizing heat loss
2. **Low cost** – Avoids the need for DAC chips or complex analog circuits
3. **Built into microcontrollers** – Easily generated with timers
4. **Ideal for power control** – Used in motor drives, LED dimmers, switch-mode supplies

---

Limitations of PWM

- **Not a true analog voltage** – Requires filtering to smooth out ripples and harmonics
- **Resolution & ripple tradeoff** – Finite timer resolution leads to stepped voltage levels; filtering reduces ripple but limits bandwidth
- **Switching noise** – Loud harmonics, EMI, potential interference with sensitive circuits
- **Not ideal for high fidelity** – Audio, precision instrumentation, and communication require clean continuous output — better served by a DAC



Sources : Adafruit Learning System+15Wikipedia+15Core Electronics Forum+15, Adafruit Learning System+3Reddit+3Arduino Forum+3, Arduino Forum, Arduino Forum+4Core Electronics Forum+4TI

7-

LED Working Principle

- An LED (Light-Emitting Diode) is a **semiconductor diode** that emits light when forward-biased (current flows from anode to cathode).
- It has a characteristic **forward voltage drop** (V☒☒), typically ~1.8 V (red) to ~3.3 V (blue/white). Once above this threshold, current increases rapidly with small voltage increases
- Electrons recombine with holes in the semiconductor, releasing energy as photons (light).

Controlling LEDs with the ESP32

- The ESP32’ s GPIO pins are **digital** — they output either 0 V (LOW) or 3.3 V (HIGH).
- **PWM (Pulse Width Modulation)** is used to simulate analog brightness control: it rapidly switches the GPIO HIGH and LOW, varying the **duty cycle** to adjust average power
- ESP32 uses its **LEDC peripheral**, allowing up to 16 channels, frequencies around 5 kHz, and resolutions up to 13 bits (~0.012% granularity)
- Code example for fading an LED:

ledcSetup(channel, freq, resolution);

ledcAttachPin(ledPin, channel);

ledcWrite(channel, duty);

Thus turning an LED ON/HIGH means duty=100% (fully lit), OFF/LOW means duty=0%, and any intermediate value dims it smoothly.

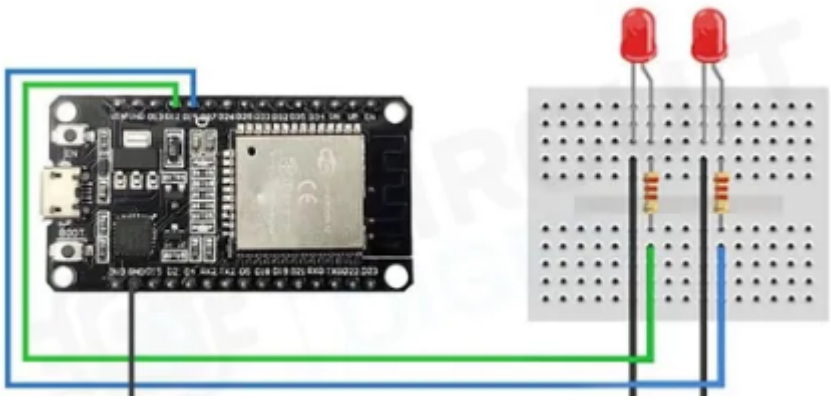
Role of a Current-Limiting Resistor

- The resistor is **essential**: LEDs have a non-linear I–V curve. Slight voltage changes cause large current jumps beyond safe limits

---

What Happens If You Omit the Resistor

- Without current limiting, the LED draws **excessive current**, enters a low-resistance state, and **overheats** until it burns out. The ESP32 pin may also be **overloaded and damaged**
- Even if not immediate, the LED or MCU may enter a **stressed state**, reducing lifespan or causing malfunction.



Sources : https://en.wikipedia.org/wiki/LED_circuit , https://circuitdigest.com/microcontroller-projects/esp32-pwm-tutorial-controlling-brightness-of-led

---

8-

How do internal ADCs in the ESP32 work?

- The ESP32 contains **two SAR (Successive Approximation Register) ADCs**, known as ADC1 and ADC2, offering a total of up to **18 analog input channels** across specific GPIOs
- Each ADC supports configurable resolution: **9-bit**, **10-bit**, **11-bit**, or **12-bit**, with **12-bit (0–4095)** being the default
- The ADC measures input voltages relative to an internal reference (**Vref ≈ 1.1 V**). Using built-in **attenuation settings** (0 dB, 2.5 dB, 6 dB, 11 dB), the measurable input range can be extended from about **0 V to ~0.95 V, 1.25 V, 1.75 V, and ~2.45 V**, respectively
- Many tutorials describe it as measuring **0–3.3 V** range, though this effectively depends on your attenuation configuration and board's wiring

Vout=DmaxDout×Vmax

where Dmax=4095D_{\max} = 4095Dmax=4095 at 12-bit resolution and VmaxV_{\max}Vmax depends on the chosen attenuation

Note: ADC2 channels are **unavailable when Wi-Fi is active**, so ADC1 pins are preferred for reliability

Keep in mind that the ESP32 ADC exhibits **nonlinearity**, **noise**, and often limited effective accuracy — many hobbyists find its true performance to be closer to ~6–8 bits or around ~8 bit effective resolution unless oversampling and calibration are used

Sources : https://docs.espressif.com/projects/esp-idf/en/v4.4.1/esp32/api-reference/peripherals/adc.html , https://www.electronicshub.org/esp32-adc-tutorial/

9-

Potentiometer used for user input

Smart thermostat / temperature setting knob

In consumer devices like smart thermostats or espresso machines, a **rotary potentiometer** acts as an intuitive user **input control**: turning the knob adjusts setpoints such as desired temperature. One Reddit user described tapping the pot’ s output voltage into a microcontroller to display the inferred temperature setting on a screen

IoT/industrial control panels

In industrial or home automation setups, potentiometers serve as adjustable analog inputs—for example, to set motor speed, lighting level, or audio volume. These wiper voltages feed into the ESP32’ s ADC, enabling digital tracking of user adjustments and enabling remote monitoring/control capabilities

LDR (Light Dependent Resistor) for environmental sensing

Smart street lighting

IoT-enabled smart street-light systems commonly employ LDRs as light sensors. The LDR’ s resistance changes with ambient daylight. When light drops below a threshold (e.g. dusk), the system—often an ESP-based controller—turns street lights on; when light returns in the morning, lights are turned off. Some systems also adjust brightness levels via PWM based on ambient intensity

Smart home/outdoor lighting automation

LDRs are integrated in IoT home lighting controls to automatically adjust indoor lighting based on natural ambient light levels (e.g. dimming or brightening smart bulbs). Light level data can be tracked in the cloud, improving energy savings and user comfort

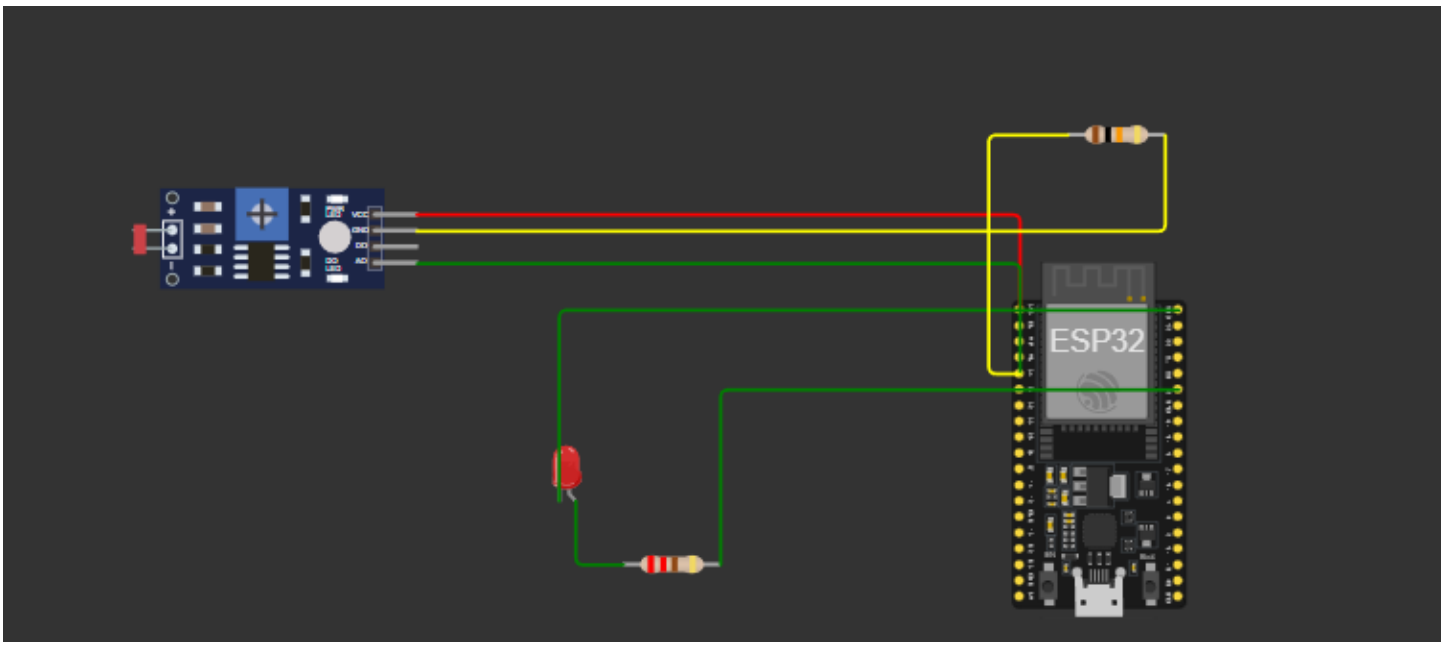
Sources : rfwireless-world.comcavliwireless.com, Medium+4ijert.org+4discovery.researcher.life+4,

---

10-

1. LDR Generates an Analog Voltage

- The **LDR (Light Dependent Resistor)** is part of a **voltage divider** with a fixed resistor (e.g., 10 kΩ).
- The output of the divider (connected to an ADC pin like GPIO 34) produces a voltage that changes with **light intensity**:
  - **Bright light** → LDR resistance is **low** → output voltage is **low**
  - **Darkness** → LDR resistance is **high** → output voltage is **high**



Project 1

Project Link: Potentiometer-Based LED - Wokwi ESP32, STM32, Arduino Simulator

Simulation video link: drive.google.com

Hardware implementation: drive.google.com

Project 2

Project Link: LDR-Controlled Night Light - Wokwi ESP32, STM32, Arduino Simulator

Simulation Video Link: drive.google.com

Hardware implementation : drive.google.com

Project 3

Simulation : finaltask3 - Wokwi ESP32, STM32, Arduino Simulator

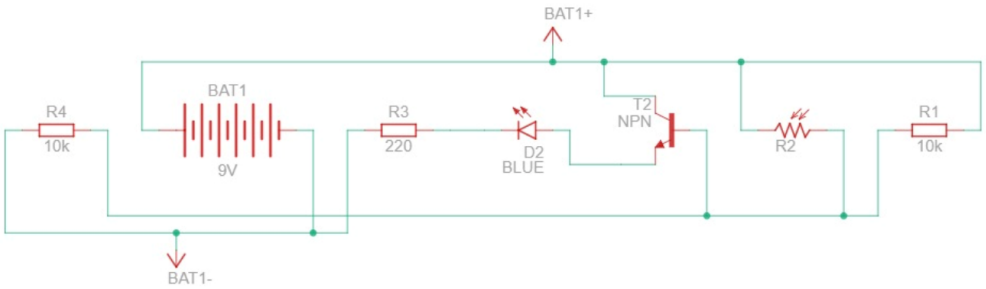
Hardware implementation : drive.google.com

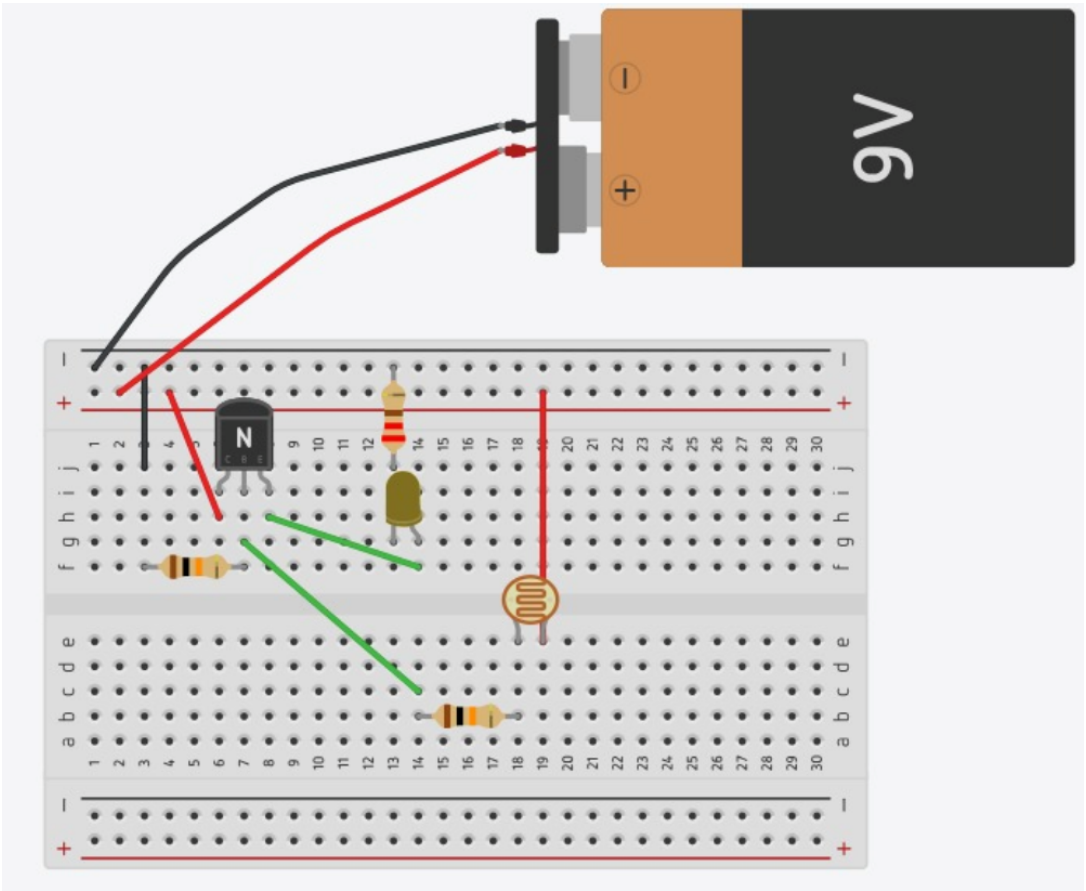
Project 4

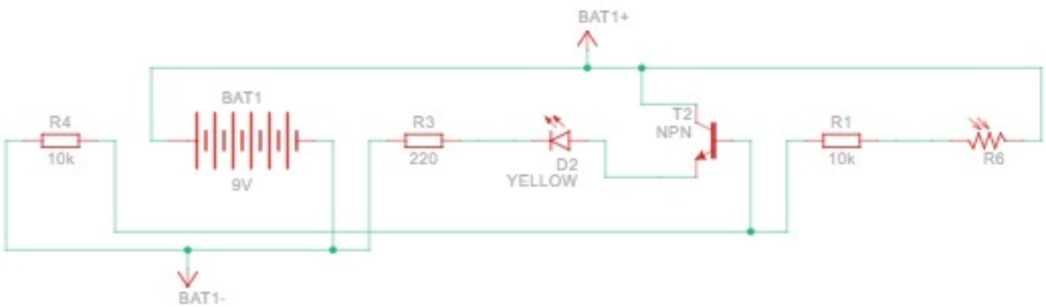
Simulation : all - Wokwi ESP32, STM32, Arduino Simulator

Hardware implementation :drive.google.com

Project 6







Project 7

Simulation Video Link: drive.google.com