Practical Sheet Nº 1 (Blinking LED)

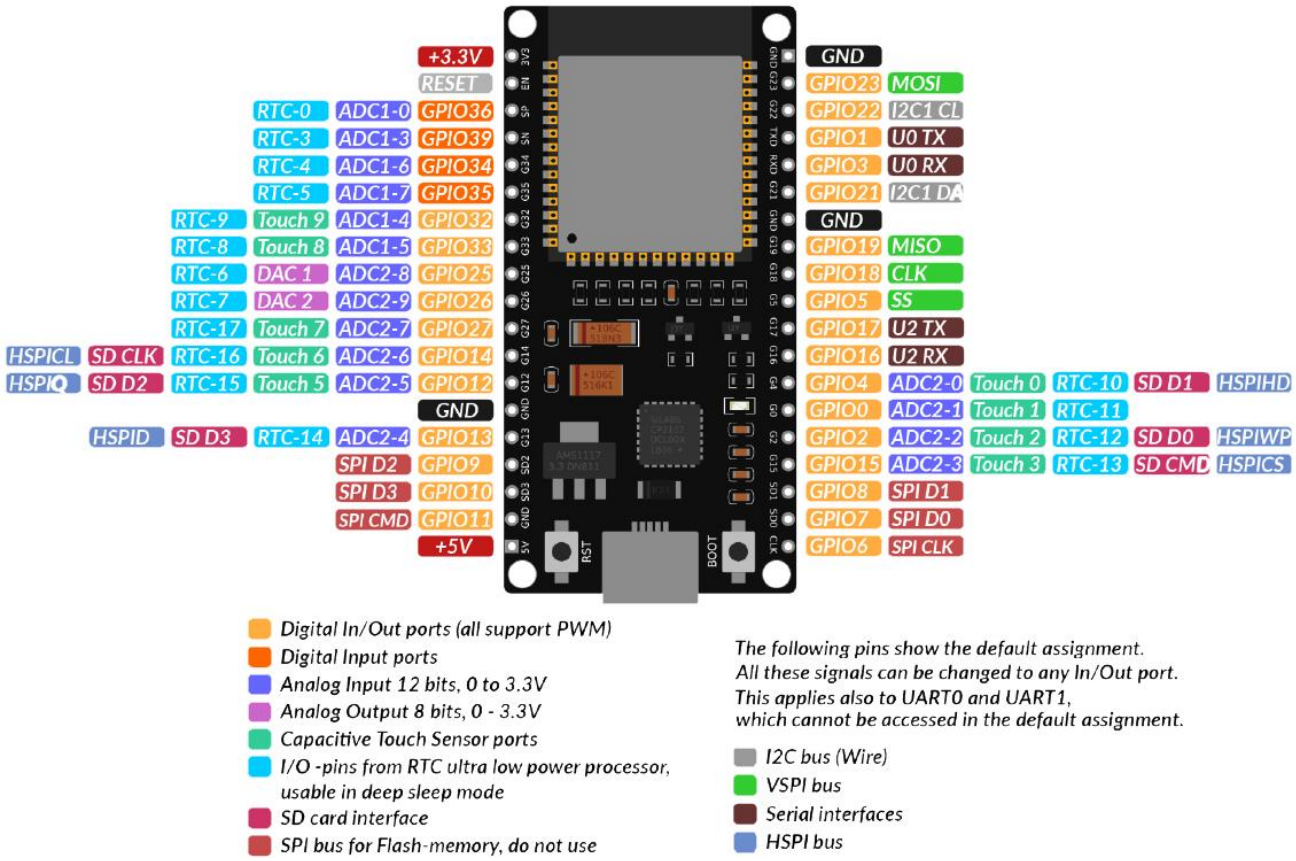
Objectives	Duration
At the end of the training, you will be able to: • Use the Arduino IDE to compile and run a program on the ESP32 kit. • Implement a simple program written in Arduino C and perform debugging via UART.	30min.

Preparation of the practical training

The development kit (DEVKIT) used in the practical classes is the ESP32s. The kit is implemented around the ESP32 and based on the processor from the manufacturer ESPressif:

https://ESPressif.com/en/products/hardware/esp32/overview

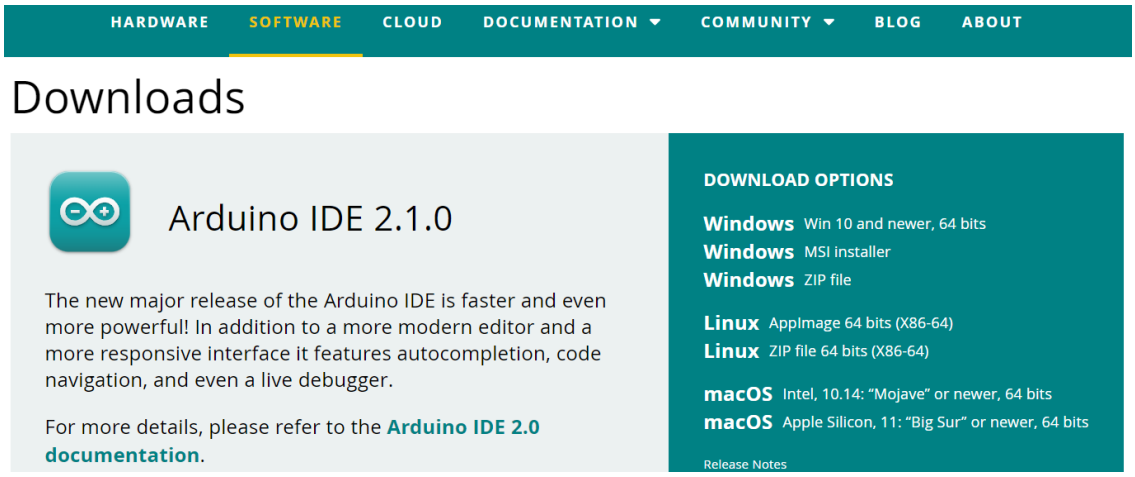
With the ESP32, it is possible to decide which pins are used for UART, I2C or SPI modules and this can be done programmatically by the user. This functionality is possible due to the existence of internal multiplexers that allow a more customized configuration of all GPIOs and modules available. If no prior settings are made by the programmer, the pins will be used in standard mode as illustrated in the following figure.



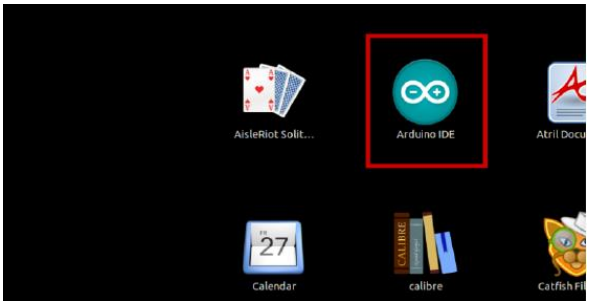
Procedure

1. If not installed, download, and install the Arduino IDE from the following link:

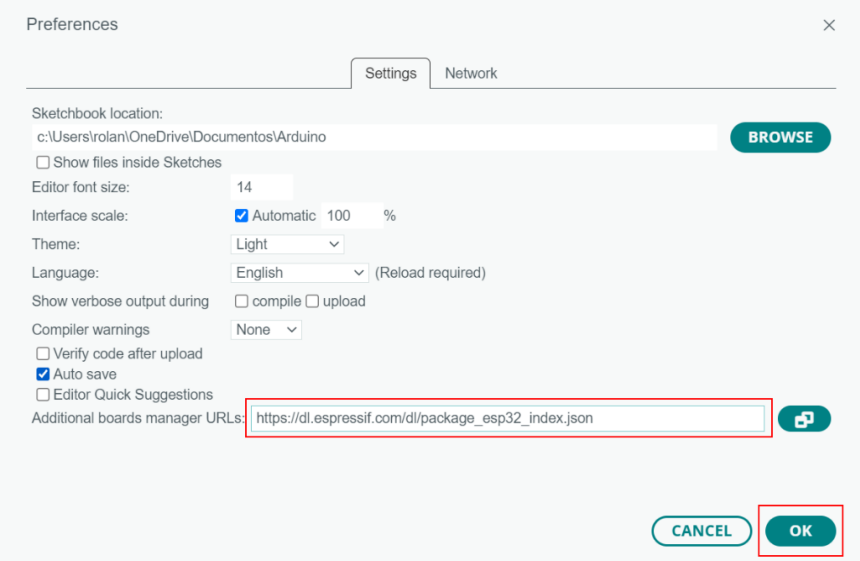
   https://www.arduino.cc/en/software



- For Windows users, double click on the downloaded ***.exe*** file and follow the instructions in the installation window.
- For Linux users, download a file with the extension ***.tar.xz***, which must be extracted. When it is extracted, go to the extracted directory, and open the terminal in that directory. Two ***.sh*** scripts must be executed, the first called ***arduino-linux-setup.sh*** and the second called ***install.sh***.
  To run the first script in the terminal, open the terminal in the extracted directory and run the following command: **sh arduino-linux-setup.sh user_name**
  user_name - is the name of a superuser in the Linux operating system. A password for the superuser must be entered when the command is started. Wait few minutes for the script to complete everything.
  The second script, called **install.sh**, must be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**
  After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.
- All operating systems come with a text editor preinstalled (for example, Windows comes with Notepad, Linux Ubuntu comes with Gedit, Linux Raspbian comes with Leafpad). All these text editors are perfectly fine for the purpose of this training.
- After the installation of these scripts, go to the All Apps, where the Arduino IDE is installed.

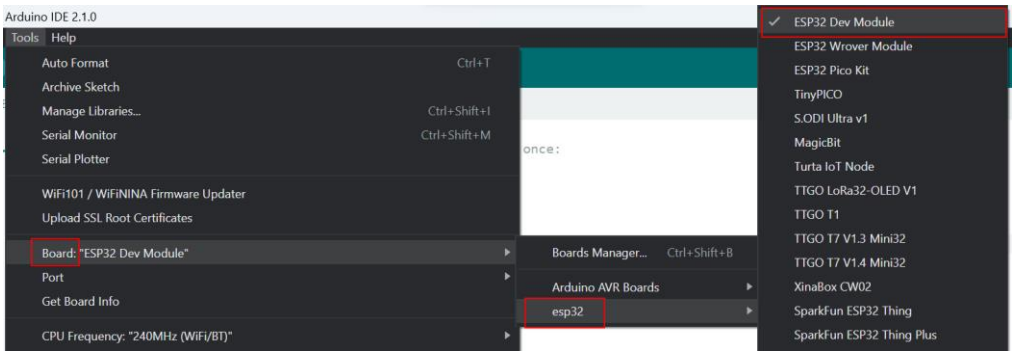


2. Next to install support for ESP32 platform, open Arduino IDE and go to:
   File > Preferences and find ***Additional Bords Manager URL.***
   Copy the following URL:
   https://dl.espressif.com/dl/package_esp32_index.json



3. In the Arduino IDE go to:
   Tools > Board > Boards manager
   When the window opens, type ESP32 in the search box and install the board called ***esp32 made by ESPressif Systems***, as shown in the image below:

4. To select the ESP32 board, go to:
   Tools > Board > ESP32 Dev Module



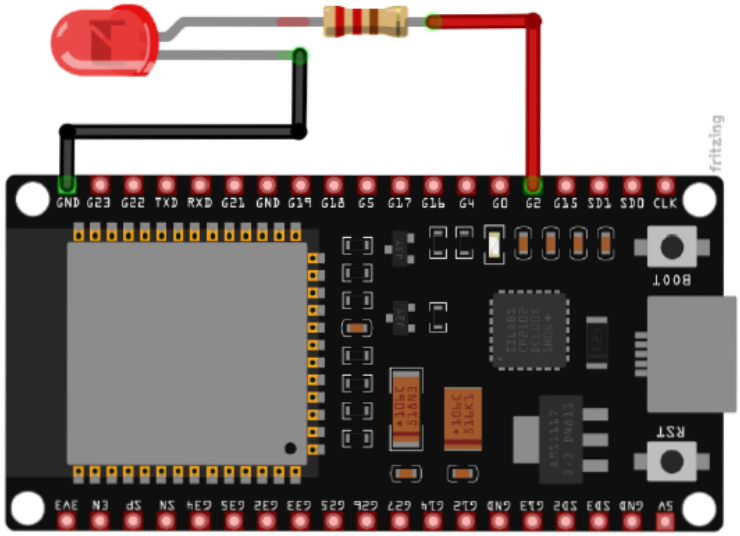
5. Create a new Sketch (File **> *New Sketch***) and paste the following code:

```
int ledPin = 2;

void setup() {
      pinMode(ledPin, OUTPUT);
}
void loop() {
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
}
```

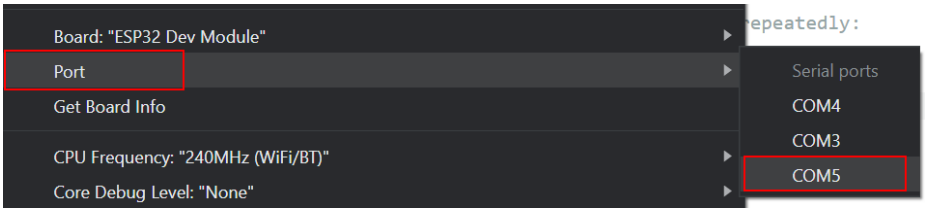
Save the Sketch (File **> *Save as***)

6. Connect the ESP32 Dev Kit with a LED and a 220Ω resistor as shown on the following diagram:



ESP32 Dev Kit C V2 pin	LED pin	Wire color
GPIO2 (pin2)	Anode (+) through resistor	**Red wire**
GND	Cathode (-)	**Black wire**

7. Connect the board to a USB port and to select the port on the Arduino IDE go to:
Tools > Port > {port name}

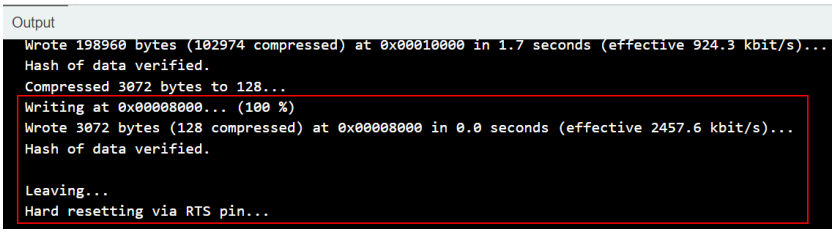


Select the correct port name for your device (COM5 is an example).

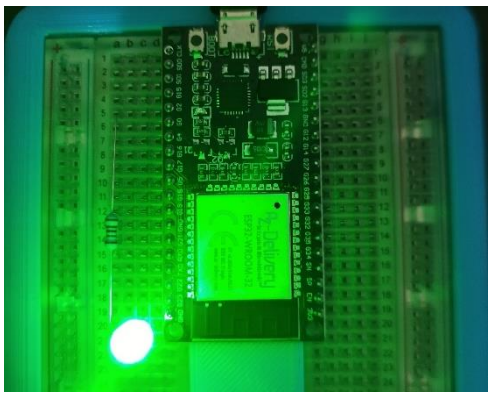
8. Compile the code to check for any mistakes:

9. Upload the code to the ESP32:

The upload is successful if you have the following message:



10. After the reboot of the device, you should see the led blinking every two seconds:



11. With this procedure, you are now able to set up the ESP connection to the Arduino IDE and write, compile, and upload code to the device.

12. Now you can try the following code and see the difference:

```
#define LEDC_CHANNEL_0 0
#define LEDC_TIMER_13_BIT 13
#define LEDC_BASE_FREQ 5000
#define LED_PIN 2

int brightness = 0;
int fadeAmount = 5;
```

```
void ledcAnalogWrite(uint8_t channel, uint32_t value, uint32_t valueMax = 255)
{
        uint32_t duty = (8191 / valueMax) * min(value, valueMax);
        ledcWrite(channel, duty);
}

void setup() {
        ledcSetup(LEDC_CHANNEL_0, LEDC_BASE_FREQ, LEDC_TIMER_13_BIT);
        ledcAttachPin(LED_PIN, LEDC_CHANNEL_0);
}

void loop() {
        ledcAnalogWrite(LEDC_CHANNEL_0, brightness);
        brightness = brightness + fadeAmount;
        if (brightness <= 0 || brightness >= 255) {
                fadeAmount = -fadeAmount;
        }
        delay(30);
}
```