

Green
STEAM
Incubator

**Module on
Microcontrollers:
30 hours lessons**

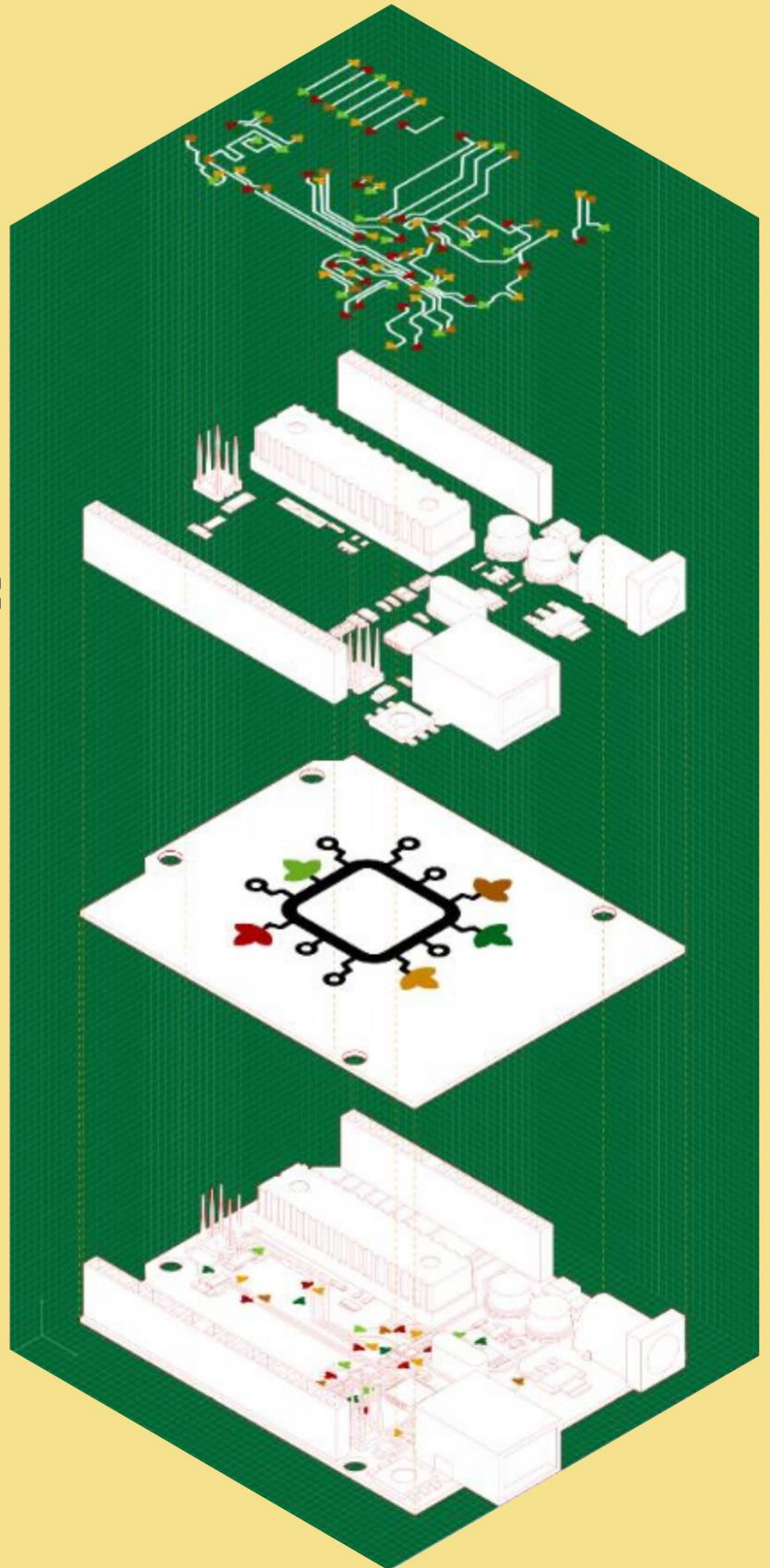


Table of Contents

INTRODUCTION	2
INTRODUCTION TO MICROCONTROLLERS.....	3
LESSON – WHAT IS A MICROCONTROLLER	3
LESSON – WHAT IS ARDUINO (AND TYPES OF ARDUINO) AND HOW TO USE IT	9
LESSON – CONCEPTS: INPUT, OUTPUT, ANALOG, DIGITAL	22
LESSON – ARDUINO SOFTWARE	26
MICROCONTROLLER AND ARDUINO FUNDAMENTALS.....	32
LESSON – HOW TO PROGRAM IN ARDUINO, AND UPLOAD PROGRAMS TO THE BOARD ...	32
LESSON – BREADBOARD ASSEMBLIES	42
LESSON- DIGITAL OUTPUT: BLINKING LED WITH ARDUINO	48
LESSON – ANALOG OUTPUT: FADING LED WITH ARDUINO.....	54
LESSON – DIGITAL INPUT: READING BUTTONS	58
LESSON – ANALOG INPUT: POTENTIOMETER	63
WIFI FUNDAMENTALS AND IOT	68
LESSON – WIRING THE ESP8266.....	68
LESSON – WiFi OVERVIEW: ESP8266 CAPABILITIES	72
LESSON – WiFi CONNECTION WITH ESP8266	76
LESSON – CONNECTING ESP8266 WITH TELEGRAM.....	88
LESSON – CONNECTING ESP8266 WITH FIREBASE	100

INTRODUCTION

This module was written within Intellectual Output 3 of the Green STEAM Incubator project and contains chapters regarding microcontrollers and the various applications. The material was created in order to provide participants with STEAM-oriented knowledge, capacities and skills.

Every partner of the consortium worked on the different chapters that make up this module, from the research phase to the writing one. The objective was to devise learning material that focuses on microcontrollers and the various ways they can be applied, with consideration for the agricultural sector. In the end, this module is about how agriculture can benefit from technology and some of its components.

The module has three main chapters, and in each one, there is a subchapter that contains themes that serve as a lesson, and in total, there are 30 hours of lessons formulated in a logical order.



INTRODUCTION TO MICROCONTROLLERS

LESSON – WHAT IS A MICROCONTROLLER

- **Study section:** Introduction to microcontrollers
- **Lesson duration:** 1:00h
- **Educational objectives:**
 - Understand what a microcontroller is
 - Get to know different types of microcontrollers (e.g., Arduino, Scratch; Micro bits; Logic Gates, Raspberry PI)
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Understand what a microcontroller is • Get to know different types of microcontrollers (e.g., Arduino, Scratch; Micro bits; Raspberry PI) 	<ul style="list-style-type: none"> • Recognise the main elements and characteristics of a microcontroller 	<ul style="list-style-type: none"> • Become able to identify microcontrollers available on the market.

- **Main Keyword(s):**
 - Internet of Things (IoT)
 - Microcontroller
 - Arduino
 - Scratch
 - Micro bits
 - Raspberry PI
- **Required material and resources:**
 - Computer
 - Internet access

Brief definition of **Internet of Things**

Internet of Things (IoT) describes the network of physical objects—“things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the Internet.

Brief definition of **Microcontroller**

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Microcontrollers

As an introduction to this session, the facilitator trickers the participants' interest with the below introductory text:

Have you ever looked at some gadget and wondered how it really worked? Maybe it was a remote-control boat, the system that controls an elevator, a vending machine, or an electronic toy? Or have you wanted to create your own robot or electronic signals for a model railroad, or perhaps you would like to capture and analyze weather data over time? Where and how do you start?

Microcontrollers can help you find some of the answers to the mysteries of electronics in a hands-on way.

Then, the facilitator starts the lesson by explaining to the participants what microcontrollers are. The facilitator begins as follows:

A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor.

Microcontrollers are used in a wide array of systems and devices.

An example of the application of microcontrollers in vertical farming automation systems is provided.

The example: a soil moisture sensor measures the soil resistance at intervals, sends the information as a digital signal to the sensor; the microcontroller then decides whether or not the soil resistance is high or not (depending on the pre-set thresholds and soil type). If the soil moisture level is lower than the pre-set threshold, the microcontroller actuates the



submersible water pump and solenoid valves which provide water to the soil via drip irrigation method (TE & COA, 2018).

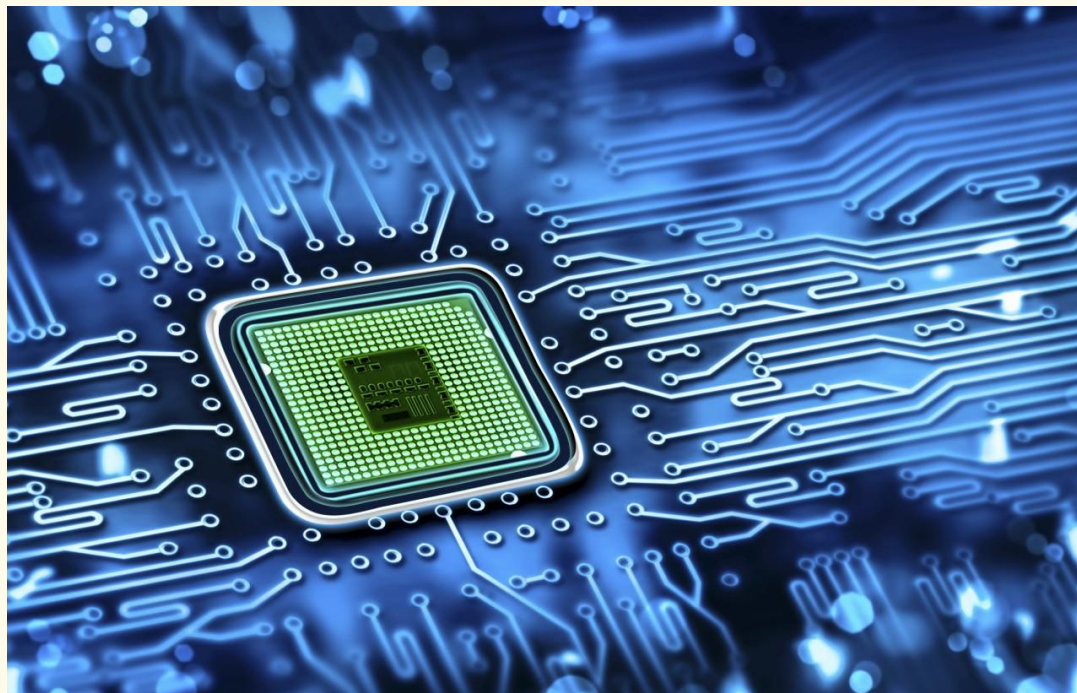


Figure 1 A microcontroller, source: <https://predictabledesigns.com/microcontroller-or-microprocessor-which-is-right-for-your-new-product/>

Then, the facilitator will briefly present the core elements of a microcontroller, which are listed below:

- The processor (CPU): A processor processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
- Memory: A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:
 - o Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
 - o Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.
- I/O peripherals: The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.

Examples of microcontrollers

There are several microcontrollers available in the market. Brief information about some of those are given below. In the lessons to follow, the facilitator will focus on Arduino.

Arduino

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. Arduino is a great platform for prototyping projects and inventions. In the lessons to follow we will focus more on Arduino use and exploitation.

Also, for more information: <https://www.arduino.cc/en/software>

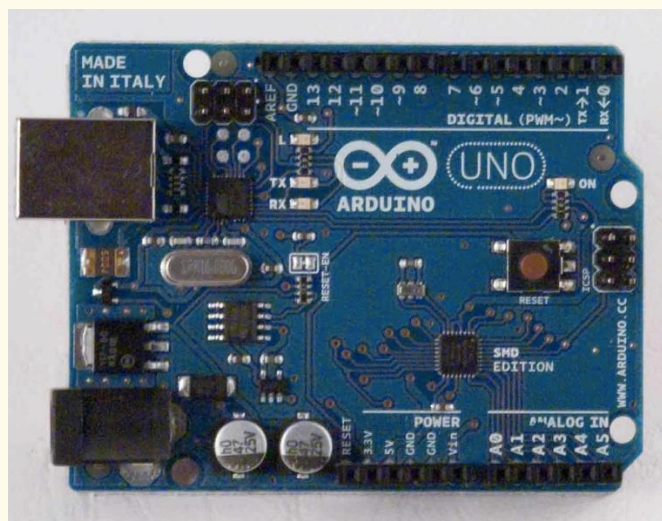


Figure 2 Arduino, source: <https://www.arduino.cc/en/Main/ArduinoBoardUnoSMD>

Scratch

Scratch is a free programming language and online community where users can create their own interactive stories, games, and animations, and share their creations with others in an online community. Scratch helps young people learn to think creatively, reason systematically, and work collaboratively — essential skills for life in the 21st century.

For more information: <https://scratch.mit.edu/>

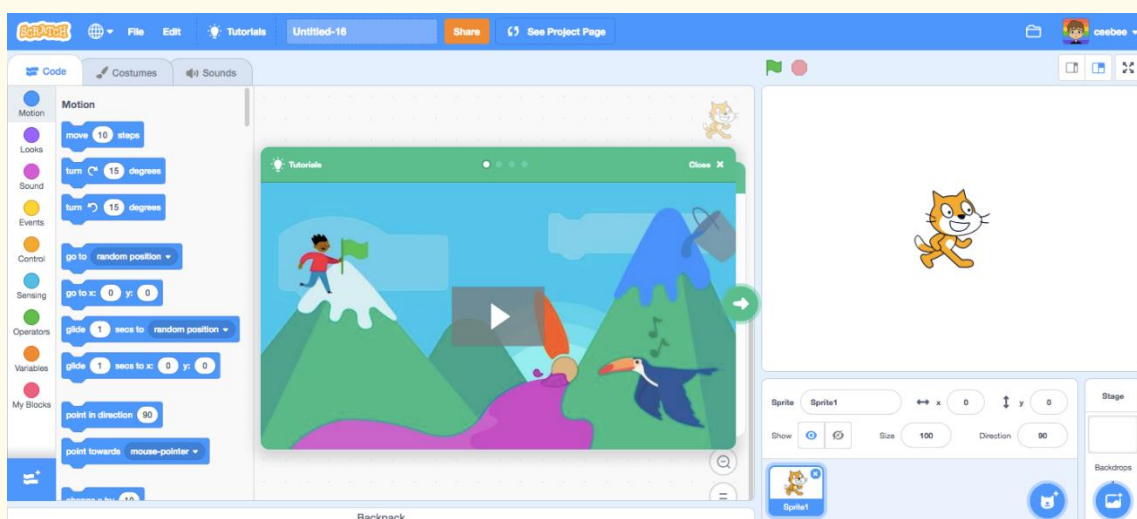


Figure 3 Scratch, source: <https://scratch.mit.edu/discuss/topic/326861/>

Micro bits

The BBC micro:bit is a pocket-sized computer that can help young people learn basic programming knowledge. It is an open development board that allows the user to run code on it and have access to all of the hardware. For more information: <https://microbit.org/>

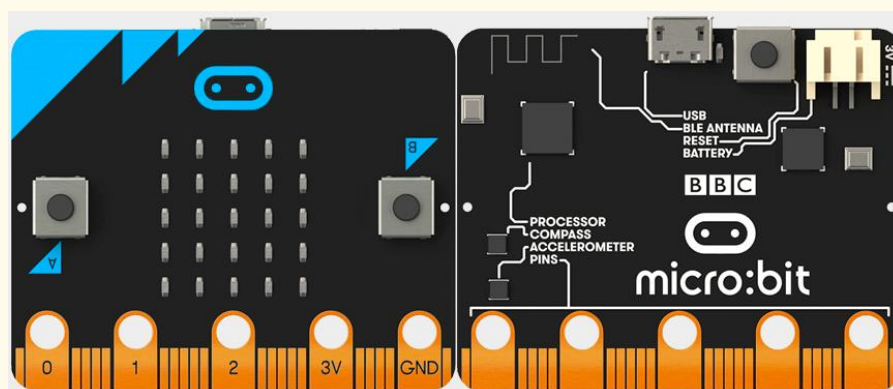


Figure 4 The BBC micro:bit, source: <https://www.drobot.com/blog-734.html>

Raspberry Pi

The Raspberry Pi is a tiny and affordable computer that the user can use to learn programming through fun, practical projects. Several generations of Raspberry Pis have been released. The first generation (Raspberry Pi Model B) was released in February 2012, while the latest generation (Raspberry Pi 400) was released in November 2020. It features a custom board that is derived from the existing Raspberry Pi 4 (previous release), specifically remodeled with a keyboard attached.

For more information: <https://www.raspberrypi.org/>

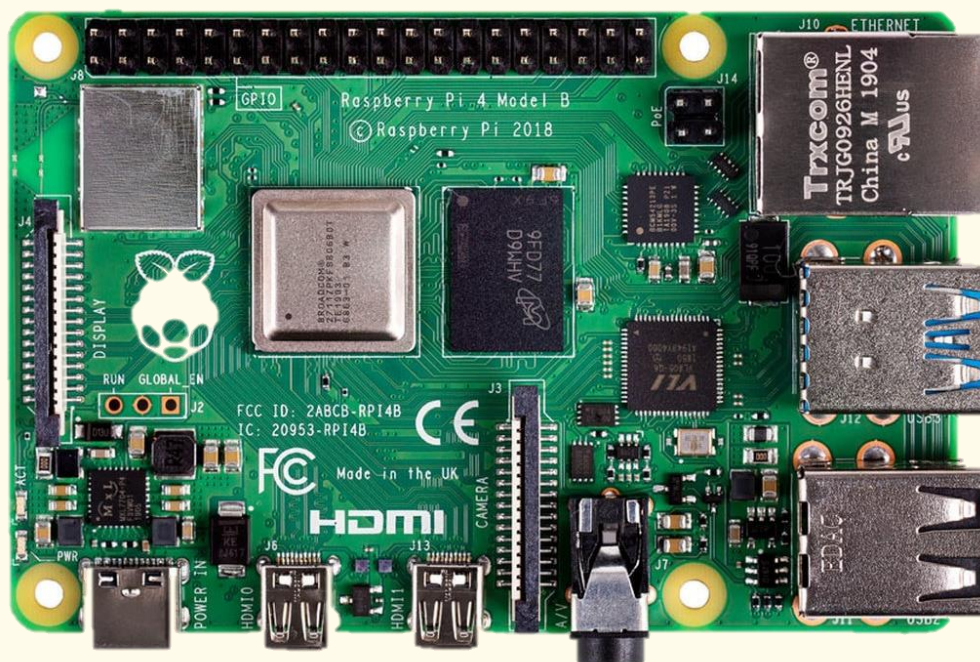


Figure 5 Raspberry Pi, source: <https://www.raspberrypi.org/>

- **Bibliographic references:**

Arduino (2020). Arduino website. <https://www.arduino.cc/en/software>

Arduino (2020). Arduino Uno SMD. Retrieved January 2021, from: <https://www.arduino.cc/en/Main/ArduinoBoardUnoSMD>

DF Robot (Sep 7, 2017). 5 Easy Steps for you to Quick Start with BBC microbit. Retrieved January 2021, from: <https://www.dfrobot.com/blog-734.html>

Microbit (2020). Micro: bit website: <https://microbit.org/>

Raspberry Pi (2020). Raspberry Pi website: <https://www.raspberrypi.org/>

Scratch (2020). Scratch website: <https://scratch.mit.edu/>

Scratch (Jan 2, 2019). Discuss Scratch. Retrieved January 2021, from: <https://scratch.mit.edu/discuss/topic/326861/>

TE, S., & COA, A. (2018). Microcontroller-based Vertical Farming Automation System. *International Journal of Electrical & Computer Engineering* (2088-8708), 8(4).

LESSON – WHAT IS ARDUINO (AND TYPES OF ARDUINO) AND HOW TO USE IT

- **Study section:** Introduction to microcontrollers
- **Lesson duration:** 1:00h
- **Educational objectives:**
 - Understand what an Arduino is
 - Learn about types of Arduino
 - Acquire basic knowledge on how to use Arduino IDE
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Understand what Arduino is and how it can be used • Acquire Basic knowledge on how to use Arduino UNO board and Arduino IDE 	<ul style="list-style-type: none"> • Recognise the main parts of Arduino UNO board and explain their functionality • Navigate to Arduino IDE and understand its main parts 	<ul style="list-style-type: none"> • Use Arduino software autonomously

- **Main Keyword(s):**
 - Arduino
 - Types of Arduino boards
 - Arduino UNO
 - Arduino IDE
- **Required material and resources:**
 - Computer
 - Internet Connection
 - Arduino UNO
 - Software Arduino IDE
 - Arduino USB cable

Brief definition of **Arduino**

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE.

Brief definition of **Arduino IDE**

Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

What is Arduino

The facilitator then introduces to the participants a particular type of microcontroller, Arduino, and the different types of Arduino. The facilitator begins as follows:

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programmed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board. Arduino is a great platform for prototyping projects and inventions.

The Arduino board (shown in Figure 2) can help you find some of the answers to the mysteries of electronics in a hands-on way. The Arduino system was created by Massimo Banzi and David Cuartielles in 2005 and offers an inexpensive way to build interactive projects, such as remote-controlled robots, GPS tracking systems, and electronic games.

The main Arduino components are listed below:

- software - used to compose your programs and communicate with the hardware, called an integrated development environment (Arduino IDE). You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE.
- hardware - refers to the boards themselves (e.g. Arduino Uno). Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.
- programming language - the Arduino programming language uses a simplified version of C++.

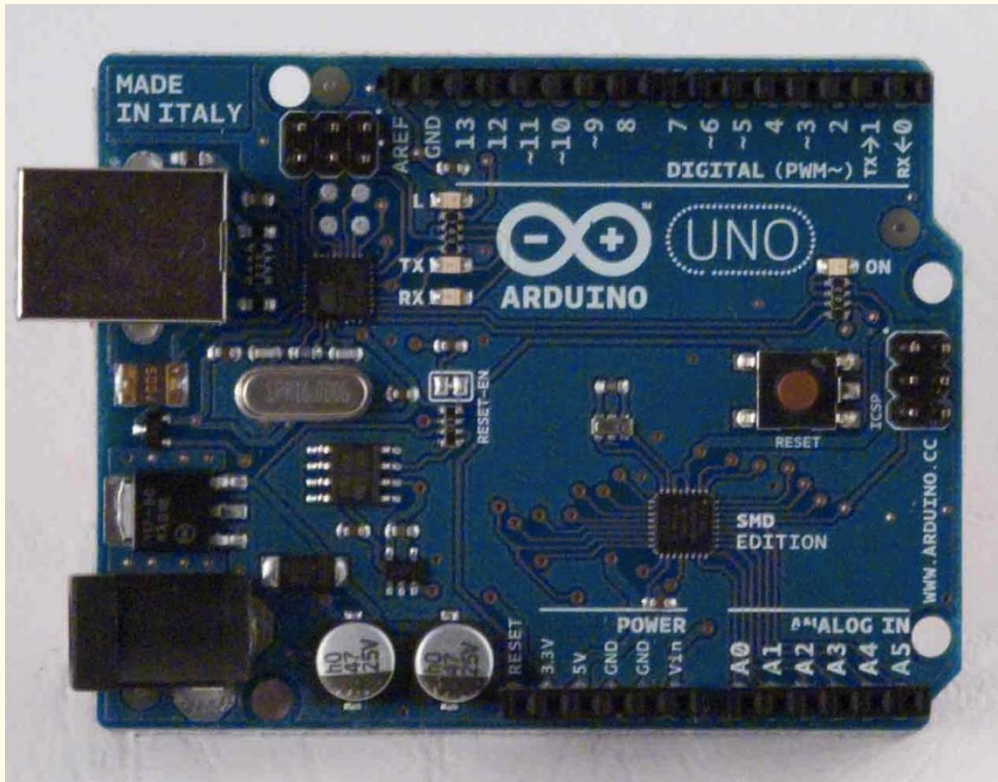


Figure 1 Arduino, source: <https://www.arduino.cc/en/Main/ArduinoBoardUnoSMD>

Types of Arduino

The facilitator will introduce the participants into different types of Arduino boards. If available, the facilitator can provide few different types of Arduino boards to the participants for them to observe them. The following text is given:

Various kinds of Arduino boards are available depending on different microcontrollers used. The differences are based on the number of inputs and outputs (the number of sensors, LEDs, and buttons you can use on a single board), speed, operating voltage, form factor etc. Some boards are designed to be embedded and have no programming interface (hardware). Some can run directly from a 3.7V battery, others need at least 5V. Still, all Arduino boards have one thing in common: they are programmed through the Arduino IDE.

Below are a few examples of the different types of Arduino boards. A complete list of different Arduino boards is provided in the Appendix.

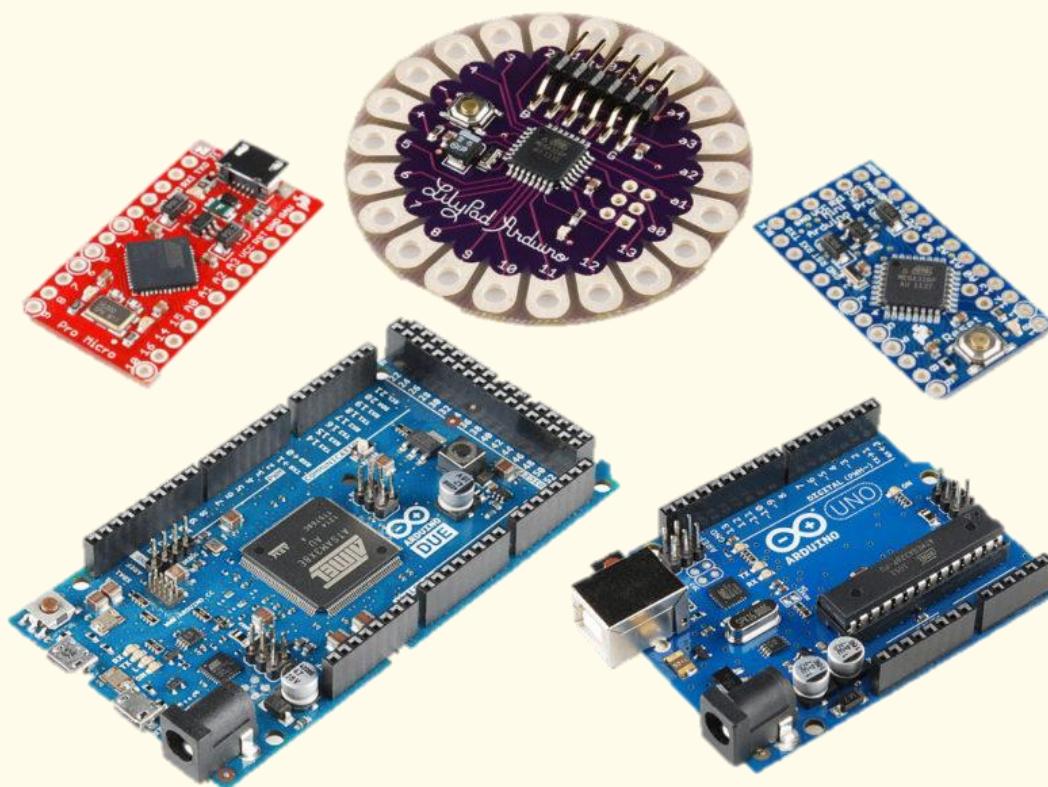


Figure 2 Different types of Arduino boards, Image credit – Sparkfun.com

One of the most popular Arduino boards is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market.

The facilitator provides an Arduino Uno to the participants, and explains to them the various components of the board and their main functions.

Here are the components that make up an Arduino board and what each of their functions are (see figure 3).

1. Reset Button – This will restart any code that is loaded to the Arduino board
2. AREF – Stands for “Analog Reference” and is used to set an external reference voltage
3. Ground Pin – There are a few ground pins on the Arduino and they all work the same
4. Digital Input/Output – Pins 0-13 can be used for digital input or output
5. PWM – The pins marked with the (~) symbol can simulate analog output
6. USB Connection – Used for powering up your Arduino and uploading sketches
7. TX/RX – Transmit and receive data indication LEDs
8. ATmega Microcontroller – This is the brains and is where the programs are stored
9. Power LED Indicator – This LED lights up anytime the board is plugged in a power source
10. Voltage Regulator – This controls the amount of voltage going into the Arduino board
11. DC Power Barrel Jack – This is used for powering your Arduino with a power supply
12. 3.3V Pin – This pin supplies 3.3 volts of power to your projects
13. 5V Pin – This pin supplies 5 volts of power to your projects
14. Ground Pins – There are a few ground pins on the Arduino and they all work the same

15. Analog Pins – These pins can read the signal from an analog sensor and convert it to digital

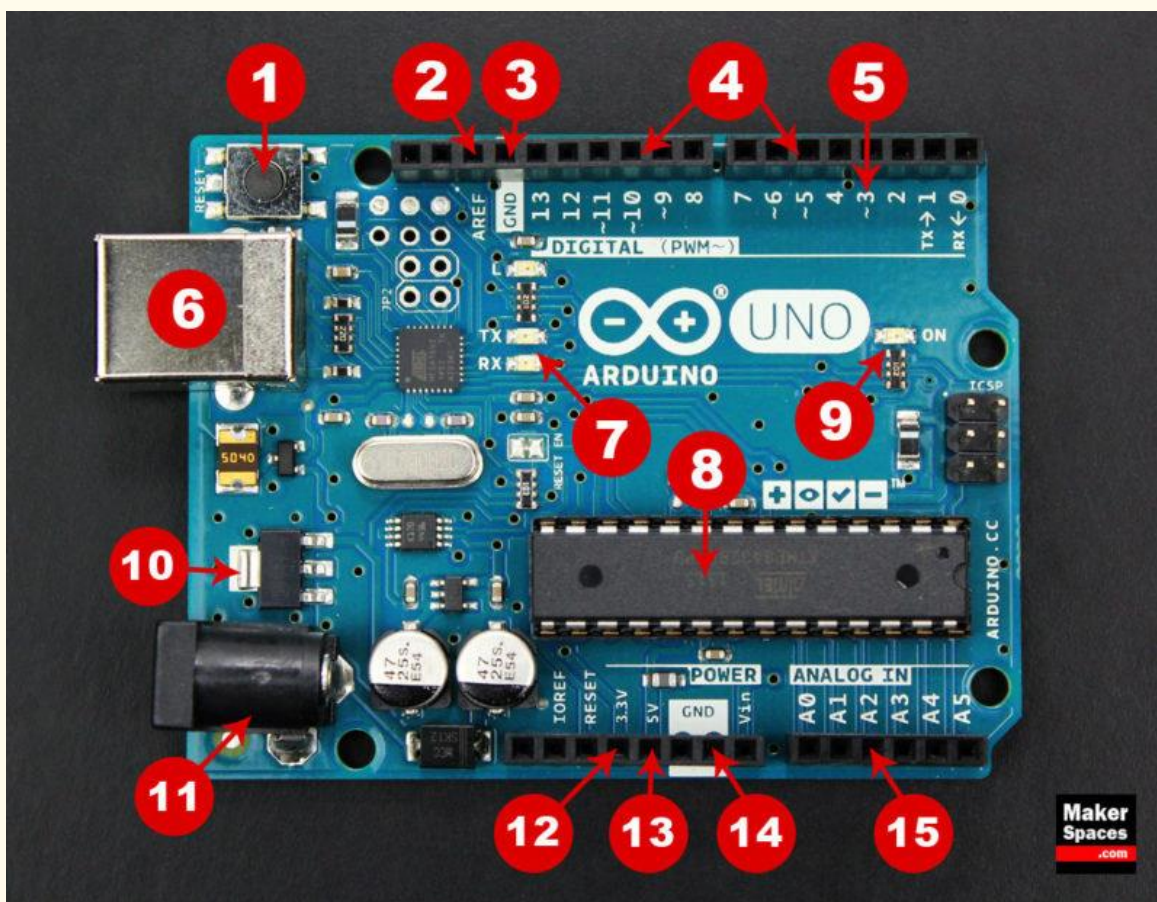


Figure 3 Arduino UNO, source: <https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. The board can be connected directly to a computer via a USB cable. In case of mobile projects, one can consider using a 9V battery pack. The last method would be to use a 9V AC power supply.



Figure 4 Arduino UNO power source, source: <https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

The facilitator also introduces another important item when working with Arduino, the breadboard. S/he explains to the participants that this device allows one to prototype an Arduino project without having to permanently solder the circuit together. The following text is given:

Using a breadboard allows one to create temporary prototypes and experiment with different circuit designs. Inside the holes (tie points) of the plastic housing, are metal clips which are connected to each other by strips of conductive material. The breadboard is not powered on its own and needs power brought to it from the Arduino board using jumper wires. These wires are also used to form the circuit by connecting resistors, switches and other components together.

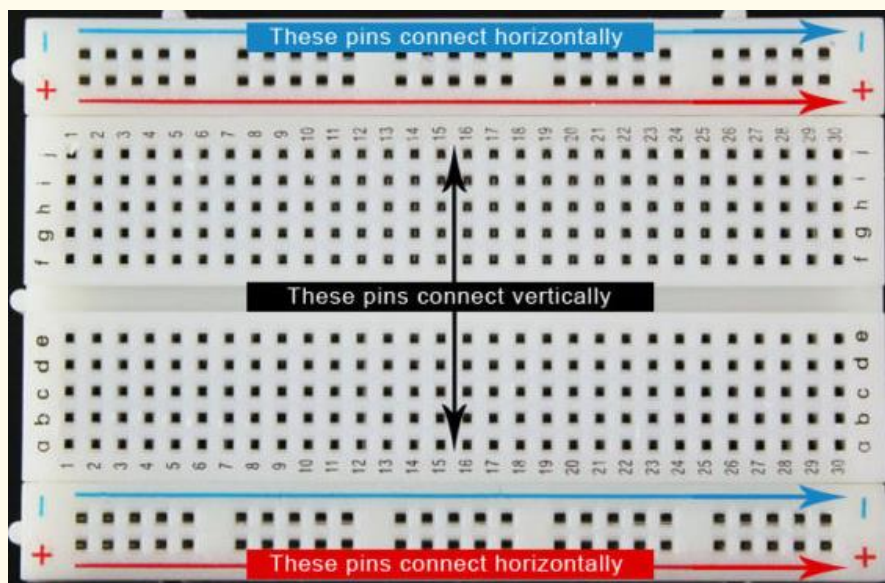


Figure 5 Arduino Breadboard, source: <https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

The facilitator provides a visual of what a completed Arduino circuit looks like when connected to a breadboard.

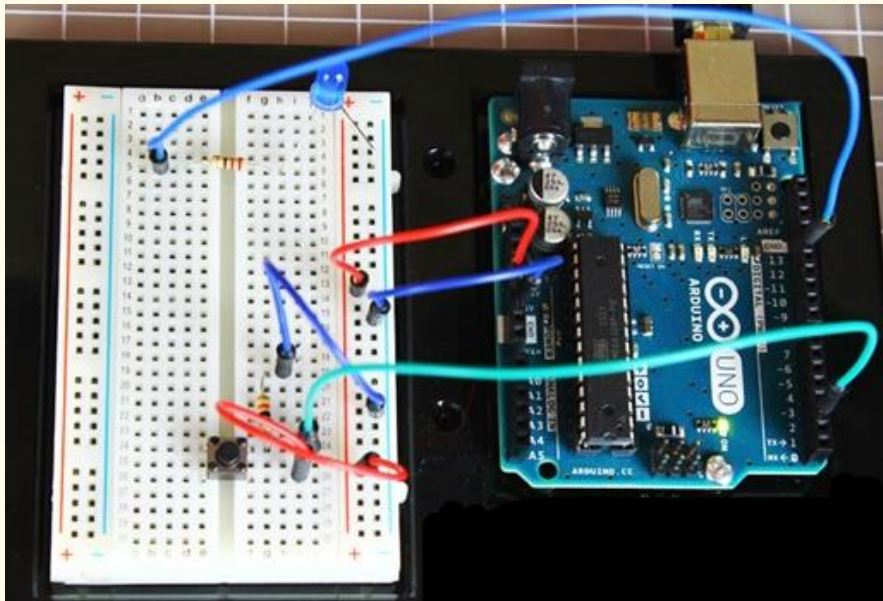


Figure 6 A complete Arduino circuit, source: <https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

How to use Arduino IDE

The facilitator introduces the Arduino IDE to the participants. As a first step, the facilitator guides the participants on how to download the software. The following instructions are provided:

Instructions for downloading Arduino IDE

1. Go to <https://www.arduino.cc/>
2. Click on Software > Downloads > Arduino IDE 1.8.13 (see figure 7)
3. Click on the download option that suits to your operating system (see figure 8). If you have doubts, click on 'Getting Started' to read more information about installation of the software for your computer.

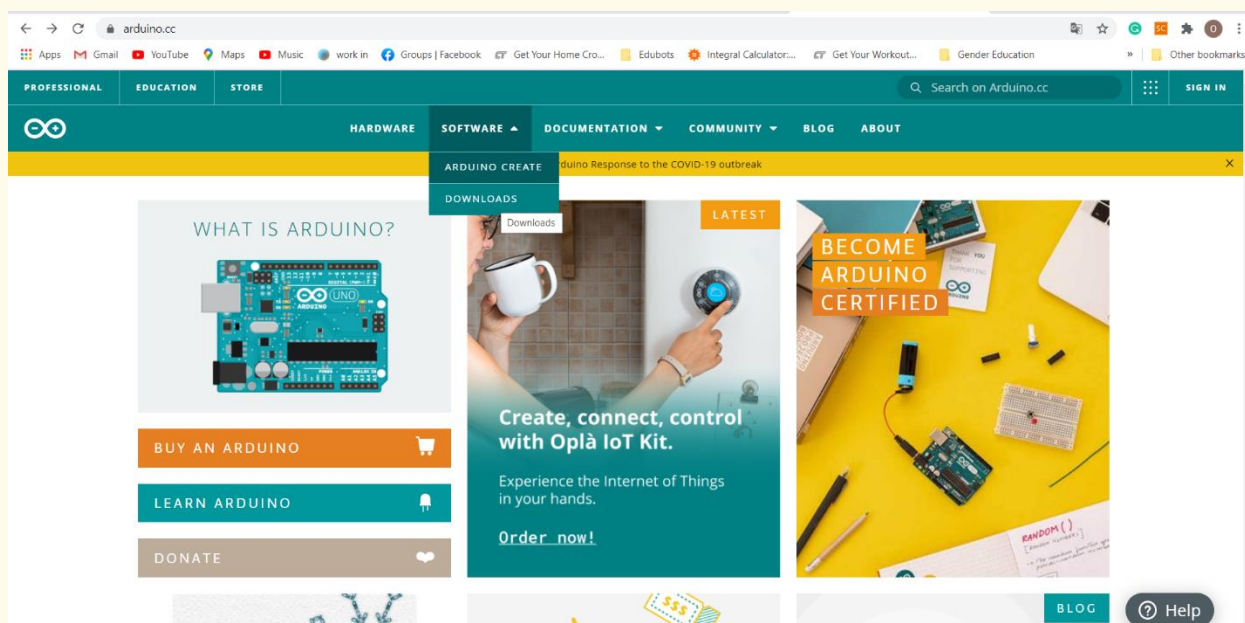


Figure 7 How to download Arduino IDE

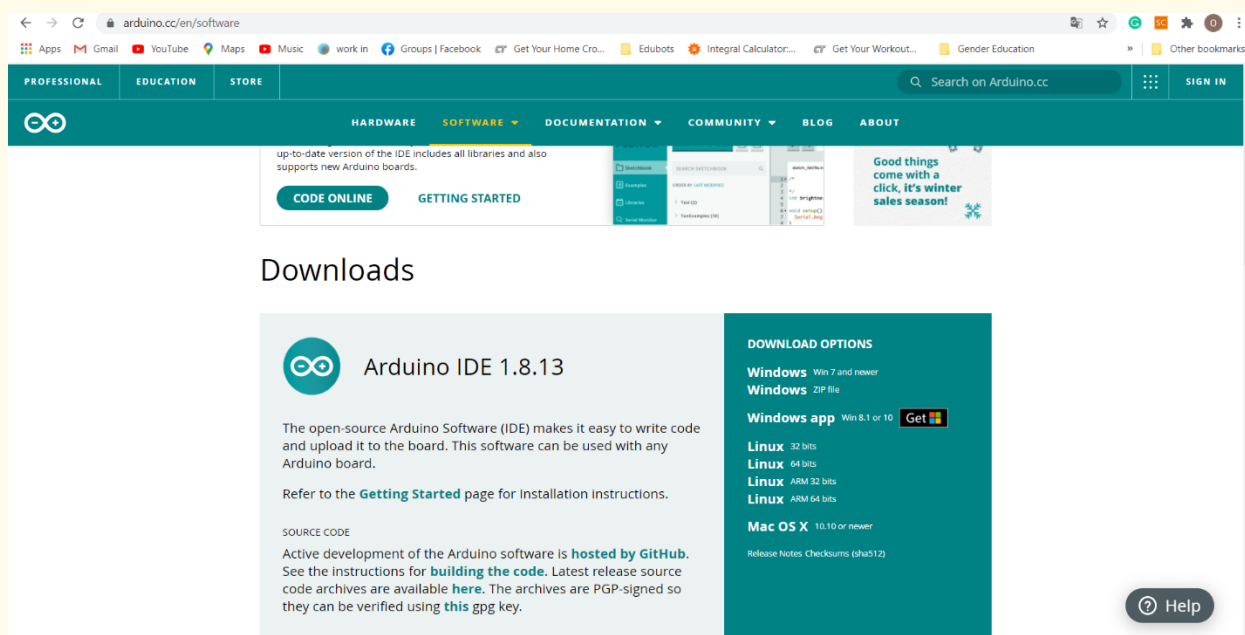


Figure 8 How to download Arduino IDE

As soon as the participants have downloaded Arduino IDE, the facilitators first provides general information in Arduino IDE and then guides the participants in the main areas of IDE. The following text is provided.

As already mentioned, the Arduino software is open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.



Figure 9 The Arduino IDE on Windows (picture on the left, source: Boxall, 2013), and Mac (picture on the right, source: Smith, 2011).

The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment. We can use Arduino IDE on the computer to create, open, and change sketches (the Arduino program is called “sketch”). Sketches define what the board will do. You can either use the buttons along the top of the IDE or the menu items.

As shown in Figure 9, the Arduino IDE resembles a simple word processor. The IDE is divided into three main areas: the command area, the text area, and the message window area.

Menu items in Windows IDE

Command area: The command area includes the title bar, menu items, and icons. The title bar displays the sketch’s filename, as well as the version of the IDE. Below this is a series of menu items (File, Edit, Sketch, Tools, and Help) and icons, as described next.

As with any word processor or text editor, you can click one of the menu items to display its various options.

- File: contains options to save, load, and print sketches; a thorough set of example sketches to open; as well as the Preferences submenu
- Edit: contains the usual copy, paste, and search functions common to any word processor
- Sketch: contains the function to verify your sketch before uploading to a board, and some sketch folder and import options
- Tools: contains a variety of functions as well as the commands to select the Arduino board type and USB port
- Help: contains links to various topics of interest and the version of the IDE

The icons: Below the menu toolbar are six icons. Mouse over each icon to display its name. The icons, from left to right, are as follows:

- Verify: click this to check that the Arduino sketch is valid and does not contain any programming mistakes.
- Upload: click this to verify and then upload your sketch to the Arduino board.
- New: click this to open a new blank sketch in a new window.
- Open: click this to open a saved sketch.
- Save: click this to save the open sketch. If the sketch does not have a name, you will be asked to provide one.
- Serial monitor: click this to open a new window for use in sending and receiving data between your Arduino and the IDE.

Menu items in MAC IDE

Parts of the IDE (from left to right, top to bottom) as illustrated on the right side of Figure 8, are given below:

- Compile: before your program “code” can be sent to the board, it needs to be converted into instructions that the board understands. This process is called compiling.
- Stop: This stops the compilation process.
- Create new Sketch: This opens a new window to create a new sketch.
- Open Existing Sketch: This loads a sketch from a file on your computer.
- Save Sketch: This saves the changes to the sketch you are working on.
- Upload to Board: This compiles and then transmits over the USB cable to your board.

In addition:

- Tab Button: This lets you create multiple files in your sketch. This is for more advanced programming.
- Sketch Editor/ Text area: This is where you write or edit sketches. You will enter the contents of your sketch here as you would in any text editor.
- Text Console/ Message Window Area: This shows you what the IDE is currently doing and is also where error messages display if you make a mistake in typing your program (often called a syntax error).
- LineNumber: This shows you what line number your cursor is on. It is useful since the compiler gives error messages with a line number.

How to change language

The facilitator provides further instructions to the participants on how to change the language (optional).

The instructions are given below:

1. Click on FILE and select PREFERENCES.
2. Next to the Editor Language there is a dropdown menu of currently supported languages.
3. Select your preferred language from the menu.
4. Restart the software to use the selected language.



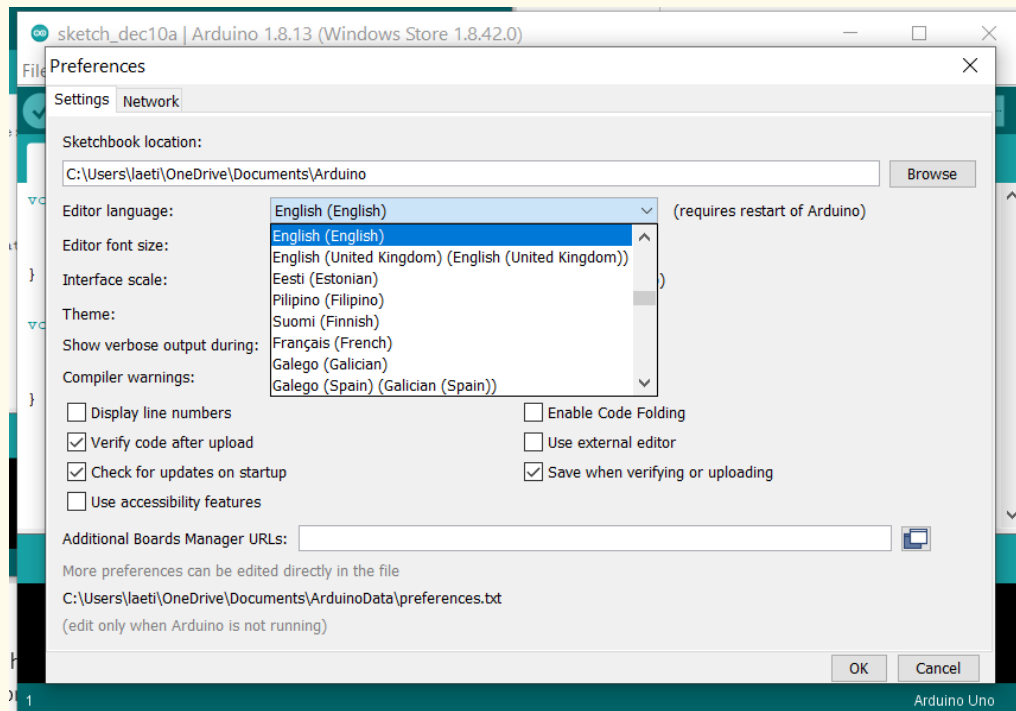


Figure 10 How to change language

- **Bibliographic references:**

Arduino (2020). Arduino IoT Cloud. Retrieved from: <https://www.arduino.cc/en/IoT/HomePage>

Autodesk, Inc. (2020). What You'll Learn. Retrieved from: <https://www.instructables.com/Tools-and-Materials-for-Arduino/>

Boxall, J. (2013). *Arduino workshop: A Hands-On introduction with 65 projects*. No starch press.

Makerspaces (2020). Arduino For Beginners. Retrieved from: <https://www.makerspaces.com/arduino-uno-tutorial-beginners/>

Smith, A. G. (2011). *Introduction to Arduino*. Alan G. Smith.

- Appendix

Table 1. Arduino boards based on ATMEGA328 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Uno R3	5V	16MHz	14	6	6	1	USB via ATmega16U2
Arduino Uno R3 SMD	5V	16MHz	14	6	6	1	USB via ATmega16U2
Red Board	5V	16MHz	14	6	6	1	USB via FTDI
Arduino Pro 3.3v/8 MHz	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
Arduino Pro 5V/16MHz	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino mini 05	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 3.3v/8mhz	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
Arduino Pro mini 5v/16mhz	5V	16MHz	14	8	6	1	FTDI-Compatible Header
Arduino Ethernet	5V	16MHz	14	6	6	1	FTDI-Compatible Header
Arduino Fio	3.3V	8MHz	14	8	6	1	FTDI-Compatible Header
LilyPad Arduino 328 main board	3.3V	8MHz	14	6	6	1	FTDI-Compatible Header
LilyPad Arduino simple board	3.3V	8MHz	9	4	5	0	FTDI-Compatible Header

Table 2. Arduino boards based on ATMEGA32u4 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Leonardo	5V	16MHz	20	12	7	1	Native USB
Pro micro 5V/16MHz	5V	16MHz	14	6	6	1	Native USB
Pro micro 3.3V/8MHz	5V	16MHz	14	6	6	1	Native USB
LilyPad Arduino USB	3.3V	8MHz	14	6	6	1	Native USB

Table 3. Arduino boards based on ATMEGA2560 microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	5V	16MHz	54	16	14	4	USB via ATmega16U2B
Mega Pro 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro 5V	5V	16MHz	54	16	14	4	FTDI-Compatible Header
Mega Pro Mini 3.3V	3.3V	8MHz	54	16	14	4	FTDI-Compatible Header

Table 4. Arduino boards based on AT91SAM3X8E microcontroller

Board Name	Operating Volt	Clock Speed	Digital i/o	Analog Inputs	PWM	UART	Programming Interface
Arduino Mega 2560 R3	3.3V	84MHz	54	12	12	4	USB native

LESSON – CONCEPTS: INPUT, OUTPUT, ANALOG, DIGITAL

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 1:00h
- **Educational objectives:**
 - Understand the concept of input and output (I/O)
 - Differentiate Analog and Digital I/O
 - Identify if an I/O is analog or digital on an Arduino
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Recognize and identify inputs and outputs • Identify if an I/O is digital or analog • Be able to select the correct pins for any I/O 	<ul style="list-style-type: none"> • Apprehend digital or analog pins • Configure inputs and outputs 	<ul style="list-style-type: none"> • Understand the very basics of circuit, analog and digital I/O

- **Main Keyword(s):**
 - Digital
 - Analog
 - Input
 - Output

Important definitions

Concept: Input/Output (I/O)

Inputs: any signal or information going into / entering the board.

Most of the time inputs cannot be seen and they are processed by sensors.

Outputs: any signal or action exiting the board.

Most of the time outputs are visible results and are revealed by LED, motors, alarms, or any action.



Brief definition of Analog and Digital I/O

The analog signals are continuous. They are sent to a system when I/O have multiple states which refer to variables.
Most common analog signals: temperature, level, rate or flow.
Example: an analog signal will inform how bright the LED (output) is (intensity)

The digital signals are used in situations where I/O only have 2 states like ON/OFF, Open/Close, Start/Stop...
A digital signal will inform if the LED is making light or not (On/off)

All analog signals include digital signals: example about the LED intensity analog will signal will refer to 0%= LED Off and 100%= LED On in addition to other values like 25% On – 85% On...

Inputs vs Outputs

The facilitator will start the lesson by explaining to the participants what inputs and outputs are and help them differentiate Analog and Digital.

The facilitator begins as follow:

What are Inputs or Outputs often written I/O in the programming language.

I/Os are the results of robot/machine interaction with the real world.

In simple terms, we could say that inputs are invisible while outputs are the visible results.

We, therefore, dissociate the inputs which are most of the time transmitted by sensors (switches, potentiometers, cameras, etc.) and the outputs which refer to the motors which will start, the LED which will light up, the alarm which will be triggered...

Examples:

When you type on the keys of your keyboard you perform Inputs. In fact, you type information, but you do not see what happens when this information is recorded. The machine processes this information to translate it on the screen. What you see (the letters that appear on the screen) is therefore an Output.

Example with Arduino: if you program a button that, if activated, should light a led, your button will be an input (you do not see the information that enters and is processed) and the led that lights up or turns off will be an output.



There are two types of Inputs/Outputs: The Analog or Digital I/O.

Analog vs Digital I/O

The facilitator gives the definition and some examples of Analog and Digital signals and explains the points below:

There are two ways to differentiate if a signal is Analog or Digital:

1. By the type of sensor used
2. By the processing method

1. Analog vs digital sensors

Let's take an example of a fading LED.

The state of the LED can be ON, OFF or variable intensity: ON but low intensity or ON with high intensity.

- If the sensor is a switch placed on the LED then it will detect whether the LED is ON or OFF and will not detect anything else. This is a digital input.
- If the sensor is a LDR (light dependent resistor) it will convert the light of the LED and tell us if the LED is 0% (off), 100% (full brightness) and many other variables 20,23%, 86% related to the dimming stages of the LED.

2. Analog vs Digital Processing

Another way to differentiate Analog and Digital is to check how both are processed.

- The time:

Analog is continuously processing information and whenever an input is modified, the output is also instantly changed according to the input. The update is immediate. For Digital, processing will have a short delay until the system registers the change. This delay is established by the "sampling frequency" and controlled by a clock.
- The resolution:

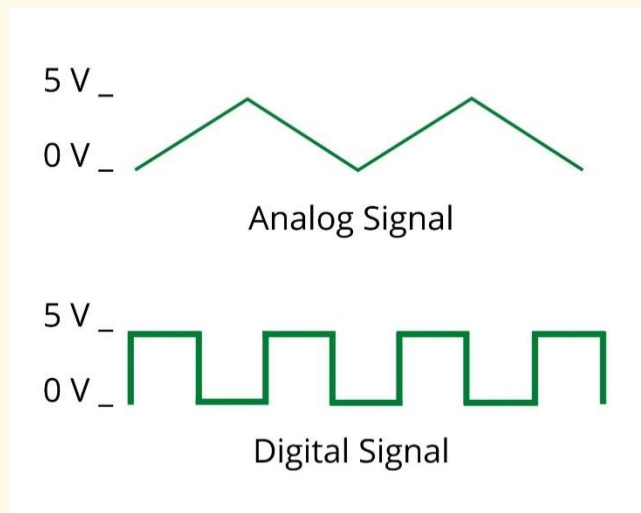
As mentioned above, Analog processing has infinite resolutions because the signal never stops and values are different from each other. Whereas for Digital processing, signals are either 0 or 1 (binary numbers) which means they need to be converted into only two values.



Example with the LED:

-> Switch it on/off with analog processing: when the dimer is activated, the circuit will change the intensity of the LED accordingly immediately.

-> Switch it on/off with digital processing: with a set sampling frequency, every 5 seconds, the system will read the switch and indicate if the light is on or off, no matter how intense and bright the light is. When you change the dimer within this 5 seconds, the LED's light will not change.



Conclusion

The real world is always about analog signals, however, for robotic purposes digital signals and digital processing are more recommended.

First because digital processing is cheaper and more flexible compared to analog processing.

Second, because programs work with digital signals (binary system).

And third because analog signals are more likely to be affected by electrical circuit noise. For this reason, most of the time, even analogue signals are converted into digital signals.

Finally, we need to keep in mind that all analog signals from inputs and outputs can be converted into digital signals while the contrary does not apply.

- Bibliographic references:

<https://blog.robotiq.com/whats-the-difference-between-digital-and-analog-i/o>

<https://www.mines.edu/epics/wp-content/uploads/sites/99/2018/01/John-Steele-Basics-of-Arduino-Presentation.pdf>

LESSON – ARDUINO SOFTWARE

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 1:00h
- **Educational objectives:**
 - Download Arduino software (IDE)
 - Learn the basics of the software
 - First steps with Arduino software
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Arduino software • Create a new project • Check / export... • 	<ul style="list-style-type: none"> • Download run and use Arduino software 	<ul style="list-style-type: none"> • Use Arduino software autonomously

- **Main Keyword(s):**
 - Arduino software
 - Arduino IDE 1.8.13
- **Required material and resources:**
 - Arduino UNO
 - Computer
 - Arduino USB cable
 - Software Arduino IDE
 - Internet Connection

What is Arduino?

Arduino is an **open-source electronics platform** based on easy-to-use hardware and software. The Arduino boards can read **inputs** (like light on a sensor, a finger on a button...) and turn them into **outputs** (activating a motor, turning on an LED.)

To program an Arduino board (give instructions to microcontrollers on the board) you need to use what **Arduino programming language** (based on **Wiring**), and the **Arduino Software (IDE)**, based on **Processing**.

It is intended for anyone who want to make interactive projects

Definition from: <https://www.arduino.cc/en/Guide/Introduction>

Arduino Software (IDE)

IDE stands for Integrated Development Environment.

Thanks to the Arduino software, it is possible to write programs and codes and upload the corresponding sketches to any Arduino board.

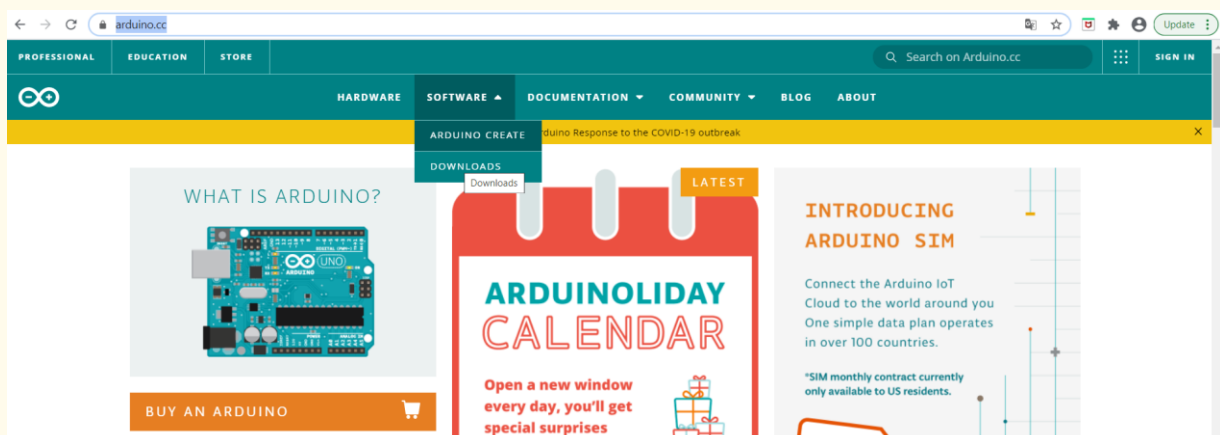
Download Arduino IDE

The facilitator will start the lesson by explaining to the participants what Arduino is, what it can be used for and how to use it.

The facilitator begins with the following steps:

To download the software, go on: <https://www.arduino.cc/>

Click on SOFTWARE -> DOWNLOADS



The version ARDUINO IDE 1.8.13

Click on the Download option that corresponds to your computer.

If you have doubts, click on **Getting Started** to read more information about installation of the software for your computer.

Downloads



Arduino IDE 1.8.13

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is hosted by [GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer

Windows ZIP file

Windows app Win 8.1 or 10 [Get](#)

Linux 32 bits

Linux 64 bits


Linux ARM 32 bits

Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#)

[Checksums \(sha512\)](#)




Arduino IDE

Arduino LLC • Outils de développement

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

Plus



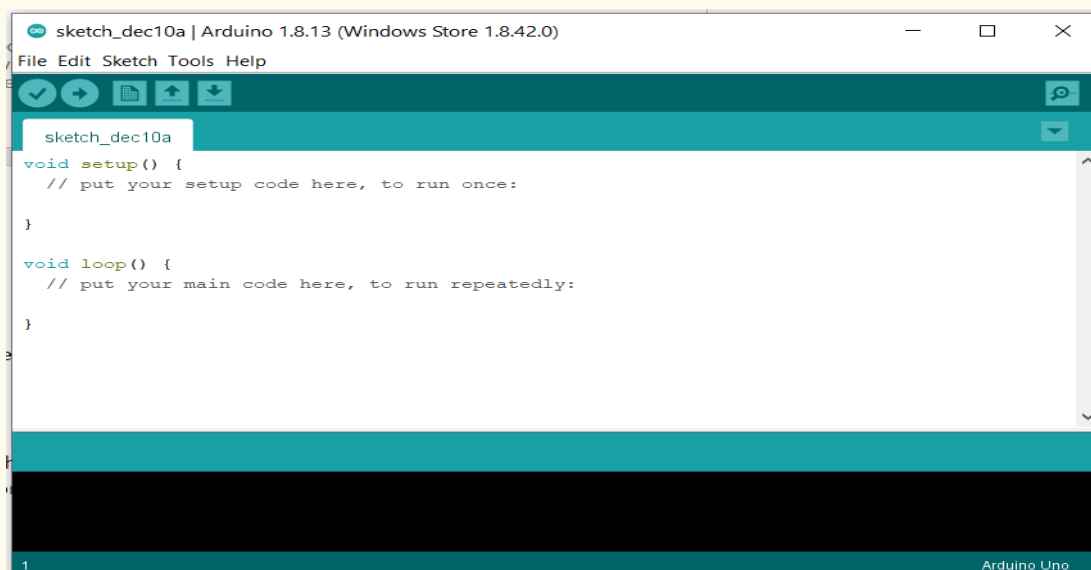
Gratuit

[Télécharger](#)

⚠ Voir la configuration requise

Quick overview of Arduino software

When the download is over, the below page will automatically open:



```

sketch_dec10a | Arduino 1.8.13 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help
sketch_dec10a
void setup() {
  // put your setup code here, to run once:
}

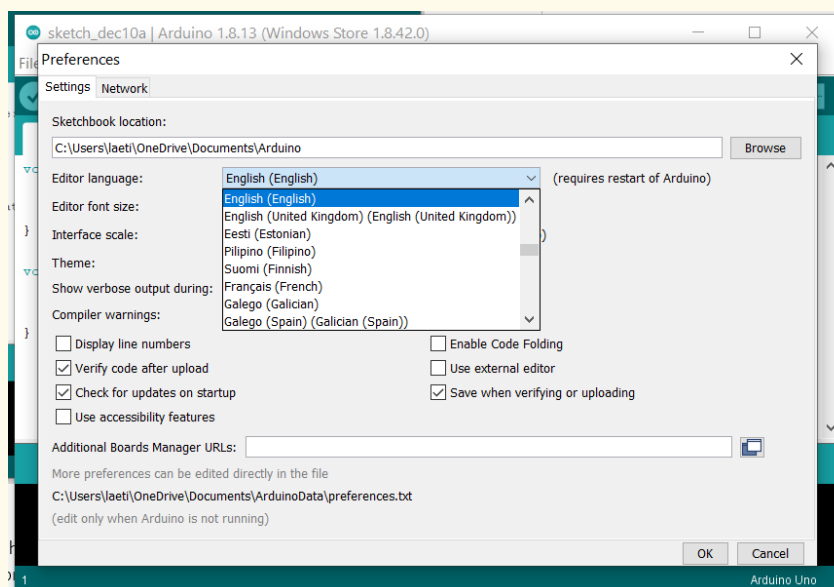
void loop() {
  // put your main code here, to run repeatedly:
}

Arduino Uno
  
```

The facilitator will guide the participants with step-by-step instructions on understanding how does the software work and how to create projects.

To change the language:

1. Click on FILE and select PREFERENCES.
2. Next to the Editor Language there is a dropdown menu of currently supported languages.
3. Select your preferred language from the menu.
4. Restart the software to use the selected language.

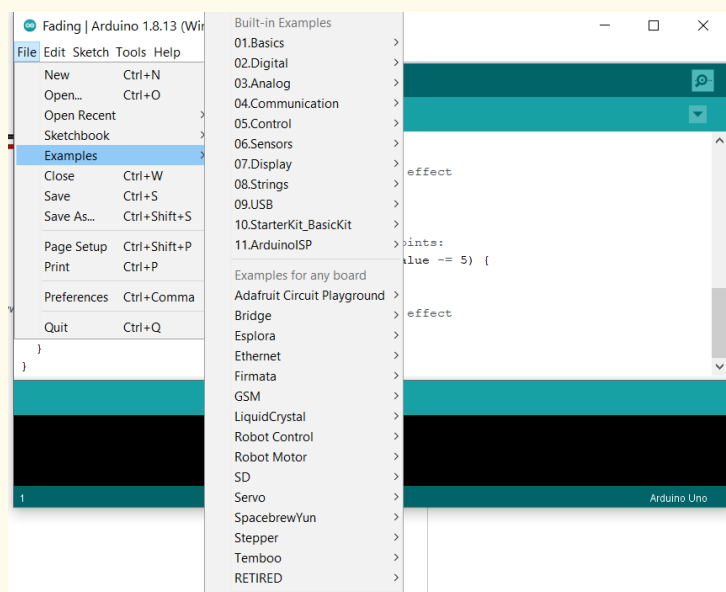


To create a new project:

File -> New

It is also possible to use already developed examples to have some inspiration or to reproduce them:

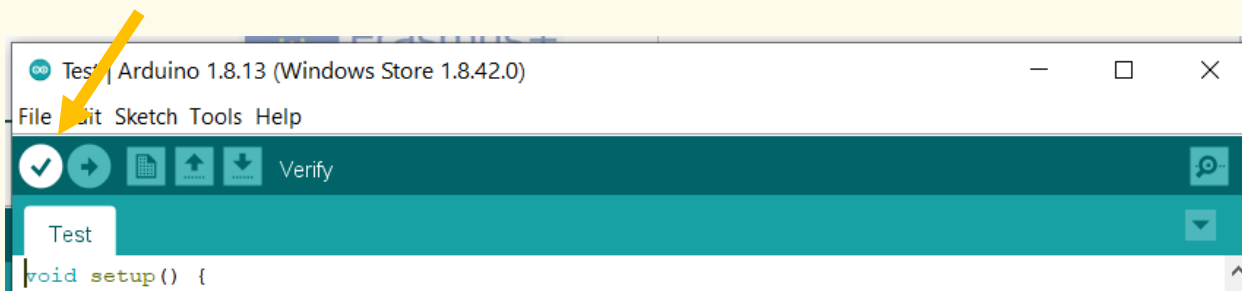
File -> Examples



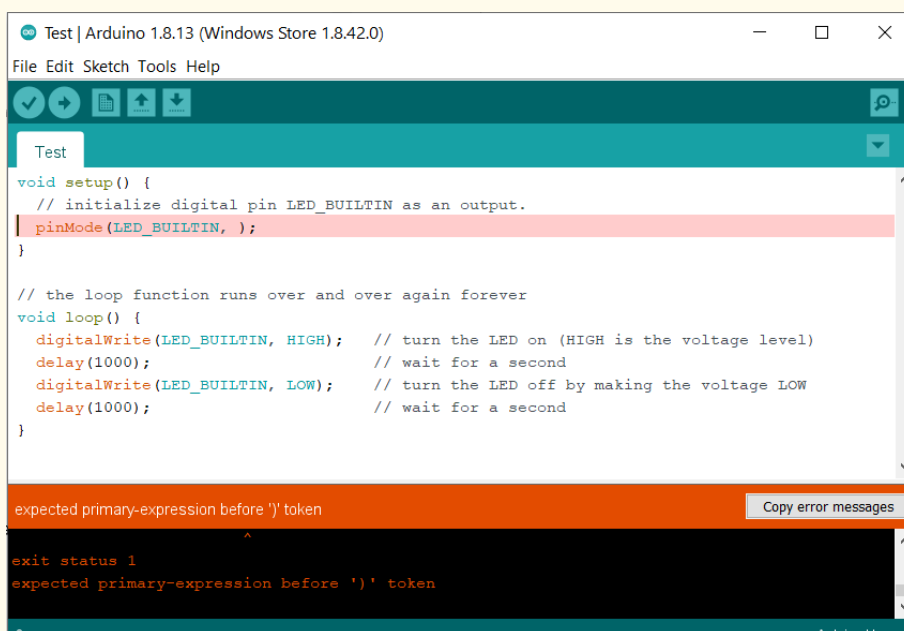
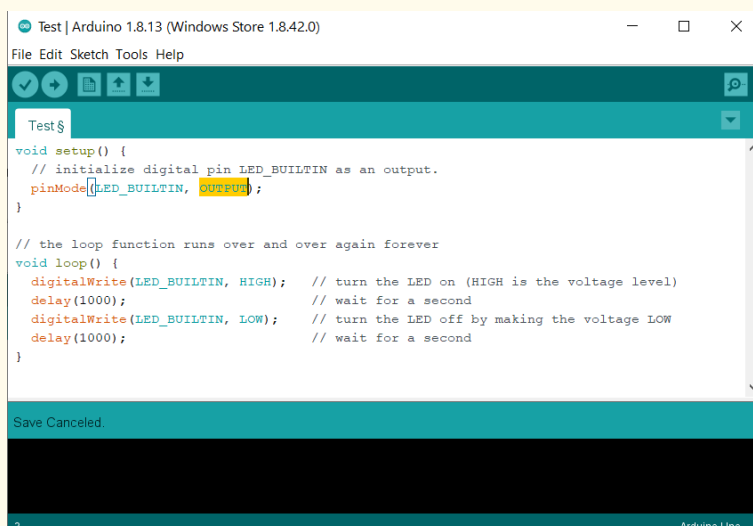
To verify the project:

Click on the TICK on the left below the Tab FILE.

It will turn orange while it is compiling information. Wait until it comes back to its initial color.



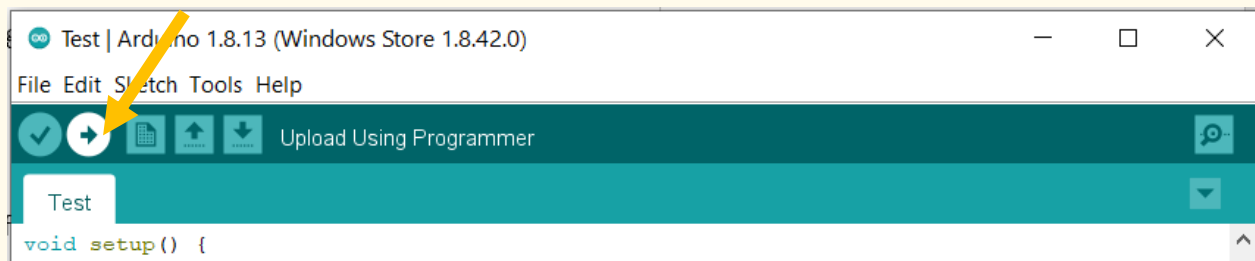
For this example, we have selected a sketch about blinking LED and we have removed the information about "output" so the "verify function" will display the error:



To upload the program:

To upload the program, the Arduino board should be plugged into your computer with a USB cable.

To upload the program, you should click on the horizontal arrow:



When the program is uploaded, the arrow will come back to its initial color.

If you have plugged and set up your Arduino board according to what you have programmed, you will be able to watch your program in action.

- Bibliographic references:

<https://www.arduino.cc/en/Guide>

https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-on-various-platforms-4b3e4a

MICROCONTROLLER AND ARDUINO FUNDAMENTALS

LESSON – HOW TO PROGRAM IN ARDUINO, AND UPLOAD PROGRAMS TO THE BOARD

- **Study section:** Introduction to microcontrollers
- **Lesson duration:** 1:00h + 1:h00
- **Educational objectives:**
 - Learn the basics on how to program Arduino
 - Learn how to upload programs to Arduino board
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • How to program Arduino • How to upload programs (sketches) to Arduino board 	<ul style="list-style-type: none"> • Program Arduino and create a simple sketch • Upload a sketch to Arduino board 	<ul style="list-style-type: none"> • Be capable of creating and uploading a simple sketch

- **Main Keyword(s):**
 - Arduino UNO board
 - Arduino IDE
 - Program / Sketch
- **Required material and resources:**
 - Arduino UNO
 - Software Arduino IDE
 - Computer
 - Arduino USB cable

Brief definition of Sketches

Arduino calls programs are called “sketches”. An Arduino sketch is a set of instructions that you create to accomplish a particular task; in other words, a sketch is a program.

How to programme Arduino

The facilitator first provides essential pieces of information on Arduino programs structures and then guides the participants in creating their first sketch in the IDE. The following text is provided:

Arduino programs can be divided in three main parts: **Structure, Values (variables and constants), and Functions**. In this session, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Arduino – Programme Structure

Let us start with the Structure. Software structure consist of two main functions:

Setup() function

Loop() function

```
Void setup ( ) {
}
```

PURPOSE – The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board.

INPUT – -

OUTPUT – -

RETURN – -

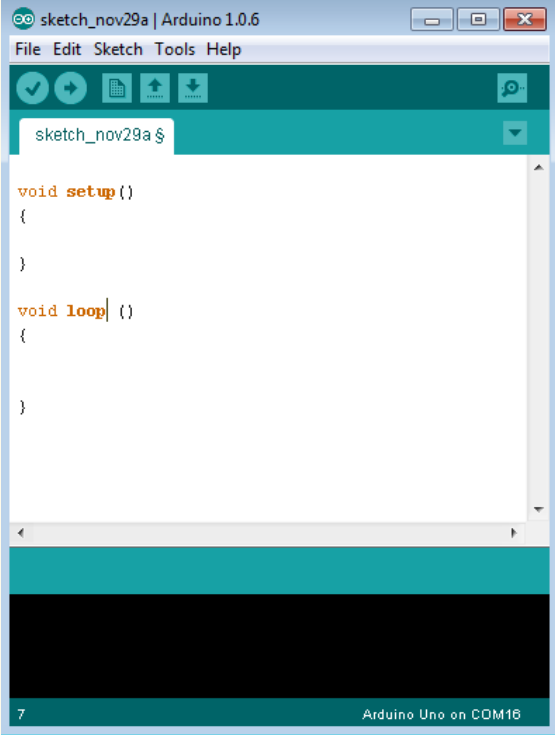
```
Void Loop ( ) {
}
```

PURPOSE – After creating a `setup()` function, which initializes and sets the initial values, the `loop()` function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

INPUT – -

OUTPUT – -

RETURN – -



```

sketch_nov29a $
void setup ()
{
}
void loop ()
{
}
  
```

Figure 1 Programme structure, source:

https://www.tutorialspoint.com/arduino/arduino_program_structure.htm

Arduino – Data Types

Data types in C refers to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.

The data types that you can use during Arduino programming are listed below.

void	Boolean	char	Unsigned char	byte	int	Unsigned int	word
long	Unsigned long	short	float	double	array	String-char array	String-object

Arduino – Variables

Variables in C programming language, which Arduino uses, have a property called scope. A scope is a region of the program and there are three places where variables can be declared. They are:

- Inside a function or a block, which is called local variables.
- In the definition of function parameters, which is called formal parameters.
- Outside of all functions, which is called global variables.

Example of local variables:

```
Void setup () {
}

Void loop () {
  int x , y ;
  int z ; Local variable declaration
  x = 0;
  y = 0; actual initialization
  z = 10.
```

Example of global variables:

```
Int T , S ;
float c = 0 ; Global variable declaration

Void setup () {
}

Void loop () {
  int x , y ;
  int z ; Local variable declaration
  x = 0;
  y = 0; actual initialization
  z = 10;
}
```

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators:

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators



More details about each type of operators can be retrieved from here:

https://www.tutorialspoint.com/arduino/arduino_operators.htm

Arduino – Control statements

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program. It should be along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found in most of the programming languages:

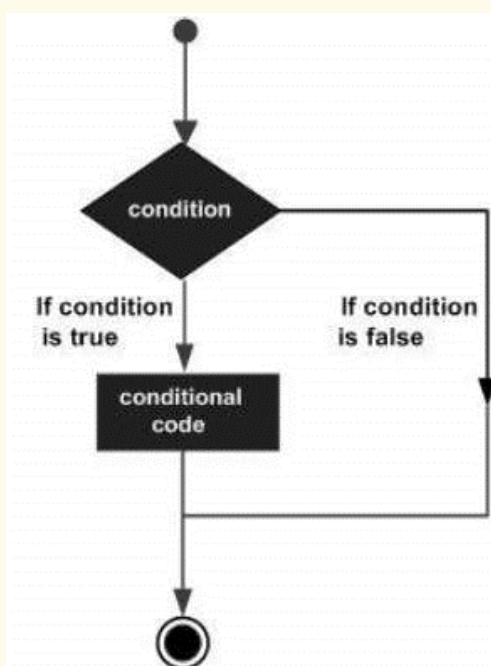


Figure 2 The general form of a typical decision-making structure found in programming languages

Control Statements are elements in Source Code that control the flow of program execution. Those are listed below.

No	Control Statement & Description
1	<p><u>If statement</u></p> <p>It takes an expression in parenthesis and a statement or block of statements. If the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.</p>
2	<p><u>If ...else statement</u></p> <p>An if statement can be followed by an optional else statement, which executes when the expression is false.</p>
3	<p><u>If...else if ...else statement</u></p> <p>The if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.</p>

4	<u>switch case statement</u> Similar to the if statements, switch...case controls the flow of programs by allowing the programmers to specify different codes that should be executed in various conditions.
5	<u>Conditional Operator? :</u> The conditional operator?: is the only ternary operator in C.

Arduino – Loops

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages.

C programming language provides the following types of loops to handle looping requirements.

No.	Loop & Description
1	<u>while loop</u> while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit.
2	<u>do...while loop</u> The do...while loop is similar to the while loop. In the while loop, the loop-continuation condition is tested at the beginning of the loop before performed the body of the loop.
3	<u>for loop</u> A for loop executes statements a predetermined number of times. The control expression for the loop is initialized, tested and manipulated entirely within the for loop parentheses.
4	<u>Nested Loop</u> C language allows you to use one loop inside another loop. The following example illustrates the concept.
5	<u>Infinite loop</u> It is the loop having no terminating condition, so the loop becomes infinite.

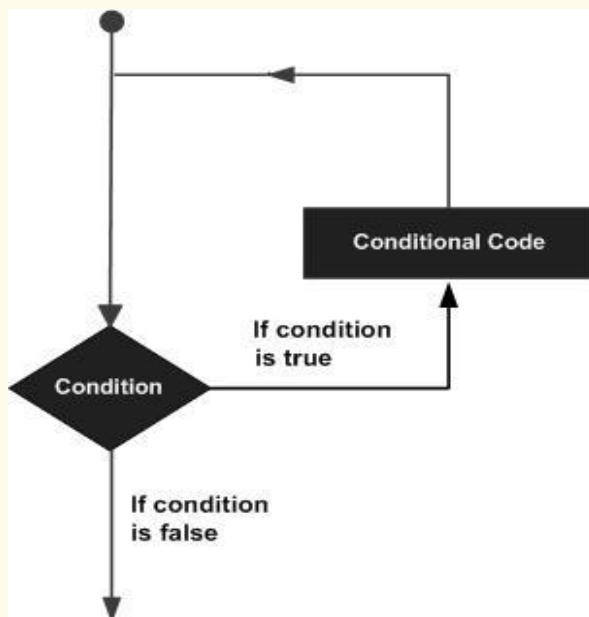


Figure 3 The general form of a loop statement in most of the programming languages

Creating your first sketch in the Arduino IDE

The facilitator guides the participants to create and upload a simple sketch. The following text can be used:

In this part of the session you will create and upload a simple sketch that will cause the Arduino's LED to blink repeatedly, by turning it on and then off for 1 second intervals.

To begin, connect your Arduino to the computer with the USB cable. Then open the IDE, choose Tools4Serial Port, and make sure the USB port is selected. This ensures that the Arduino board is properly connected.

The first stage in creating any sketch is to add the void setup() function. This function contains a set of instructions for the Arduino to execute once only, each time it is reset or turned on. To create the setup function, add the following lines to your sketch:

```
void setup()
{
}
```

Our program will blink the user LED on the Arduino. The user LED is connected to the Arduino's digital pin 13. A digital pin can either detect an electrical signal or generate one on command. In this small project, we will generate an electrical signal that will light the LED. Enter the following into your sketch between the braces ({ and }):

```
pinMode(13, OUTPUT); // set digital pin 13 to output
```

The number 13 in the listing represents the digital pin you're addressing. You are setting this pin to OUTPUT, which means it will generate (output) an electrical signal. If you wanted it to detect an incoming electrical signal, then you would use INPUT instead. Notice that the function `pinMode()` ends with a semicolon (;). Every function in your Arduino sketches will end with a semicolon. Save your sketch again to make sure that you don't lose any of your work, by choosing File > Save As. Enter a short name for your sketch, and then click OK.

Remember that our goal is to make the LED blink repeatedly. To do this, we will create a loop function to tell the Arduino to execute an instruction over and over until the power is shut off or someone presses the RESET button.

Enter the code shown in boldface after the `void setup()` section in the following listing to create an empty loop function. Be sure to end this new section with another brace (}), and then save your sketch again.

```

/*
Arduino Blink LED Sketch
by Mary Smith, created 09/09/12
*/
void setup()
{
pinMode(13, OUTPUT); // set digital pin 13 to output
}
void loop()
{
// place your main loop code here:
}

```

Next, enter the actual functions into `void loop()` for the Arduino to execute.

Enter the following between the loop function's braces, and then click Verify to make sure that you have entered everything correctly:

```

digitalWrite(13, HIGH); // turn on digital pin 13
delay(1000); // pause for one second
digitalWrite(13, LOW); // turn off digital pin 13
delay(1000); // pause for one second

```


The `digitalWrite()` function controls the voltage that is output from a digital pin: in this case, pin 13 to the LED. By setting the second parameter of this function to `HIGH`, a “high” digital voltage is output; then current will flow from the pin and the LED will turn on.

With the LED turned on, the light pauses for 1 second with `delay(1000)`. The `delay()` function causes the sketch to do nothing for a period of time— in this case, 1,000 milliseconds, or 1 second.

Next, we turn off the voltage to the LED with `digitalWrite(13, LOW)`; Finally, we pause again for 1 second while the LED is off, with `delay(1000)`; The completed sketch should look like this:

```

/*
Arduino Blink LED Sketch
by nickname, created 28/12/20
*/
void setup()
{
pinMode(13, OUTPUT); // set digital pin 13 to output
}
void loop()
{
digitalWrite(13, HIGH); // turn on digital pin 13
delay(1000); // pause for one second
digitalWrite(13, LOW); // turn off digital pin 13
delay(1000); // pause for one second
}

```

Before you do anything further, save your sketch.

Next, you should verify your sketch. When doing so, you ensure that it has been written correctly in a way that the Arduino can understand. To verify your complete sketch, click `Verify` in the IDE and wait a moment. Once the sketch has been verified, a note should appear in the message window, as shown in Figure 4.



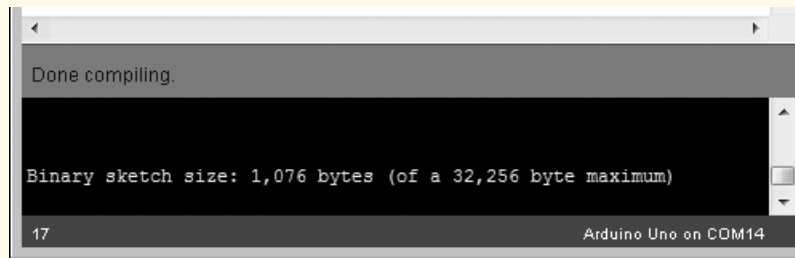


Figure 4 The sketch has been verified.

This “Done compiling” message tells you that the sketch is okay to upload to your Arduino. It also shows how much memory it will use (1,076 bytes in this case) of the total available on the Arduino (32,256 bytes).

In case that your sketch is not OK, e.g., you forgot to add a semicolon at the end of the second `delay(1000)` function, then when you click Verify, the message window should display a verification error message similar to the one shown in Figure 5.



Figure 5 The message window with a verification error.

The message tells you that the error occurs in the void loop function, lists the line number of the sketch where the IDE thinks the error is located (blinky:16, or line 16 of your blinky sketch), and displays the error itself (the missing semicolon, error: expected ';' before '}' token). Furthermore, the IDE should also highlight in yellow the location of the error or a spot just after it. This helps you easily locate and rectify the mistake.

Uploading and Running Your Sketch

Once having saved your sketch, ensure that your Arduino board is connected, and click Upload in the IDE. The IDE may verify your sketch again and then upload it to your Arduino board. During this process, the TX/RX LEDs on your board should blink, indicating your program functions properly.

- **Bibliographic references:**

Arduino (2020). Arduino Tutorial. Retrieved from:
<https://www.tutorialspoint.com/arduino/index.htm>

Boxall, J. (2013). *Arduino workshop: A Hands-On introduction with 65 projects*. No starch press.

Smith, A. G. (2011). *Introduction to Arduino*. Alan G. Smith.



LESSON – BREADBOARD ASSEMBLIES

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Discovering the breadboard assemblies
 - Understand the basics of breadboard assemblies
 - Discovers the parts to create a circuit
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● What is a Breadboard assemblies ● How to use it ● How does it work ● The different parts 	<ul style="list-style-type: none"> ● Power a breadboard assemblies ● Build a circuit ● Recognize the parts 	<ul style="list-style-type: none"> ● Be capable of building a simple circuit on a breadboard

- **Main Keyword(s):**
 - Breadboard
 - Parts: jumper, push buttons, resistors, LEDs...
- **Required material and resources:**
 - Breadboard Arduino Compatible (BBAC)
 - A 9V battery
 - A battery clip
 - A LEDs
 - Jumpers

What is a Breadboard?

A **breadboard** is a thin plastic board used to hold electronic components (transistors, resistors, chips, etc.) that are wired together to create a circuit. It is used to develop prototypes of electronic circuits.

Breadboards can be used to create one-of-a-kind systems but rarely become commercial products. A breadboard can be reused for several different projects.

Definition from: <https://www.pcmag.com/encyclopedia/term/breadboard>

Present the breadboard

The facilitator will start the lesson by showing different type of breadboard and explain to participants what it is used for, and how to use it.

The facilitator begins with the following steps:

THE BREADBOARD also called SOLDERLESS BREADBOARD

There are several kinds of breadboard also called Solderless Breadboard with different size, shape and purpose. The one we will focus on for this lesson is the breadboard Arduino compatible that looks like the image below:

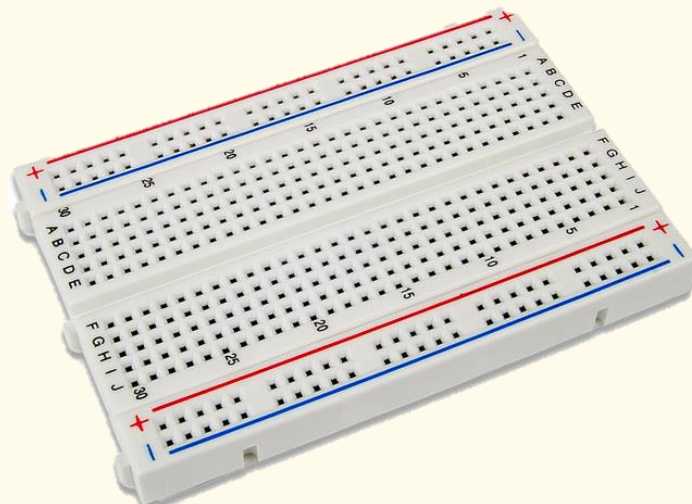
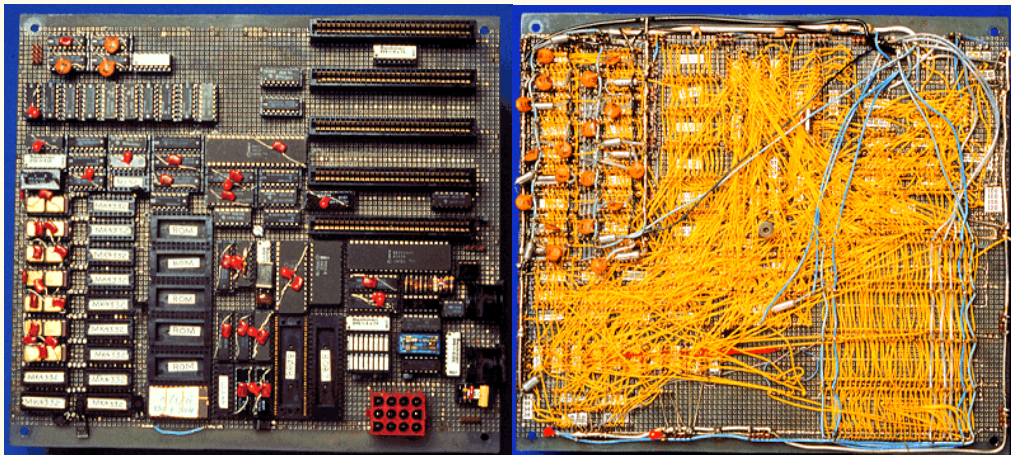


Image: <https://blog.digilentinc.com/a-breadboard-for-every-circuit/>

More complex breadboards



Source: <https://www.pcmag.com/encyclopedia/term/breadboard>

WHAT IS THE BREADBOARD USED FOR?

A breadboard is used to construct simple electronic circuits which do not require to use a soldering iron. Components like pins, microcontrollers, LEDs, jumpers... are pushed/ plugged into the holes (sockets) of the breadboard.

Wires called jumpers enable the connection (power) within the different elements.

How does the breadboard work?

The most common breadboard is composed of rows and columns that are electrically connected.

You have two types of rails:

The vertical rails or columns:

The vertical lines (red and blue) with + and – that run the length of the board. They are traditionally used for power and ground.

The horizontal rails or lines:

1,2,3...up to 30 in our below example.
a-b-c-d-e of each line are connected together.

But 1 – 2 – 3 ...30 are separated by a gap so they are not connected.

To have a better understanding, it is worth having a look at the back of a breadboard with the metal connections.

Caution: do not rip the back off your breadboard, it will ruin the breadboard.

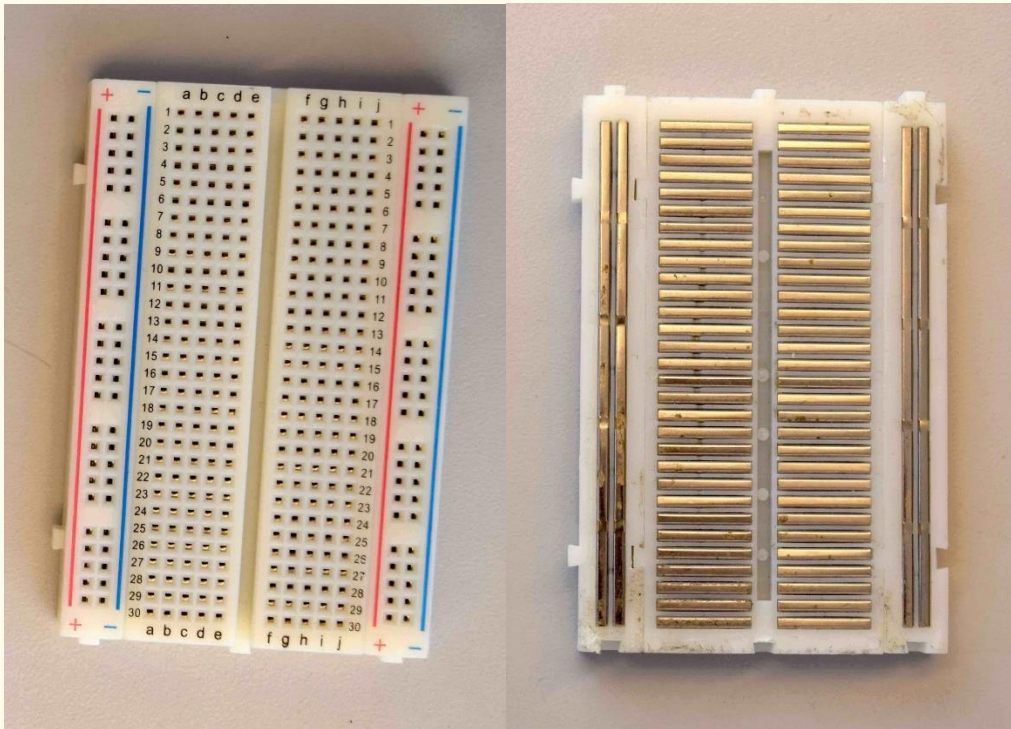


Image source: <https://hcie.csail.mit.edu/classes/2018-fall-6810/6810-electronics.html>

TO HAVE A WORKING CIRCUIT

To develop a working circuit, your breadboard needs to be alimented with power. In the next lessons we will have power through an USB cable connected to Arduino, but for this lesson we will have a look at a classic 9V battery.

The components:

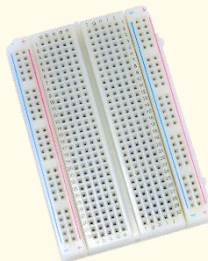
A battery clip



A 9V battery



A breadboard



To connect the breadboard to power, jumpers (wires) from the battery should be plugged in the vertical lines.

The red jumper in the red line (the +) and the black jumper in the ground (the -)

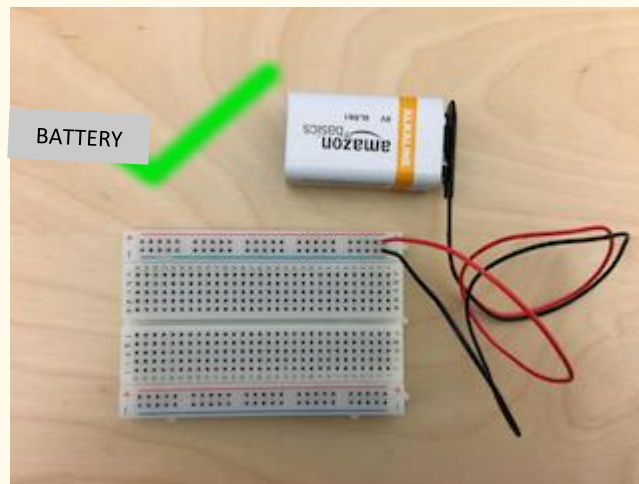


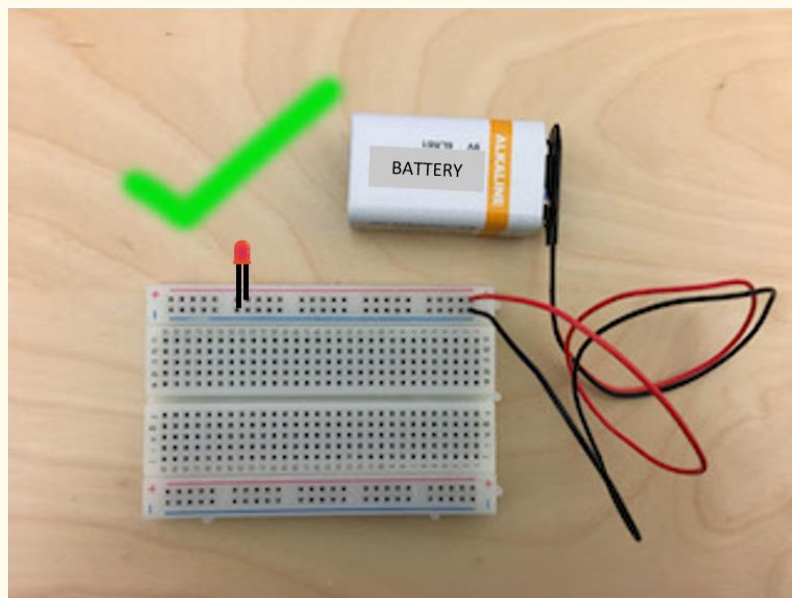
image: <https://hcie.csail.mit.edu/classes/2018-fall-6810/6810-electronics.html>

Example

At this stage, facilitator should display what are the bad practices and what should not be done:

- Plugging the red jumper in the minus (ground) and the black jumper in the plus.
- Plugging the jumpers in different locations than the vertical lines (1-A and 5-B or 4-A and 4-C)

Then facilitator could show how a LED can be switched on thanks to breadboard connections:



- Bibliographic references:

<https://www.seeedstudio.com/blog/2020/01/06/how-to-use-a-breadboard-wiring-circuit-and-arduino-interfacing/>

<https://learn.adafruit.com/lesson-0-getting-started/breadboard>

<https://hcie.csail.mit.edu/classes/2018-fall-6810/6810-electronics.html>

<https://www.arduino.cc/en/main/standalone>

<https://www.digikey.si/en/maker/projects/learn-to-breadboard/43bde9f32dad493182278f0441466a6e>

Images sources (components):

<https://core-electronics.com.au/solderless-breadboard-300-tie-points-zy-60.html>

<https://www.allekabels.be/blok-batterij/7289/1217862/blok-batterij-alkaline.html?gclid=Cj0KCQiAzsz->

[BRCCARIsANotFgM9Sxa3miWc_qzn_BUEsmQscHz6gp0mDBfN4itGROwA4fGiv99eo2IaAkViEALw_wcB](https://www.allekabels.be/blok-batterij/7289/1217862/blok-batterij-alkaline.html?gclid=Cj0KCQiAzsz-BRCCARIsANotFgM9Sxa3miWc_qzn_BUEsmQscHz6gp0mDBfN4itGROwA4fGiv99eo2IaAkViEALw_wcB)

<https://www.parts-express.com/snap-type-battery-clip-with-wire-leads-for-9v-batteries-battery-holders--090-805>

LESSON- DIGITAL OUTPUT: BLINKING LED WITH ARDUINO

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Build a circuit to have a blinking LED (breadboard & Arduino)
 - Program Arduino to trigger the blinking LED
 - The components
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • How to install a circuit • The digital outputs • How to program the Arduino to have a blinking LED 	<ul style="list-style-type: none"> • Know the components needed • Program/code the outputs • Build a circuit 	<ul style="list-style-type: none"> • Be capable of building a blinking LED circuit on a breadboard using an Arduino

- **Main Keyword(s):**
 - Breadboard
 - Components: jumpers, LED, resistors, LEDs...
 - Arduino
 - Digital output
- **Required material and resources:**
 - Arduino Uno R3
 - Arduino IDE software
 - Computer/Laptop
 - Connection cable to connect Arduino and Laptop
 - Breadboard Arduino Compatible (BBAC)
 - A LED
 - 1 Resistor (220ohm)
 - 2 Jumpers



Objective of the exercise

The facilitator will start the lesson by explaining what the objective of the lesson is:

In this lesson we want to develop something simple that can be developed with an Arduino, that is to say we want to build a circuit and program the Arduino so we will obtain a blinking LED every 1 second.

The signal we will receive is a digital output: a LED switching from on and off every 1 second.

We will therefore use the DIGITAL PINS of the Arduino.

If necessary, the facilitator can briefly present the concepts of I/O as well as the Arduino Uno and its functionalities.

Present the components and circuit

Then, the facilitator will present the different components that will be used during the lesson and show them how components must be manipulated to build this circuit.

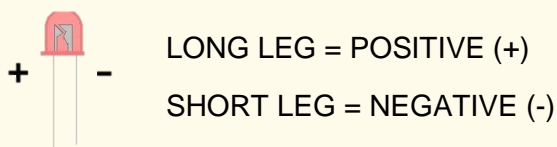
The facilitator begins with the following steps:

- **THE LED and the RESISTOR**

LEDs are small, powerful lights that can be used for many different applications. The LED has two legs, one + and one -.

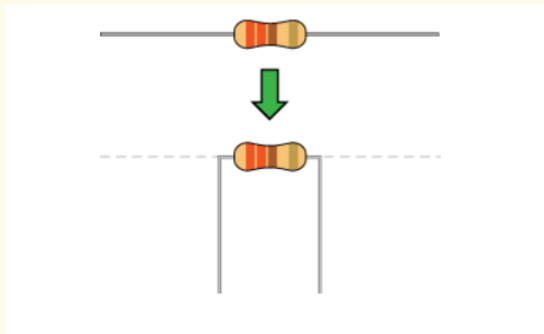
In our example, the legs of the LED can be either plugged in the pin of the Arduino board or on the breadboard.

To determine the positive and the negative leg you have to remember that:



Most of the time the LED is used with a RESISTOR

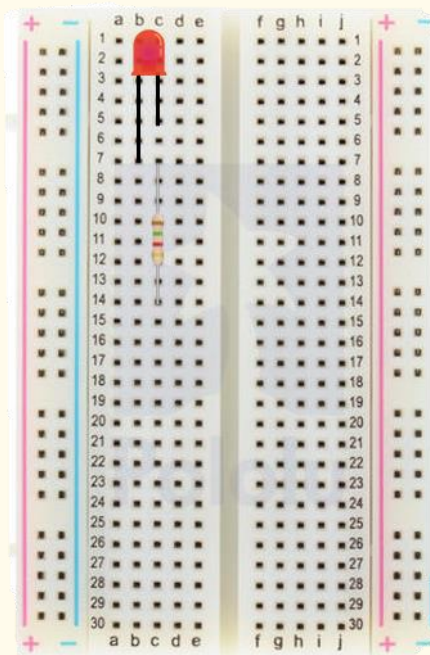
A **RESISTOR** is a component that allows to reduce the supply voltage (like a transformer delivering 12V DC) in order to obtain the necessary voltage to illuminate a led (for example 2.3V). Attention, the value of a resistor is calculated in relation to what you want to supply.



To be able to plug the resistor in the breadboard, you need to bend the terminals (legs) of the resistor into 90° angles, so it fits the sockets.

Remember that you can also cut the legs to obtain the appropriate length for your circuit.

Let's plug our LED and RESISTOR on the BREADBOARD:



LED

The long leg (+) of the LED is in 7B
The short leg of the LED is in 5C

RESISTOR

Terminals in 7C and 14C

- ARDUINO BOARD – BREADBOARD AND JUMPERS

To build your circuit you will first need to plug the jumpers: red and black wires into the pin of the Arduino and make the connections with the breadboard.

IMPORTANT NOTE: The BLACK jumper is the GROUND (-), so it is ALWAYS plugged in the GDN PIN of the Arduino board and the RED jumper is (+)

You need to determine whether you are in presence of Digital or Analog I/O so you place your jumpers in the correct pin.

In our example we will plug the Black jumper in the PIN GDN of the Digital

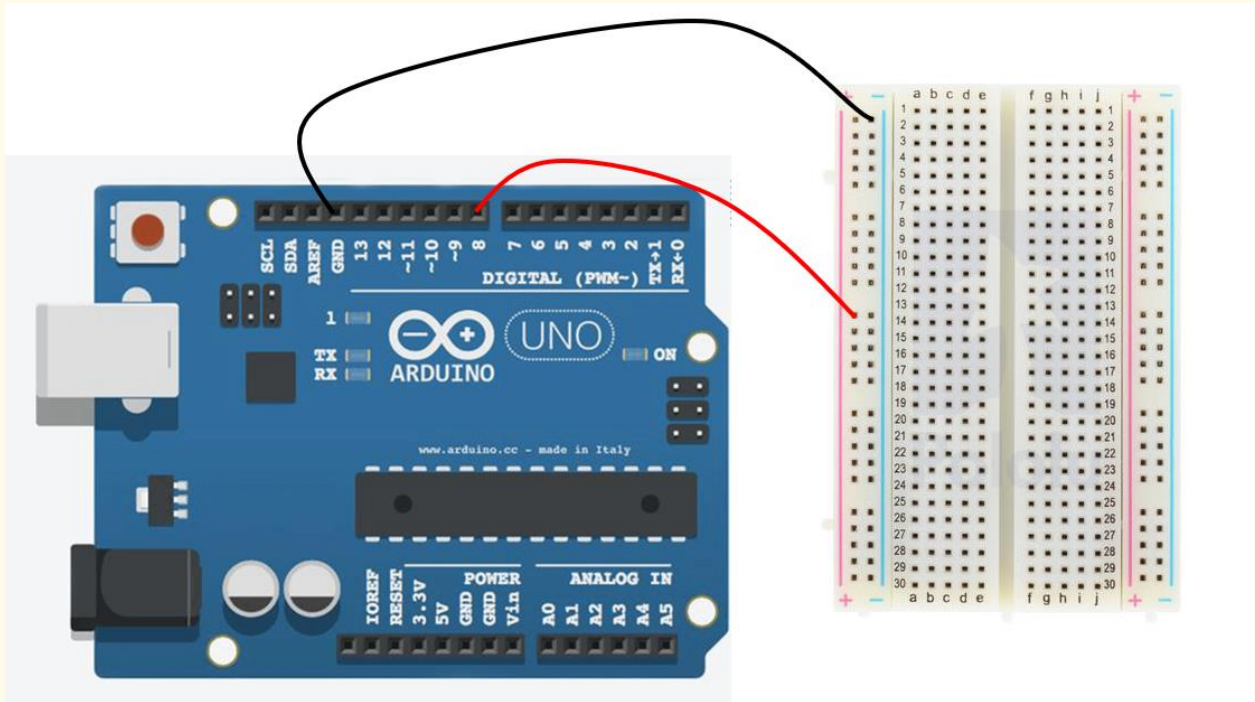
And the Red jumper in the PIN 9 of Digital.

Jumpers on the breadboard:

The Black (-) jumper going from GDN pin to the Minus (Blue) column

The Red (+) jumper going from 8 Digital PIN to the Plus (Red) column

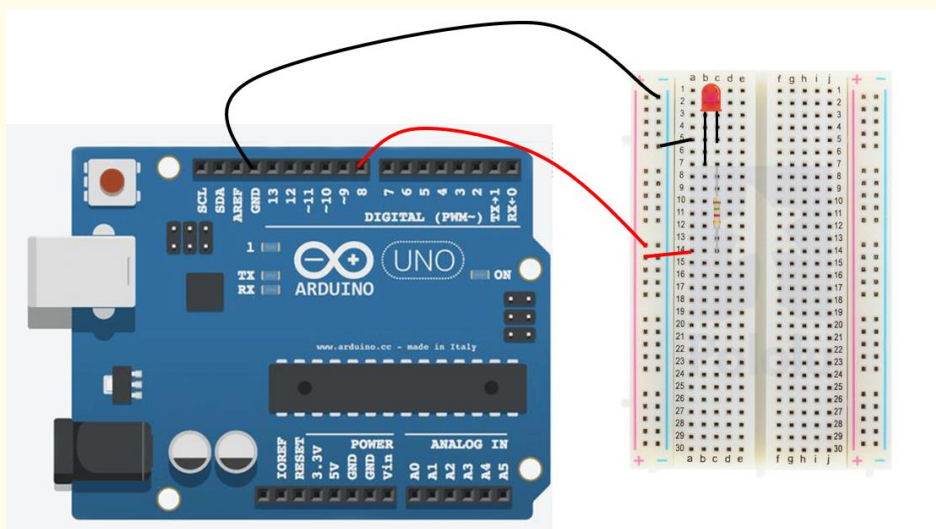
See below:



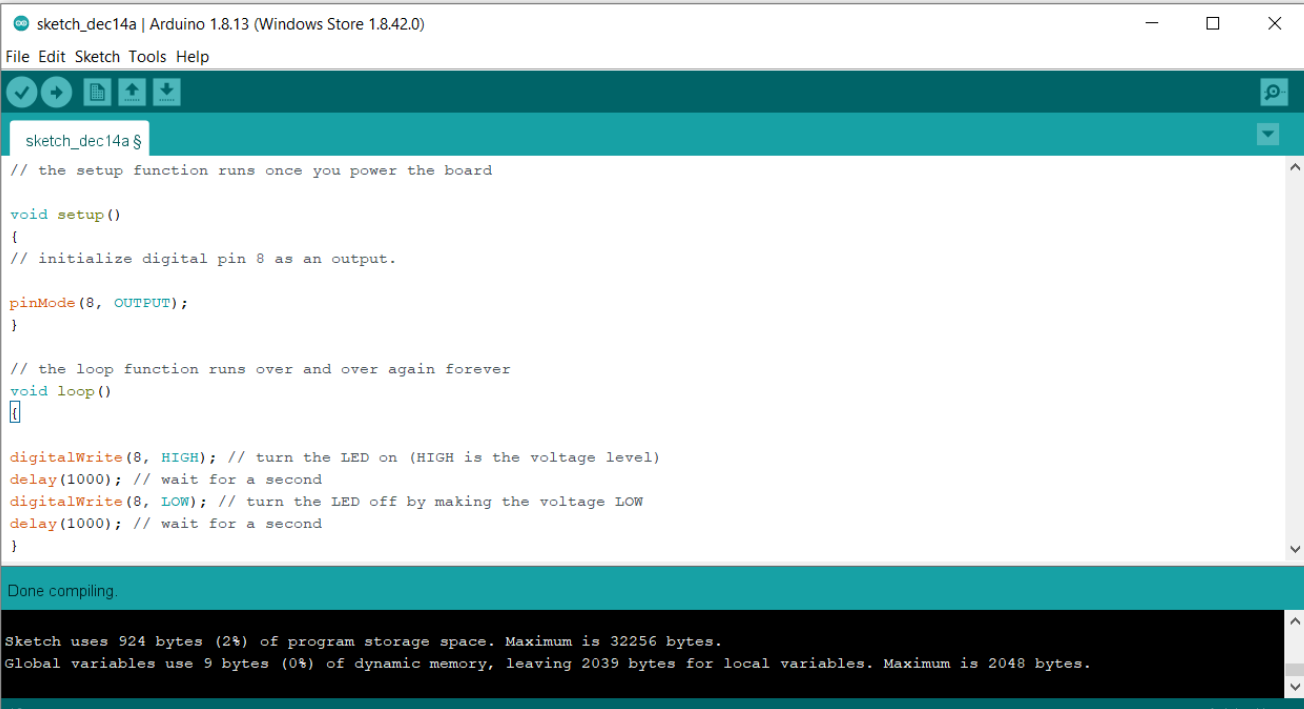
IMPORTANT NOTE:

To have power running in your circuit, and to be able to set up the program, your Arduino board should be plugged with the USB cable to your laptop/computer.

- ARDUINO BREADBOARD CONNECTIONS



Arduino code:



```

sketch_dec14a | Arduino 1.8.13 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help
sketch_dec14a $
// the setup function runs once you power the board

void setup()
{
  // initialize digital pin 8 as an output.

  pinMode(8, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(8, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(8, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}

Done compiling.

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

18 Arduino Uno

```

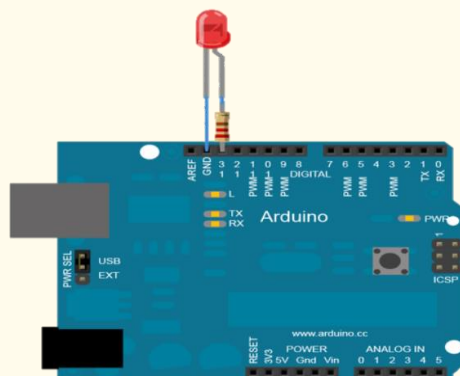
Additional notes:

pinMode(2, OUTPUT) – Before you can use one of Arduino’s pins, you need to tell Arduino Uno R3 whether it is an INPUT or OUTPUT. We use a built-in “function” called pinMode() to do this.

digitalWrite(2, HIGH) – When you are using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts), or LOW (output 0 volts).

To have a blinking LED, we could also have attached the LED to the PIN of the Arduino without using a breadboard.

See example below:



Source: <https://www.instructables.com/How-to-Blink-LED-Using-Arduino/>

- Bibliographic references:

<https://www.instructables.com/How-to-Blink-LED-Using-Arduino/>

https://www.tutorialspoint.com/arduino/arduino_blinking_led.htm

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink>

<https://www.hackerearth.com/blog/developers/a-tour-of-the-arduino-uno-board/>

<https://docs.allthingstalk.com/examples/kits/arduino-rdk/>



The European Commission support for the production of this document does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Co-funded by the
Erasmus+ Programme
of the European Union

LESSON – ANALOG OUTPUT: FADING LED WITH ARDUINO

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Build a circuit to have a fading LED (breadboard & Arduino)
 - Program Arduino to change the intensity of the LED
 - The components
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● How to install the circuit ● How to program the Arduino ● The analogWrite() function ● The concept of PWM ● Program a fading LED on an Arduino 	<ul style="list-style-type: none"> ● Know the components needed ● Program/code the outputs ● Learn about how to use PWM 	<ul style="list-style-type: none"> ● Be capable of building a fading LED circuit on a breadboard ● Code a fading LED program

- **Main Keyword(s):**
 - Breadboard
 - Components: jumpers, LED, resistors, LEDs...
 - analogWrite() function
 - Pulse Width Modulation (PWM)
 - Digital output
- **Required material and resources:**
 - Arduino Uno R3
 - Arduino IDE software
 - Computer/Laptop
 - Connection cable to connect Arduino and Laptop
 - Breadboard Arduino Compatible (BBAC)
 - A LED
 - 1 Resistor (220ohm)
 - 2 Jumpers



Definition: Pulse Width Modulation (PWM)

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means.

The Fading example demonstrates the use of analog output (PWM)

AnalogWrite uses pulse width modulation (PWM), turning a digital pin on and off very quickly with different ratio between on and off, to create a fading effect.

Objective of the exercise

The facilitator will start the lesson by explaining what the objective of the lesson is:

In this lesson we will see how to make an LED fade. In order for an LED to fade, it has to change from 0% OFF (0 Volt) to 100% ON (5 Volt) very quickly so the human eye sees a nuance of intensity.

To do this we will use the `analogWrite()` function. `AnalogWrite` uses pulse width modulation (PWM).

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means.

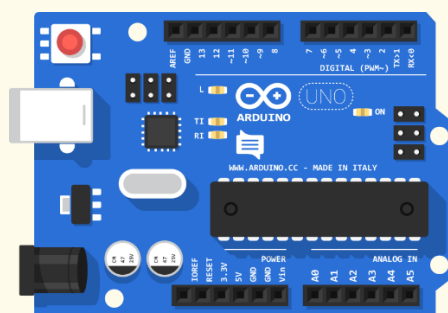
The PWM pins

The Arduino board is composed of several pins.

There are Analog and Digital Pins, and among the digital pins, some are what we call with PWM, that is to say PULSE WIDTH MODULATION.

The pins with PWM can be identified with the “~” sign like ~3, ~5, ~6, ~9, ~10 and ~11.

See on the image below:



Source: <https://docs.allthingstalk.com/examples/kits/arduino-rdk/>

To know more about the PINS and Arduino board, check the lessons about “Introduction to microcontrollers”

Present the components and circuit

The components used for fading a LED are the same as for blinking a LED. Please refer to the previous lesson to have more details about the components.

- THE LED and the RESISTOR (see lesson about Blinking LED)
- ARDUINO BOARD – BREADBORD AND JUMPERS (see lesson about Blinking LED)
- ARDUINO BREADBOARD CONNECTIONS

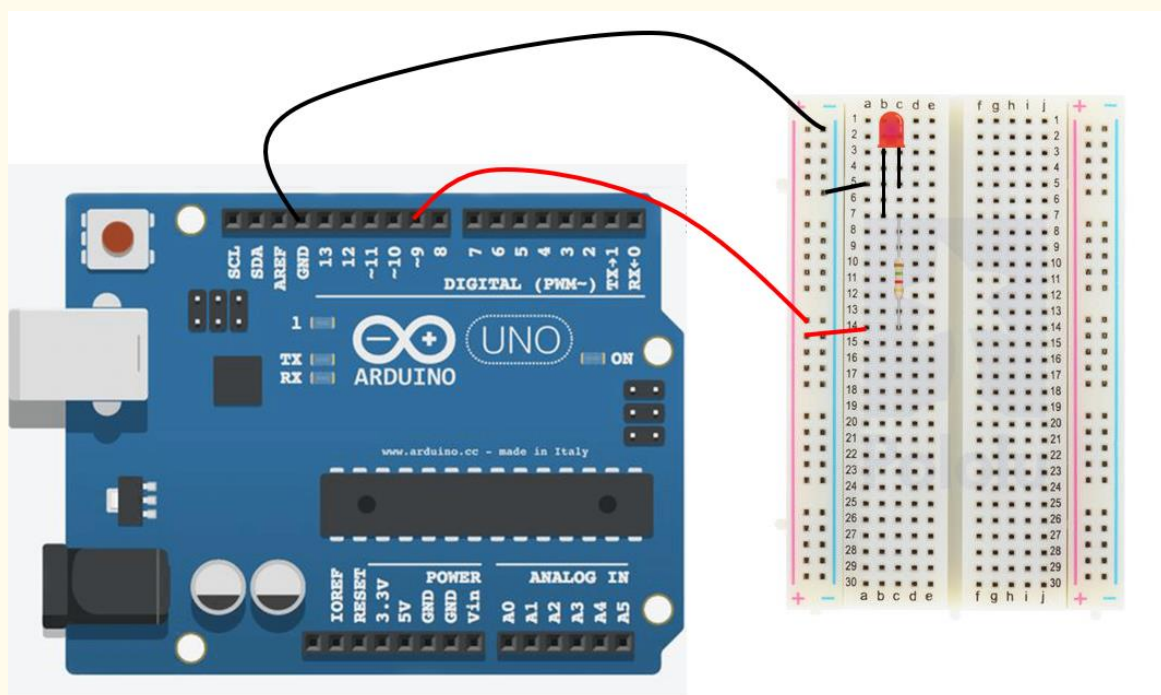
Unlike the circuit for blinking LED, for the fading LED, the output will be in PIN 9 because it is with PWM.

Connect the positive leg (the long one) of the LED to digital output pin 9 on your board through a 220 ohm resistor. Connect the negative leg (the shorter) directly to ground.

+LEG of the LED in 7B with resistor in 7C connected to the ~9 PIN

– LEG of the LED in 5C connected to the Ground.

See the circuit below:



Arduino code:

This example shows how to fade an LED on pin 9 using the `analogWrite()` function.

The `analogWrite()` function uses PWM, so if you want to change the pin you're using, be sure to use another PWM capable pin. On most Arduino, the PWM pins are identified with a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

This example code is in the public domain:

<http://www.arduino.cc/en/Tutorial/Fade>

```

Fade §
|
int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;   // how bright the LED is
int fadeAmount = 5;   // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}

Done compiling.

Sketch uses 1144 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 13 bytes (0%) of dynamic memory, leaving 2035 bytes for local variables. Maximum is 2048 bytes.

1 Arduino Uno

```

- Bibliographic references:

<https://www.arduino.cc/en/Tutorial/Foundations/PWM>

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/Fade>

<https://create.arduino.cc/projecthub/glowascii/fade-an-led-arduino-basics-eebff7>

<https://www.instructables.com/How-to-Fade-an-LED-Arduino-Tutorial/>



The European Commission support for the production of this document does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



LESSON – DIGITAL INPUT: READING BUTTONS

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Describe inputs in Arduino
 - Understand Digital Inputs
 - Program Arduino to read a button's value
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● Recognize a Digital Input ● Read a switch's on/off state ● Use digitalRead function 	<ul style="list-style-type: none"> ● Program and configure digital pins ● Blink an LED by closing a switch 	<ul style="list-style-type: none"> ● Solve problems autonomously

- **Main Keyword(s):**
 - Digital Inputs
 - Arduino Programming
 - LED and Buttons
- **Required material and resources:**
 - Arduino UNO
 - LED Red
 - Resistor 1K and 10K
 - A momentary switch, button, or toggle switch
 - Hook-up wires
 - Breadboard
 - Computer
 - Arduino USB cable
 - Software Arduino IDE
 - Internet Connection



Brief definition of Digital Input

The digital inputs are signals or external values sent to a system having two states: HIGH or LOW.

Digital signals are used in situations where the input or output will have one of those two values. For example, a digital signal is used to turn an LED on or off by reading a button's value.

Digital Inputs

The facilitator will start the lesson by explaining to the participants what inputs are in Arduino and, more precisely, digital inputs. The facilitator begins as follows:

Inputs can be signals or external values sent to a system. For instance, the keyboard is an input for the computer because it sends data to the system. Another example of an input is the microphone that captures audio and sends it to a computer. On Arduino, by default, all the pins are already pre-configured as input. Unlike analog inputs, which may take on any value within a range of values, digital signals have two states: 0 or 1.

Digital signals are used in situations where the input or output will have one of those two values. For example, a digital signal is used to turn an LED on or off by reading a button's value. Buttons connect two points in a circuit when you press them, as shown in Figure 1. When the button/switch is unpressed, there is no connection between the button's two legs, so the pin is connected to 5 volts (through the pull-up resistor), and we read a HIGH. When the button/switch is closed/pressed, it makes a connection between its two legs, connecting the pin to the ground, so that we read a LOW.

In Arduino programming, the function `digitalRead()` is used to read a value from a digital pin, which can only be either HIGH (1) or LOW (0).

Syntax:

`digitalRead(pin)`, where "pin" is the number of the digital pin you want to read (int)

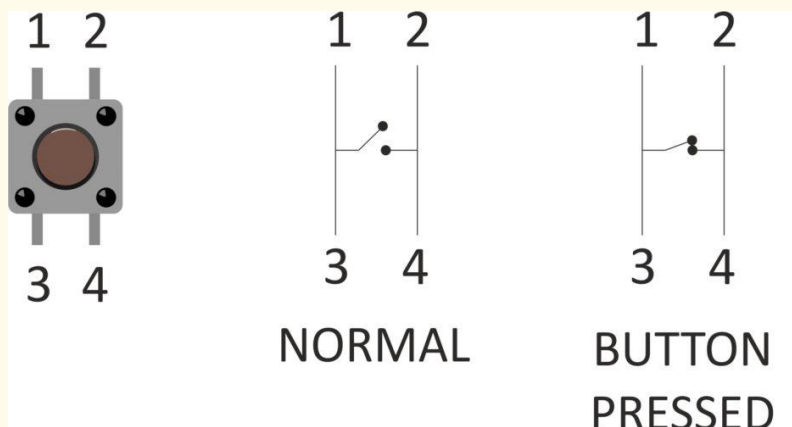


Figure 1 Buttons Diagram (Source: <https://roboindia.com/>)

Program Arduino to turn On and Off a LED using a button

The facilitator will guide the participants with step-by-step instructions on making the connections on the breadboard and then explaining the Arduino code to them.

Step-by-step Instructions:

- Connect an Arduino GND pin to one of the long power rails on the breadboard
- Connect the short leg of the LED to the same ground rail on the breadboard and connect the long leg to a different row on the breadboard.
- Connect the 1K resistor, as shown in Figure 2, in the same row where the LED's long leg is attached.
- Place the pushbutton on the breadboard.
- Connect a jumper wire from the 5-volt pin to one side of the pushbutton and a jumper wire from pin 9 to the other side of the pushbutton.
- Connect one side of the 10k ohm resistor to the ground rail on the breadboard and the other side to the pushbutton in row 24 (as shown in Figure 2).
- Plug the Arduino board into your computer with a USB cable.
- Open the Arduino IDE.
- The code can be found in Figure 3.
- Click the Verify button on the top left. It should turn orange and then back to blue.



- Click the Upload button. Once the uploading has finished, it will turn blue.



- Press the button and watch how the LED reacts.
- Arduino breadboard connections:

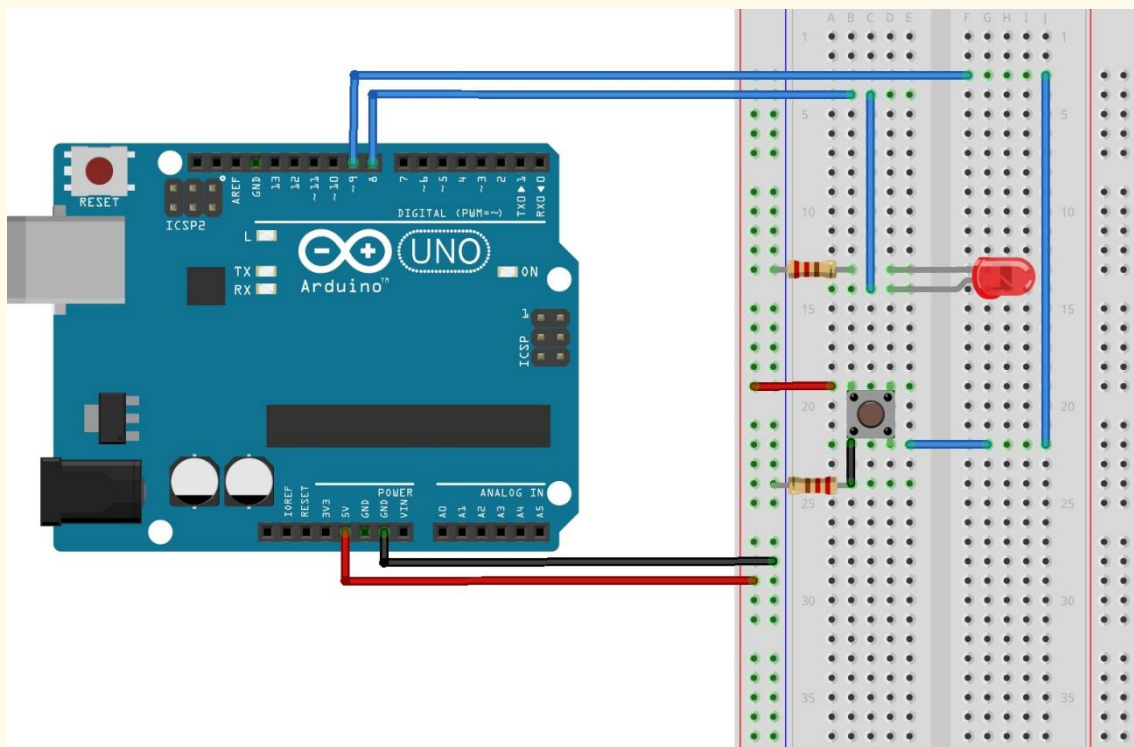



Figure 2 Turn On and Off a LED using a button (Source: <https://www.allaboutcircuits.com/>)

- Arduino Code:



```

DigitalInput | Arduino 1.8.13
File Edit Sketch Tools Help
DigitalInput
const int BUTTON = 9; // Pushbutton Input to Pin No.9
const int LED = 8; // Output LED to Pin No. 8

int BUTTONState = 0; // To store input status

void setup() {
  pinMode(LED, OUTPUT); // Define LED pin as output.
  pinMode(BUTTON, INPUT); // Define BUTTON pin as Input.
}

void loop(){
  BUTTONState = digitalRead(BUTTON); // Reading input from Button Pin.

  if (BUTTONState == HIGH) // Checking if Input from button is HIGH (1/+5V)
  {
    digitalWrite(LED, HIGH); // If input is High make LED ON (HIGH/1/+5V)
  }
  else
  {
    digitalWrite(LED, LOW); // For every other condition make LED OFF (0/GND/LOW)
  }
}

Done Saving.
1 Arduino Uno

```

Figure 3 Arduino code to turn off and on an LED using push button

- Bibliographic references:

<https://roboindia.com/tutorials/digital-input-how-to-use-the-button-with-arduino/>

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/DigitalReadSerial>

<https://www.tutorialspoint.com/arduino/index.htm>

Setting up a circuit with a push button and a LED:

<https://www.youtube.com/watch?v=SYMe1wWthlw>

Circuits with multiple push buttons: <https://www.youtube.com/watch?v=kzImojF3tBo>

<https://www.youtube.com/watch?v=OCJabRA3m4U>

Timed LED with a button: <https://www.youtube.com/watch?v=oPDcVnPfjdk>

LESSON – ANALOG INPUT: POTENTIOMETER

- **Study section:** Microcontroller and Arduino Fundamentals
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Describe inputs in Arduino
 - Understand Analog Inputs
 - Program Arduino to read a potentiometer's values
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● Recognize an Analog Input ● Read the value of a potentiometer ● Use pinMode(), delay(), map(), digitalWrite(), analogWrite() and analogRead() functions 	<ul style="list-style-type: none"> ● Program and configure analog pins ● Change a LED brightness using the potentiometer 	<ul style="list-style-type: none"> ● Solve problems autonomously

- **Main Keyword(s):**
 - Analog Inputs
 - Arduino Programming
 - LED and Potentiometer
- **Required material and resources:**
 - Arduino UNO
 - LED Red
 - Resistor 220 ohm
 - Potentiometer
 - Hook-up wires
 - BreadBoard
 - Computer
 - Arduino USB cable
 - Software Arduino IDE



Brief definition of Analog Input

Analog inputs are signals or external values that convert a voltage level into a digital value that can be stored and processed in a computer.

Analog Inputs

The facilitator will start the lesson by explaining to the participants what analog inputs are. The facilitator begins as follows:

An analog input converts a voltage level into a digital value stored and processed in a computer. In Arduino, the function `analogRead()` is used to read a value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. That means that it will map input voltages between 0 and 5 Volts into integer values between 0 and 1023.

A potentiometer is an example of analog input (Figure 1). It consists of a simple knob that provides a variable resistance, which it can be read into the Arduino board as an analog value. If we use the LED example, the analog value coming from the potentiometer controls the level of the LED's brightness.

By turning the potentiometer's shaft, we change the amount of resistance on either side of the wiper connected to the potentiometer's center pin. This changes the relative "closeness" of that pin to 5 volts and ground, giving us a different analog input. When the shaft is turned in one direction, 0 volts go to the pin, and we read 0. When the shaft is rotated in the other direction, 5 volts go to the pin, and we read 1023. In between, `analogRead()` returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

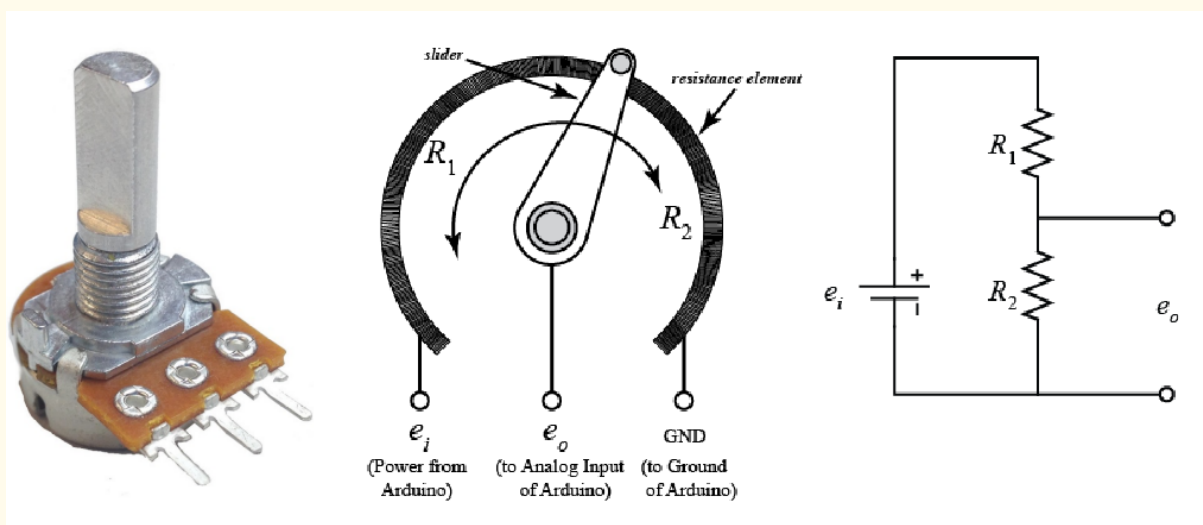


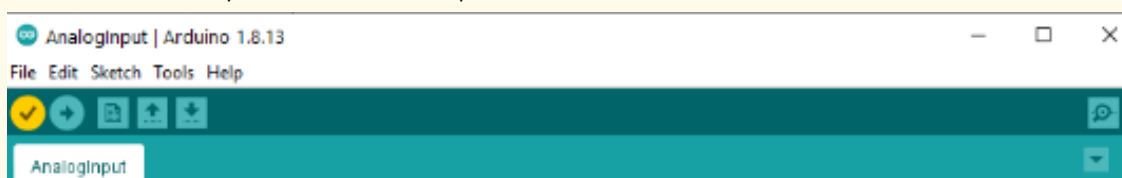
Figure 1 Potentiometer diagram (Source: <https://hackerfarm.jp/tutorials/intro-to-arduino-for-indoor-grow-lights/>)

Use a potentiometer to change the LED's brightness

The facilitator will guide the participants with step-by-step instructions on making the connections on the breadboard and then explaining the Arduino code to them.

Step-by-step Instructions:

- The potentiometer has three terminals. Connect terminal 1 to the ground and terminal 2 to the 5V of the Arduino card, as shown in Figure 2.
- Connect terminal 3 (the one in the middle) to the Analog Input (A0) in the Arduino card. This is where we get a value between 0 and 5 Volts.
- Place the one end of the 220-ohm resistor into pin 9 and the LED's short leg into the ground pin (GND).
- Connect the long leg of the LED to the other end of the resistor.
-
- Use the USB cable to plug the Arduino into your computer.
- Open the Arduino IDE and copy the code given in Figure 4.
- Click the Verify button on the top left side of the screen.



- Then click the Upload button. It will turn orange and then back to blue once it has finished.



- As you adjust the knob of the potentiometer, the brightness of the LED should vary.

- Arduino breadboard connections:

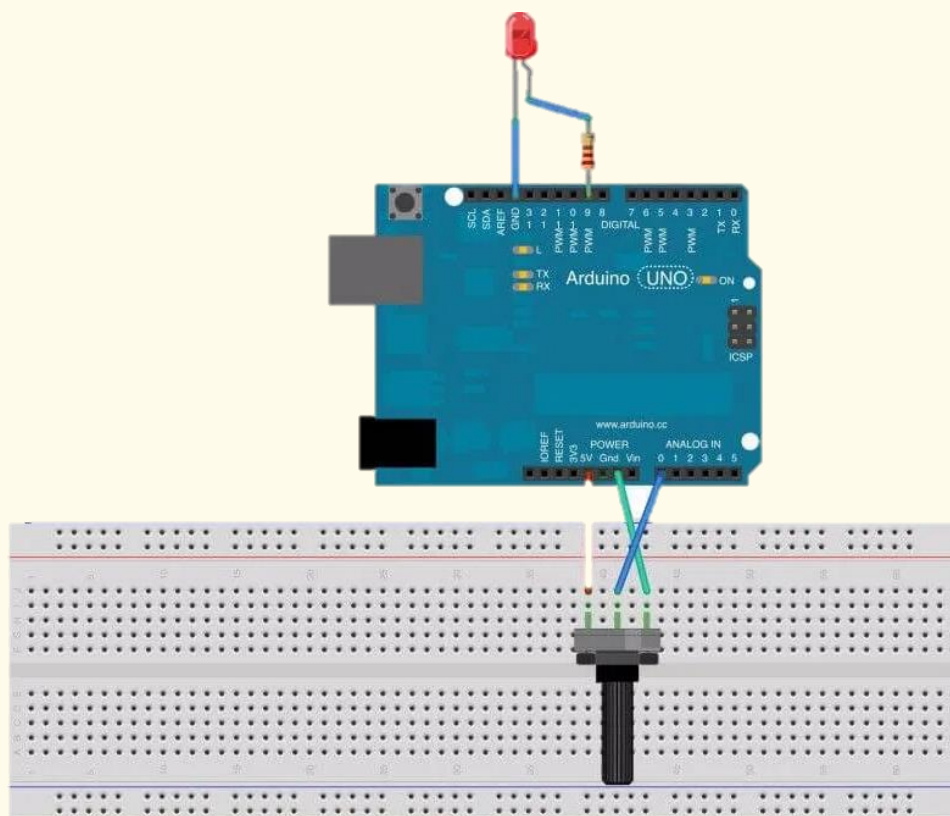


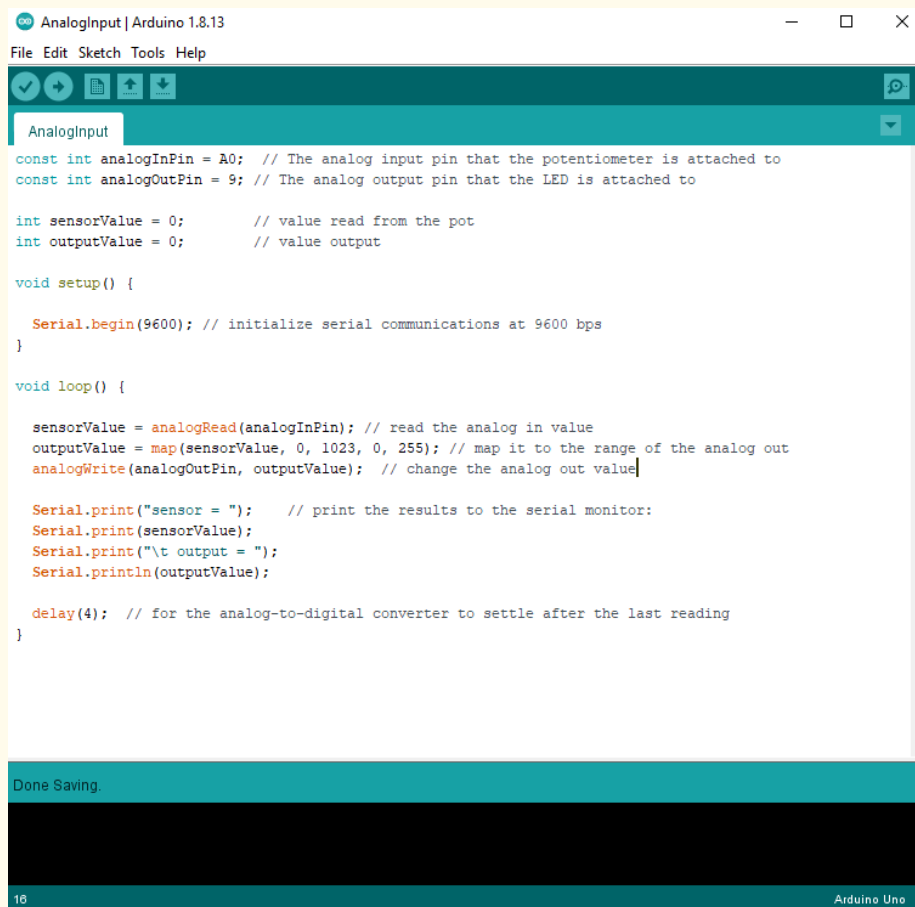
Figure 2 Reading Analog inputs in Arduino using a Potentiometer (Source:<https://www.arduino.cc/>)

- Arduino Code:

At the end the facilitator will explain the code in brief:

- Read analog value from potentiometer
sensorValue=analogRead(analogInPin)
- Map analog values 0-1024 to pwm values 0-255
outputValue = map(sensorValue, 0, 1023, 0, 255);

- Change the analog out value `analogWrite(analogOutPin, outputValue);`



```

AnalogInput
const int analogInPin = A0; // The analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // The analog output pin that the LED is attached to

int sensorValue = 0; // value read from the pot
int outputValue = 0; // value output

void setup() {
  Serial.begin(9600); // initialize serial communications at 9600 bps
}

void loop() {

  sensorValue = analogRead(analogInPin); // read the analog in value
  outputValue = map(sensorValue, 0, 1023, 0, 255); // map it to the range of the analog out
  analogWrite(analogOutPin, outputValue); // change the analog out value

  Serial.print("sensor = "); // print the results to the serial monitor:
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  delay(4); // for the analog-to-digital converter to settle after the last reading
}

```

Done Saving.

16 Arduino Uno

Figure 2 Arduino code that change the brightness of a LED using a potentiometer

- Bibliographic references:

<https://www.programmingelectronics.com/tutorial-21-analog-input-old-version/>

<https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogInput>

<https://www.instructables.com/How-to-use-Potentiometer-Arduino-Tutorial/>

Schwartz, M. (2014). Arduino Networking

WIFI FUNDAMENTALS AND IOT

LESSON – WIRING THE ESP8266

- **Study section:** Wi-Fi Fundamentals and IoT
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Describe ESP8266
 - Understand ESP8266 wiring
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Recognize an ESP8266 • Wire an ESP8266 	<ul style="list-style-type: none"> • Wiring different modules to Arduino UNO 	<ul style="list-style-type: none"> • Solve problems autonomously

- **Main Keyword(s):**
 - ESP8266
 - Wi-Fi network
 - Microcontroller units (MCUs)
- **Required material and resources:**
 - An ESP8266 board
 - Arduino UNO
 - A computer that can run the Arduino IDE
 - A USB-to-Serial converter (it is essential to use a 3.3V model)
 - A USB cable
 - A 3.3V power supply or voltage regulator
 - A Wi-Fi network
 - Breadboard
 - Computer
 - Arduino USB cable
 - Software Arduino IDE
 - Internet Connection



Brief definition of ESP8266

The ESP8266 module is a low-cost Wi-Fi microchip that enables microcontrollers to connect to Wi-Fi, using the IEEE 802.11 protocol.

ESP8266

The facilitator will start the lesson by explaining to the participants what the ESP8266 module is and connecting it to Arduino. The facilitator begins as follow:

The ESP8266 module is a low-cost Wi-Fi microchip that enables microcontrollers to connect to Wi-Fi, using the IEEE 802.11 protocol. It can provide Wi-Fi connectivity to external host microcontroller units (MCUs) or be used as a self-sufficient MCU. The module has a full TCP/IP stack and provides data processing, reads, and controls of General Purpose Input/OGPIOs. ESP8266 Arduino core comes with libraries to communicate over Wi-Fi using TCP and UDP, use a file system in flash memory, work with SD cards, and I2C peripherals. It can be used in several applications such as smart power plugs, home automation, industrial wireless control, IP Cameras, sensor networks, wearable electronics, Wi-Fi location-aware devices, and security ID tags.

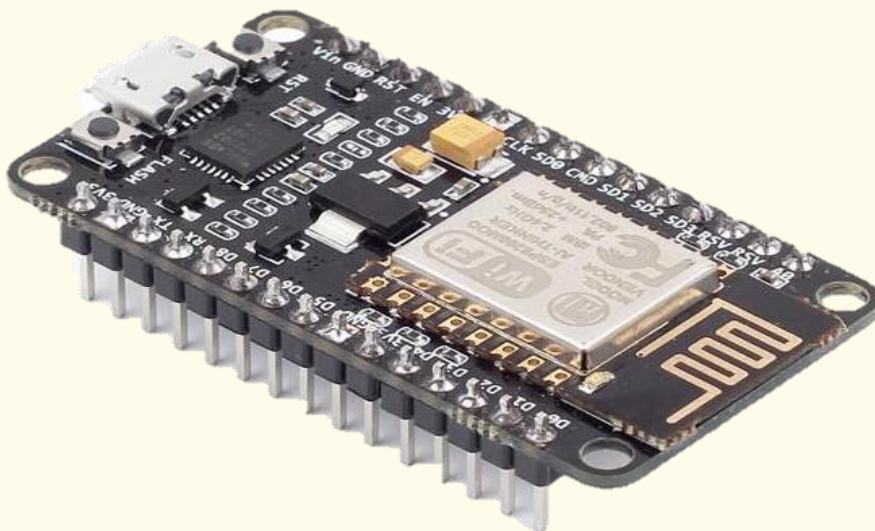


Figure 1 ESP8266 WiFi based microcontroller board (Source: www.antratek.com)

Connect ESP8266 with Arduino

The facilitator will guide the participants with step-by-step instructions on connecting the ESP8266 module with Arduino Uno.

- **Requirements to program ESP8266 with Arduino IDE:**
 - An ESP8266 board
 - A computer that can run the Arduino IDE
 - A USB-to-Serial converter (it is essential to use a 3.3V model)
 - A USB cable
 - A 3.3V power supply or voltage regulator
 - A Wi-Fi network

- **Steps to connect the ESP8266 to Arduino:**
 - Power the ESP8266 with the 3.3V power supply. You can also power ESP8266 using two AA batteries as shown in Figure 2. Positive from batteries to VCC of ESP and GND to GND of ESP 8266.

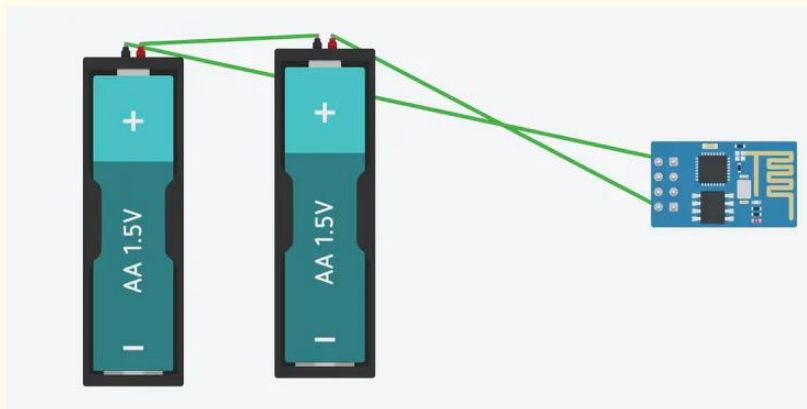


Figure 23 Powering ESP8266 with 2 AA batteries (Source <https://www.hackster.io/PatelDarshil/things-you-should-know-before-using-esp8266-wifi-module-784001>).

- Connect the Arduino to a breadboard.
- Connect the Arduino's 3.3V output to the red line of a breadboard (The ESP8266 needs 3.3V. If you connect it to a 5V power supply, you'll destroy it).
- Connect the ground (GND) to the blue line.
- Connect the RES or RESET pin to the blue line. When you ground the reset pin, the Arduino works as a USB to a serial connector, which we want to talk to the ESP8266.
- Connect the RXD pin of the Arduino to the RX pin of the ESP8266.
- Connect the TXD pin of the Arduino to the TX pin of the ESP8266.
- Connect the GND pin of the ESP to the blue line and the VCC pin to the red line.
- Connect the CH_PD to the red line.

- Arduino breadboard connections:

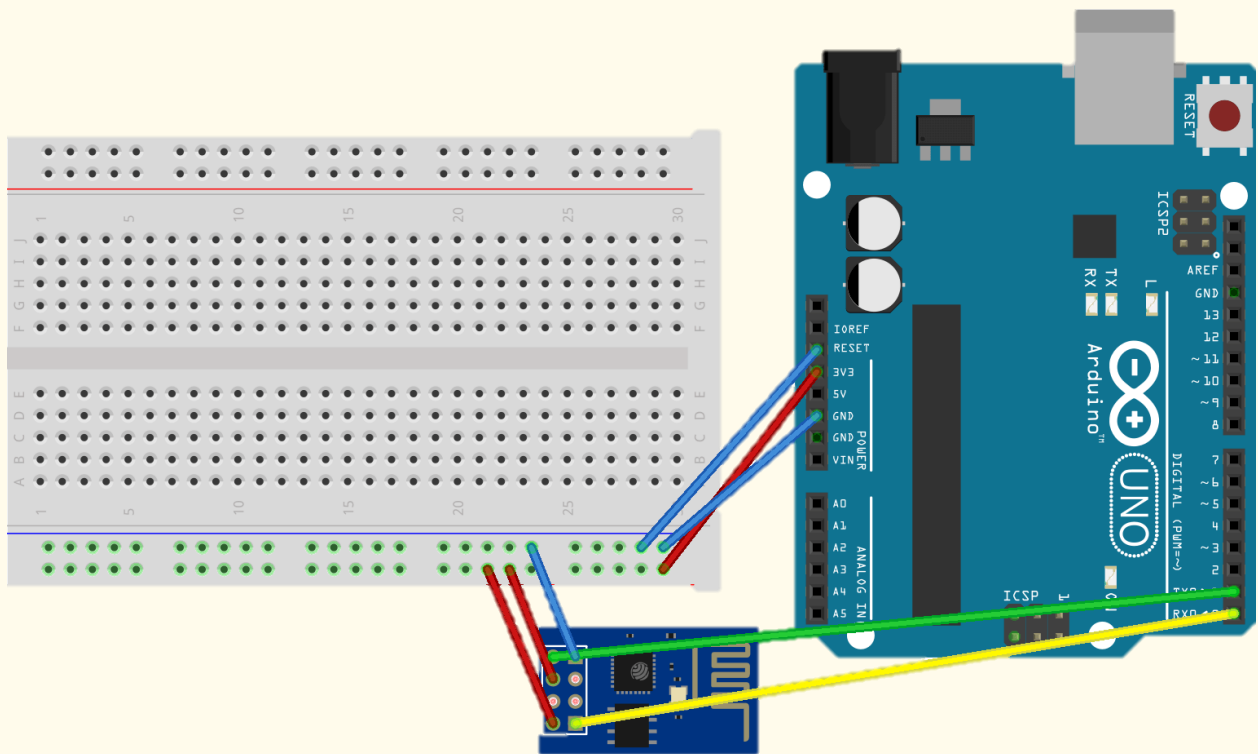


Figure 24 Wiring ESP8266 to Arduino UNO (Source: <http://www.teomaragakis.com/hardware/electronics/how-to-connect-an-esp8266-to-an-arduino-uno/>)

- Bibliographic references:

<http://www.teomaragakis.com/hardware/electronics/how-to-connect-an-esp8266-to-an-arduino-uno/>

<http://dexterous-programmer.blogspot.com/2018/09/arduino-uno-wifi-module-esp8266.html>

https://la.mathworks.com/help/supportpkg/arduino/ug/connect-esp8266-to-arduino-hardware.html#mw_d1735291-00d0-410d-a9a5-3beb76a24286

<https://create.arduino.cc/projecthub/turaib/esp8266-arduino-communication-1fdd40>

LESSON – WIFI OVERVIEW: ESP8266 CAPABILITIES

- **Study section:** Wi-Fi Fundamentals and IoT
- **Lesson duration:** 2:00h
- **Educational objectives:**
 - Describe TCP/IP stack
 - Understand layers and protocols
 - Understand ESP8266 capabilities
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> • Understand the difference between layer and protocol • Understand the difference between an access point and a station and between a client and a server. • Explain the ESP8266 capabilities 	<ul style="list-style-type: none"> • Determine the capabilities of an ESP8266 	<ul style="list-style-type: none"> • Communicate concepts and ideas clearly

- **Main Keyword(s):**
 - ESP8266
 - Wi-Fi network
 - TCP/IP
 - Layers and protocols
 - Access point and station
 - Client and server
- **Required material and resources:**
 - An ESP8266 board

Brief definition of ESP8266 capabilities

ESP8266 works in two modes: Station and Access Point (AP). AP mode allows it to create its own network and have other devices connect to it. Station mode allows the ESP8266 to connect to a Wi-Fi network .

ESP8266 can function as a client, as an access point or both.

ESP8266

The facilitator will start the lesson by explaining to the participants what the ESP8266 module is and connecting it to Arduino. The facilitator begins as follow:

As previously mentioned, the reason why most people use ESP8266 is its Wi-Fi capabilities. To understand how ESP8266 communicates with other devices on the network, we need to understand the different layers and protocols (Table 1).

Layer	Protocol
Application	HTTP,FTP, etc.
Transport	TCP, UDP
Internet	IP
Link	Ethernet, Wi-Fi

Table 1 The TCP/IP stack

The link layer is the physical link between two devices, an Ethernet cable or a Wi-Fi connection. To connect an ESP8266 to the network, you have to physically connect the ESP8266 to other devices either through actual wires (Ethernet) or through Wi-Fi. The Internet Protocol (IP) involves giving each device an IP address to make it possible that the devices can send messages and "talk" to each other. Nevertheless, IP can't guarantee that the messages (packets) arrive in the same order they were sent in. This means that we can't reliably send messages by only using the link and the Internet layer. So the Transport layer is needed.

The Transmission Control Protocol (TCP) makes sure that all packets are received, that the packets are in order, and corrupted packets are re-sent, and the User Datagram Protocol (UDP) only checks for errors (faster with lower latency).

The application layer of protocols is the one that "translates" the messages for two programs to understand each other.

- **ESP8266 Capabilities:**
 - Station mode: Station mode is used to get the ESP8266 module connected to a Wi-Fi network established by an access point (router). As shown in Figure 1, the

router acts as an access point, and the ESP8266 is set as a station. To control the ESP8266, we need to be connected to the router.

- o An access point (AP) is a device that provides access to a Wi-Fi network to other devices (stations) and connects them further to a wired network. ESP8266 can provide similar functionality, except it does not have an interface to a wired network.

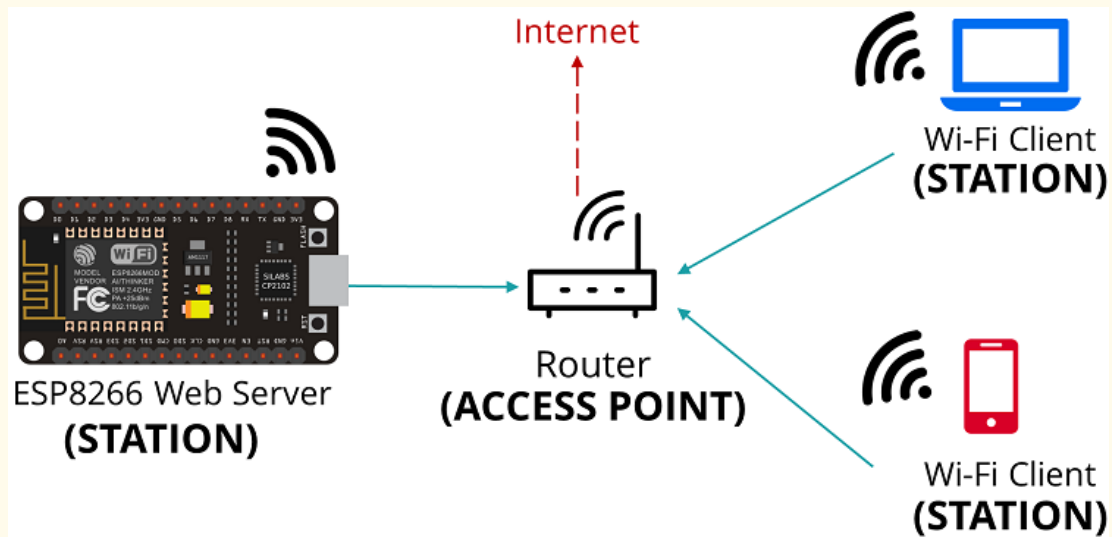


Figure 5 ESP8266 as a Station (Source: <https://randomnerdtutorials.com/esp8266-nodemcu-access-point-ap-web-server/>)

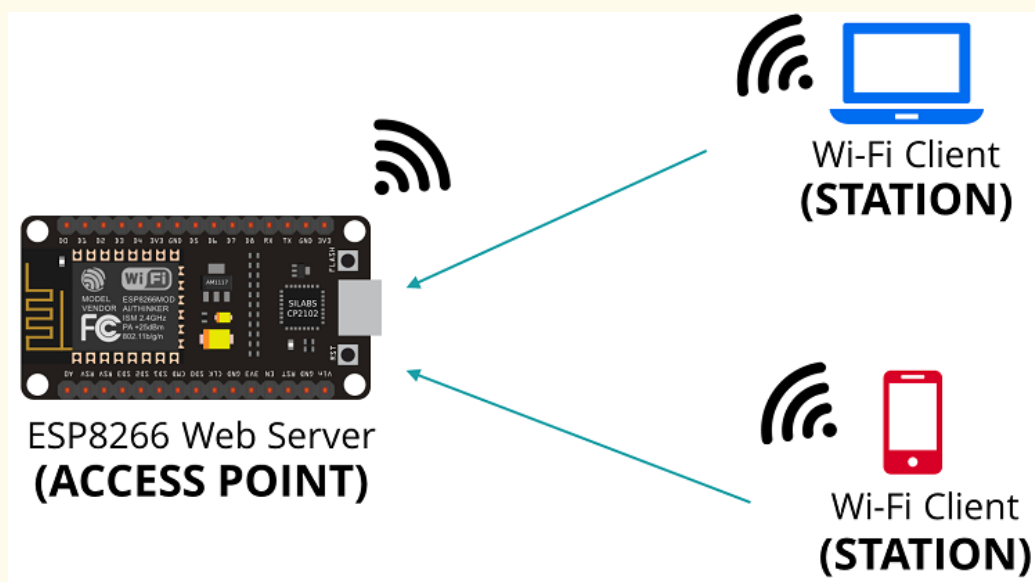


Figure 6 ESP8266 as an Access Point (Source: <https://randomnerdtutorials.com/esp8266-nodemcu-access-point-ap-web-server/>)

- Bibliographic references:

[http://onlineshouter.com/use-esp8266-wifi-modes-station-access-point/#:~:text=An%20access%20point%20\(AP\)%20is,interface%20to%20a%20wired%20network.&text=The%20maximum%20number%20of%20stations,the%20soft%20DAP%20is%20five.](http://onlineshouter.com/use-esp8266-wifi-modes-station-access-point/#:~:text=An%20access%20point%20(AP)%20is,interface%20to%20a%20wired%20network.&text=The%20maximum%20number%20of%20stations,the%20soft%20DAP%20is%20five.)

<https://medium.com/concepts-for-dummies/client-vs-server-terminology-b18355d1f9ff>

<http://www.teomaragakis.com/hardware/electronics/how-to-connect-an-esp8266-to-an-arduino-uno/>

<http://dexterous-programmer.blogspot.com/2018/09/arduino-uno-wifi-module-esp8266.html>

https://la.mathworks.com/help/supportpkg/arduino/ug/connect-esp8266-to-arduino-hardware.html#mw_d1735291-00d0-410d-a9a5-3beb76a24286

<https://create.arduino.cc/projecthub/turaib/esp8266-arduino-communication-1fdd40>

LESSON – WIFI CONNECTION WITH ESP8266

- **Study section:** WiFi Fundamentals and IoT
- **Lesson duration:** 3:00h
- **Educational objectives:**
 - Describe the usefulness of ESP8266
 - Configure the ESP8266 to access to internet
 - Build the Web Server with ESP8266
- **Main Keyword(s):**
 - ESP8266
 - Wifi
 - Web Server
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● Characterize ESP 8266 ● Recognize the hardware needed to build a webserver using ESP8266 	<ul style="list-style-type: none"> ● Determine the hardware to be used in different configurations with the ESP8266 ● Program and configure Internet access using ESP8266 	<ul style="list-style-type: none"> ● Plan and structure tasks ● Communicate concepts and ideas clearly ● Act with initiative and demonstrate analytical capacity ● Recommend solutions for problem-solving ● Demonstrate creativity, autonomy and innovative spirit

- **Required material and resources:**
 - ESP8266
 - Cable USB to microUSB
 - LED Green
 - LED Red
 - BreadBoard
 - Computer
 - Software Arduino IDE
 - Internet Connection



Brief definition of ESP8266

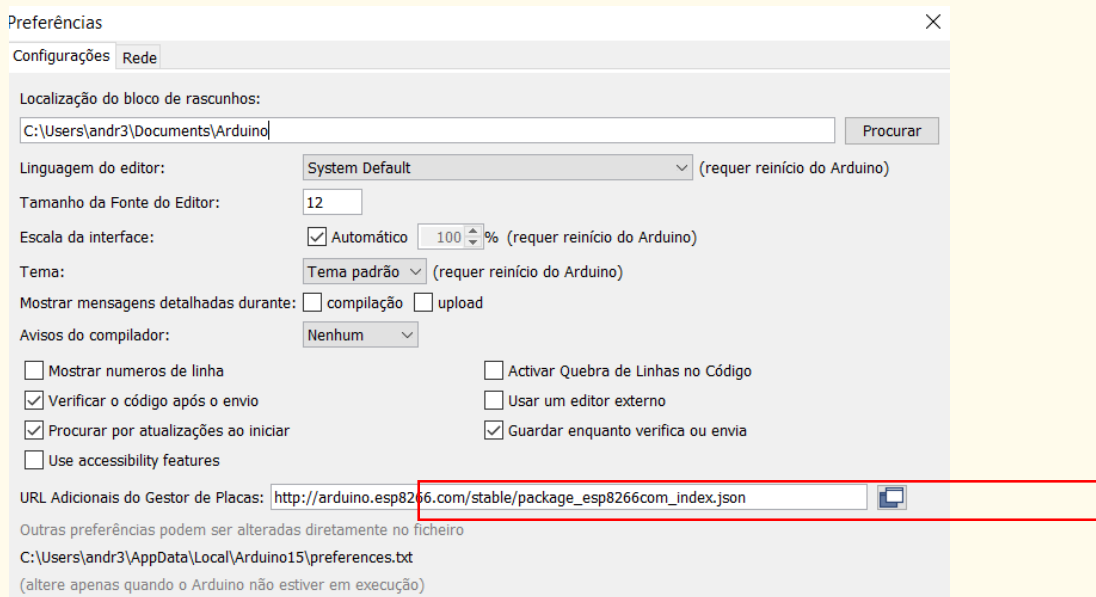
The ESP8266 is a low-cost Wi-Fi chip.

This chip revolutionized wifi connections in the development of small circuits due to its low cost, allowing for a quick dissemination. What caught our attention is that it has WiFi enabling the connection of several devices to the internet, so it can be easily configured to work as for example a wireless switch, among many other things.

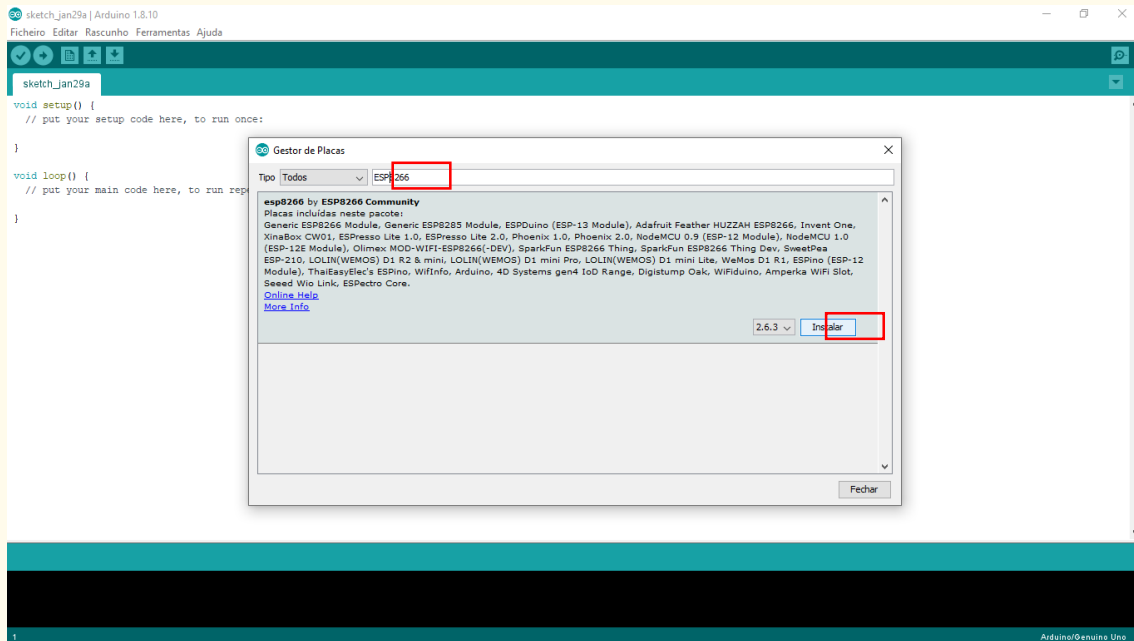
Programming ESP8266

Regarding the installation of nodemcu board, you must follow the following steps:

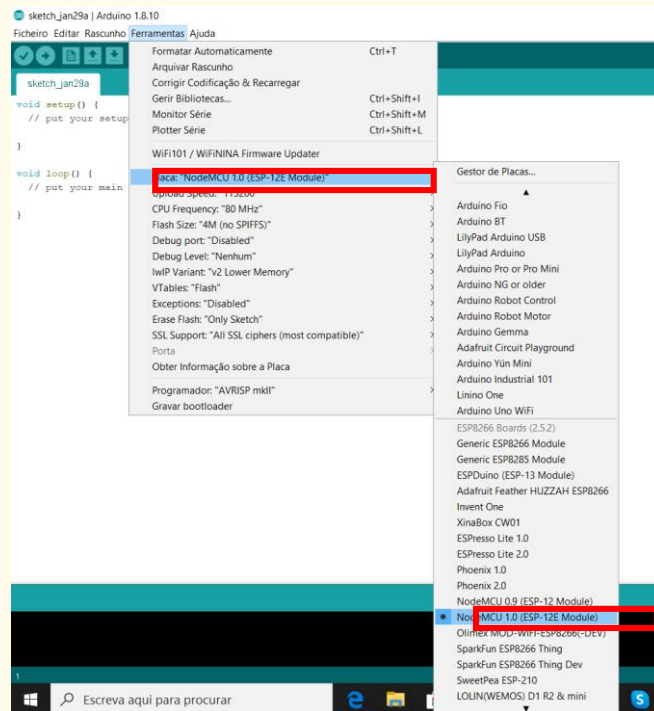
1. Access the Arduino IDE and add the URL indicated above.



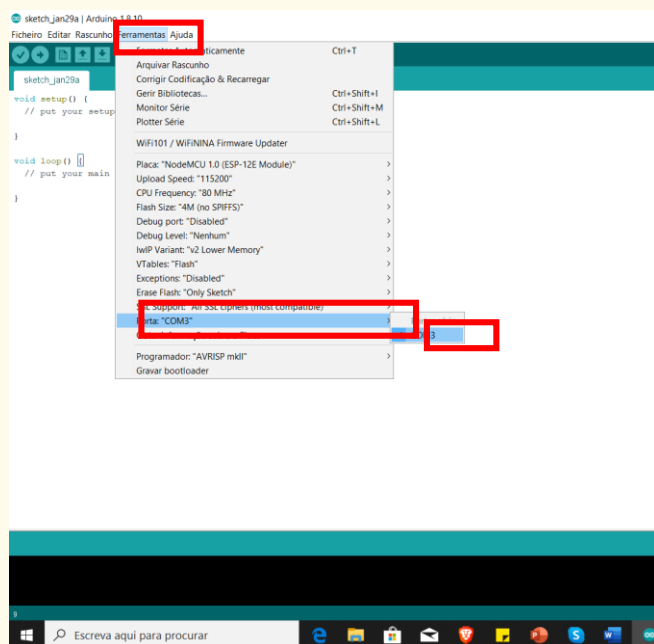
2. In plate manager add the ESP8266



3. Select the NodeMCU 1.0



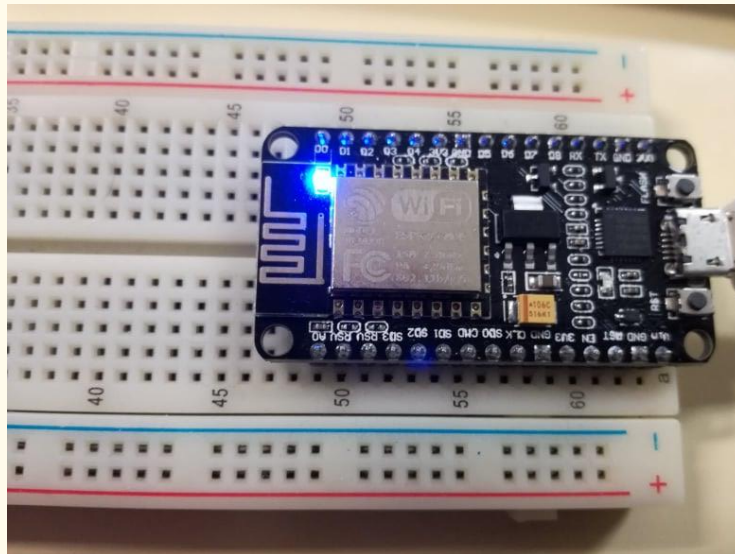
4. Connect the ESP8266 and Select the COM



In order to verify that the installation was performed correctly proceed to insert the following code and select the upload to board.



After the upload, we can observe the internal LED flash ESP8266 1 second in 1 second.



Wifi Connection –“Green Steam” Web Server

Include the WiFi library for the Esp8266: ESP8266WiFi.h.

```
sketch_nov25a | Arduino 1.8.9
Ficheiro Editar Rascunho Ferramentas Ajuda
sketch_nov25a $
#include <ESP8266WiFi.h>

void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Define a variable with your WiFi ssid and other one with your WiFi password. Begin the WiFi library with the ssid and password variables. The command is: `WiFi.begin(ssid,pass)`.

```
sketch_dec02c | Arduino 1.8.9
Ficheiro Editar Rascunho Ferramentas Ajuda
sketch_dec02c $
#include <ESP8266WiFi.h>

#define WIFI_SSID "Your SSID"//Your WiFi SSID
#define WIFI_PASSWORD "Your Pass"//Your WiFi Password

void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}
void loop() {
}
```

Initialize the Serial to a 115200 baud rate. The command is `Serial.begin (115200)`.

With that it is possible to print values on the Serial Monitor with the specific baud rate (115200).

```
void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.begin(115200);
}
```

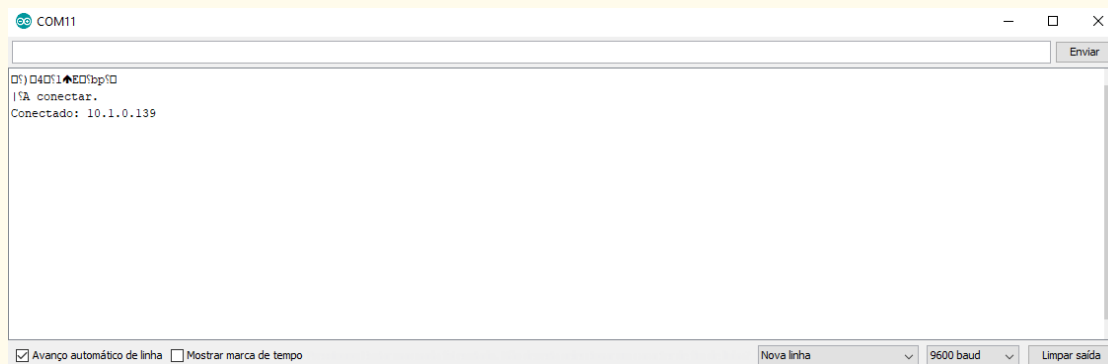
While not connected the code will be on a loop. The ESP8266 will try to connect to the WiFi network specified and will print dots "." on the Serial Monitor.

```
void setup() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.begin(115200);
  Serial.print("A conectar");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
}
```

When the ESP8266 connects to the WiFi the code will exit the loop and will run forward, printing that the ESP8266 is connected, its IP and will continue running to the void loop.

The result is the following:



Creating a web server using the ESP8266

Using the previous code that was used to connect the ESP8266 to an WiFi network define the WiFi server to port 80. The command is: `WiFiServer server(80)`.

```
WiFiServer server(80);
```

Define the following variables plus the led pin that will be used. Here it will going to be used the internal LED of the ESP8266 on the GPIO 2.

```
String header;

String State = "off";
int led_green = 2;
int led_red = 15;

unsigned long currentTime = millis();

unsigned long previousTime = 0;

const long timeoutTime = 2000;
```

Then, on the void setup define the green and red LEDs pins as OUTPUT, since signals will be emitted.

```
pinMode(led_green,OUTPUT);
pinMode(led_red,OUTPUT);
```

After the ESP8266 being connected the server must begin:

```
void setup() {

  pinMode(led_green,OUTPUT);
  pinMode(led_red,OUTPUT);

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.begin(115200);

  Serial.print("A conectar");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Conectado: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

By now, when the ESP8266 connects to the WiFi the code will exit the while loop and run forward-printing that the ESP8266 is connected, its WiFi, beginning the WiFi server to port 80 and jumps to the void loop.

Read if the ESP8266 has a connected client. If there is a new client connected the code will do a procedure.

```
void loop() {
  WiFiClient client = server.available();

  if(client){
  }
}
```

All the following code will be inside the if client condition.

Set the defined variables "currentTime" to millis(), to get every passed millisecond and "previousTime" to "currentTime".

Create a new string and call it, for example, currentLine, setting it to an empty string ("").

```
void loop() {
  WiFiClient client = server.available();

  if(client){
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");
    String currentLine = "";
  }
}
```

Create a loop while to make the code running continuously there while the ESP8266 is connected. Then, when the ESP8266 will be disconnected or timeout the header string, used to store every character received from the web, must be set to an empty string, the client must stop and it will printed on the Serial Monitor that the ESP8266 is disconnected.

```
void loop() {
  WiFiClient client = server.available();

  if(client){
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");
    String currentLine = "";

    while (client.connected() && currentTime - previousTime <= timeoutTime) {

    }
    header = "";
    client.stop();
    Serial.println("Desconectado");
    Serial.println("");
  }
}
```

While the ESP8266 is connected the variable `currentTime` is set to `millis()` and then ask if the ESP8266 receives data.

```
while (client.connected() && currentTime - previousTime <= timeoutTime) {
  currentTime = millis();
  if (client.available()) {
    }
  }
}
```

If the ESP8266 receives data, it will read every character received and store it on a variable called `c`. Then the ESP8266 prints on the Serial Monitor the read character. All the characters that will be received will be added to the variable `header`, making a String.

```
if (client.available()) {
  char c = client.read();
  Serial.write(c);
  header += c;
}
```

If the client reads a new line character, it will do a procedure. If is not a new line character but a carriage return character add it to the `currentLine` String

```
if (client.available()) {
  char c = client.read();
  Serial.write(c);
  header += c;
  if (c == '\n') {
    }
  else if (c != '\r') {
    currentLine += c;
  }
}
```

If the client reads a new line character, then ask if the length of the `currentLine` equals to 0. If so it will do a certain procedure, if not set the `currentLine` variable to an empty string.

```
if (c == '\n') {
  if (currentLine.length() == 0) {
    }
  else {
    currentLine = "";
  }
}
```

If the length of the current line equals to 0, meaning it is the end of the http request, the client will send a response. That response starts with a response code such as HTTP/1.1 200 OK and a content-type to make the client knowing what is coming. And then a send a blank line.

```
if (currentLine.length() == 0) {
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type:text/html");
  client.println("Connection: close");
  client.println();
}
```

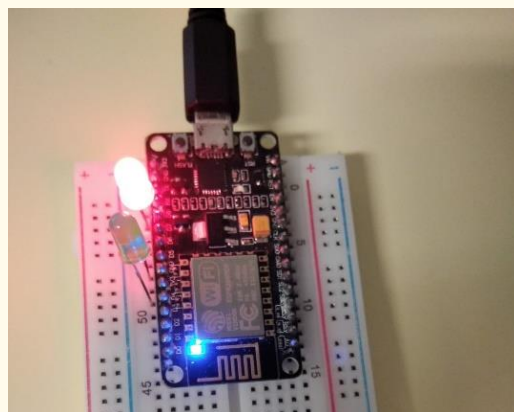
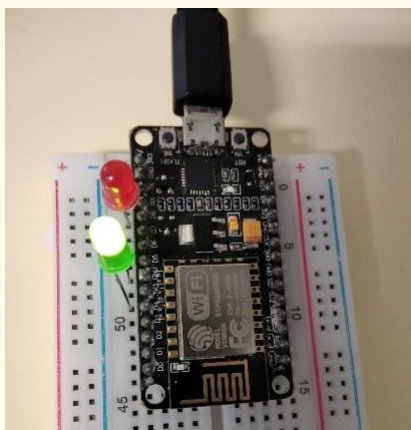
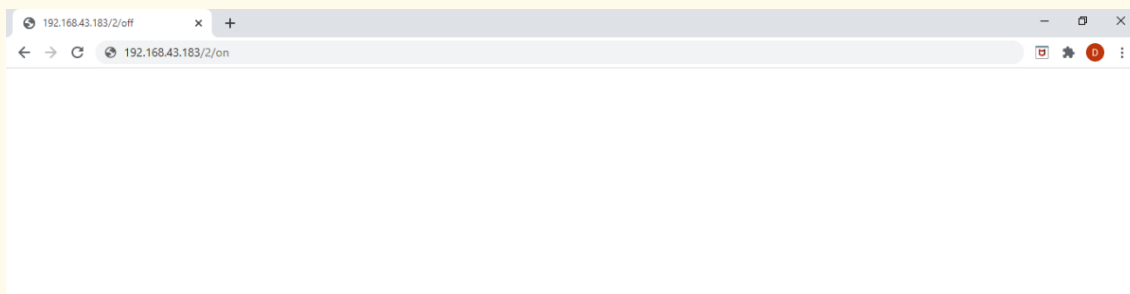
After sending the response ask if the header contains GET/2/on, turn the green LED on and the red LED off, store "on" in the State variable

```
if (header.indexOf("GET /2/on") >= 0) {
  State = "on";
  digitalWrite(led_green, HIGH);
  digitalWrite(led_red, LOW);
}
```

Otherwise, if the header contains GET/2/off, turn the LED off, store "off" in the State variable and optionally print on the Serial that the GPIO 2 is LOW as an indication.

```
else if (header.indexOf("GET /2/off") >= 0) {
  State = "off";
  digitalWrite(led_green, LOW);
  digitalWrite(led_red, HIGH);
}
```

After those conditions, it is possible to control the LED changing the values on the header, but the page is blank.



The page is build using the client to print http commands to define all the specifications required such as the buttons, their background colour, text colour, font etc...

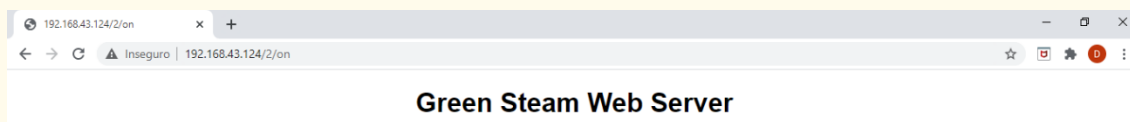
The resulting piece of code is the following:

```
// Display the HTML web page
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
client.println("<link rel=\"icon\" href=\"data:;\">");
// CSS to style the on/off buttons
// Feel free to change the background-color and font-size attributes to fit your preferences
client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;};");
client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;};");
client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;});");
client.println(".button2 {background-color: #AF4C50;};</style></head>");
```

After that write the title of the page:

```
client.println("<body><h1>Green Steam Web Server</h1>");
```

Look! The “Green Steam Web Server” is written on the web page.



After that, write the state of the LED:

```
client.println("<p>" + State + "</p>");
```

With that, the page will have a visible title and a real time indication of the led state either on or off.

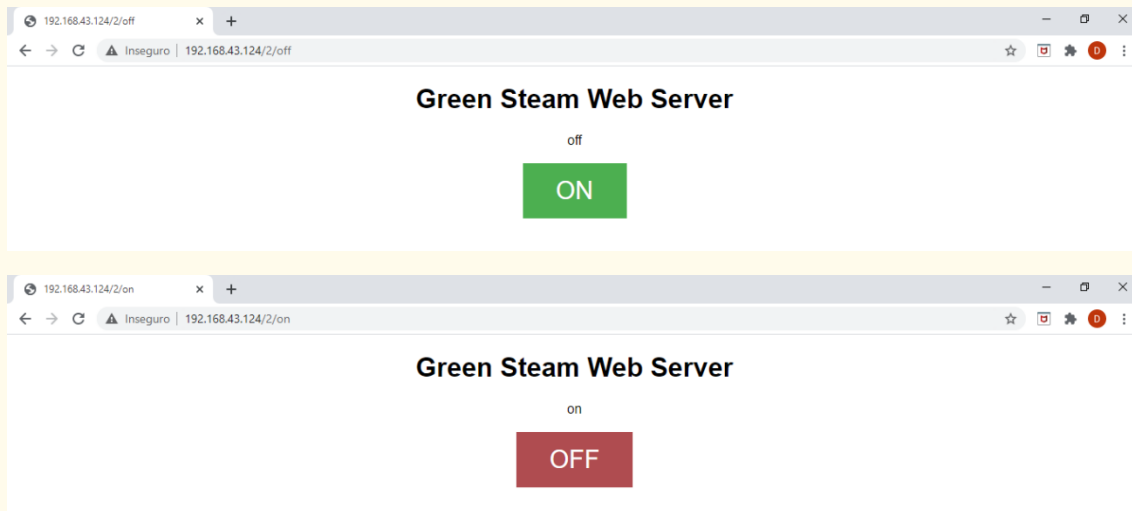


Then, if the State equals to “off” the client must print a green button with “ON” written on it. Otherwise, the client will print a red button with “OFF” written on it.

The result is the following:

```
if (State=="off") {
  client.println("<p><a href=\"/2/on\"><button class=\"button\">ON</button></a></p>");
} else {
  client.println("<p><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></p>");
}
}
```





Finally, it is just necessary to add this piece of code to finish the html code and break the while loop.

```
client.println("</body></html>");
client.println();
break;
```

- **Bibliographic references:**

- <http://www.electronicwings.com/>. (n.d.). Retrieved from http://www.electronicwings.com/public/images/user_images/images/NodeMCU/NodeMCU%20Basics%20using%20ESP8266%20IDE/NodeMCU%20MQTT%20Client/MQTT%20Broker%20nw.png.
- <https://diygeeks.org/learn/intro-to-blynk/>. (n.d.). Retrieved from <https://diygeeks.org>.
- <https://esp8266-shop.com/esp8266-guide/esp8266-nodemcu-pinout/>. (n.d.). Retrieved from <https://esp8266-shop.com/>.
- <https://realbusiness.co.uk/from-1982-coca-cola-vending-machine-to-latest-trend-what-the-internet-of-things-means-for-business/>. (n.d.).
- <https://www.embarcados.com.br/introducao-ao-blynk-app/>. (n.d.). Retrieved from <https://www.embarcados.com.br/>.
- Oliveira, S. d. (2017). *Internet das coisas*. novatec.
- Santos, R. (n.d.). <https://randomnerdtutorials.com/>. Retrieved from <https://randomnerdtutorials.com/>.
- Souza, F. (n.d.). *Controle de dispositivos remotamento com o ESP8266*.

LESSON – CONNECTING ESP8266 WITH TELEGRAM

- **Study section:** WiFi Fundamentals and IoT
- **Lesson duration:** 3:00h
- **Educational objectives:**
 - Configure the ESP8266 to communication with the Telegram
 - Configure the Telegram to communication with the ESP8266
 - Communication between Telegram and ESP8266
- **Main Keyword(s):**
 - ESP8266
 - Wifi
 - Telegram
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● Characterize ESP 8266 ● Recognize the hardware needed to connect ESP8266 with Telegram ● Recognize the importance and benefits the Telegram 	<ul style="list-style-type: none"> ● Determine the hardware to be used in different configurations with the ESP8266 ● Program and configure communication between the Telegram and ESP8266 	<ul style="list-style-type: none"> ● Plan and structure tasks ● Communicate concepts and ideas clearly ● Act with initiative and demonstrate analytical capacity ● Recommend solutions for problem-solving ● Demonstrate creativity, autonomy and innovative spirit

- **Required material and resources:**
 - ESP8266
 - Cable USB to microUSB
 - LED Red
 - BreadBoard
 - Computer
 - Software Arduino IDE
 - Internet Connection
 - Smartphone with the Telegram app



Brief definition of Telegram

Telegram is a free application that is used similarly to Whatsapp, where you can share text, photos, videos, files, etc.

Conecting ESP8266 with Telegram

Telegram can be used for more specific purposes and with professional interest, that is, it can be used in IOT systems, with the objective of controlling / monitoring intelligent devices, which will be our main objective.

In 2015 a new API was presented, which allowed the creation of bots in the telegram for ESP32 / ESP8266, where it is possible to exchange messages as if they were a person. With this functionality it is possible to send instructions, thus controlling smart devices.

In this chapter it will be presented how we can configure ESP32 / ESP8266 and Telegram so that there is a bidirectional communication between both.



To demonstrate the potential of this relationship, the control of an LED will be presented through a message by the Telegram Bot.

Programming nodemcu plate

Regarding the installation of nodemcu board, you must follow the following steps similar in the topic Programming ESP8266.

After checking the steps in this topic, we can proceed to Telegram programming.

Programming Arduino IDE to communication with Telegram

```

/*
  Project created using Brian Lough's Universal Telegram Bot
  Library: https://github.com/witnessmenow/Universal-Arduino-Telegram-
  Bot
  Example based on the Universal Arduino Telegram Bot Library:
  https://github.com/witnessmenow/Universal-Arduino-Telegram-
  Bot/blob/master/examples/ESP8266/FlashLED/FlashLED.ino
*/

#ifdef ESP32
  #include <WiFi.h>
#else
  #include <ESP8266WiFi.h>
#endif
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h> // Universal Telegram Bot
Library written by Brian Lough:
https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
#include <ArduinoJson.h>

// Your network credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

// Initialize Telegram BOT
#define BOTtoken "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
//Your Bot Token (Get from Botfather)

// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
#define CHAT_ID "XXXXXXXXXX"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

const int ledPin = 2;
bool ledState = LOW;

// Handle what happens when you receive new messages
void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i=0; i<numNewMessages; i++) {

```

```

// Chat id of the requester
String chat_id = String(bot.messages[i].chat_id);
if (chat_id != CHAT_ID){
  bot.sendMessage(chat_id, "Unauthorized user", "");
  continue;
}

// Print the received message
String text = bot.messages[i].text;
Serial.println(text);

String from_name = bot.messages[i].from_name;

if (text == "/start") {
  String welcome = "Welcome, " + from_name + ".\n";
  welcome += "Use the following commands to control your
outputs.\n\n";
  welcome += "/led_on to turn GPIO ON \n";
  welcome += "/led_off to turn GPIO OFF \n";
  welcome += "/state to request current GPIO state \n";
  bot.sendMessage(chat_id, welcome, "");
}

if (text == "/led_on") {
  bot.sendMessage(chat_id, "LED state set to ON", "");
  ledState = HIGH;
  digitalWrite(ledPin, ledState);
}

if (text == "/led_off") {
  bot.sendMessage(chat_id, "LED state set to OFF", "");
  ledState = LOW;
  digitalWrite(ledPin, ledState);
}

if (text == "/state") {
  if (digitalRead(ledPin)){
    bot.sendMessage(chat_id, "LED is ON", "");
  }
  else{
    bot.sendMessage(chat_id, "LED is OFF", "");
  }
}
}

void setup() {
  Serial.begin(115200);

#ifdef ESP8266
  client.setInsecure();

```



```

#endif

pinMode(ledPin, OUTPUT);
digitalWrite(ledPin, ledState);

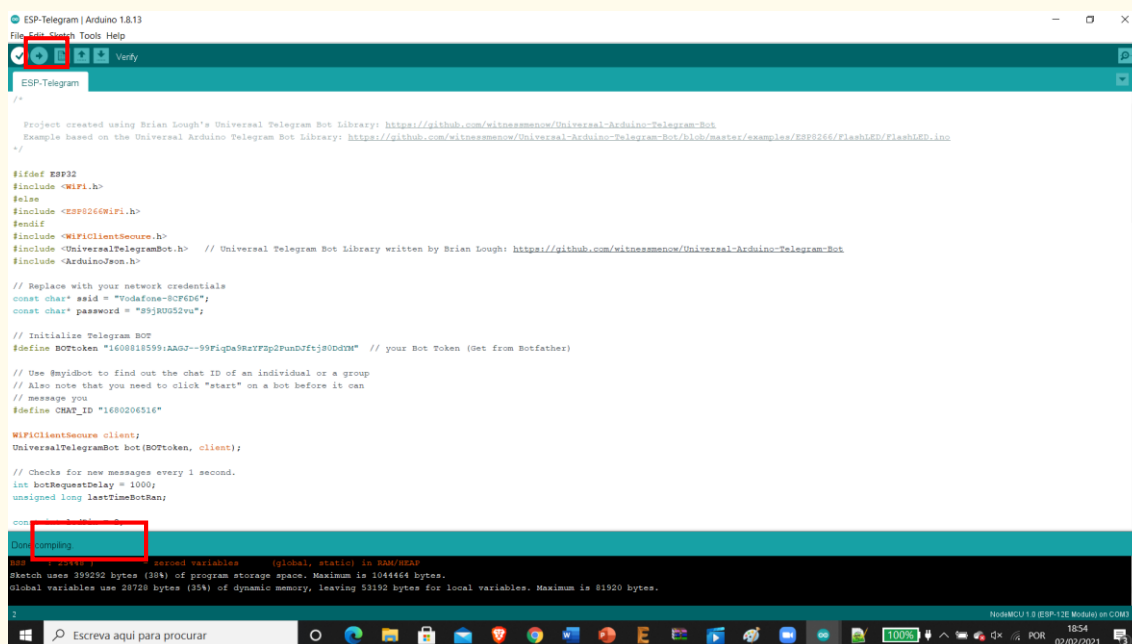
// Connect to Wi-Fi
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
}
// Print ESP32 Local IP Address
Serial.println(WiFi.localIP());
}

void loop() {
    if (millis() > lastTimeBotRan + botRequestDelay) {
        int numNewMessages = bot.getUpdates(bot.last_message_received +
1);

        while (numNewMessages) {
            Serial.println("got response");
            handleNewMessages (numNewMessages);
            numNewMessages = bot.getUpdates (bot.last_message_received +
1);
        }
        lastTimeBotRan = millis();
    }
}

```

After the copy code to Arduino IDE, you need verify:



If compiling is correct below the image show the message: Done Compiling

The next step is the transfer the code to Board ESP8266. You should not forget to select the correct board and port COM. We can see this correct operation below:

```

ESP-Telegram | Arduino 1.8.13
File Edit Sketch Tools Help
ESP-Telegram
/*
 * Project created using Brian Lough's Universal Telegram Bot Library: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
 * Example based on the Universal Arduino Telegram Bot Library: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot/blob/master/example/ESP8266/FlashESP/FlashESP.ino
 */

#ifdef ESP32
#include <WiFi.h>
#else
#include <ESP8266WiFi.h>
#endif
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h> // Universal Telegram Bot Library written by Brian Lough: https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot
#include <ArduinoJson.h>

// Replace with your network credentials
const char* ssid = "Yodafone-8CF6D6";
const char* password = "89jRU52vu";

// Initialize Telegram Bot
#define BOTtoken "1600818599:AA07--99FiQe9w7Fp2PunDvftj9Ud3M" // your Bot Token (Get from Botfather)

// Use #myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
#define CHAT_ID "1680206516"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

const int ledPin = 2;

Done uploading
Uploading...
Used resetting via RST pin...
NodeMCU 1.0 (ESP-12E Module) COM3

```

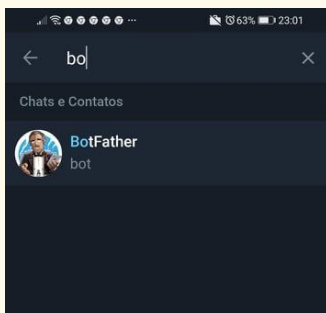
Telegram Configuration

After configuring the ESP8266 we will move to Telegram creating the necessary steps for the communication between the devices. For this we must proceed with the installation of the Telegram application as shown in the figure below:

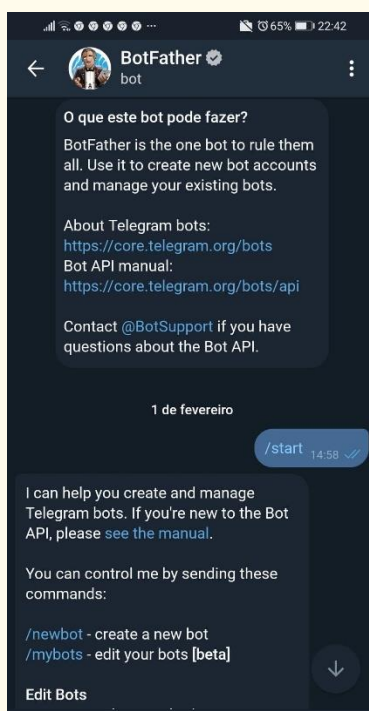


After installing the application and creating your account, we will create a bot. This bot is a system that allows you to respond to messages, however, for that to happen it has to be configured. To better understand what it is about, let's follow the tutorial step by step to observe the final result.

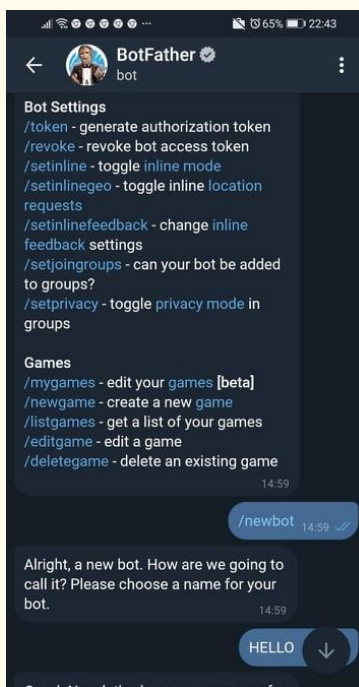
Let's start by looking for "botfather" as shown in the figure below:



The "botfather" is a bot that is already built and allows us to manage our bots. To do this we click on the botfather and type `/start`, observing some available codes.

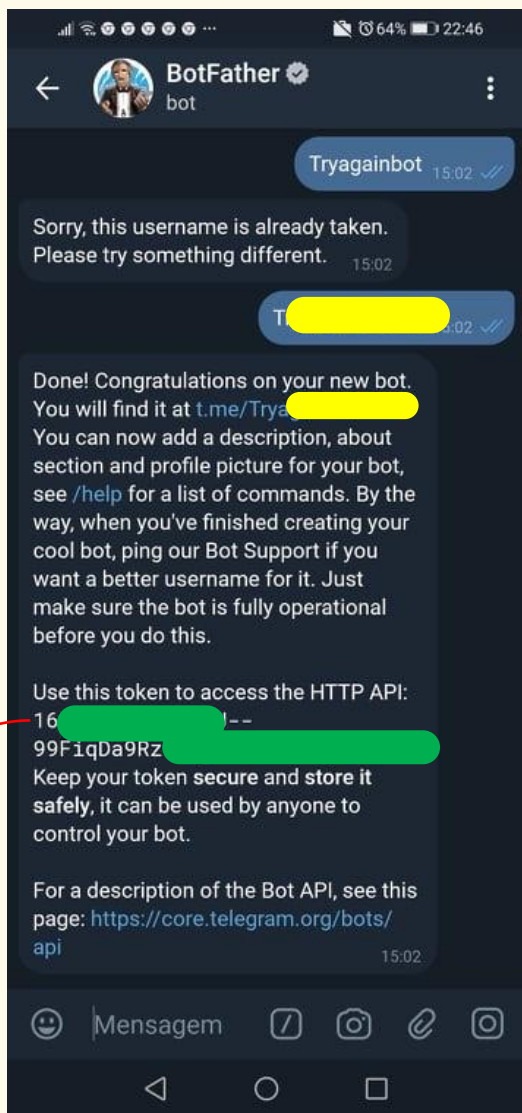


We wrote the code `/newbot` as shown in the figure below:



In response to the question asked, I put the name HELLO, thus defining the name for my bot.

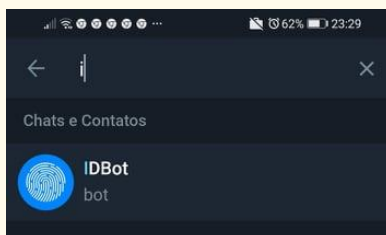
(Continues on the next page)



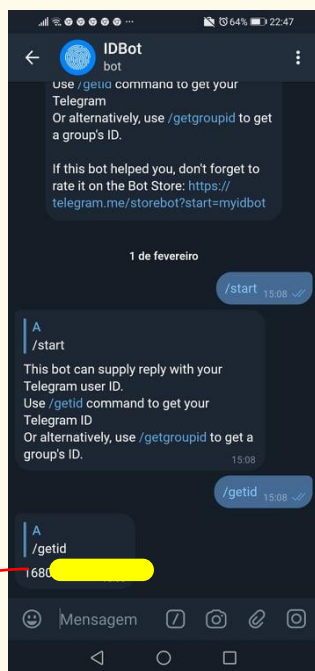
```
// Initialize Telegram BOT
#define BOTtoken "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
//Your Bot Token (Get from Botfather)
```

In the place shown in yellow, the username for the Bot is defined. If the bot is successfully created, you will receive a message similar to the one above, where an green ID is displayed that will be used to communicate with the bot's token. We copied this ID and put it in the ESP programming in the Arduino IDE as shown in the figure above.

The next step is to obtain our ID, so that in this way it is identified who is communicating with the bot. For this I will search for "idbot" as shown in the figure below:



I select IDBot and send first `/start` and then `/get id` thus obtaining my ID.

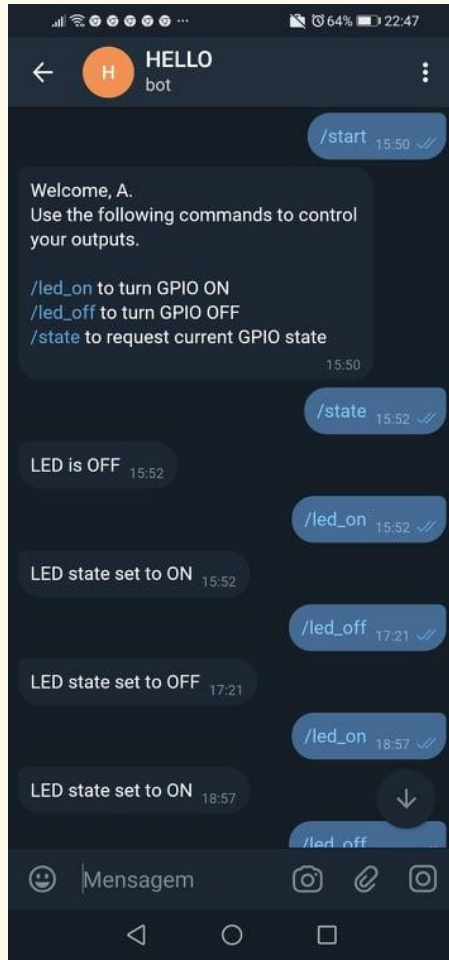


```
// message you
#define CHAT_ID "XXXXXXXXXX"

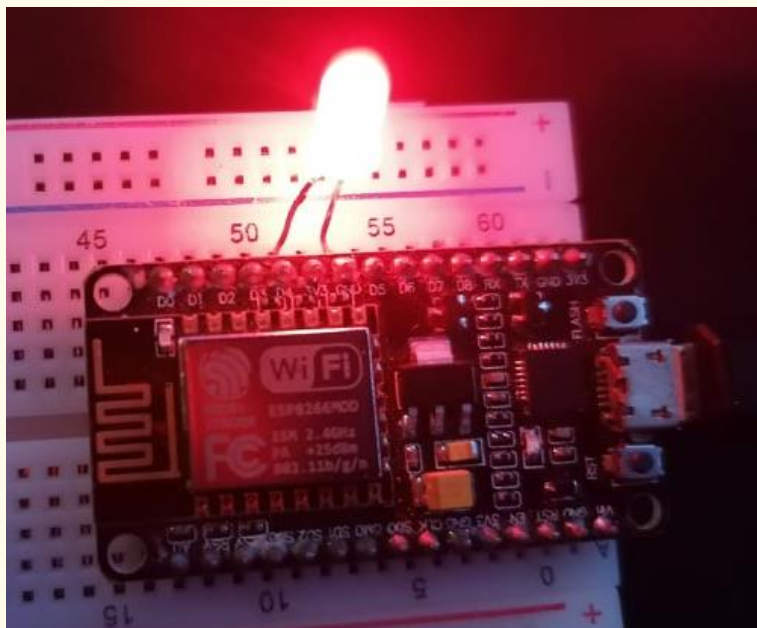
// message you
#define CHAT_ID "XXXXXXXXXX"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);
```

We look for our created bot and we can start the conversation with him. In this case to turn on and turn off an LED.



To observe the final result, an LED was placed between PIN D4 and GND, obtaining the result below:



- **Bibliographic references:**

<http://www.electronicwings.com/>. (n.d.). Retrieved from http://www.electronicwings.com/public/images/user_images/images/NodeMCU/NodeMCU%20Basics%20using%20ESPlorer%20IDE/NodeMCU%20MQTT%20Client/MQTT%20Broker%20nw.png.

<https://diygeeks.org/learn/intro-to-blynk/>. (n.d.). Retrieved from <https://diygeeks.org>.

<https://esp8266-shop.com/esp8266-guide/esp8266-nodemcu-pinout/>. (n.d.). Retrieved from <https://esp8266-shop.com/>.

<https://realbusiness.co.uk/from-1982-coca-cola-vending-machine-to-latest-trend-what-the-internet-of-things-means-for-business/>. (n.d.).

<https://www.embarcados.com.br/introducao-ao-blynk-app/>. (n.d.). Retrieved from <https://www.embarcados.com.br/>.

Oliveira, S. d. (2017). *Internet das coisas*. novatec.

Santos, R. (n.d.). <https://randomnerdtutorials.com/>. Retrieved from <https://randomnerdtutorials.com/>.

Souza, F. (n.d.). *Controle de dispositivos remotamento com o ESP8266*.

LESSON – CONNECTING ESP8266 WITH FIREBASE

- **Study section:** WiFi Fundamentals and IoT
- **Lesson duration:** 4:00h
- **Educational objectives:**
 - Configure the ESP32 to Firebase access
 - Configure the Firebase to communication with ESP32
 - Communication between Firebase and ESP32
- **Main Keyword(s):**
 - ESP32
 - Wifi
 - Firebase
- **Learning outcomes and acquired competences:**

At the end of the unit, the student should be able to:

Knowledge	Skills	Responsibility and Autonomy
<ul style="list-style-type: none"> ● Characterize ESP 8266 ● Recognize the hardware needed to connect ESP32 with Firebase ● Recognize the importance and benefits the Firebase 	<ul style="list-style-type: none"> ● Determine the hardware to be used in different configurations with the ESP32 ● Program and configure communication between the Firebase and ESP32 	<ul style="list-style-type: none"> ● Plan and structure tasks ● Communicate concepts and ideas clearly ● Act with initiative and demonstrate analytical capacity ● Recommend solutions for problem-solving ● Demonstrate creativity, autonomy and innovative spirit

- **Required material and resources:**
 - ESP32
 - Cable USB to microUSB
 - BreadBoard
 - Computer
 - Software Arduino IDE
 - Internet Connection
 - Google Account



Brief definition of Firebase

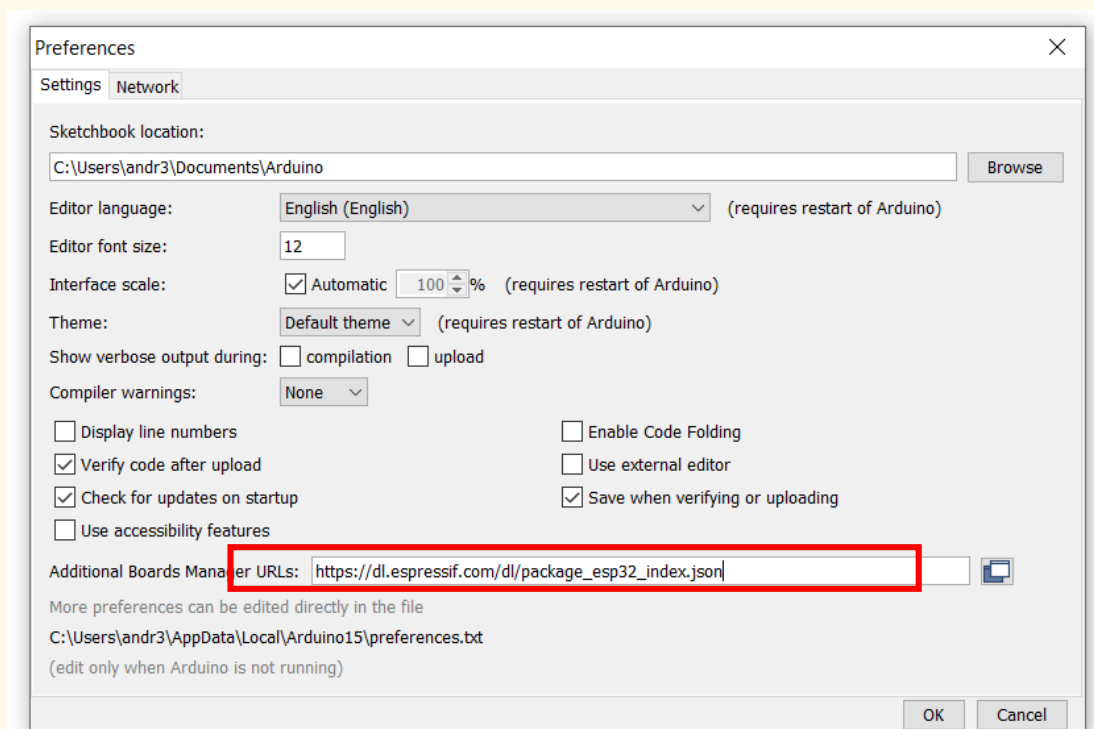
The Firebase provides developers with a variety of tools and services to help them develop quality applications, grow their user base and make a profit. It is built on Google's infrastructure.

Connecting Firebase – ESP32

In our project we will work with the Firebase Real Time Database. The Realtime Database is a cloud-hosted database. Data is synced across all clients in realtime and remains available when your app goes offline.

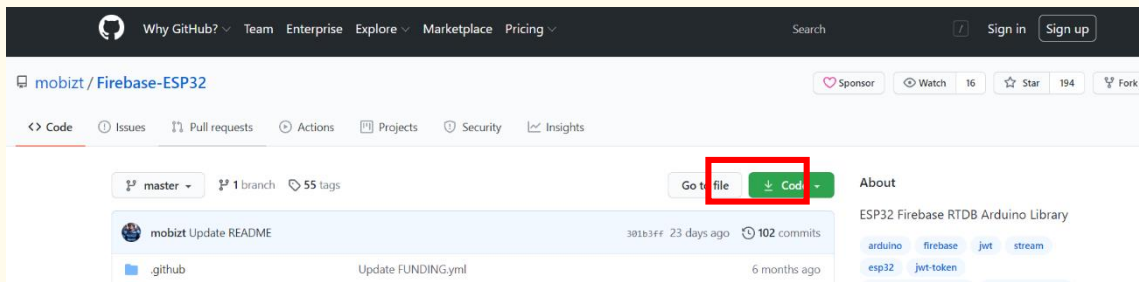
Programming ESP32

Access the Arduino IDE and add the URL indicated above.



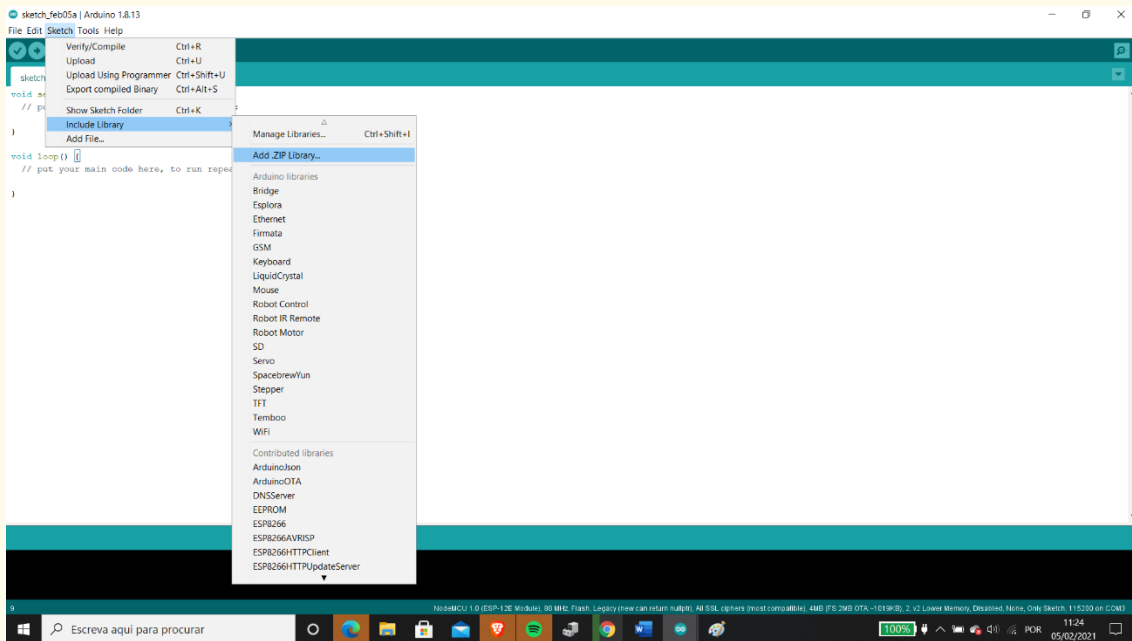
For work with ESP32 and Firebase we need to download the library FirebaseESP32.H. For this we can download it from the page indicated below:

<https://github.com/mobizt/Firebase-ESP32>

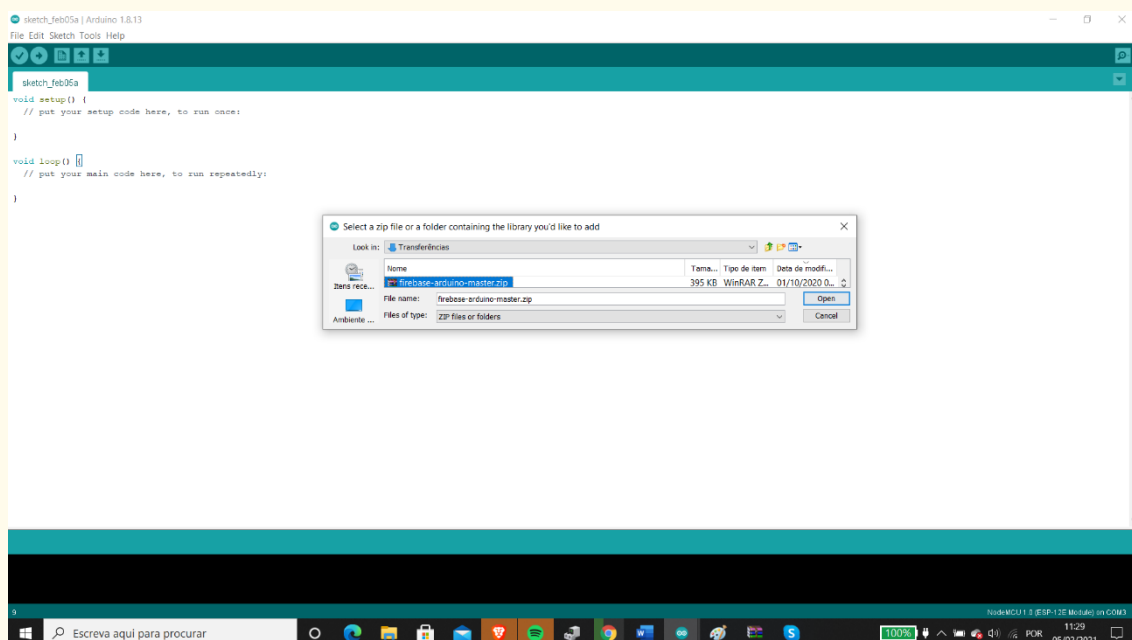


After download we need install the library. We can see the step in the images below:

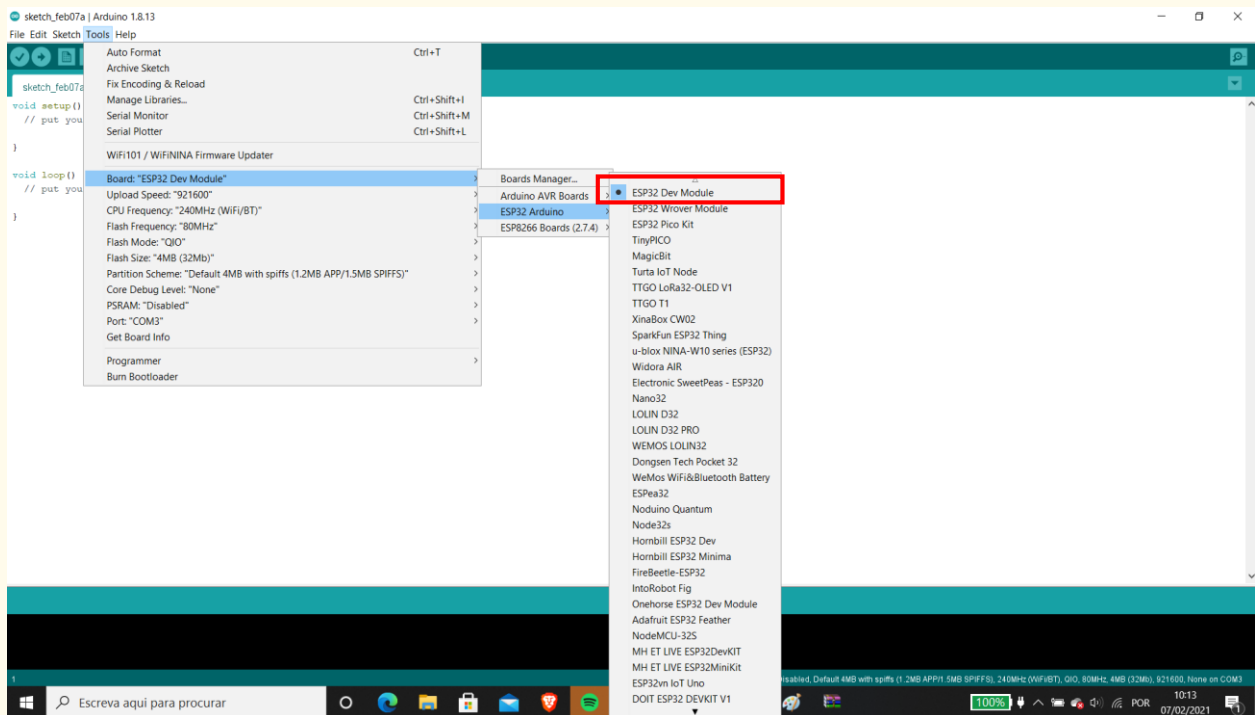
Add the library.



Select the library "firebase-arduino-master".zip



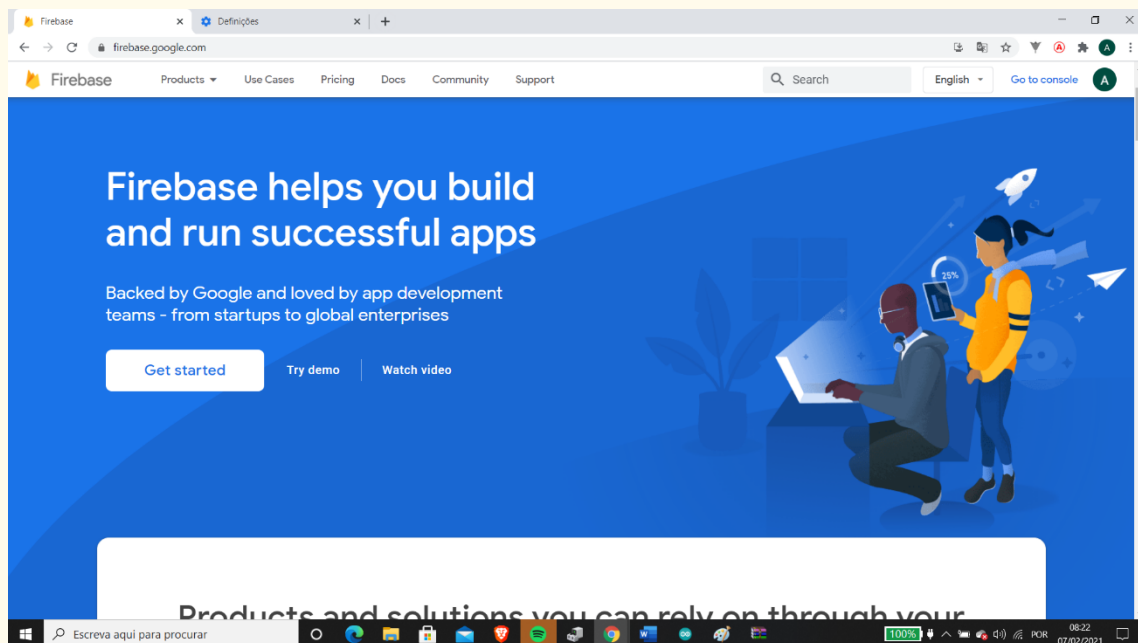
In the last step for the programming the ESP32 we select the correct board:



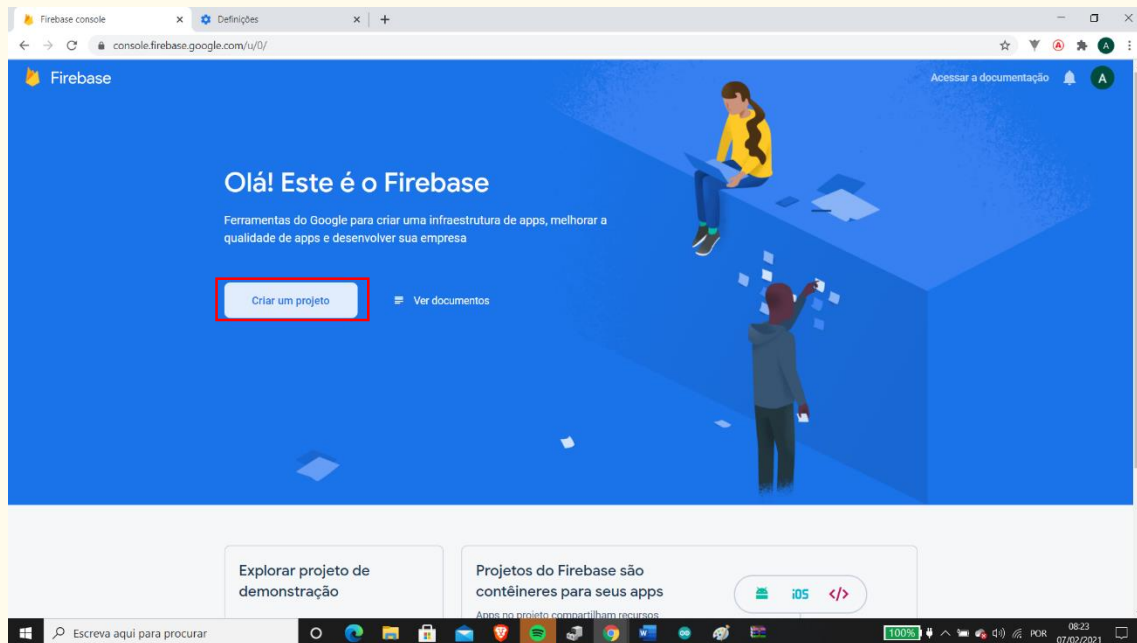
Configuration Firebase

For configuration Firebase you must follow the following steps:

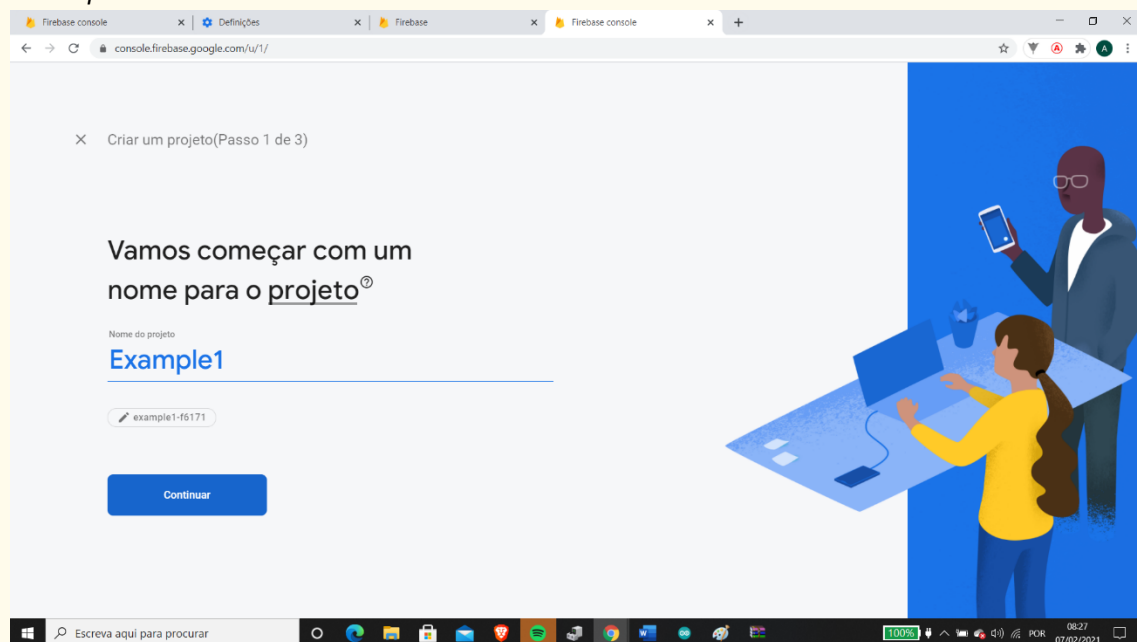
1. Go the site firebase.google.com



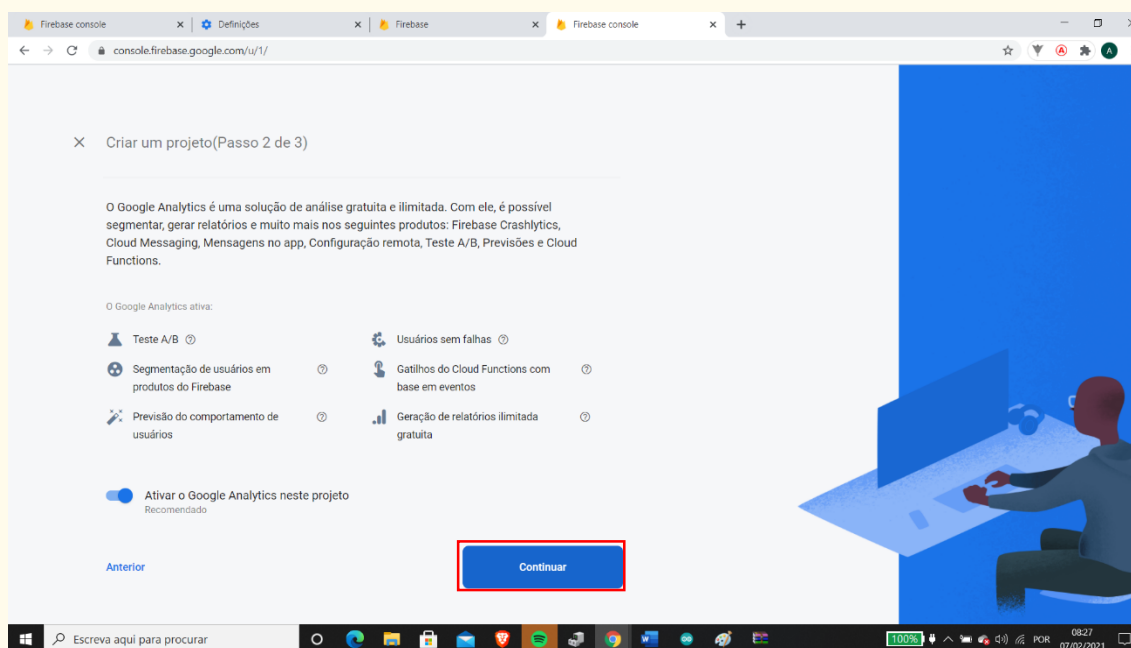
2. Select *Add new project*



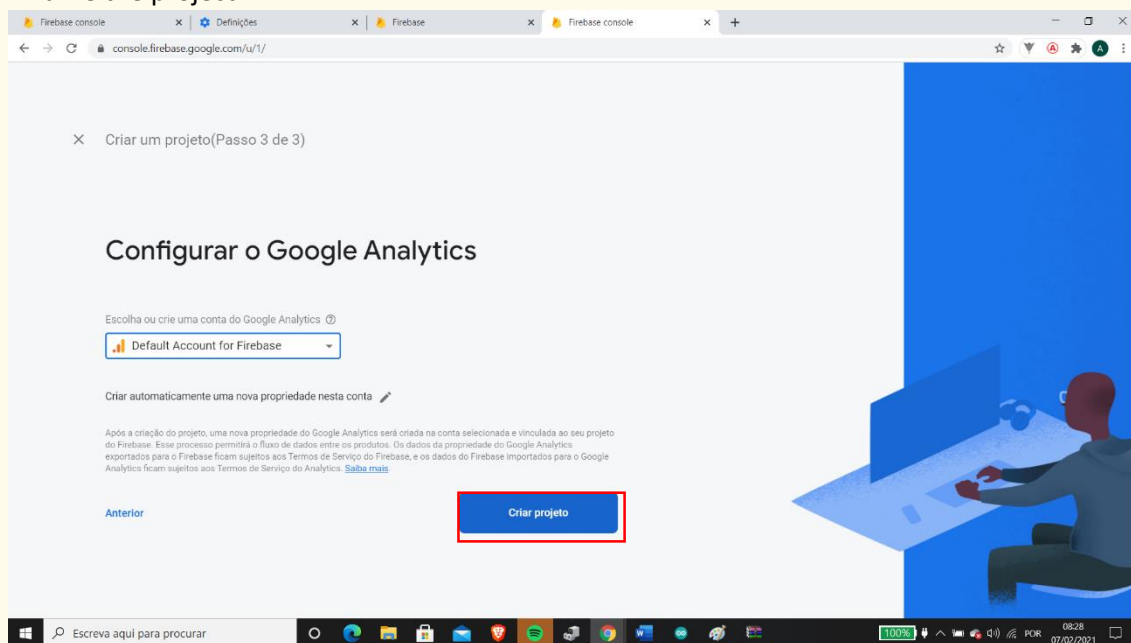
3. Choose the name for your project. In the example below the chosen name was the "Example1"



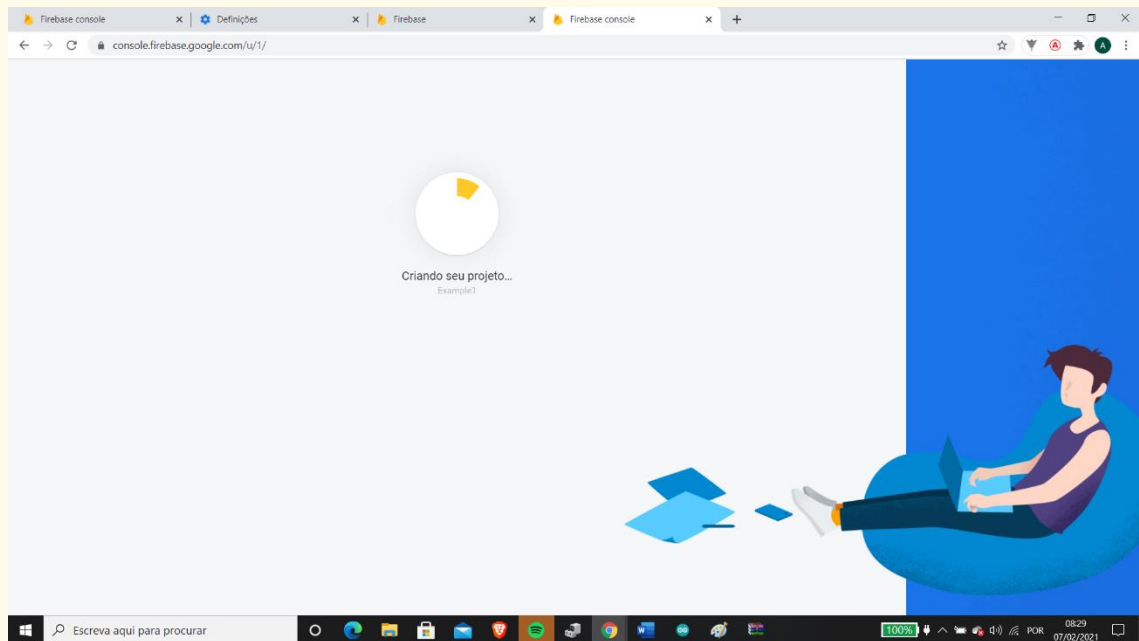
4. Confirm the creation of the project.



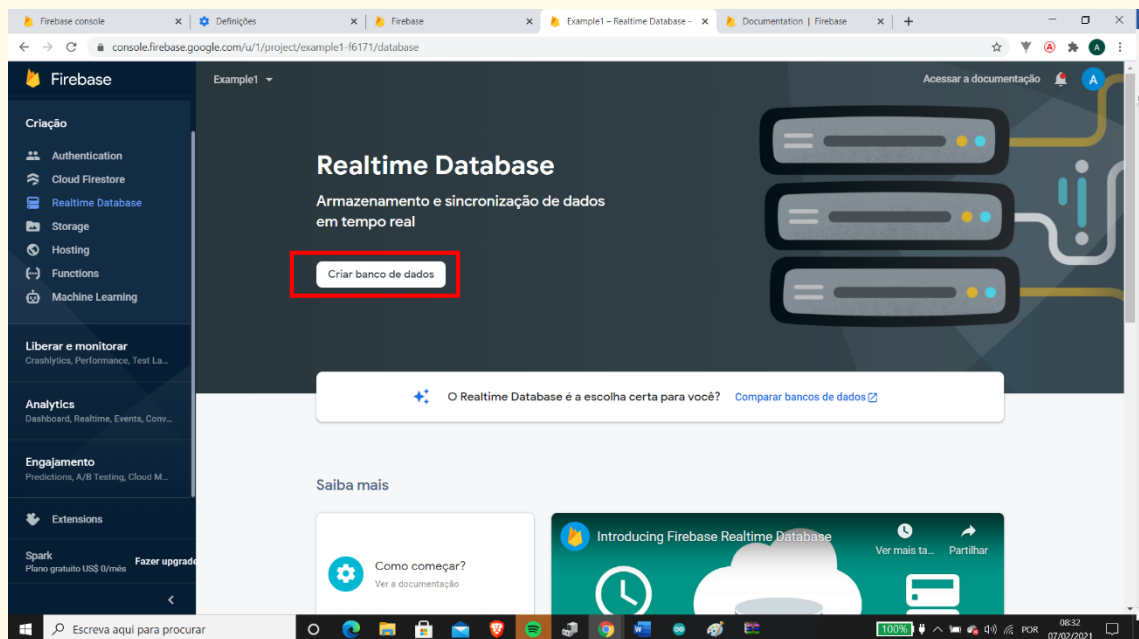
5. Finalize the project.



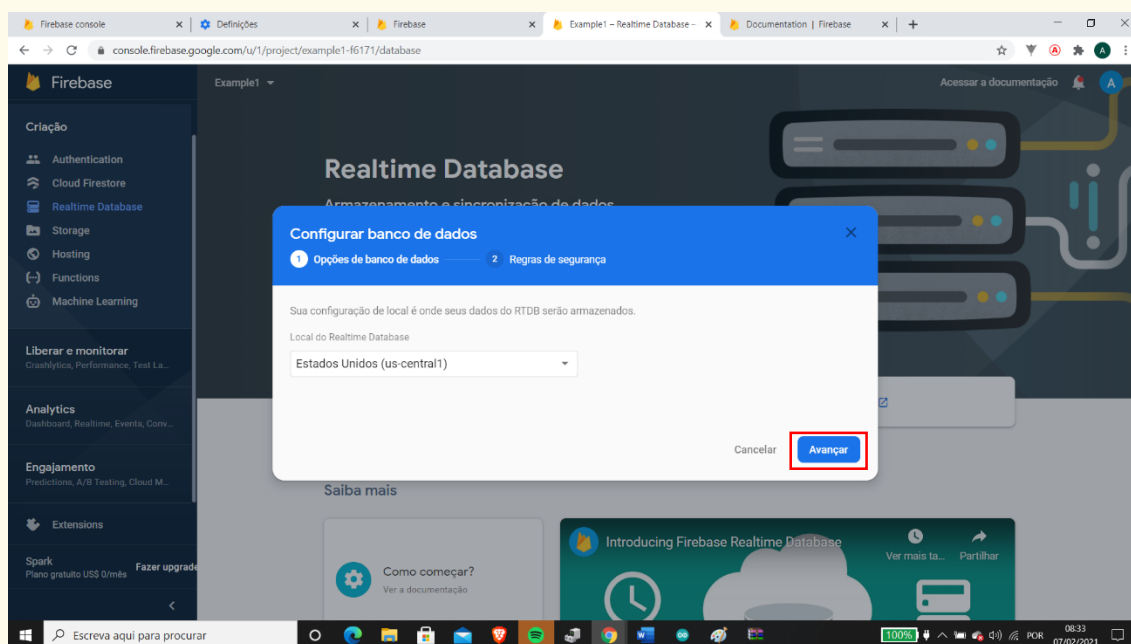
6. Wait while the project is created.



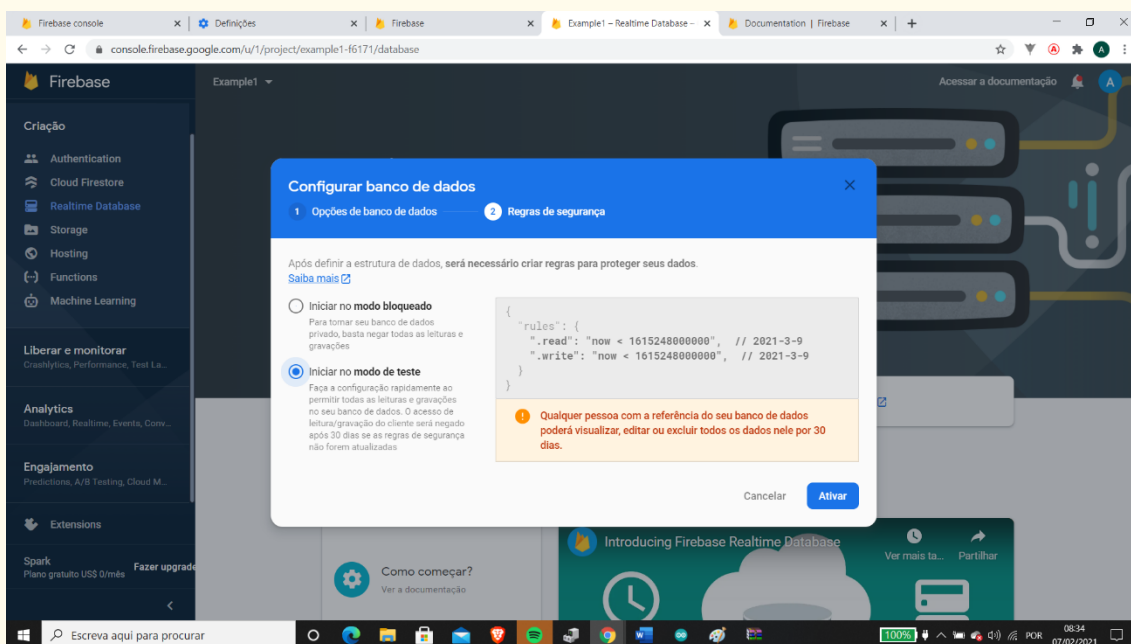
7. Configure the Real Time Database to operating with your system:



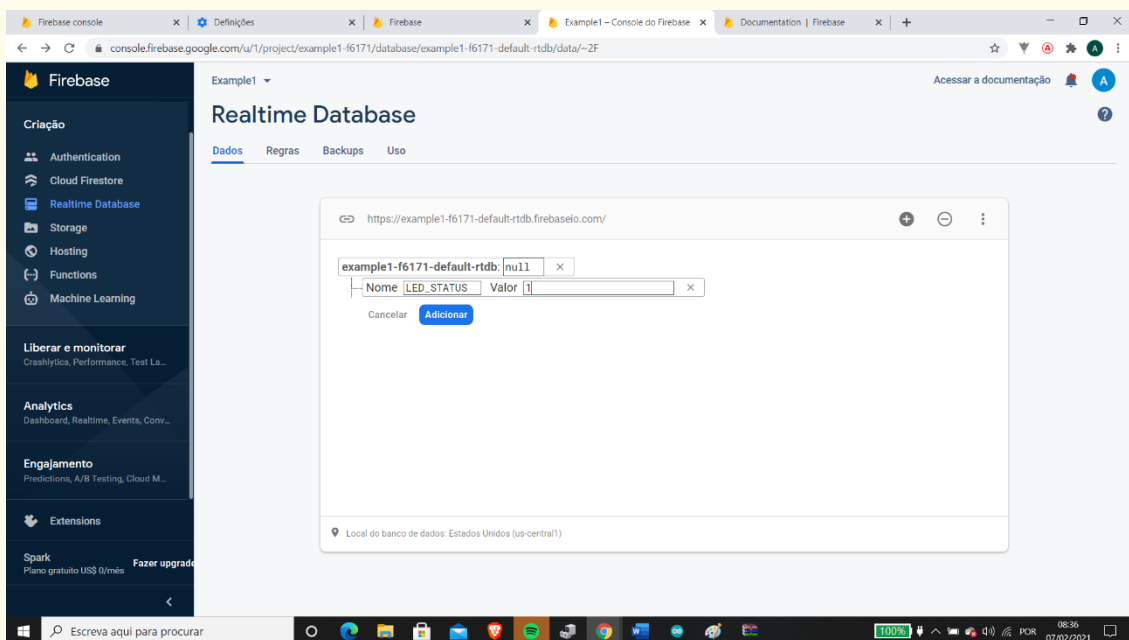
8. Select the “Configuration Database”.



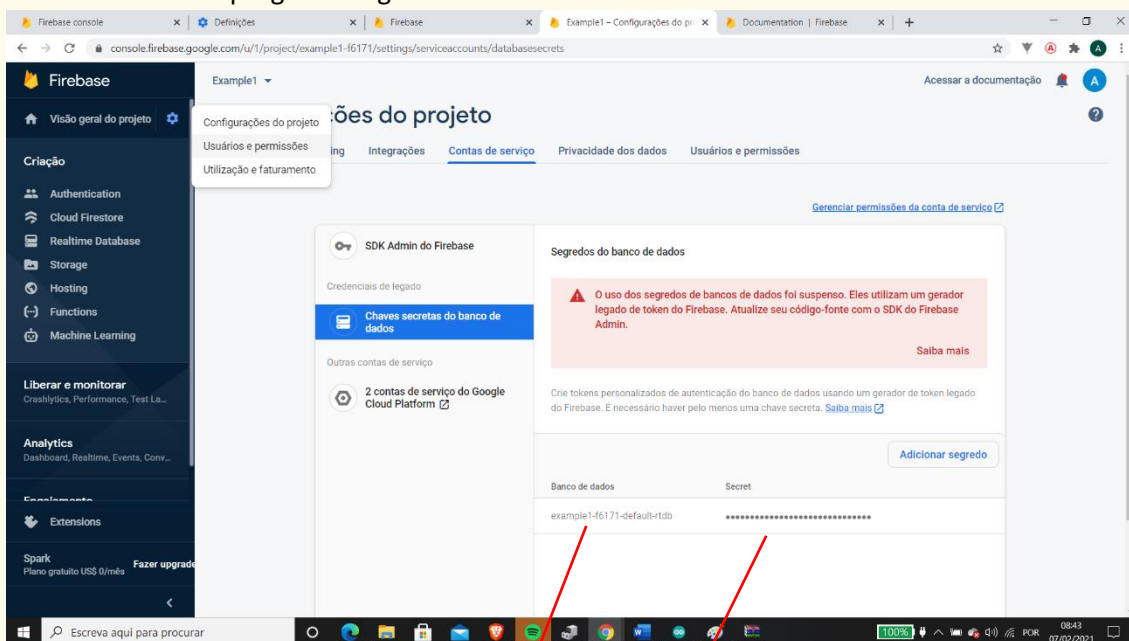
9. Select the “Test Mode”.



10. Select the Data and your example. To fill in this field you must put the name STATUS LED and assign a value of 1, as shown in the image below.



11. In the *project overview* select the *user and permissions* and copy the secret token. This token will be used in the programming ESP32 to communication with the firebase.



The Programming ESP32 to connection with Firebase

```
#include <WiFi.h>
#include <FirebaseESP32.h>

#define FIREBASE_HOST "Your Firebase Host"
#define FIREBASE_AUTH "Your Firebase Auth"
```

Now we can copy the code below to Arduino IDE:

```
#include <WiFi.h>
#include <FirebaseESP32.h>

#define FIREBASE_HOST "Your Firebase Host"
#define FIREBASE_AUTH "Your Firebase Auth"

#define WIFI_SSID "Vodafone-8CF6D6"
#define WIFI_PASSWORD "S9jRUG52vu"

FirebaseData firebaseData;
FirebaseJson json;
int led = 2;

void setup() {
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
}

void loop() {

  Firebase.getInt(firebaseData, "LED_STATUS");
  Serial.println(firebaseData.intData());

  if (firebaseData.intData() == 1) { // compare the input of led
status received from firebase
    Serial.println("Led Turned ON");
    digitalWrite(led, HIGH);
  }
}
```



```

if (firebaseData.intData() == 2) { // compare the input of led
status received from firebase
  Serial.println("Led Turned OFF");
  digitalWrite(led, LOW);
}
}

```

After the copy code to Arduino IDE, you need verify:

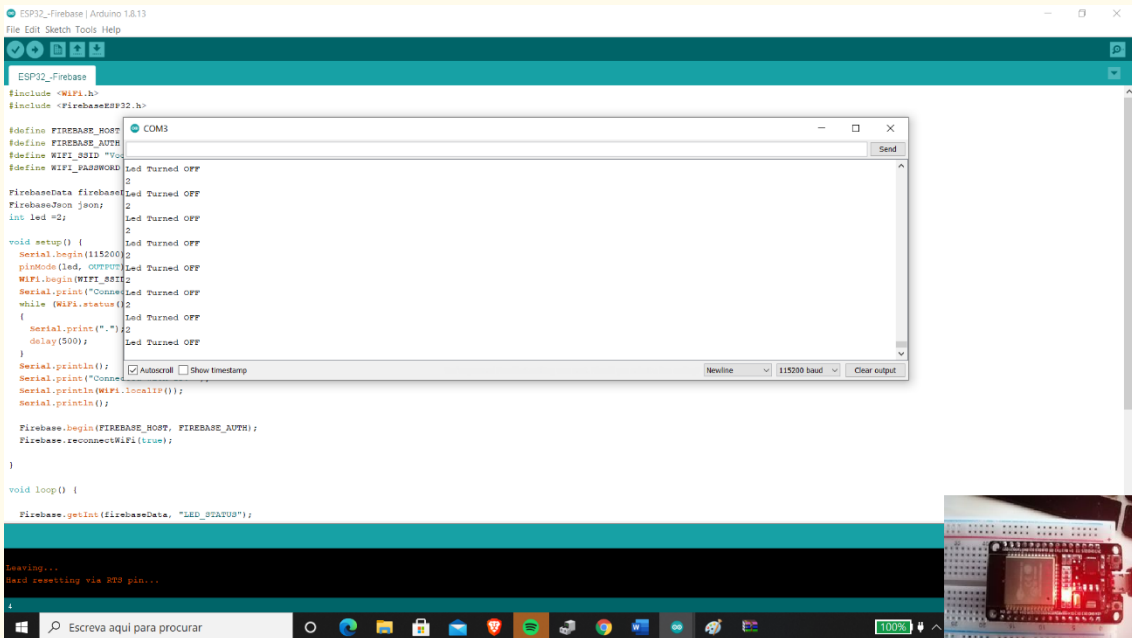
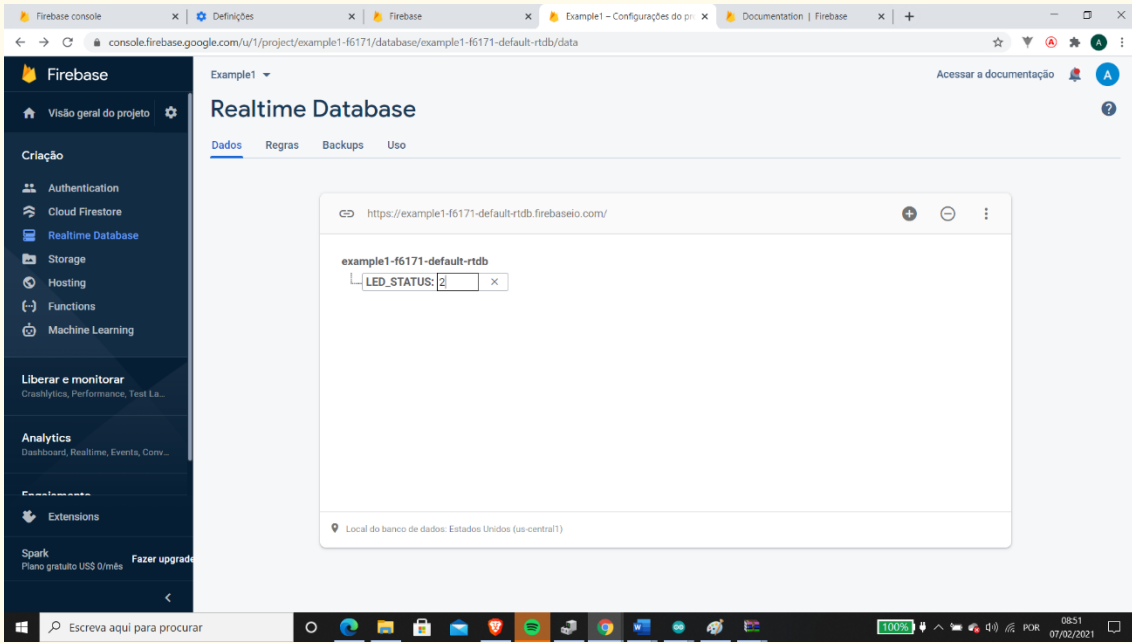
The screenshot shows the Arduino IDE interface for an ESP-Telegram project. The 'Verify' button is highlighted with a red box. The output window at the bottom shows the compilation status: 'Done Compiling'. The code in the background includes headers for ESP32, WiFi, and the UniversalTelegramBot library, along with network credentials and bot token configuration.

If compiling is correct below the image show the message: Done Compiling

The next step is the transfer the code to Board ESP32. You should not forget to verify the correct board and port COM. We can see this correct operation below:

The screenshot shows the Arduino IDE interface for an ESP32-Firebase project. The 'Upload' button is highlighted with a red box. The output window shows the upload progress and the message 'LED Turned ON'. The code in the background includes headers for ESP32, Firebase, and WiFi, along with Firebase credentials and a loop that checks for data from Firebase and controls an LED.

The LED blue turn on. If chance the value 1 in LED_SATATUS for 2 the LED blue turn off.



The European Commission support for the production of this document does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



- **Bibliographic references:**

<http://www.electronicwings.com/>. (n.d.). Retrieved from http://www.electronicwings.com/public/images/user_images/images/NodeMCU/NodeMCU%20Basics%20using%20ESPlorer%20IDE/NodeMCU%20MQTT%20Client/MQTT%20Broker%20nw.png.

<https://diygeeks.org/learn/intro-to-blynk/>. (n.d.). Retrieved from <https://diygeeks.org>.

<https://esp8266-shop.com/esp8266-guide/esp8266-nodemcu-pinout/>. (n.d.). Retrieved from <https://esp8266-shop.com/>.

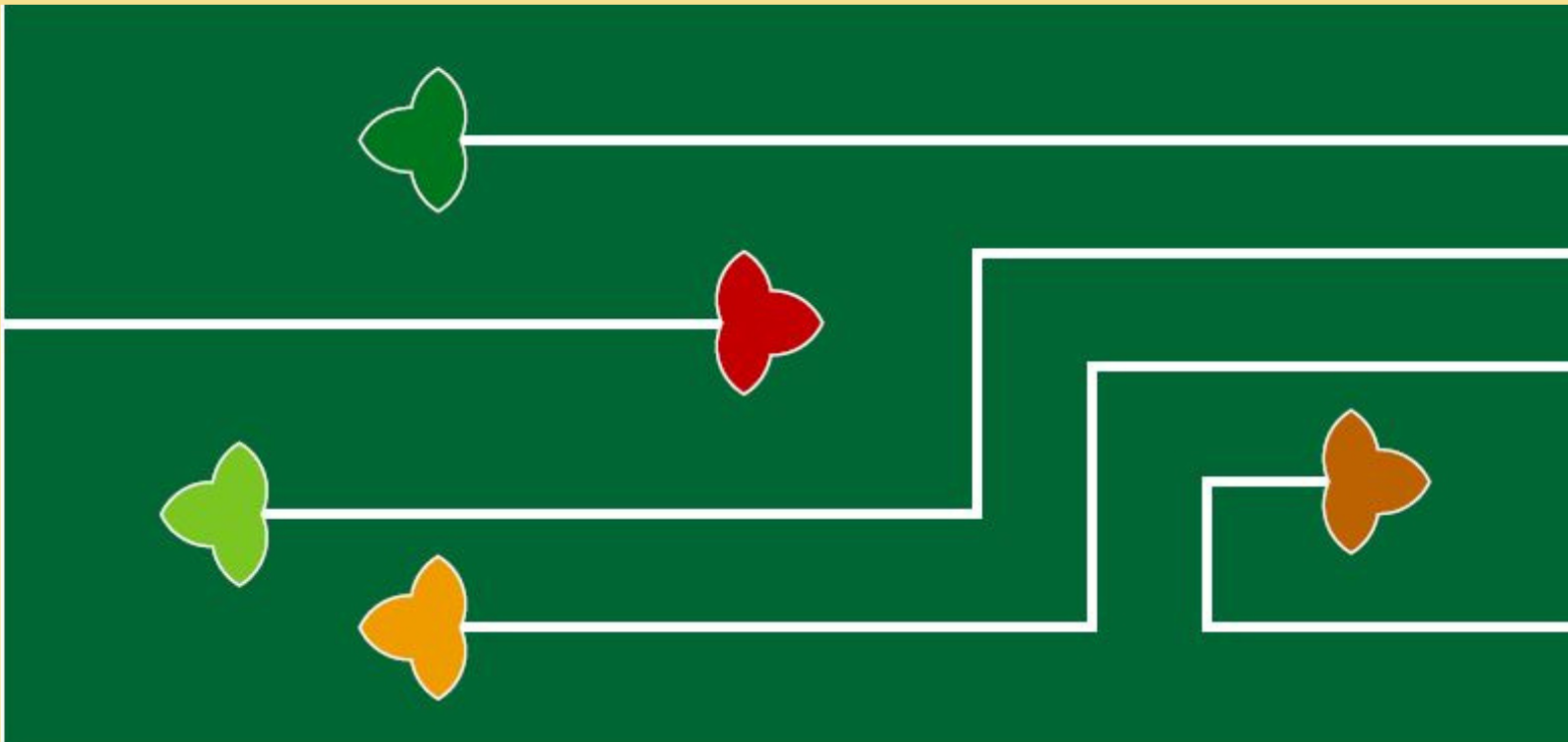
<https://realbusiness.co.uk/from-1982-coca-cola-vending-machine-to-latest-trend-what-the-internet-of-things-means-for-business/>. (n.d.).

<https://www.embarcados.com.br/introducao-ao-blynk-app/>. (n.d.). Retrieved from <https://www.embarcados.com.br/>.

Oliveira, S. d. (2017). *Internet das coisas*. novatec.

Santos, R. (n.d.). <https://randomnerdtutorials.com/>. Retrieved from <https://randomnerdtutorials.com/>.

Souza, F. (n.d.). *Controle de dispositivos remotamento com o ESP8266*.



Erasmus+

The Green Steam Incubator project has been funded with support from the European Commission. The publication reflect the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Project number: 2019-3-CY02-KA205-001692



**CITIZENS
IN POWER**



Center for Social
Innovation

