Getting Started with the ESP32 Development Board

This tutorial is a getting started guide for the ESP32 development board. If you're familiar with the ESP8266, the ESP32 is its sucessor. The ESP32 is loaded with lots of new features. The most relevant: it combines WiFi and Bluetooth wireless capabilities and it's dual core.

ESP32 DEVKIT V1

In this Tutorial, we'll be using the ESP32 DEVKIT V1 board as a reference. But the information on this page is also compatible with other ESP32 development boards with the ESP-WROOM-32 chip.

SPECIFICATIONS

When it comes to the ESP32 chip specifications, you'll find that:

- The ESP32 is dual core, this means it has 2 processors.
- It has Wi-Fi and bluetooth built-in.
- It runs 32 bit programs.
- The clock frequency can go up to 240MHz and it has a 512 kB RAM.
- This particular board has 30 or 36 pins, 15 in each row.
- It also has wide variety of peripherals available, like: capacitive touch, ADCs, DACs, UART, SPI, I2C and much more.
- It comes with built-in hall effect sensor and built-in temperature sensor.

PROGRAMMING ENVIRONMENTS

The ESP32 can be programmed in different programming environments. You can use:

- Arduino IDE
- Espressif IDE (IoT Development Framework)
- Micropython
- JavaScript
- LUA
- ...

In this Tutorial, we program the ESP32 with Arduino IDE.

PREPARING THE ESP32 BOARD IN ARDUINO IDE

There's an add-on for the Arduino IDE allows you to program the ESP32 using the Arduino IDE and its programming language. Follow one of the next tutorials to prepare your Arduino IDE:

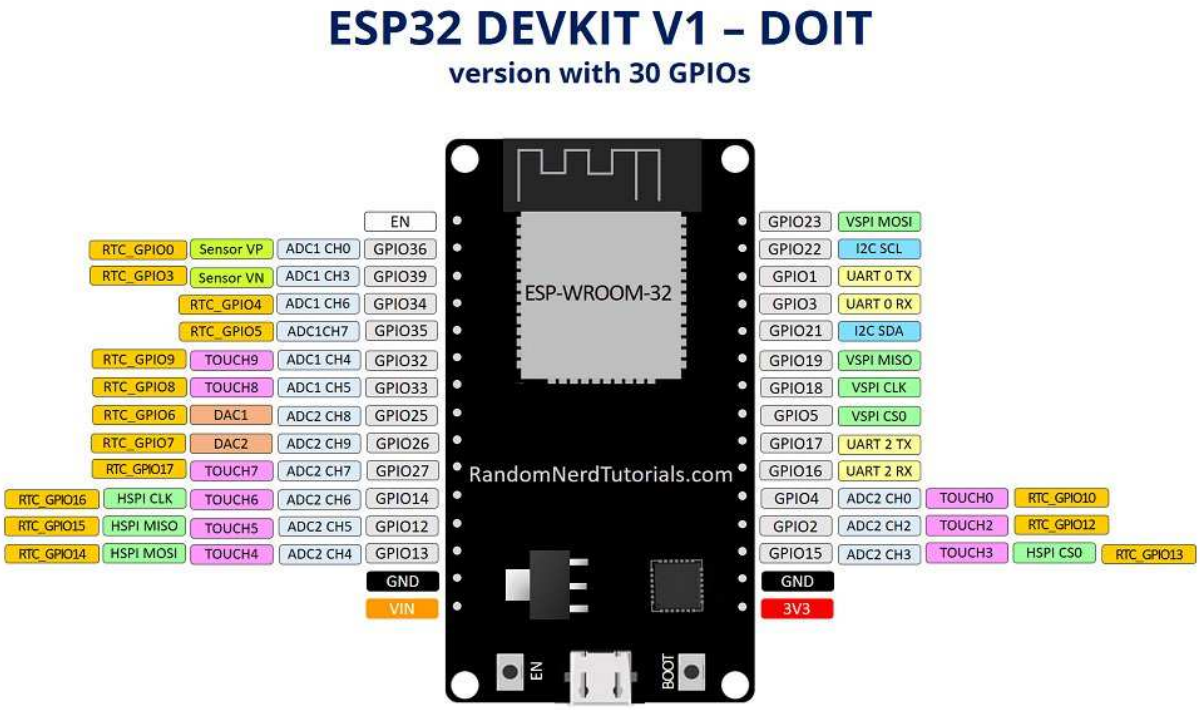
- **Windows** instructions – Installing the ESP32 Board in Arduino IDE https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

- **Mac and Linux** instructions – Installing the ESP32 Board in Arduino IDE https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/

ESP32 PINOUT GUIDE

The ESP32 has more GPIOs with more functionalities compared to the ESP826. With the ESP32 you can decide which pins are UART, I2C, or SPI – you just need to set that on the code. This is possible due to the ESP32 chip's multiplexing feature that allows to assign multiple functions to the same pin. If you don't set them on the code, the pins will be used as default – as shown in the figure below (the pin location can change depending on the manufacturer).

Version with 30 GPIOs



You can read our detailed ESP32 Pinout Reference Guide :

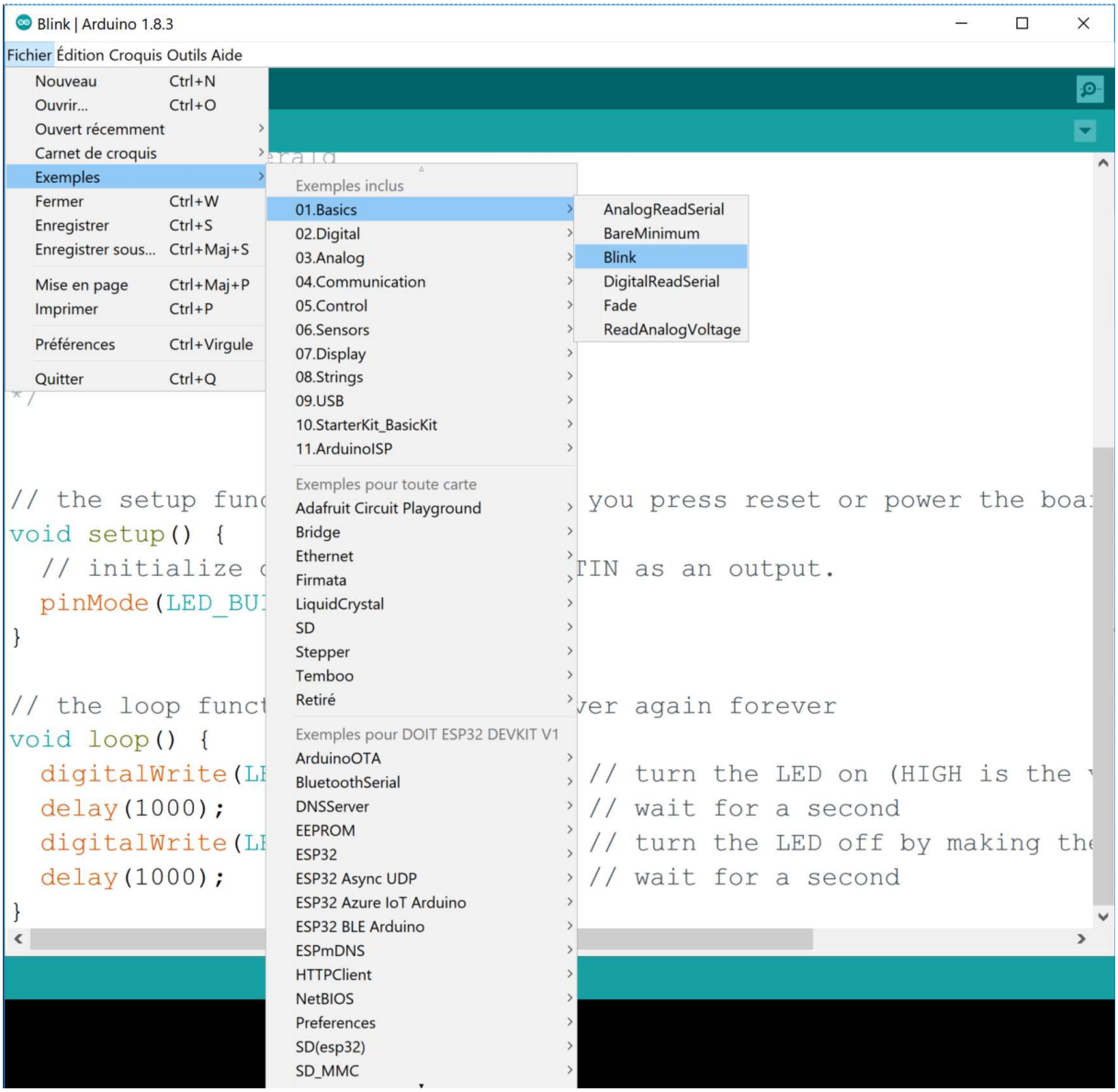
https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

UPLOAD CODE TO THE ESP32 USING ARDUINO IDE

To show you how to upload code to your ESP32 board, we'll build a simple example to blink a LED. There is a blue builtin LED on the board.

For that purpose, we gone use an Example code given by the Arduino IDE.

Go to **Files > Example > Basics** and choose **Blink**

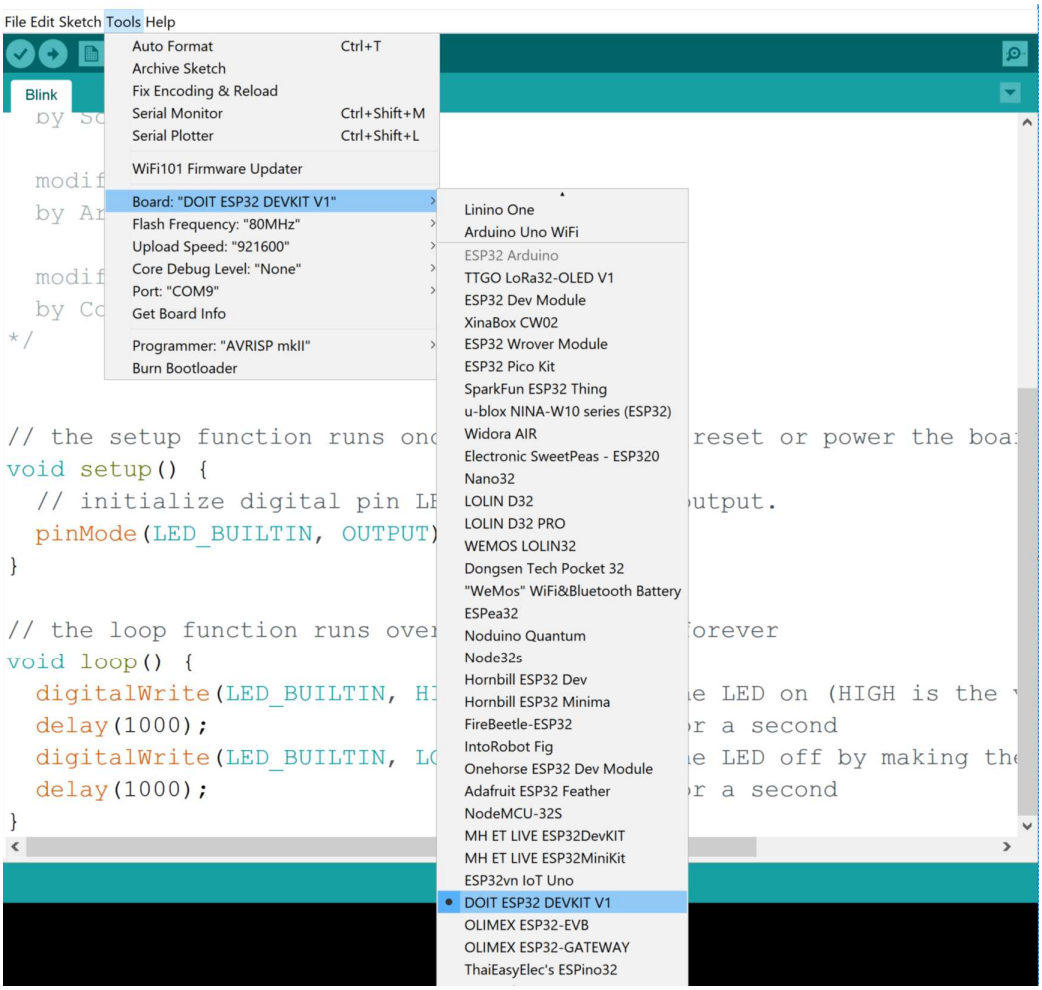


Here is the code of this exemple :

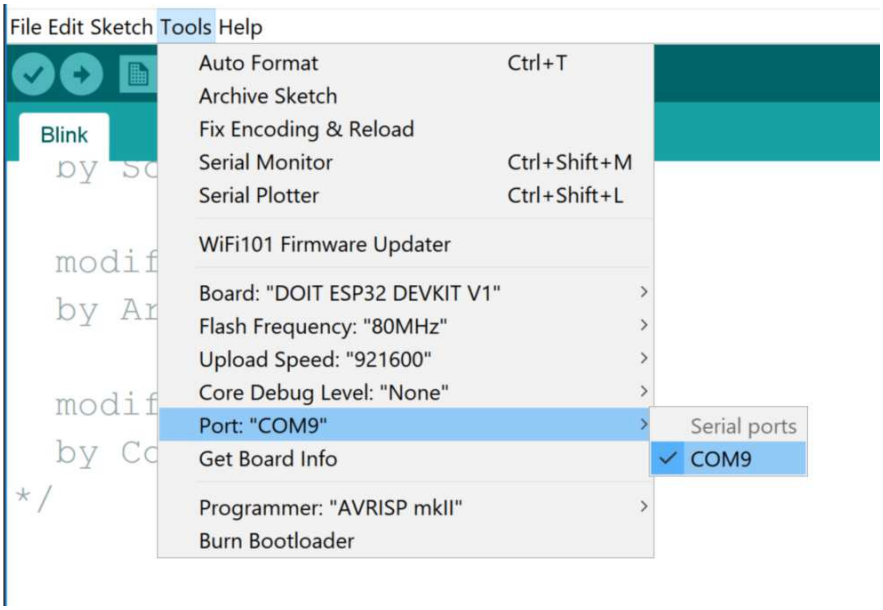
```
/*
  Blink the Builtin LED
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                     // wait for a second
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                     // wait for a second
}
```

Plug your ESP32 development board to your computer and follow these next instructions :

1) Go to **Tools** > **Board**, scroll down to the ESP32 section and select the name of your ESP32 board. In my case, it's the DOIT ESP32 DEVKIT V1 board.



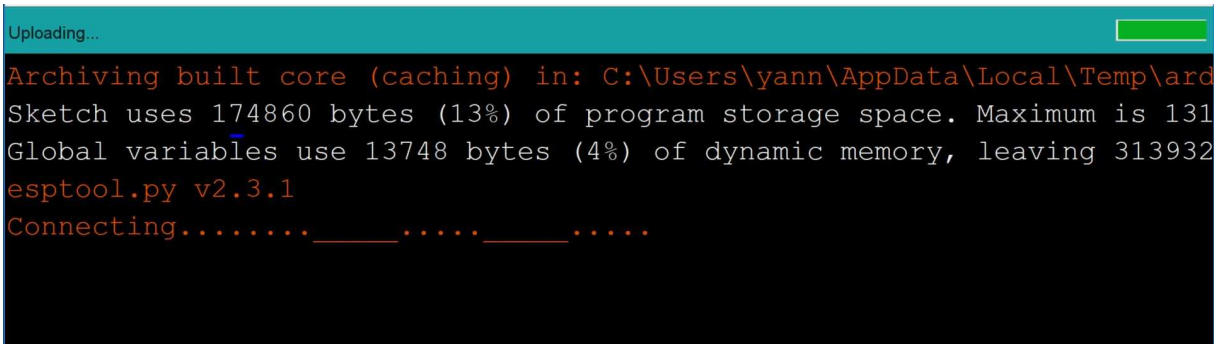
2) Go to **Tools** > **Port** and select a COM port available.

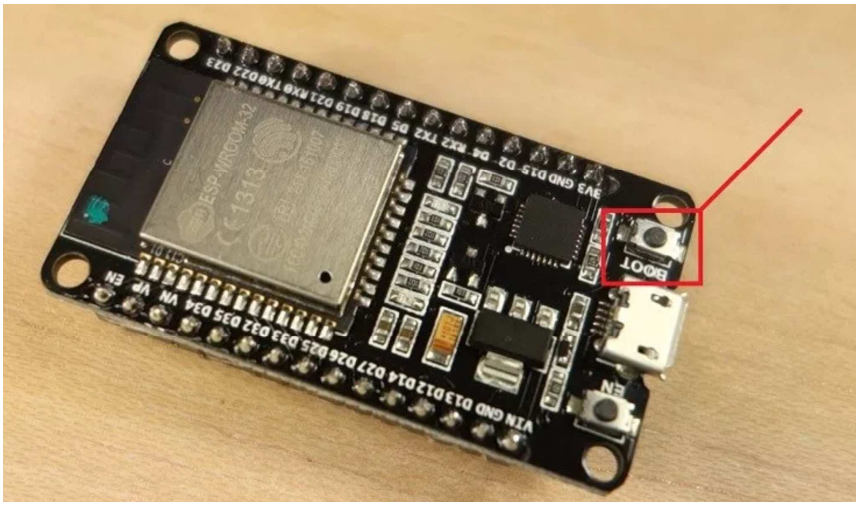


3) Press the upload button

To upload code, you need to follow the next steps

- When you see the "**Connecting….**" Message in your Arduino IDE (like the picture below), Hold-down the "**BOOT**" button in your ESP32 board :





- After you will see some messages printed in your Arduino IDE, release the finger from the "**BOOT**" button.
- After that, you should see the "**Done uploading**" message.
  You should see some messages that inform you that the firmware was write successfully to the Flash, like the picture below :

```
Done uploading.

Writing at 0x00010000... (16 %)
Writing at 0x00014000... (33 %)
Writing at 0x00018000... (50 %)
Writing at 0x0001c000... (66 %)
Writing at 0x00020000... (83 %)
Writing at 0x00024000... (100 %)
Wrote 174992 bytes (88362 compressed) at 0x00010000 in 1.8 seconds
Hash of data verified.
Compressed 3072 bytes to 144...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (144 compressed) at 0x00008000 in 0.0 seconds (ef
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

And normaly the blue LED should blink on your ESP32 DevKit.

CHANGE THE CODE TO BLINK AN EXTERNAL LED

Now we will change the code to blink an external LED on the GPIO 23

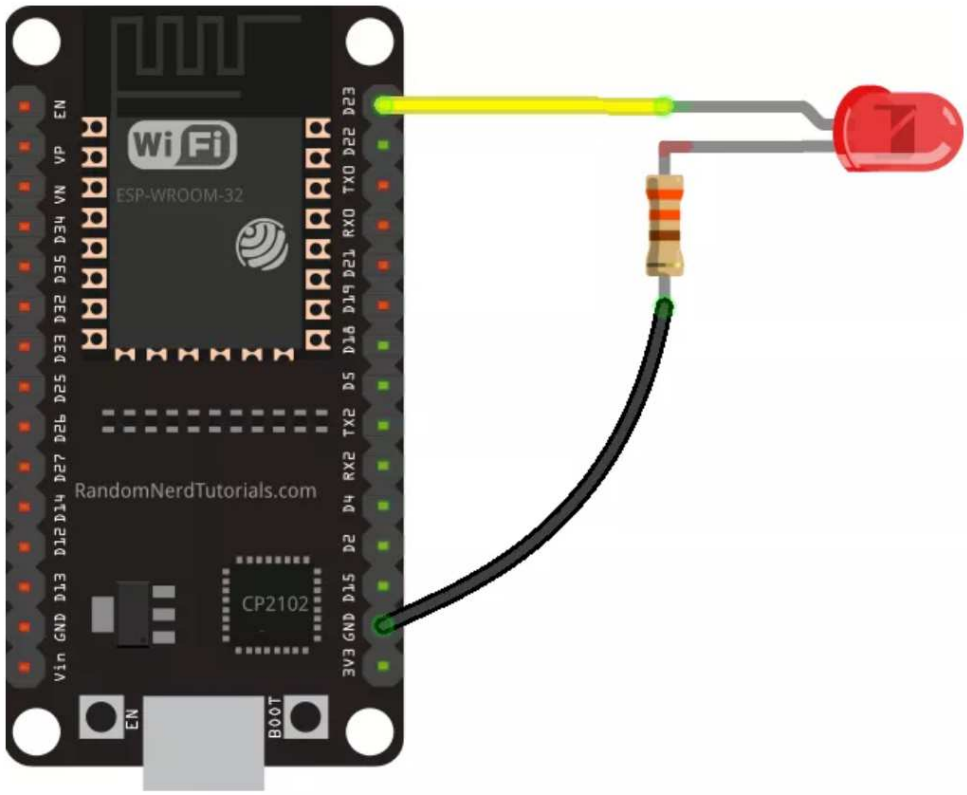
```
/*
  Blink an external LED
*/
// ledPin refers to ESP32 GPIO 23
const int ledPin = 23;

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin ledPin as an output.
  pinMode(ledPin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                  // wait for a second
}
```

In this code, we're controlling an LED connected to GPIO 23.

```
const int ledPin = 23;
```

So, connect an LED to your ESP32 by following the next schematic diagram.

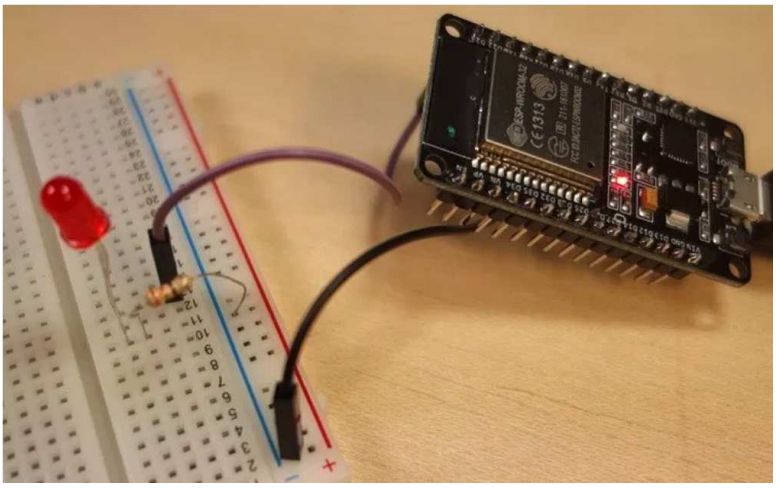
Important: always check the pinout for your specific board before building any circuit.



Here's a list of the parts you need to build this previous circuit:

- ESP32 DOIT DEVKIT V1 Board
- 5mm LED
- 330 Ohm resistor
- Jumper wires
- Breadboard (optional)

DEMONSTRATION



Y. DOUZE, source from Randomneurdtutorials.com

EXERCICES

Now you have to change the code and the wiring to blink two different LED at two different Frequency.

LED 1 : blink at 1 Hz

LED 2 : blink at 3 Hz