Lab 1: Exploring Basic Circuits on ESP32

Introduction

ESP32 is a highly integrated solution for Wi-Fi and Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, power amplifier, low noise receive amplifier, filters, and power management modules. The ESP32 boards can be programmed using many different programming languages. For example, you can program your ESP32 board in C++ language (like the Arduino) or MicroPython. And to make use of all the ESP32 features Espressif has officially provided the Espressif IoT Development Framework, or ESP-IDF (tutorial link). And using Arduino IDE is the easiest way to get started in programming the ESP32 board.

Setup for LAB experiments:

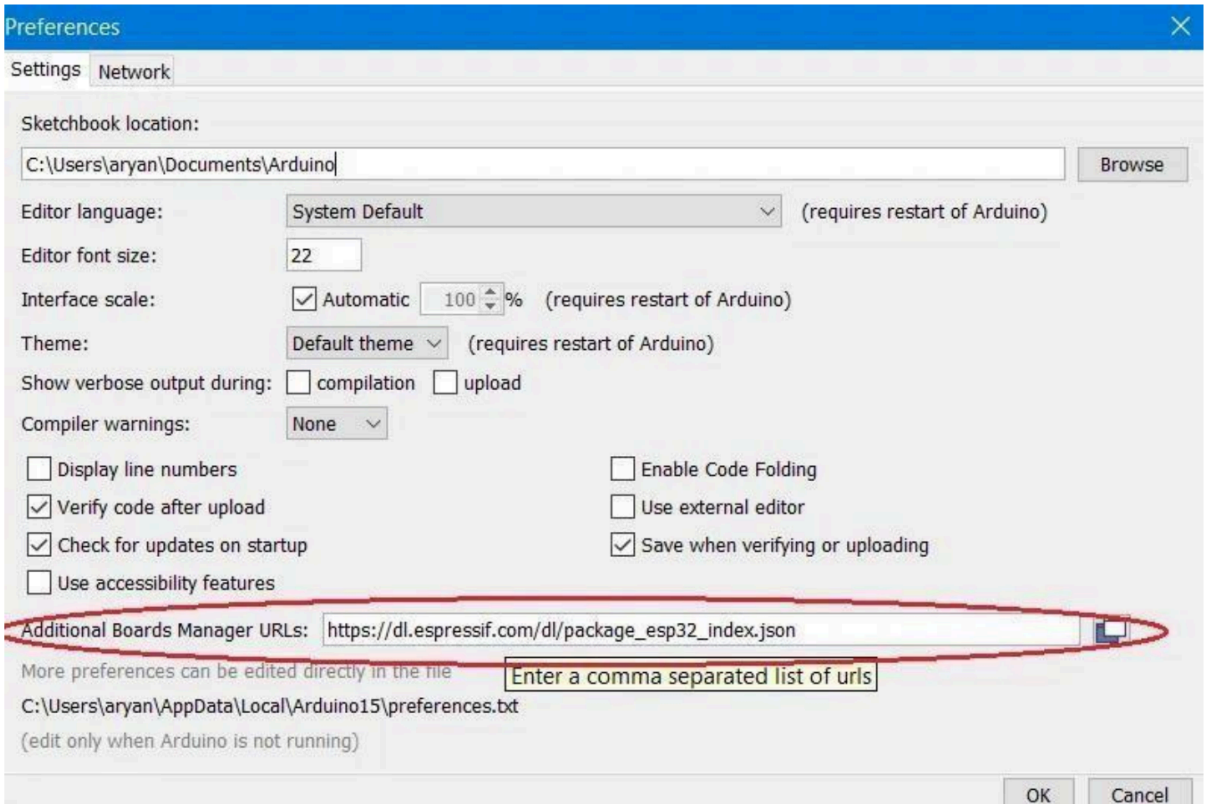
• Install the COM/Serial port driver. The new version ESP32(we use this) comes with the CP2102 serial chip, you can download and install the driver here.

• Download the Arduino IDE from the **official website of Arduino**. You can download the software for Windows, Mac, and Linux 32 bit, 64 bit.

• Install the ESP32 Board Package: (See **References**)

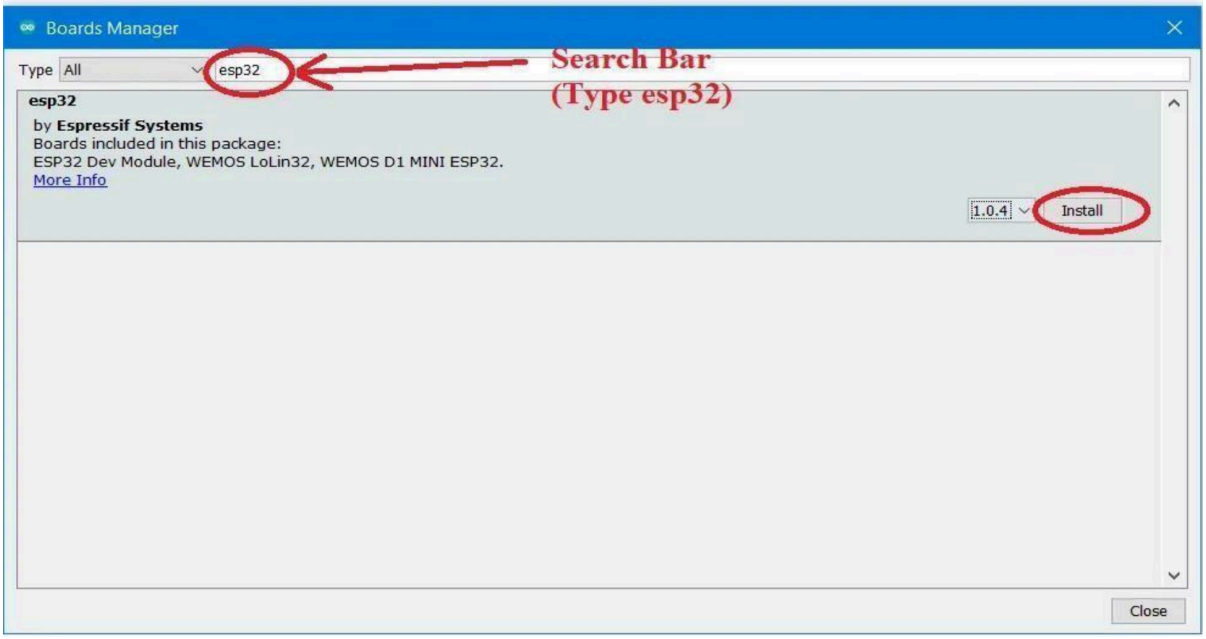
Enter Paste the URL: “https://dl.espressif.com/dl/package_esp32_index.json” into the “Additional Board Manager URLs” box as shown in the figure below. Then, click the “OK” button:

Note: In case there are already some links present in the Additional Board Manager URL, you can add new URLs by using a comma to separate them.

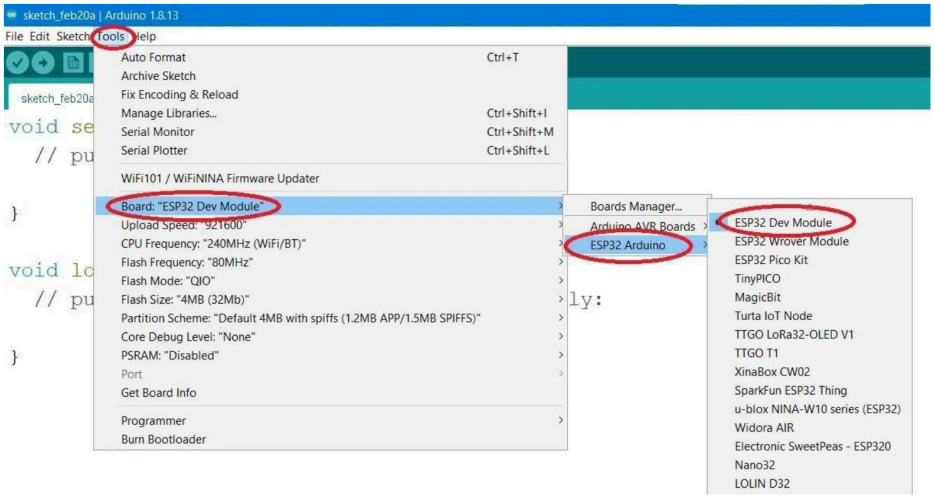


• Now open the board manager from Tools>Board>Board Manager to install ESP32 package.

• In the board manager dialog box, click on the search bar. Then type “ESP32” and hit enter. Now select and click on the install button for “ESP32 by Espressif systems”



Now after the installation, in the Tools> Board menu a new option should appear “ESP32 Arduino” as shown in the figure below. Click on this option and select the version of your ESP32 board.



Note: You’d better close the Arduino IDE and restart it again.

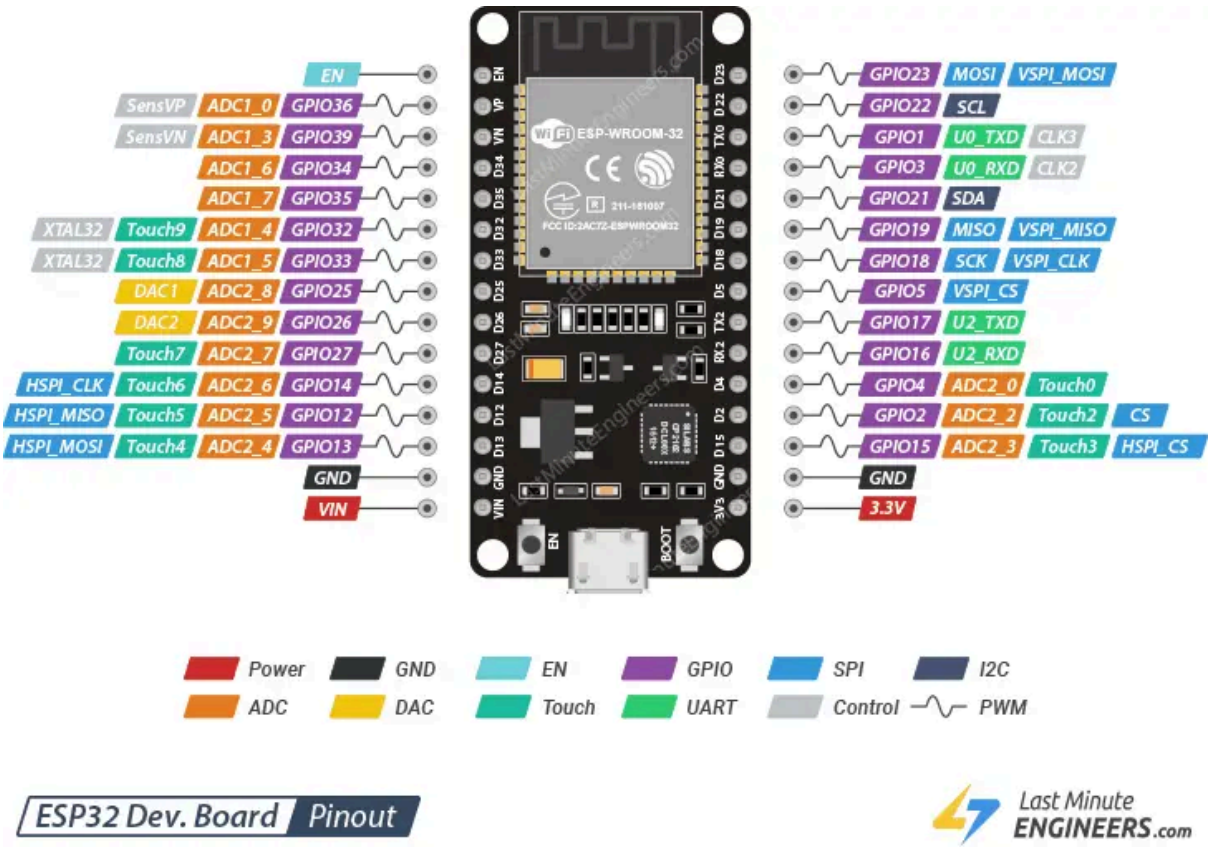
• Config the Board menu and choose the right Port for your device.

○ CPU Frequency：80MHz,

○ Flash Size：4M（3M SPIFFS），

○ Upload Speed：115200

● Use the below given PINOUT diagram of the ESP32



Experiment 1: Blinking an external LED

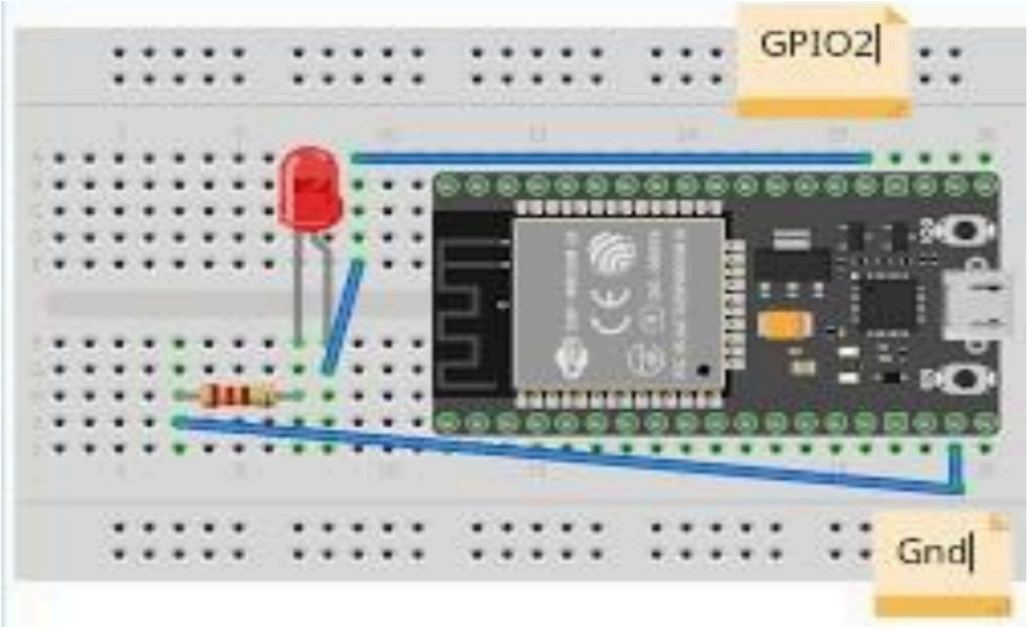
In this lab session, we will introduce how to control an external device like LED using the pins on ESP32.

● Components Required

- ESP32x 1
- LED x 1
- 200 ohm resistor x 1
- Micro USB cable x 1
- PC x 1

Connect one end of the resistor 200 ohms to D1(or any digital pin) and the other end to LED long leg (positive leg / anode). Connect the short leg (negative leg/cathode) to GND.

The value of the resistor in series with the LED may be of a different value than 200 ohm; the LED will be lit up with values up to 1K ohm.



Task:

1. Write the code to blink the LED at 1-second intervals.Modify the code to make the LED blink at variable intervals (e.g., 500ms, 1s, 2s) based on a delay value.
2. Experiment with adding a second LED and make it blink in an alternating pattern with the first LED.

Expected Output:

The external LED blinks at regular intervals, demonstrating basic control of an output device using ESP32.

Experiment 2: LED with Button (Pull-Up Resistor)

Objective

Design a circuit that lights up an LED when a button is pressed, using a pull-up resistor configuration.

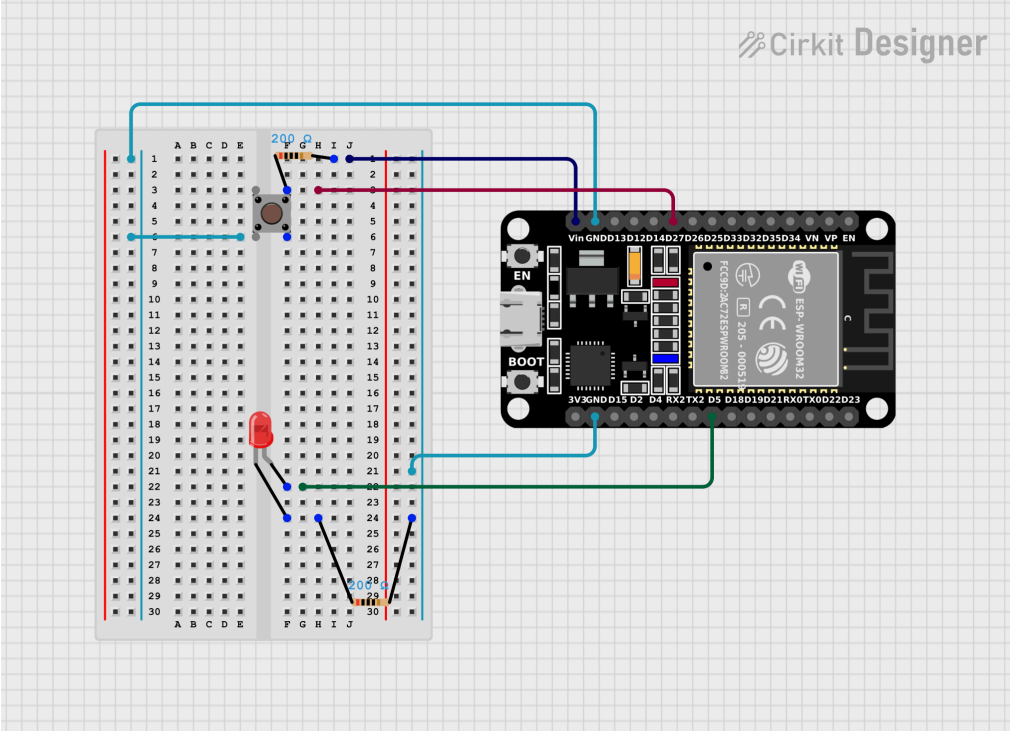
Components Required

- ESP32 Board.
- LEDs and resistor (220 Ω).
- Push buttons.
- 10 kΩ resistor (for pull-up configuration).
- Breadboard and jumper wires.

Schematic

- Connect the LED and resistor as in Experiment 1.

- Connect the button such that one leg is connected to a digital pin (e.g., pin2) and the other leg to the ground.
- Use a pull-up resistor to connect the button pin to the 5V pin of the ESP32.



Task:

Write the code to turn the LED on when the button is pressed.

Expected Output:

The LED lights up only when the button is pressed.

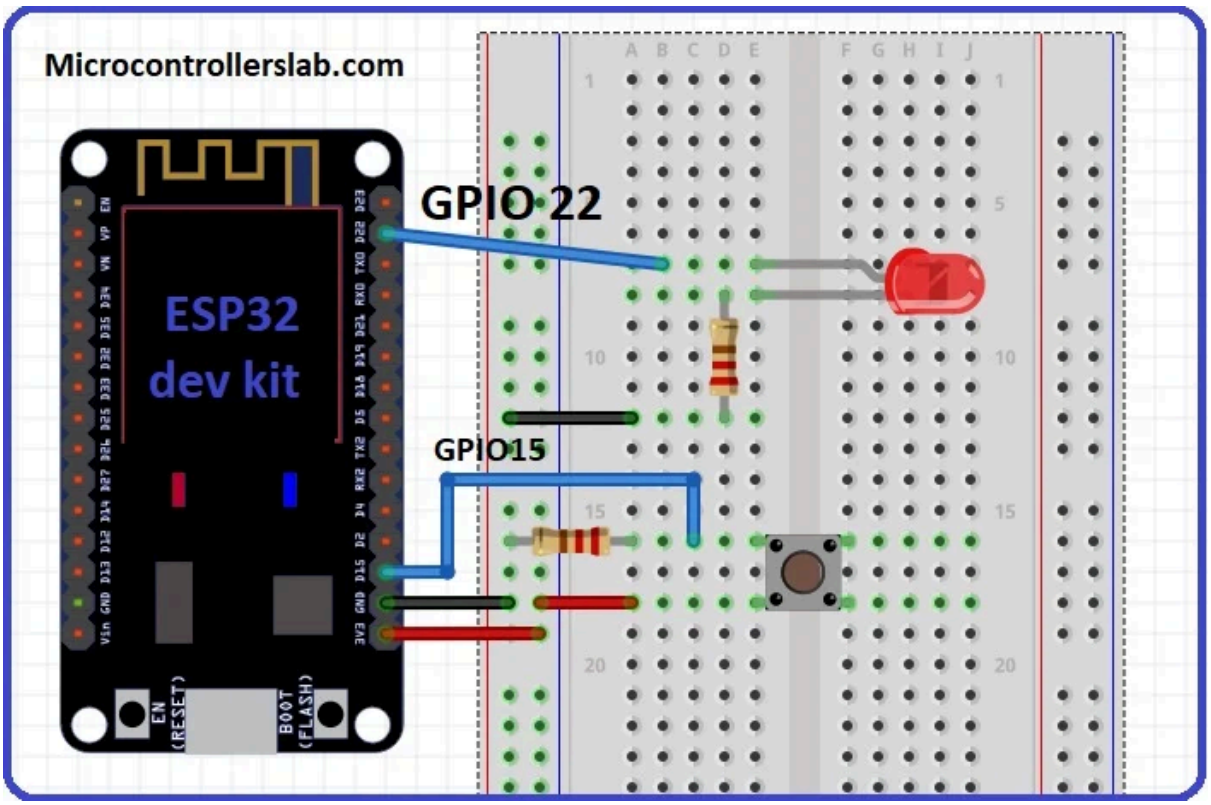
Experiment 3: LED with Button (Pull-Down Resistor)

Objective

Design a circuit that turns off an LED when a button is pressed, using a pull-down resistor configuration.

Schematic

- Modify the circuit in Experiment 2 to use a pull-down resistor:
- Connect one leg of the button to a digital pin and the other to 5V.
- Connect a 10 kΩ resistor between the button pin and ground.



Task:

Write the code to turn the LED off when the button is pressed.

Expected Output: The LED turns off only when the button is pressed.

Experiment 4a: Printing on Serial Port Monitor

In this lab session,we will introduce how to print the output of the sensor data on to serial monitor using the micro USB cable.

Components

- ESP32x 1
- Micro-USB cable x 1
- PC x1

Connect your ESP32 to the PC and put below code to your Arduino IDE

PseudoCode:

- Use Serial.begin(<baud rate>) to begin the serial communication.
- Use Serial.println(<data>) to print on the serial monitor in a new line.
- The data in the experiment can be randomly generated number using random(<min>, <max>)
- Modify the provided pseudocode to output one randomly generated number in an iterative order where the min-limit and max-limit are varying.

Consider the initial min value to be 0 and max value to be 10. In this case, the random number generated will lie between 0 and 9. Now, in the second iteration the min value=10 and max value = 20, in the third iteration min value=20, max value = 30 and so on.

Perform the above-mentioned task of random number generation and display them on the serial monitor for 20 iterations.

Experiment 4b: Reading from Serial Port Monitor

In this lab session, read the inputs using Serial.read from the serial monitor and control the LED blinking frequency.

References

Use these videos as guides to understand the experiments and concepts:

- Complete beginner's guide to using a breadboard (optional)
- Pull up/ Pull down resistor - explained
- Understanding Pushbuttons and Pull Up and Pull Down Resistors
- Installing ESP32 in Arduino IDE
- Serial port Drivers for ESP32 / Arduino
- ESP32 with Push buttons