

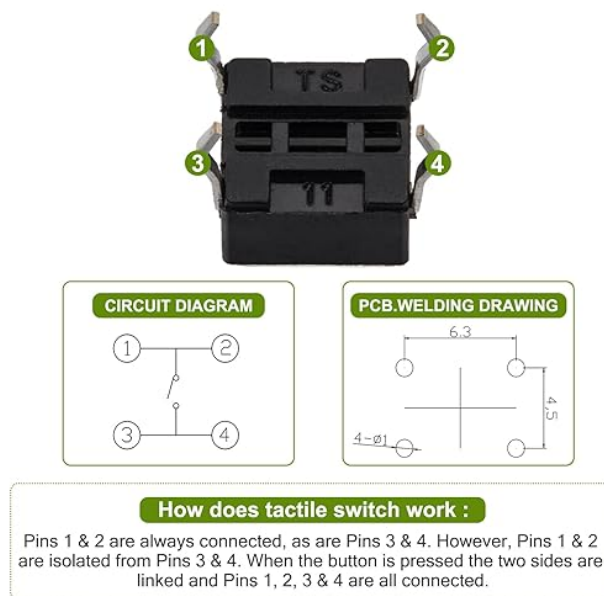
Electronics Workshop-1: Lab 6 Handout

In this lab, you'll be introduced to ESP32 through two experiments. Here are the components you'll be using in the experiments.

Push Button

A **push button** is a simple momentary switch. It makes or breaks an electrical connection only while it is being pressed. When you release it, it returns to its default state (usually open).

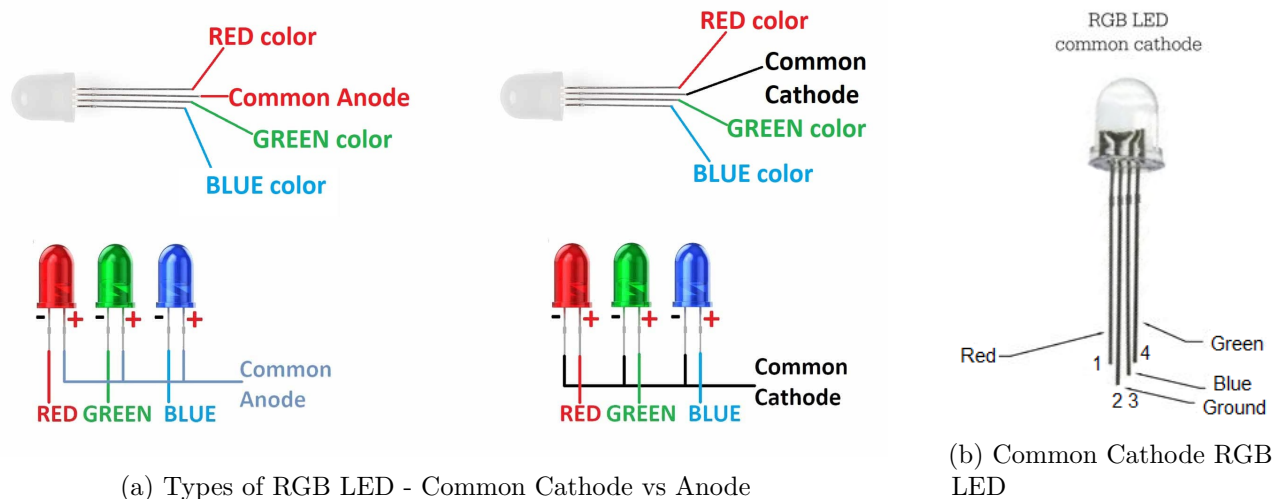
How it works: Internally, a push button connects two terminals when pressed, allowing current to flow and completing the circuit.



RGB LED

An RGB (Red, Green, Blue) LED combines three separate LEDs (one red, one green, and one blue) into a single package. There are two categories, namely

- **Common Cathode:** All three LEDs share a common ground (GND) pin. This is the type we will use.
- **Common Anode:** All three LEDs share a common power (VCC) pin.



(a) Types of RGB LED - Common Cathode vs Anode

(b) Common Cathode RGB LED

IR Sensor

An **Infrared (IR) sensor** is used for obstacle detection. It consists of two main parts: an **IR emitter** (an LED that emits infrared light) and an **IR receiver** (a photodiode that is sensitive to infrared light).

How it works: The emitter sends out a beam of IR light. If this light hits an object, it reflects back and is detected by the receiver. This detection triggers the sensor's output pin to change its state, signaling the presence of an object to the microcontroller.

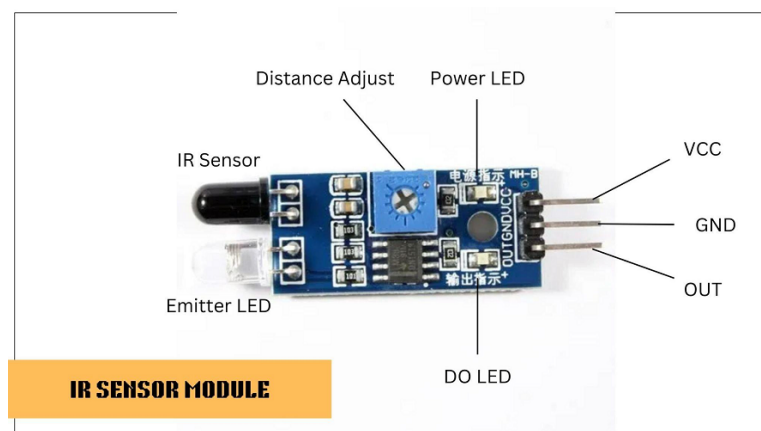


Figure 2: Pinout of an Infrared (IR) sensor

Servo Motor

A **servo motor** is a rotary actuator that allows for precise control of angular position. Unlike a standard DC motor that rotates continuously, a servo can be told to **move to and hold a specific angle** (e.g., 0°, 90°, 180°). This is achieved by sending a specific Pulse Width Modulation (PWM) signal to its signal pin. [Reference video](#)



Figure 3: Pinout of Servo Motor (Supply is +3.3V, not 5V)

ESP32

The **ESP32** is a powerful and affordable microcontroller with built-in Wi-Fi and Bluetooth. Its versatility makes it a popular choice for Internet of Things (IoT) projects and robotics. In this lab, we will use its General Purpose Input/Output (GPIO) pins to interact with electronic components.

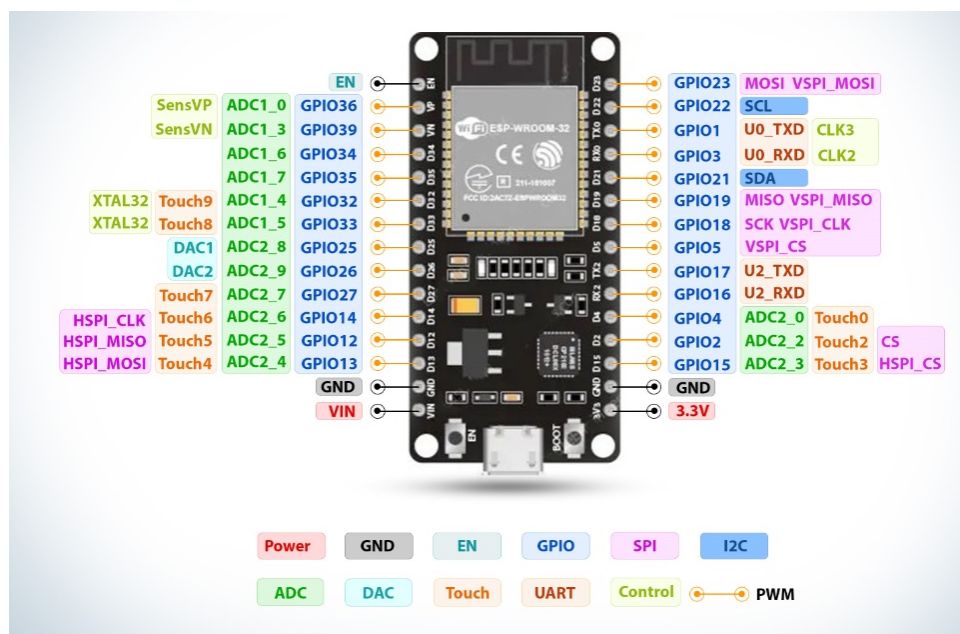


Figure 4: ESP32 Pinout Diagram

Here's a [troubleshooting guide](#) if you're facing any issues related to ESP32. Programming of the ESP32 can be done using Arduino IDE.

General Purpose Input/Output (GPIO) Functions

These functions are used to control and read the state of the ESP32's digital pins.

`pinMode(pin, mode)` Configures a specific GPIO pin to behave as an input or an output.

- **OUTPUT:** Sets the pin to be an output (e.g., for an LED).

- **INPUT**: Sets the pin to be an input (e.g., for an IR sensor).

digitalWrite(pin, value) Writes a digital value (**HIGH** or **LOW**) to a pin that has been configured as an **OUTPUT**.

digitalRead(pin) Reads the digital value (**HIGH** or **LOW**) from a specified input pin. This is used to check the state of a button or sensor.

Servo Motor Control Functions

These functions are specific to controlling a servo motor and require including the servo library at the top of the sketch with `#include <ESP32Servo.h>`.

Servo <object_name>; This line creates a “servo object” called **<object_name>**. An object is a special variable that has its own functions. You can name it anything you like (e.g., **gateServo**).

myservo.attach(pin) Connects the servo object to a specific GPIO pin on the ESP32. This pin must be capable of generating PWM (Pulse Width Modulation) signals.

myservo.write(angle) Commands the servo to rotate to a specific angle. The **angle** is typically a value between 0 and 180 degrees.

Essential Functions for Debugging

When your circuit does not work as expected, you need a way to see what the ESP32 is doing. Serial communication allows the ESP32 to send messages and data to your computer, which you can view in the Arduino IDE’s Serial Monitor (**Tools > Serial Monitor**).

Serial.begin(speed) This function must be called once in **setup()** to initialize serial communication. The **speed**, or **baud rate**, determines how fast data is sent. A speed of 115200 is standard for the ESP32.

Experiment 1: RGB LED Control Using Push Button

Objective

To control the color of an **RGB LED** using a **push button**. Each press will cycle the LED through different colors, demonstrating basic digital input and output.

Components Required

- ESP32 development board, Common cathode RGB LED, Push button
- $3 \times 220\Omega$ resistors, Jumper wires & Breadboard

Circuit Connections

Ensure there's a resistor in series with the LED. **DO NOT** connect the LED directly to a source.

Wiring for ESP32 + RGB LED + Push Button

Component Pin	Connection
RGB LED Red (R)	ESP32 GPIO through 220Ω resistor
RGB LED Green (G)	ESP32 GPIO through 220Ω resistor
RGB LED Blue (B)	ESP32 GPIO through 220Ω resistor
RGB LED Cathode (–)	ESP32 GND
Push Button one side	ESP32 GPIO
Push Button other side	ESP32's Vin/GND accordingly

Procedure

1. Assemble the circuit.
2. Open Arduino IDE. Ensure that Arduino IDE detects the connected ESP32 Dev Module.
3. Type your code in the Arduino IDE
4. Upload the code and press the button to test.

You'll observe that each button press cycles the RGB LED through: **Red** → **Green** → **Blue** → **Off**.

Fun Optional Activity

Try [generating different colors](#) by assigning different combinations of HIGH and LOW to Red, Green and Blue pins accordingly.

Experiment 2: Servo Motor Control Using IR Sensor

Objective

To use an **IR sensor** to detect an object and command a **servo motor** to move. The ESP32 relays the signal from IR sensor to servo motor.

Components Required

- ESP32 development board, IR sensor module (HW201), Servo motor (SG90), Jumper wires

Circuit Connections

Wiring for ESP32 + IR Sensor + Servo Motor

Component Pin	Connection
IR Sensor VCC	ESP32 Vin(3.3V) pin
IR Sensor GND	ESP32 GND pin
IR Sensor OUT	ESP32 GPIO
Servo VCC (Red)	ESP32 Vin pin
Servo GND (Brown)	Common GND with ESP32
Servo Control (Orange)	ESP32 GPIO

Procedure

1. Install the **ESP32Servo** library by Kevin Harrington, John K. Bennett from the Arduino Library Manager.
2. Assemble the circuit
3. Type your code in the Arduino IDE. Rotate the servo motor by 90° when an object is detected
4. Redo it for different angles