

INTRODUCTION TO IoT

Spring 2022

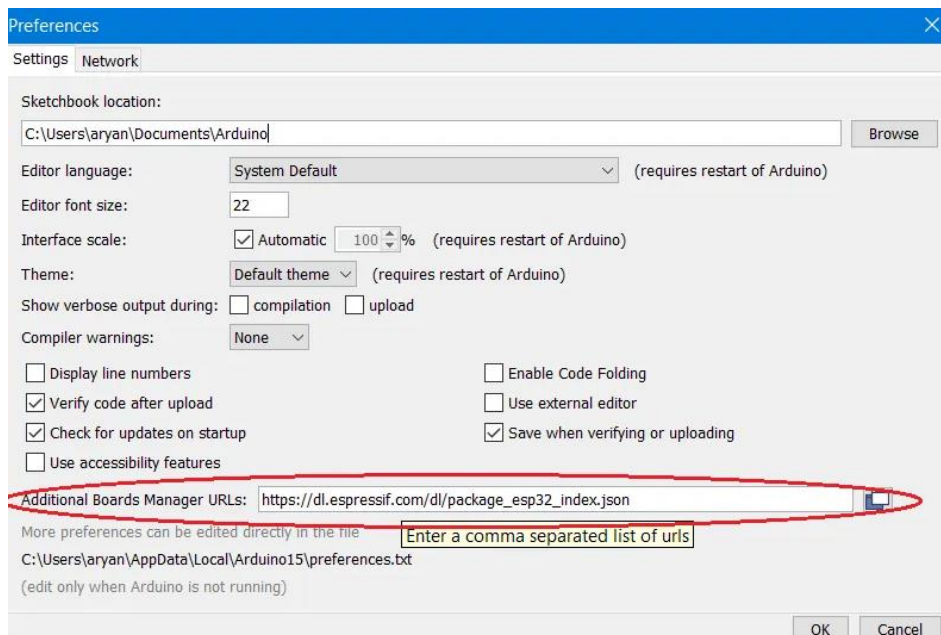
LAB-1

Introduction

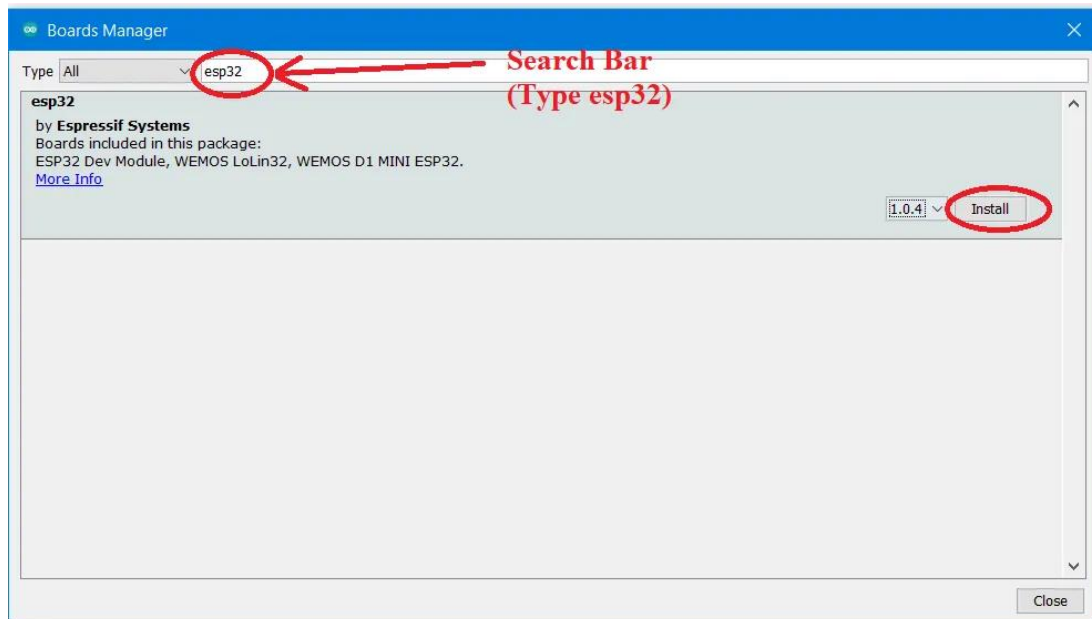
ESP32 is a highly integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, power amplifier, low noise receive amplifier, filters, and power management modules. The ESP32 boards can be programmed using many different programming languages. For example, you can program your ESP32 board in C++ language (like the Arduino) or MicroPython. And to make use of all of 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:

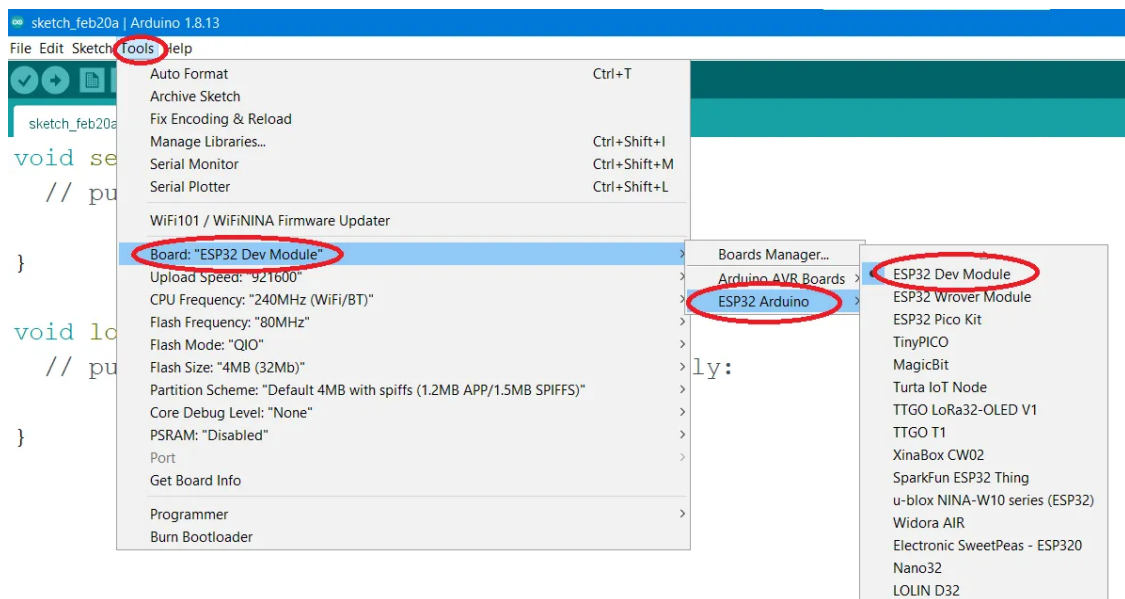
- Install the COM/Serial port driver
 - The new version ESP32(we use this) comes with the CP2102 serial chip, you can download and install [the driver here](#).
- Download the Arduino IDE from the [official website of Arduino](#). You can download the software for Windows, Mac, and Linux 32 bit, 64 bit.
- Install the ESP32 Board Package
 - 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 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

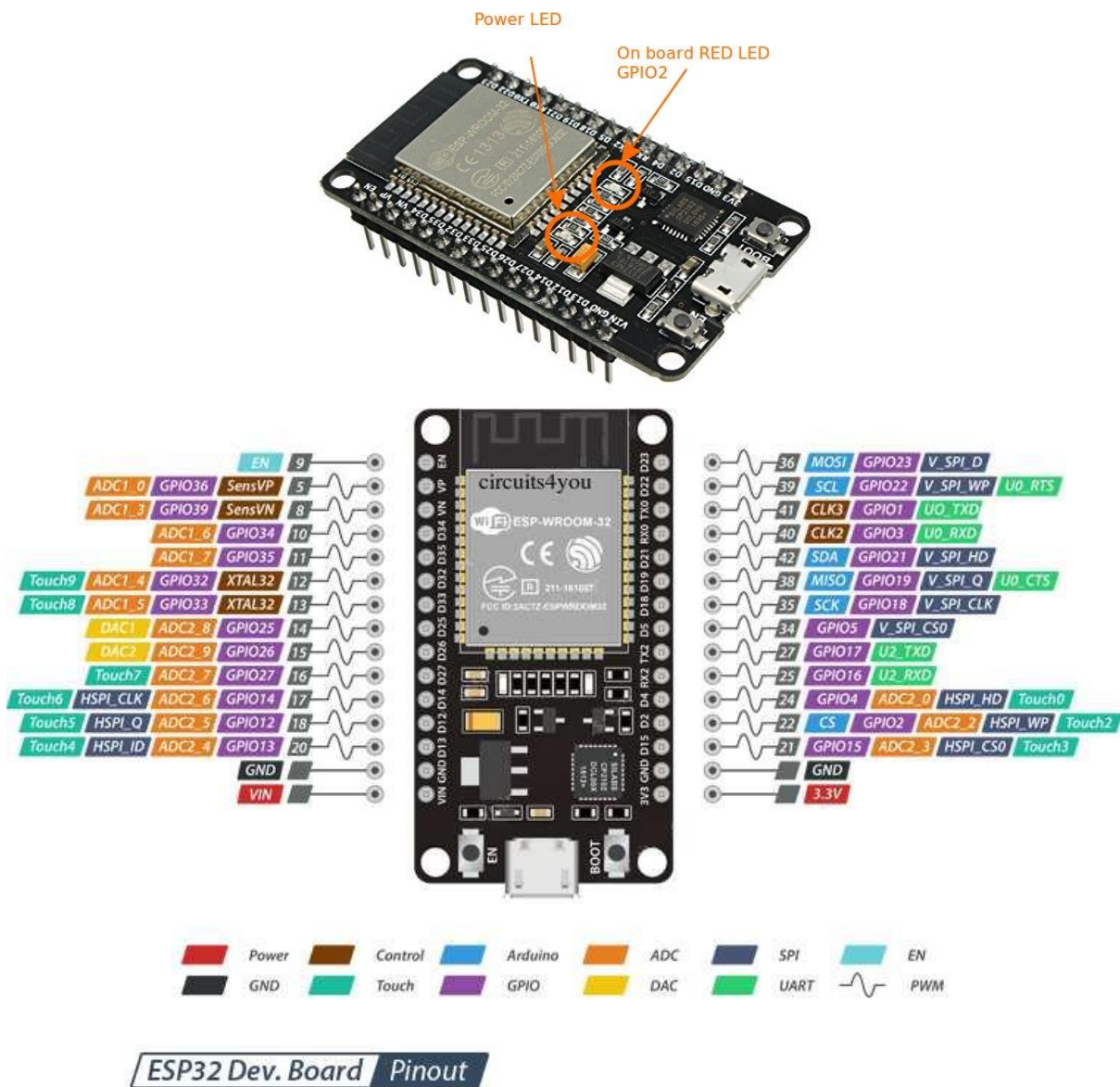


- Note: You’d better close the Arduino IDE and restart it again.
- Config the Board menu and choose the right Port for your device.
 - CPU Frequency : 80MHz,
 - Flash Size : 4M (3M SPIFFS) ,
 - Upload Speed : 115200

LAB EXPERIMENT- 1_a

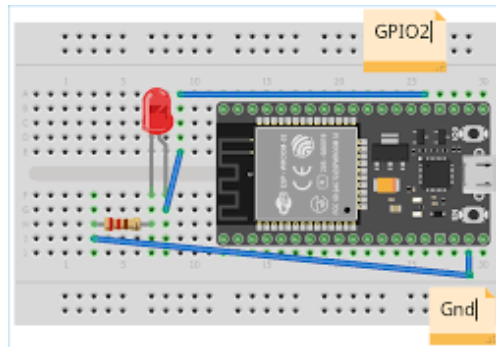
- In this lab session, we will introduce how to control an external device like LED and BUZZER using the pins on ESP32.

- Hardware Required
 - ESP32x 1
 - LED x 1
 - 200 ohm resistor x 1
 - Micro USB cable x 1
 - PC x 1
- Blink the onboard LED connected to pin 2 on the ESP32 .
- Follow the above-mentioned board config to use ESP32 from Tools->Board dropdown
- Use the below given PINOUT diagram of the ESP32



- Pseudocode:
 - Setup block
 - Declare the led pin 2 as output using pinMode(<pin number>, <pin type>)
 - function
 - Loop block
 - Make the LED blink using the functions digitalWrite(<pin number>, <output>) and delay(<time in milliseconds>) with appropriate delay in milliseconds.

- Blink an external LED
 - 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 lit up also with values up to 1K ohm.



- Pseudocode
 - Same as internal LED except for LED pin now is D1(or any other digital pin).

LAB EXPERIMENT- 1_b

- 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.
- Hardware Required
 - ESP32x 1
 - Micro USB cable x 1
 - PC x 1
- Connect your ESP32 to the PC and put below code to your Arduino IDE
- **Hardware Required**
 - ESP32x 1
 - Micro USB cable x 1
 - PC x 1
- 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 a randomly generated number using `random(<min>,<max>)`

LAB EXPERIMENT- 1_c

- In this lab session, we will introduce how to read the inputs using `Serial.read` from the serial monitor and controlling the buzzer's frequency using the `tone` function.
- Hardware Required
 - ESP32 x 1
 - Micro USB cable x 1
 - PC x 1
 - Passive Buzzer MH-FMD
- Difference between active and passive buzzer:
 - An active buzzer will generate a tone using an internal oscillator, so all that is needed is a DC voltage. A passive buzzer requires an AC signal to make a sound.

It is like an electromagnetic speaker, where a changing input signal produces the sound, rather than producing a tone automatically.

- With a "passive" device or speaker, you must send it an AC "sound signal" and you can control the sound. The Arduino needs to generate the "tone". With enough hardware & software you could make it play music. etc. The only downside (besides a bit of extra programming) is that it takes processor-time to generate the sound.