

Internet de las Cosas

LAB 1 – *Arduino*

Simulación, Implementación y Programación.

Sesión de Laboratorio

Objetivo de sesión aquí



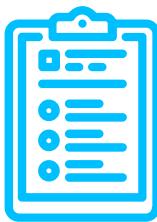
> Reinventa el mundo <

En esta experiencia de laboratorio, exploraremos diversas prácticas orientadas al aprendizaje activo de los principios fundamentales de los circuitos eléctricos y su simulación. A lo largo de las sesiones, los estudiantes desarrollarán competencias prácticas mediante el uso de plataformas digitales y componentes reales. Aquí se presentan los objetivos principales formulados de manera clara y didáctica:

1. **Simulación de Circuitos.** Utilizar Tinkercad y Falstad para construir y analizar circuitos eléctricos de forma virtual.
1. **Conexiones de Resistencias.** Aplicar configuraciones en serie y paralelo para calcular la resistencia equivalente.
1. **Montaje de Componentes Básicos.** Integrar resistencias, LEDs, potenciómetros y botones en circuitos prácticos.
1. **Diseño de un Semáforo.** Implementar un sistema de control de semáforo con lógica programada y entradas digitales.
1. **Uso de Instrumentos de Medición.** Medir voltaje, corriente y resistencia con instrumentos adecuados y conexiones correctas.

TRANSFOR**TEC**

1.



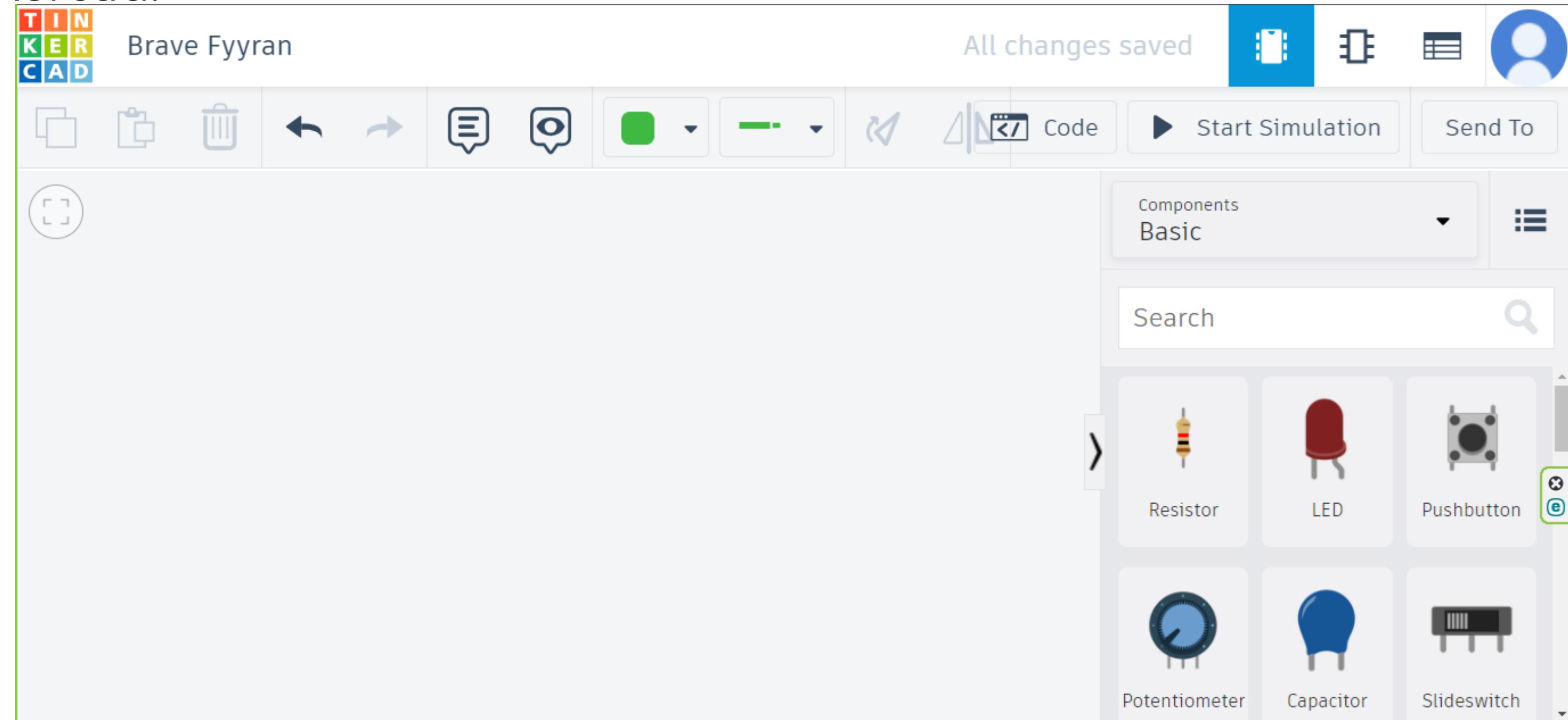
Simulación de Circuitos Eléctricos

Tinkercad y Falstad



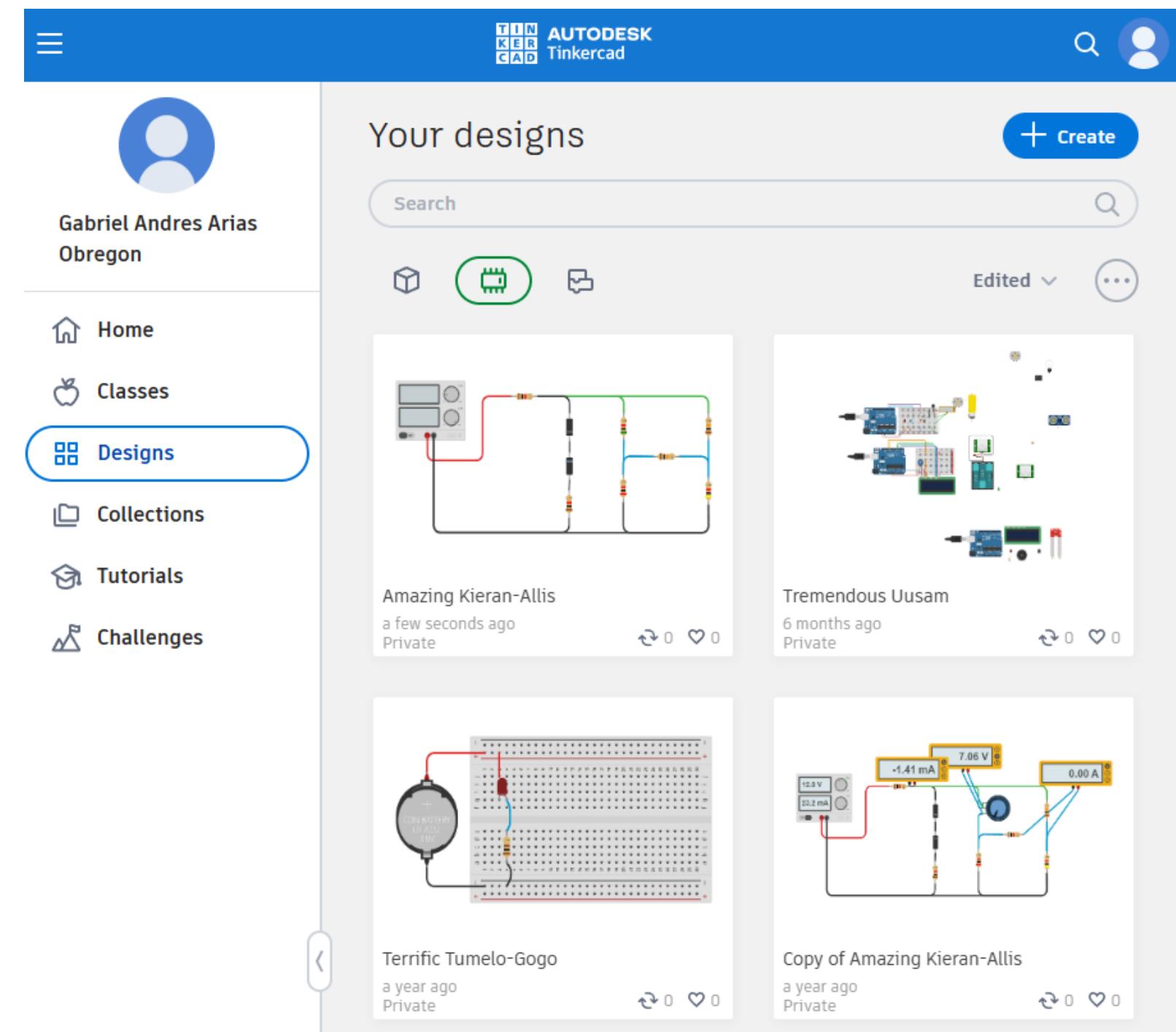
Espacio de Trabajo

En Tinkercad.



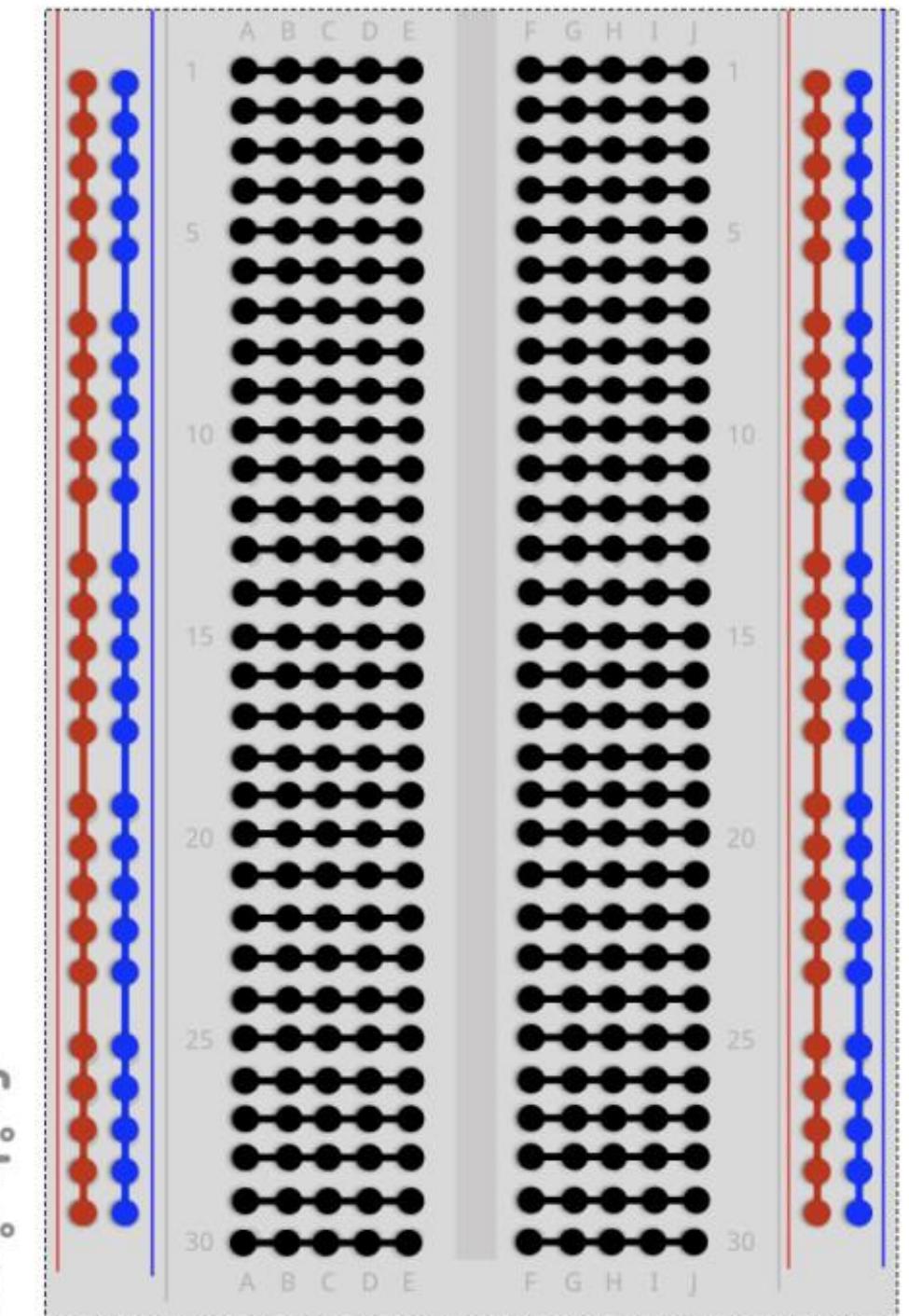
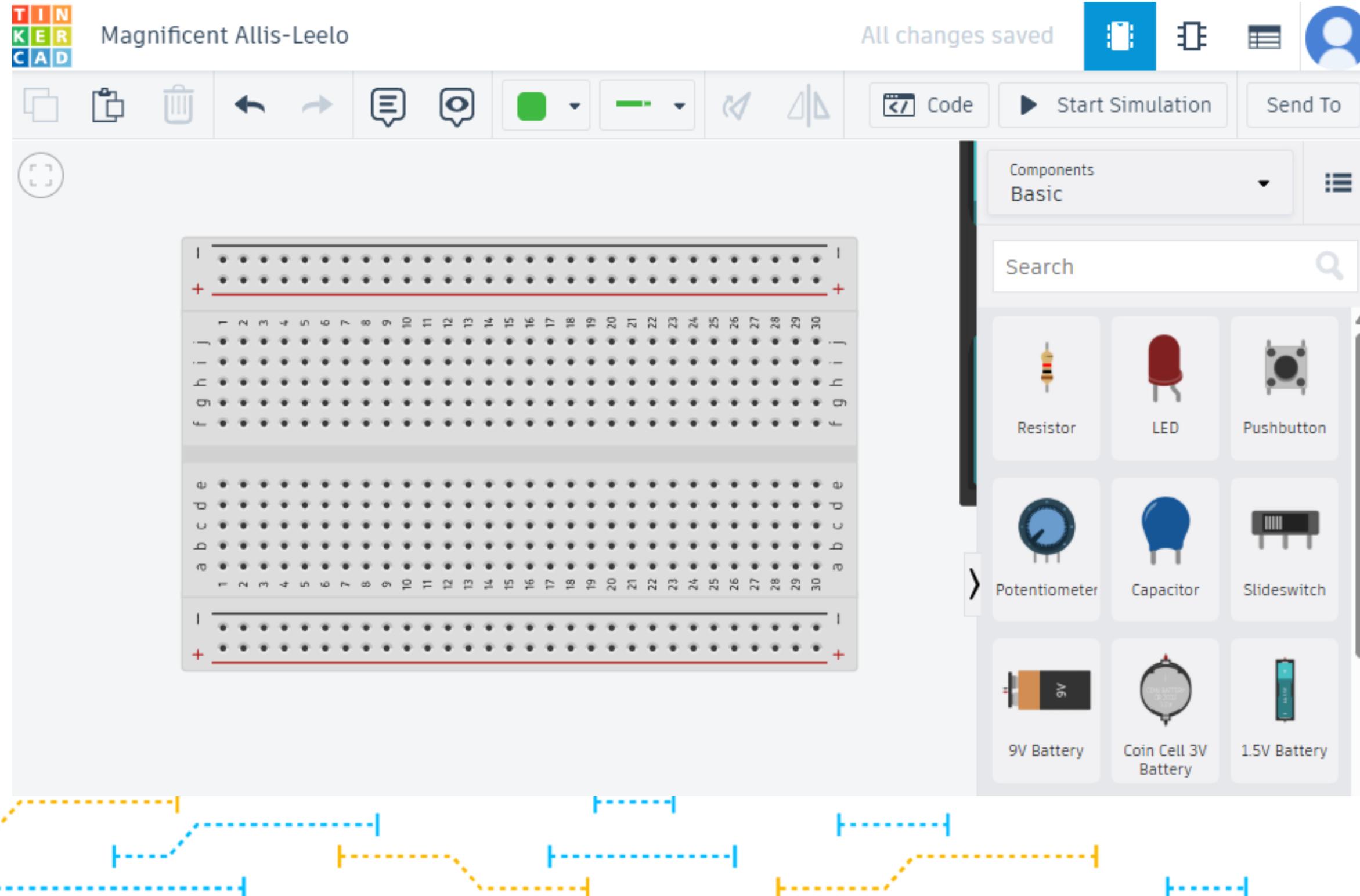
Espacio de Trabajo

En Tinkercad.



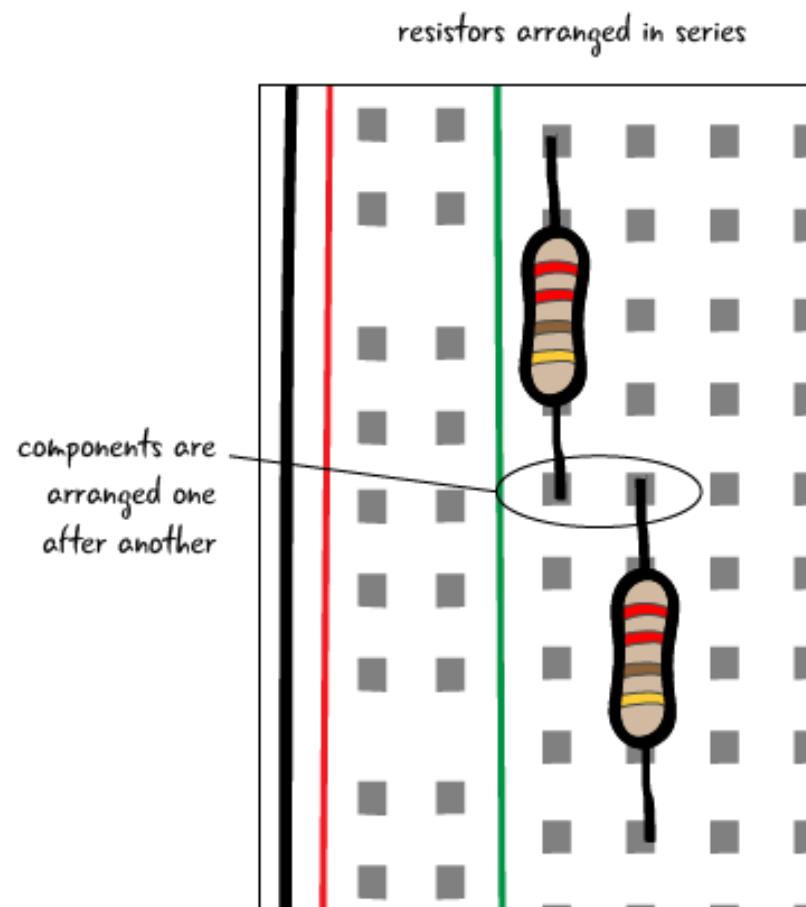
Espacio de Trabajo

En Tinkercad.



Conexiónado de Resistencias

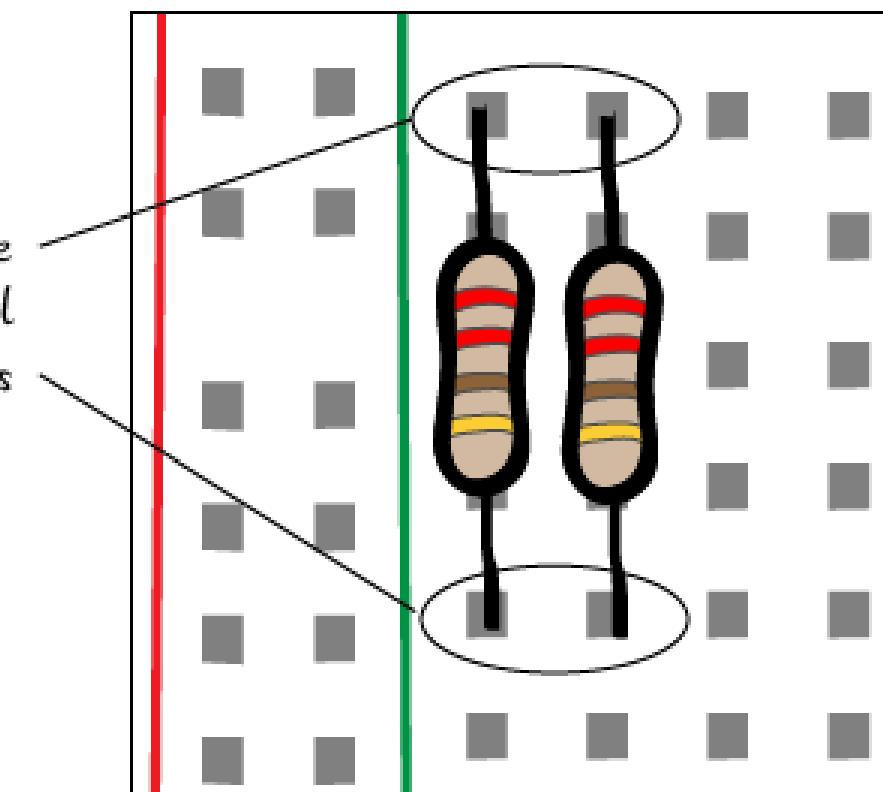
En protoboard.



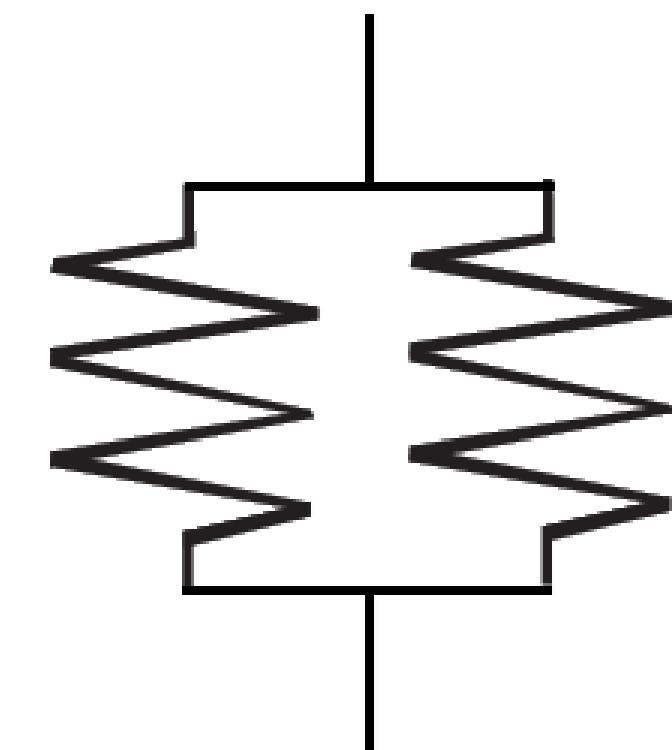
schematic of resistors
arranged in series



resistors arranged in parallel



schematic of resistors
arranged in parallel



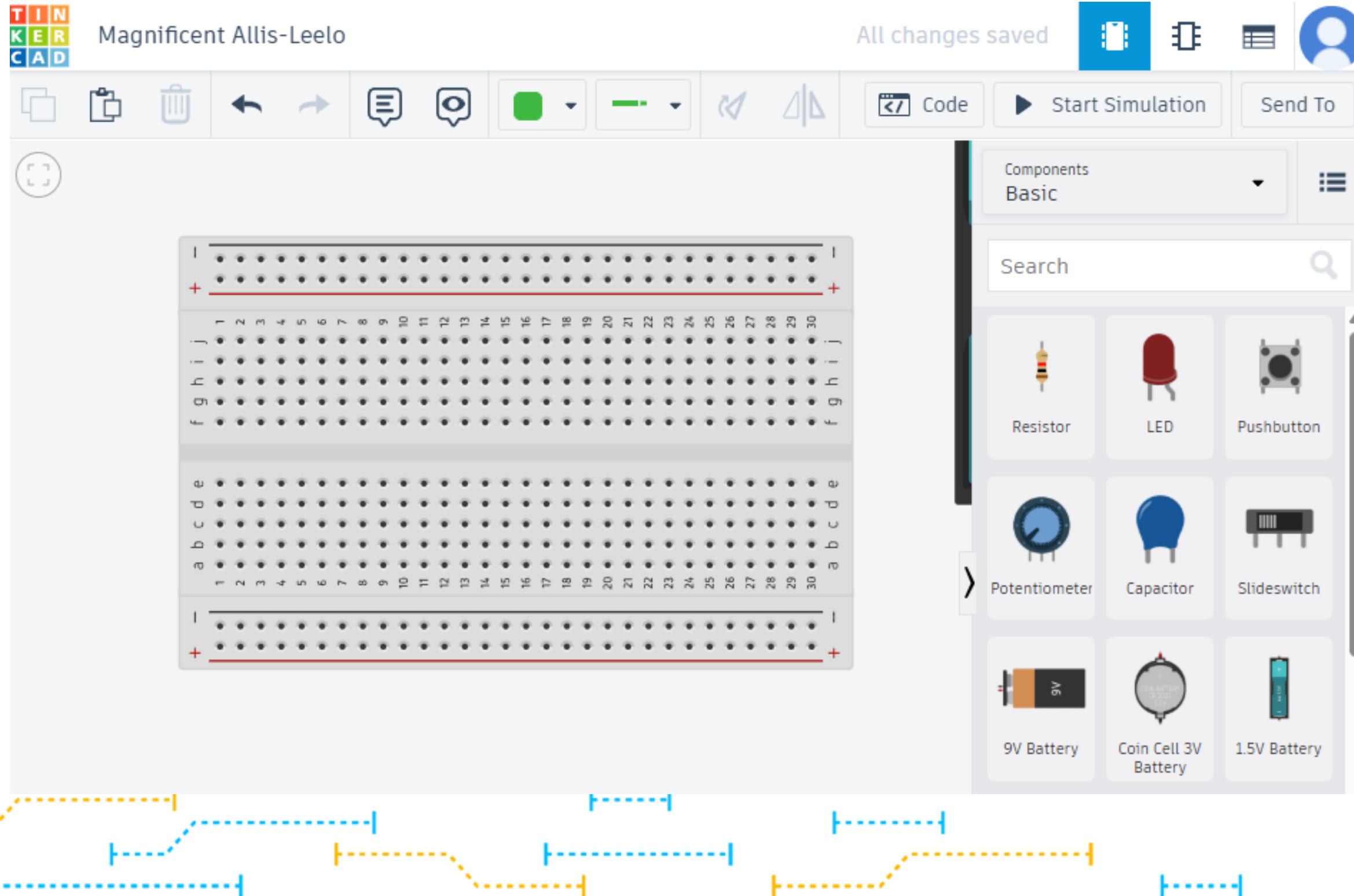
Espacio de Trabajo

En Tinkercad.

Magnificent Allis-Leelo

All changes saved

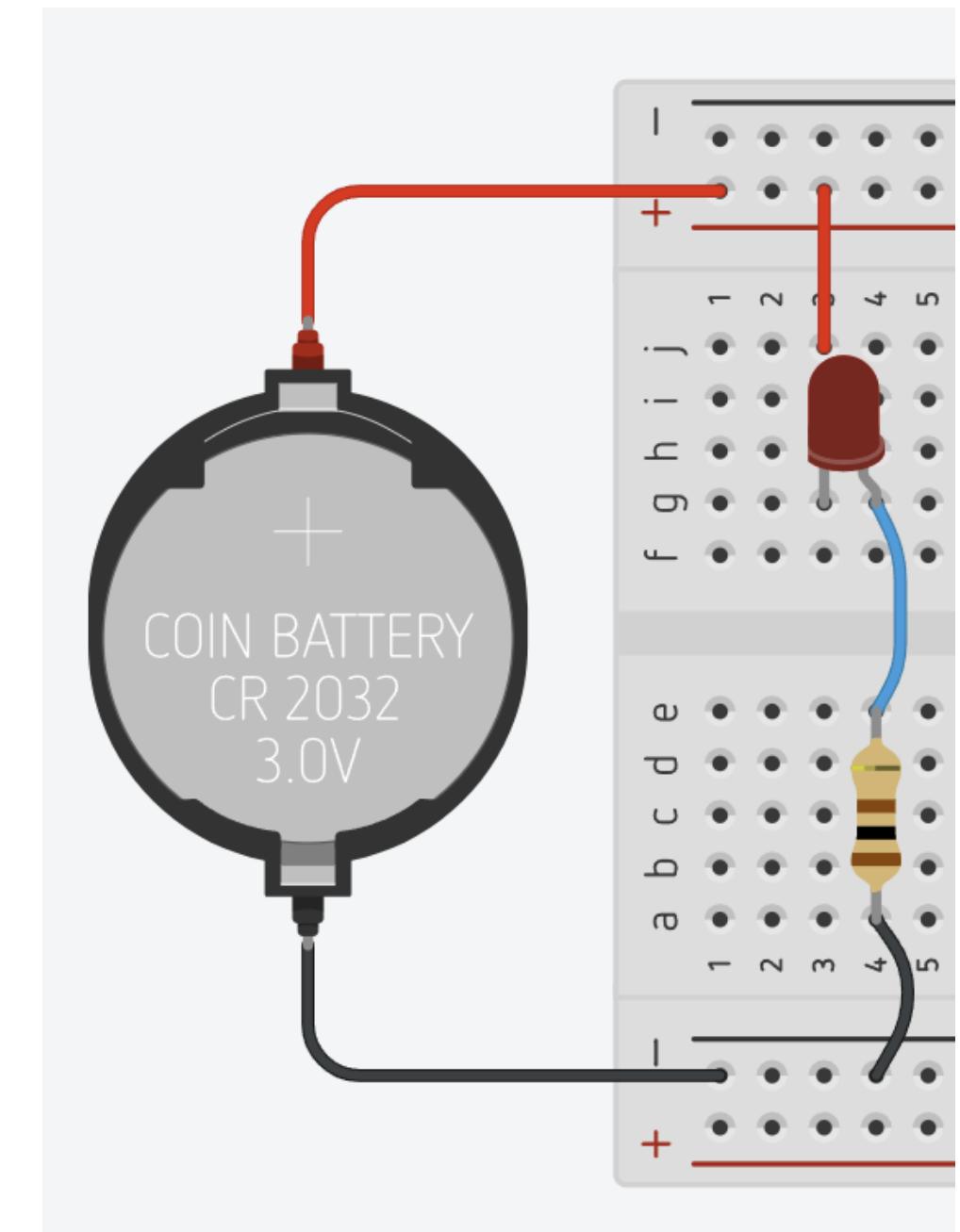
TIN KER CAD



Components Basic

Search

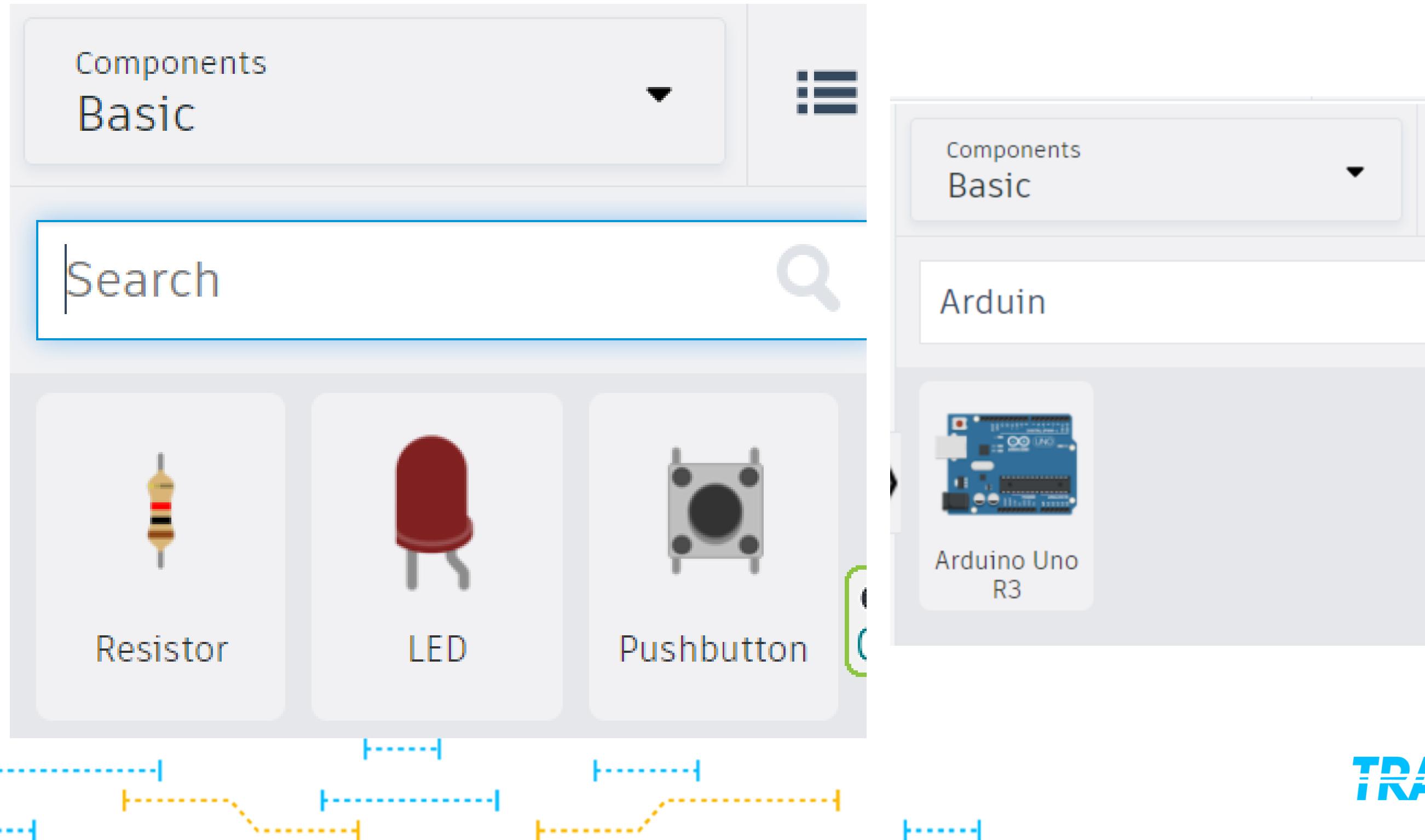
- Resistor
- LED
- Pushbutton
- Potentiometer
- Capacitor
- Slideswitch
- 9V Battery
- Coin Cell 3V Battery
- 1.5V Battery



¿Funciona?

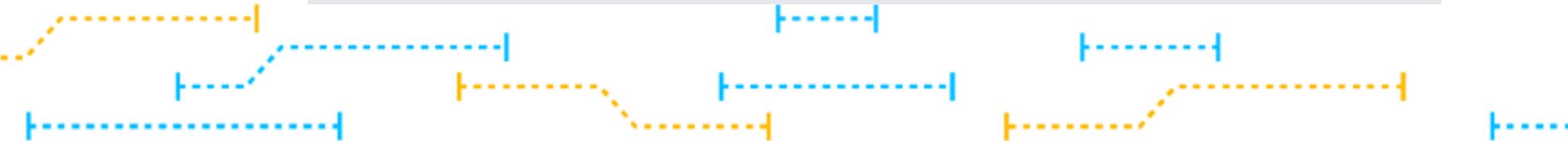
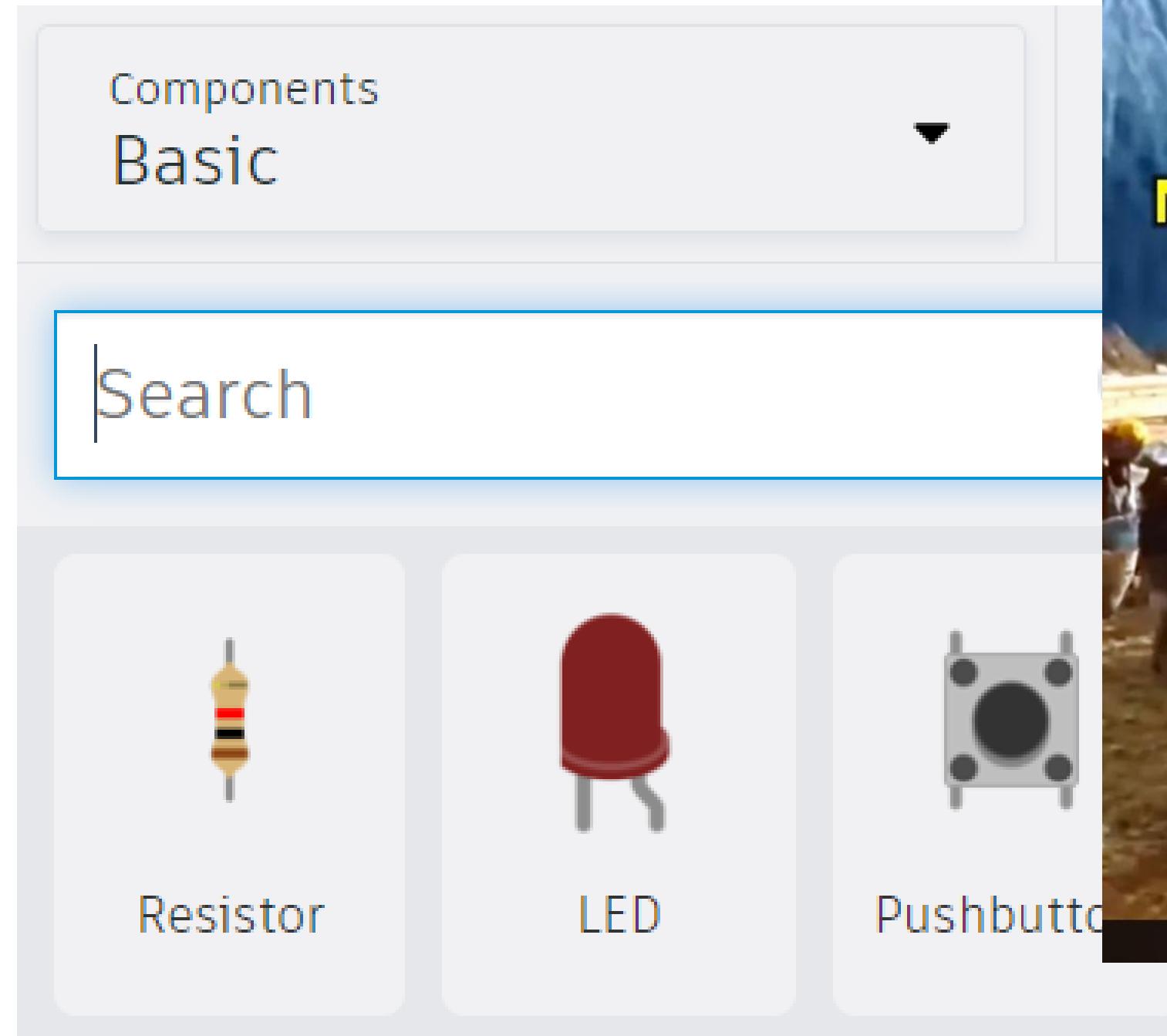
Espacio de Trabajo

En Tinkercad.



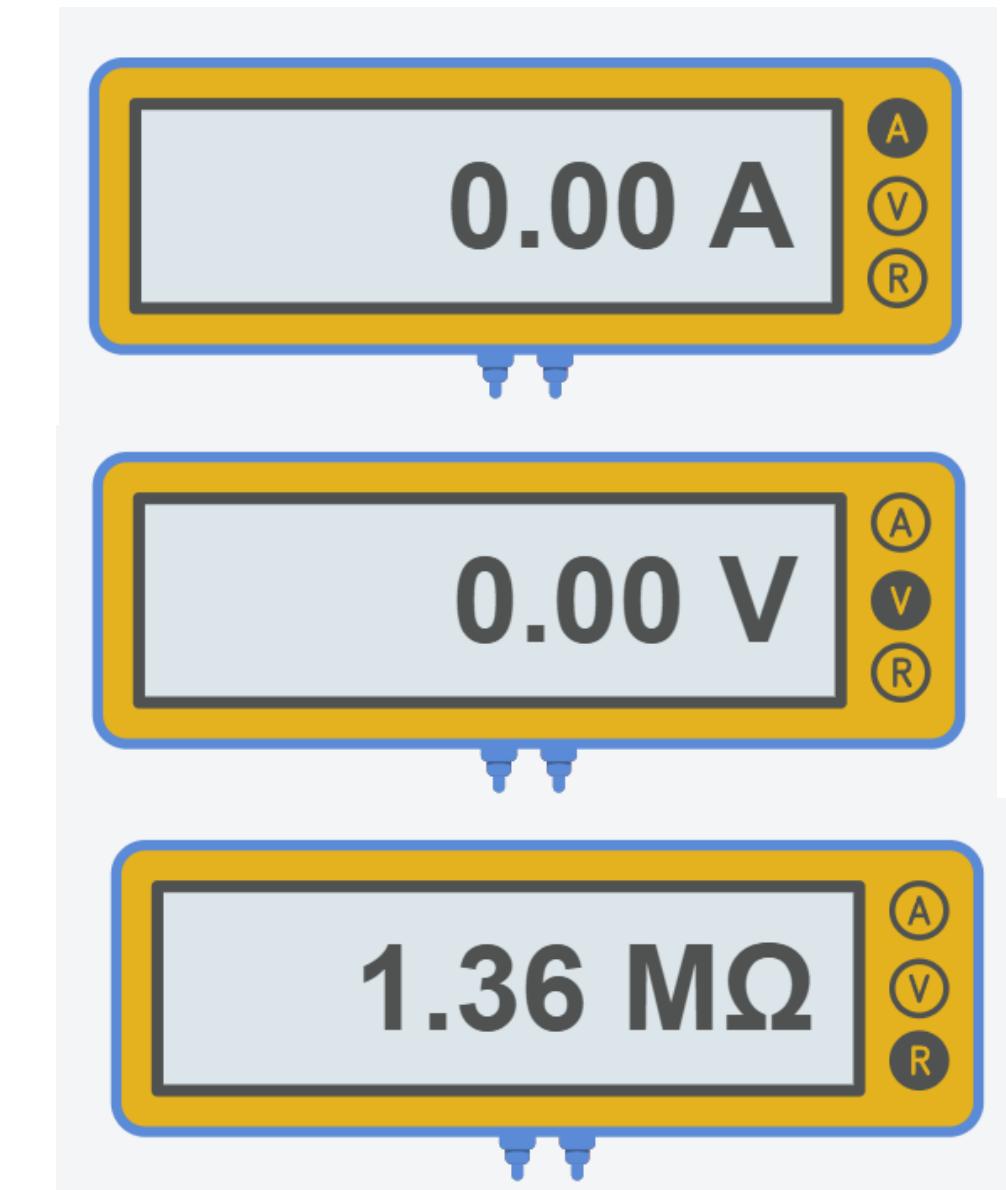
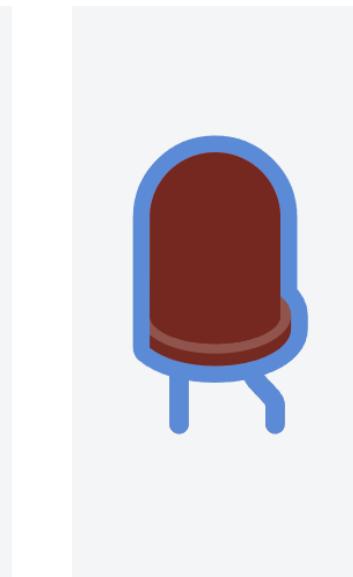
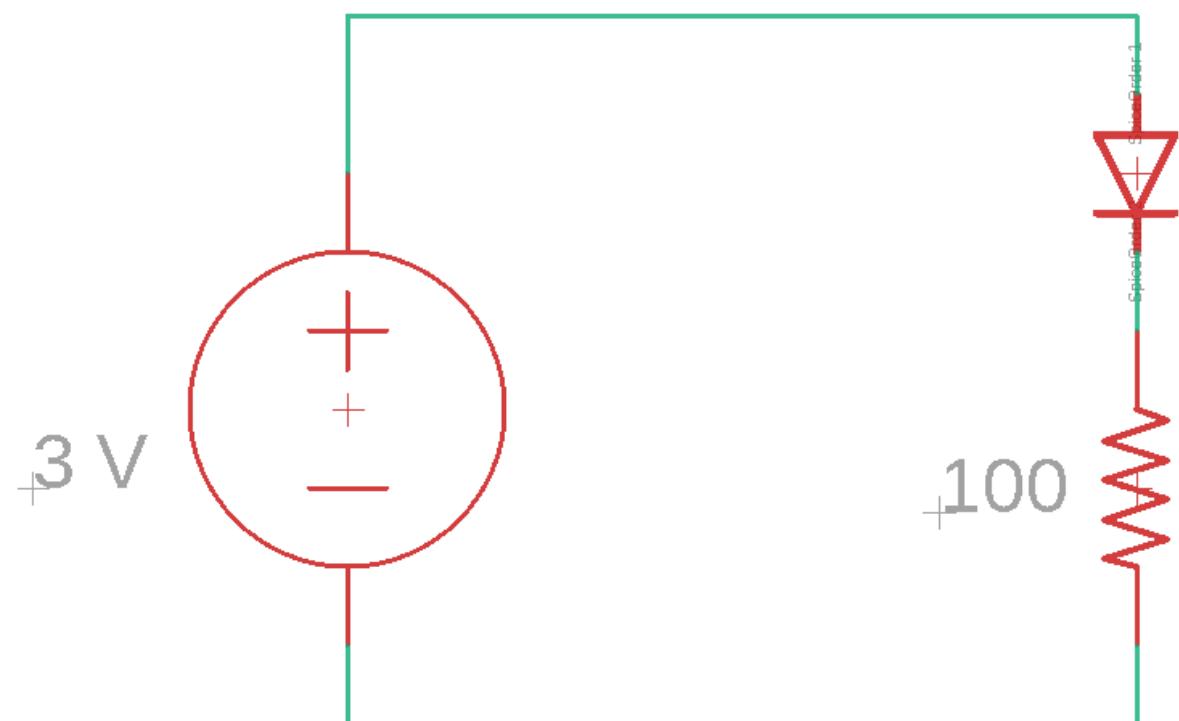
Espacio de Trabajo

En Tinkercad.

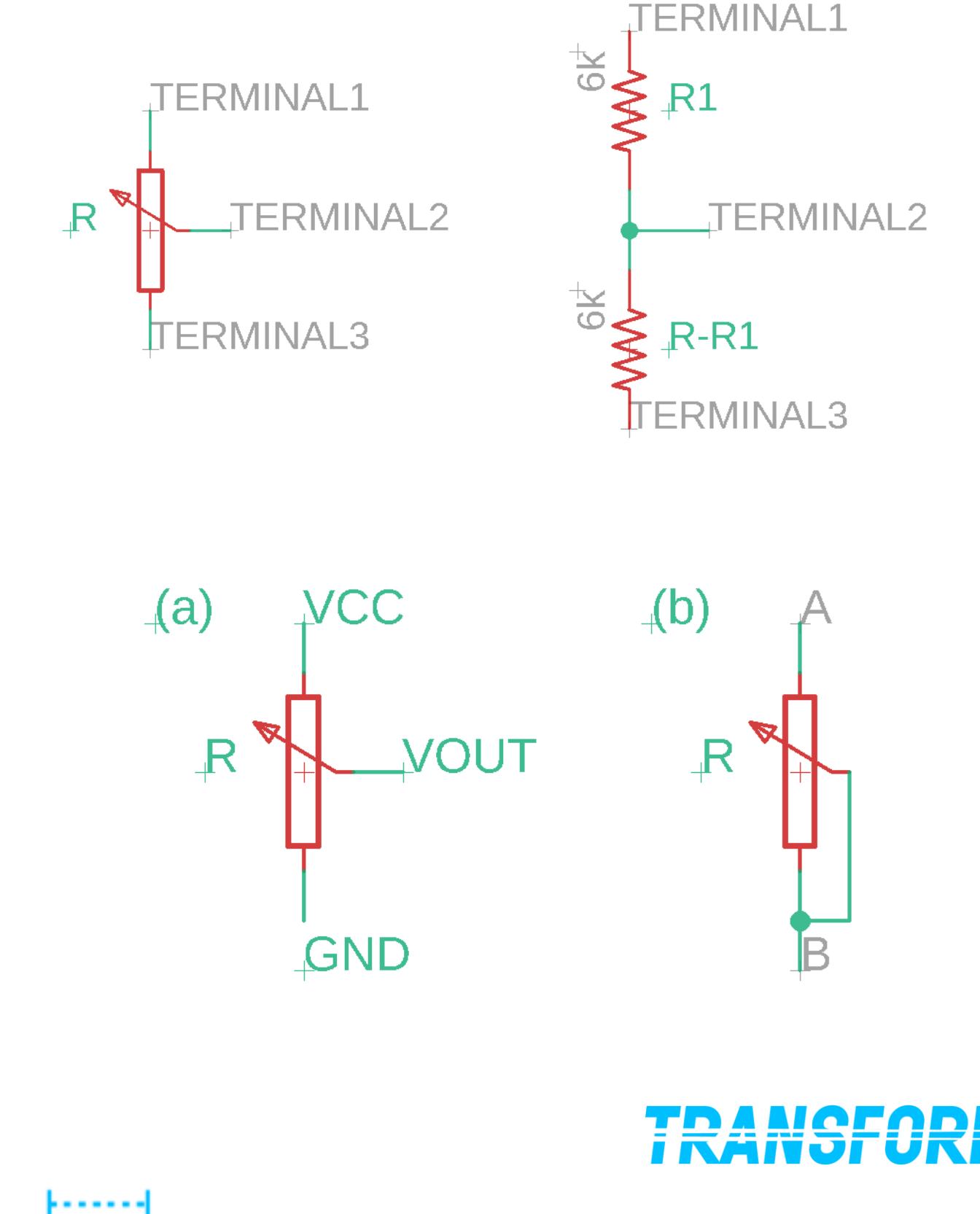
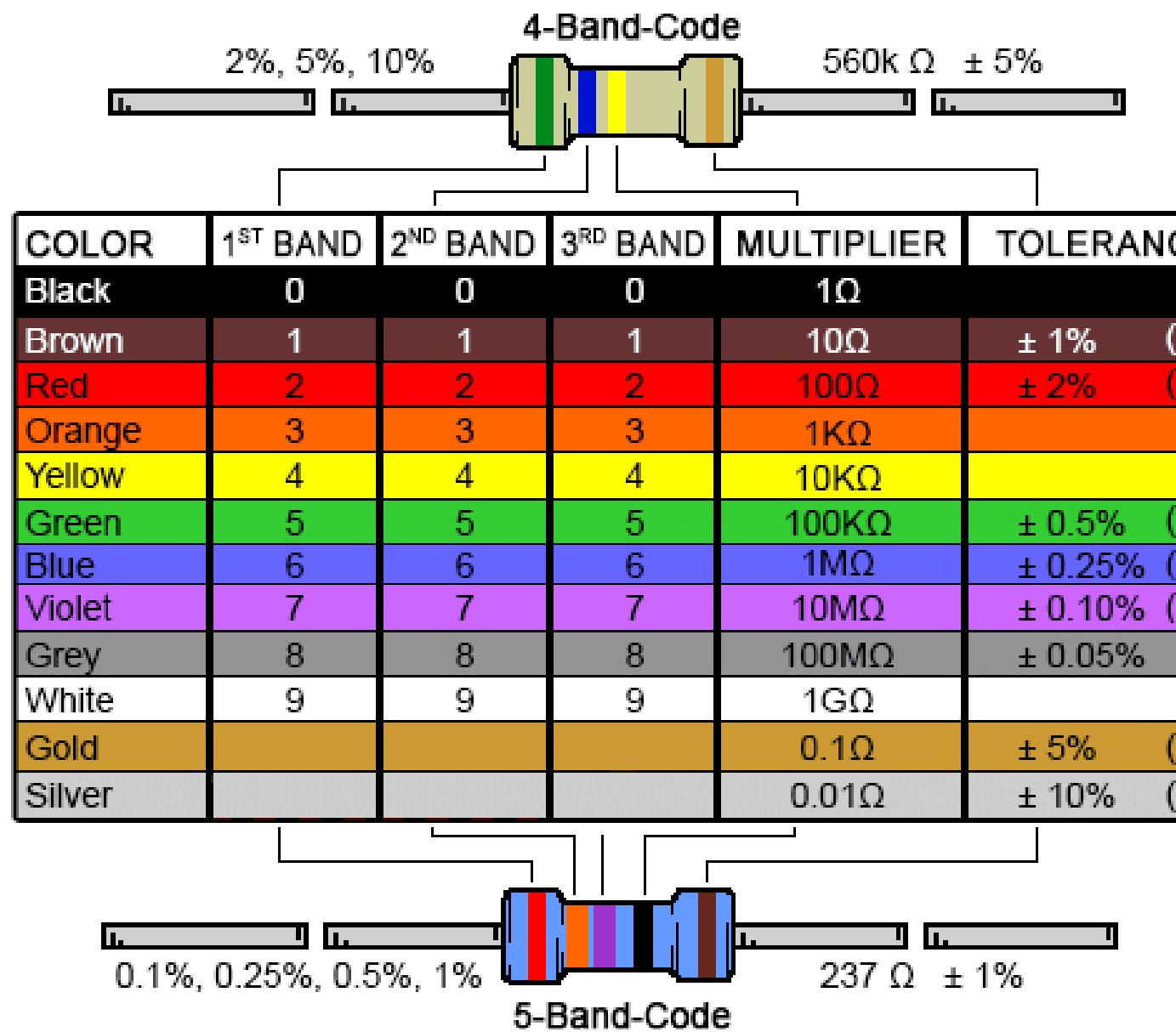


Simulación en Tinkercad

Encendido de LED y medición con herramientas.



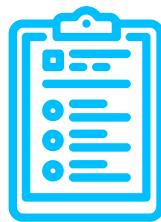
Anexo: Resistencias



Recuperación de sesiones



2.



Programación en Arduino

Tinkercad



Introducción: Microcontroladores

¿Qué es un Microcontrolador?

Una unidad de microcontrolador es esencialmente una computadora pequeña en un solo chip. Está diseñado para administrar tareas específicas dentro de un sistema integrado sin requerir un sistema operativo complejo.

Estos circuitos integrados compactos (IC) contienen un núcleo de procesador (o núcleos), memoria de acceso aleatorio (RAM) y memoria de solo lectura programable borrable eléctricamente (EEPROM) para almacenar los programas personalizados que se ejecutan en el microcontrolador, incluso cuando la unidad está desconectada de una fuente de alimentación.



Microcontrolador PIC16F876A de Microchip



Introducción: Microcontroladores

¿Cuál es la diferencia con un microprocesador?

Un **microprocesador** (como los de una computadora) necesita componentes externos para funcionar, como memoria y dispositivos de entrada/salida.

Los **microcontroladores** integran periféricos de procesamiento, memoria y entrada/salida (E/S), incluidos temporizadores, contadores y convertidores de analógico a digital (ADC), en una unidad independiente eficiente y rentable.



Microcontrolador PIC16F876A
de Microchip



Microprocesador Intel Core i7

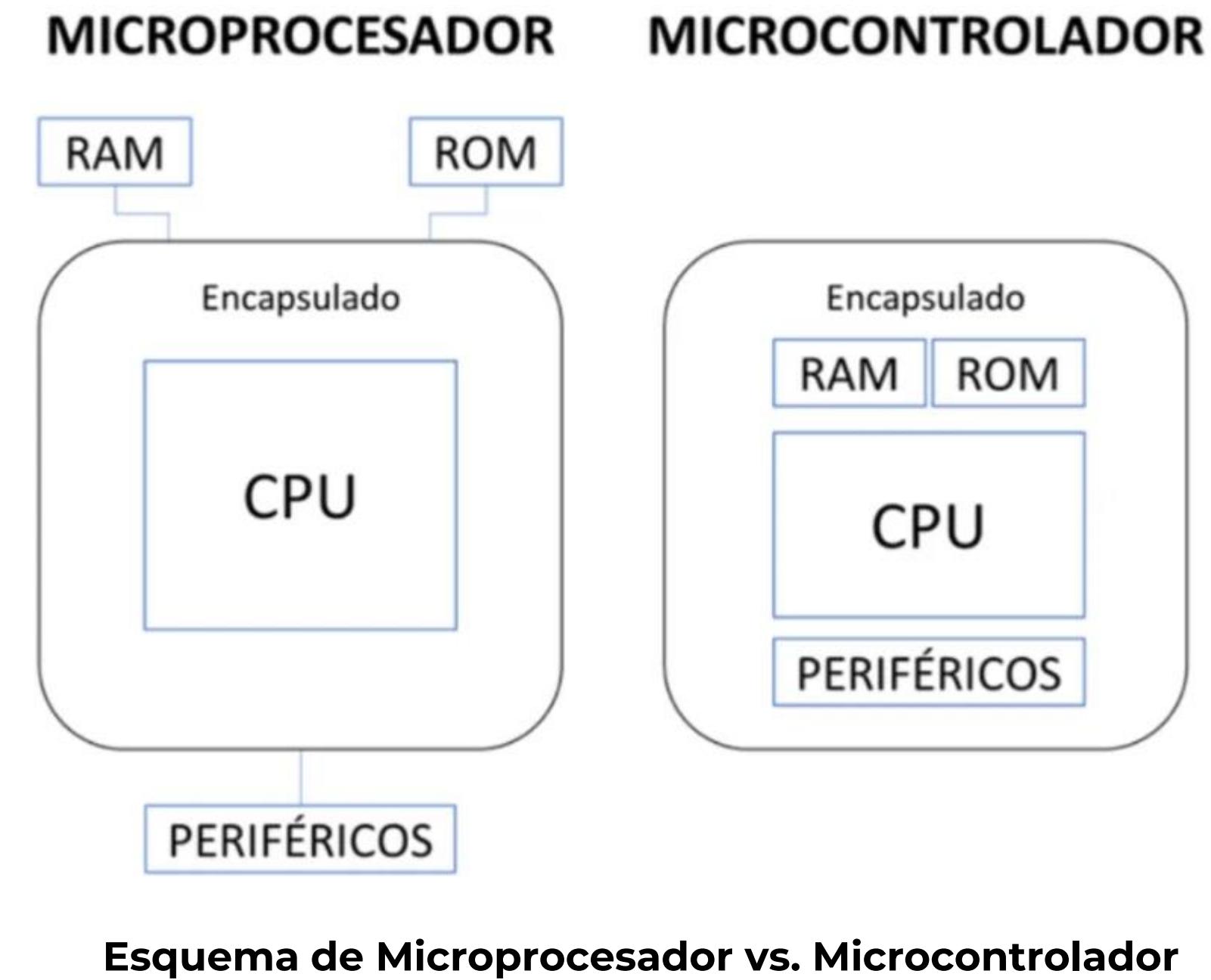


Introducción: Microcontroladores

¿Cuál es la diferencia con un microprocesador?

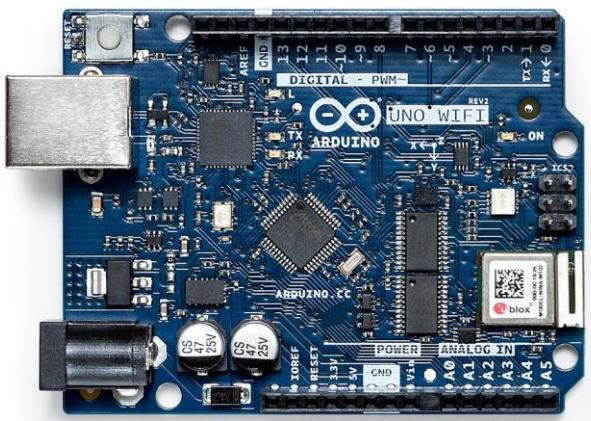
Un **microprocesador** (como los de una computadora) necesita componentes externos para funcionar, como memoria y dispositivos de entrada/salida.

Los **microcontroladores** integran periféricos de procesamiento, memoria y entrada/salida (E/S), incluidos temporizadores, contadores y convertidores de analógico a digital (ADC), en una unidad independiente eficiente y rentable.

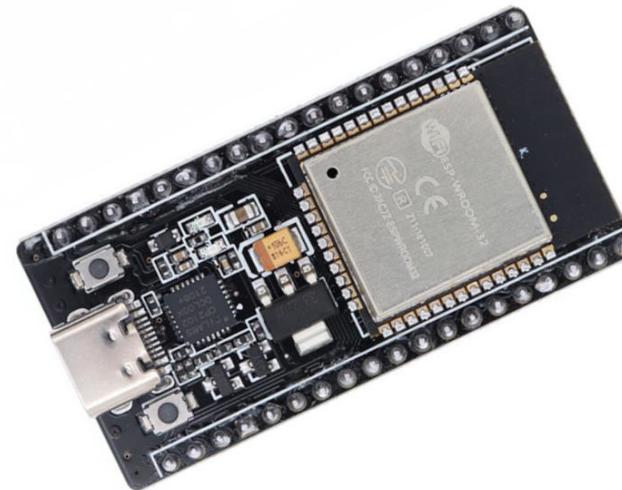


Introducción: Microcontroladores

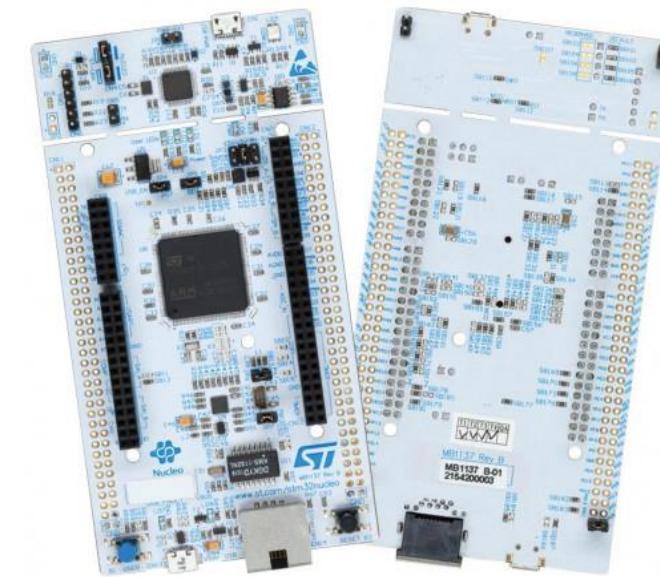
Diversas placas de desarrollo



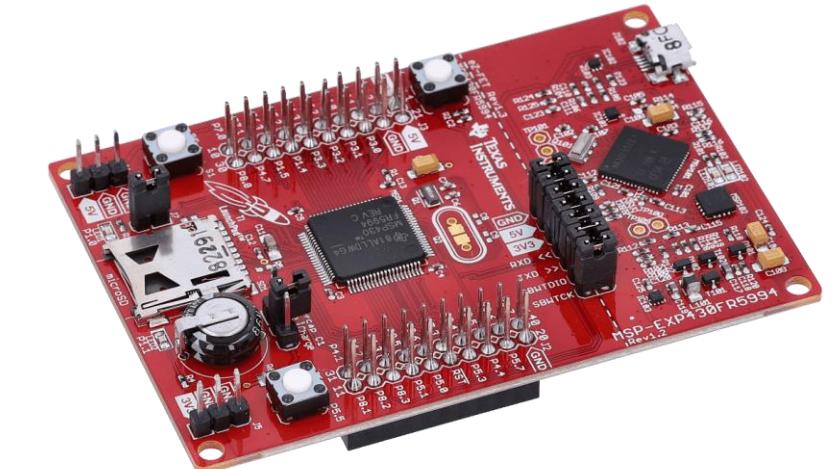
Arduino UNO



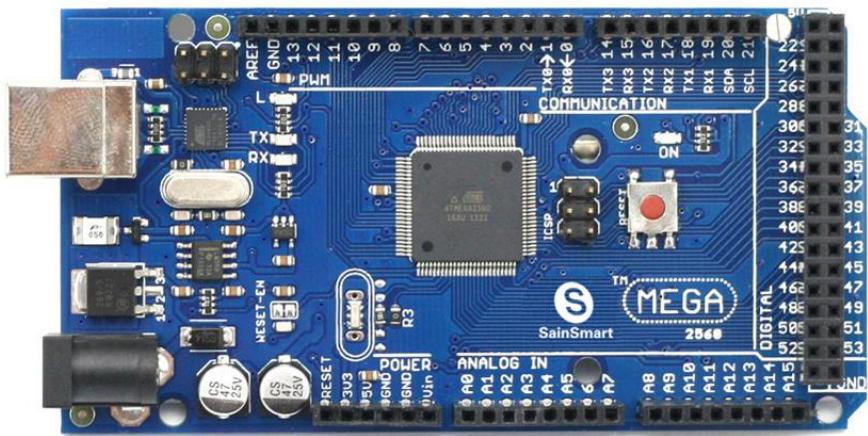
ESP32



STM Nucleo



MSP-ESP430G2ET (MSP430)



Arduino MEGA



Raspberry Pi



PIC18 Explorer



RAKWisblock (STM)

Introducción a Arduino

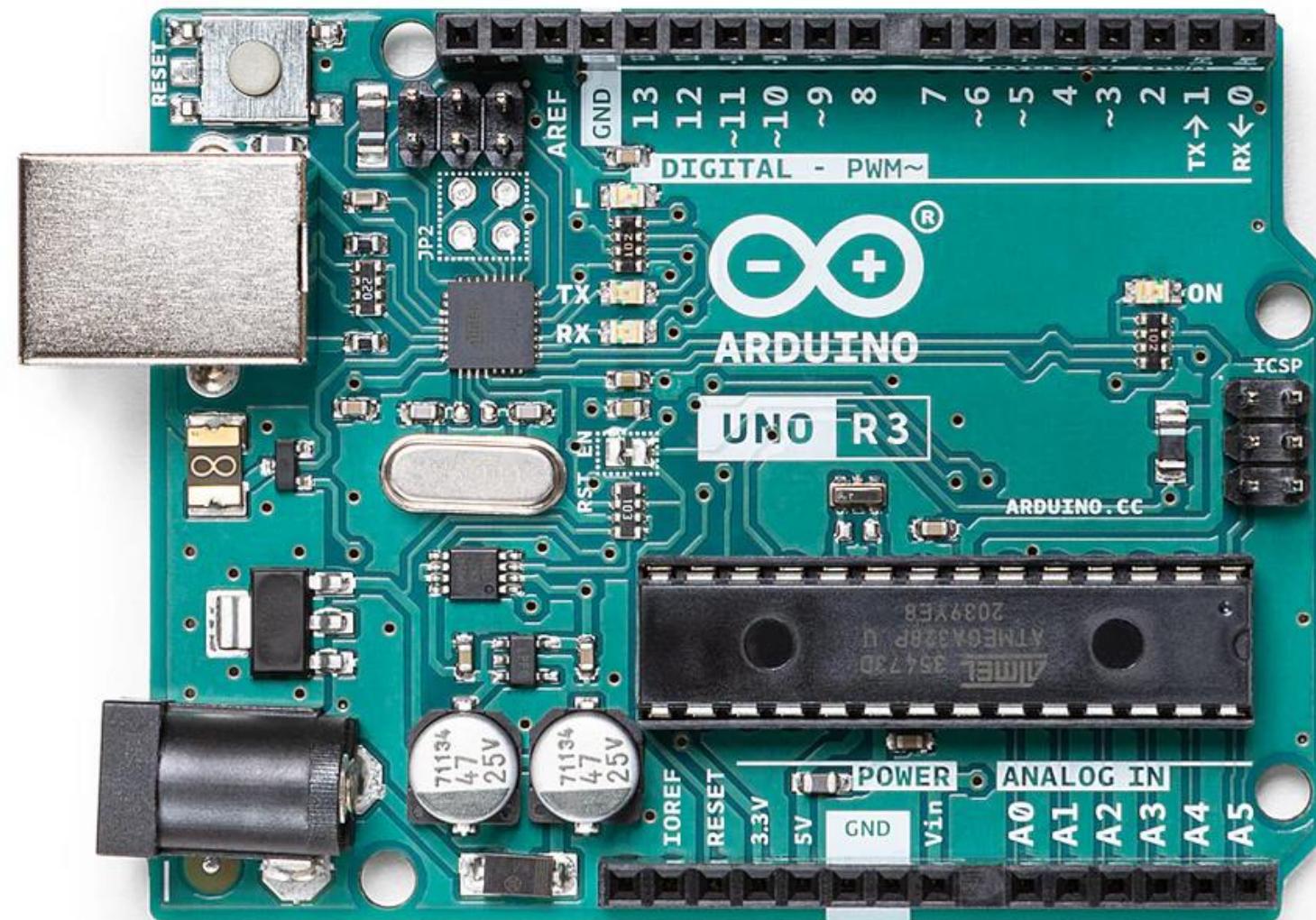
¿Qué es Arduino?

- Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar.
- Es una plataforma de prototipado electrónico de código abierto que consta de hardware y software fácil de usar.
- El hardware de Arduino generalmente se refiere a la Breakout Board o conocido también como placa de desarrollo, que es una PCB que incluye un microcontrolador y una serie de pines de entrada/salida que permiten a los usuarios conectar diversos componentes electrónicos y sensores.
- El software de Arduino IDE consiste en un entorno de desarrollo integrado que facilita la escritura, carga y ejecución de código en la placa Arduino.

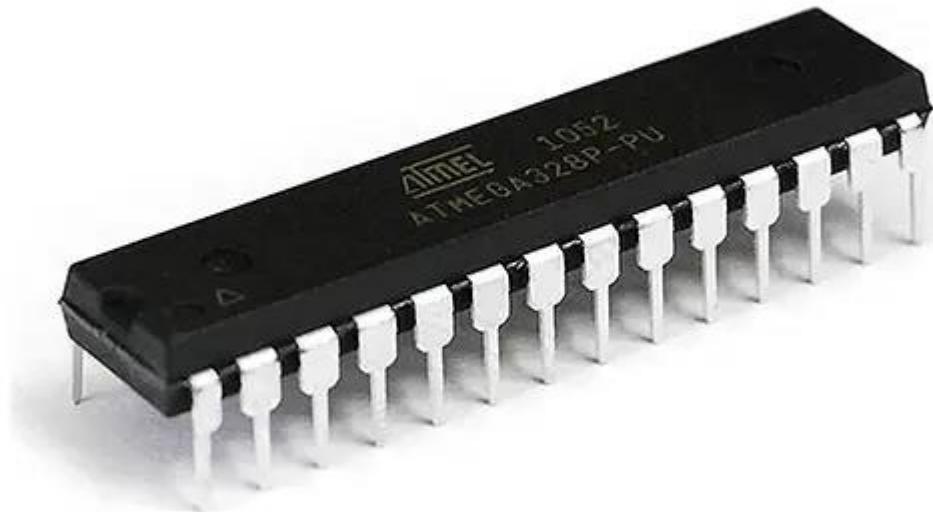


Introducción a Arduino

¿Qué es Arduino?



Arduino Uno Rev3

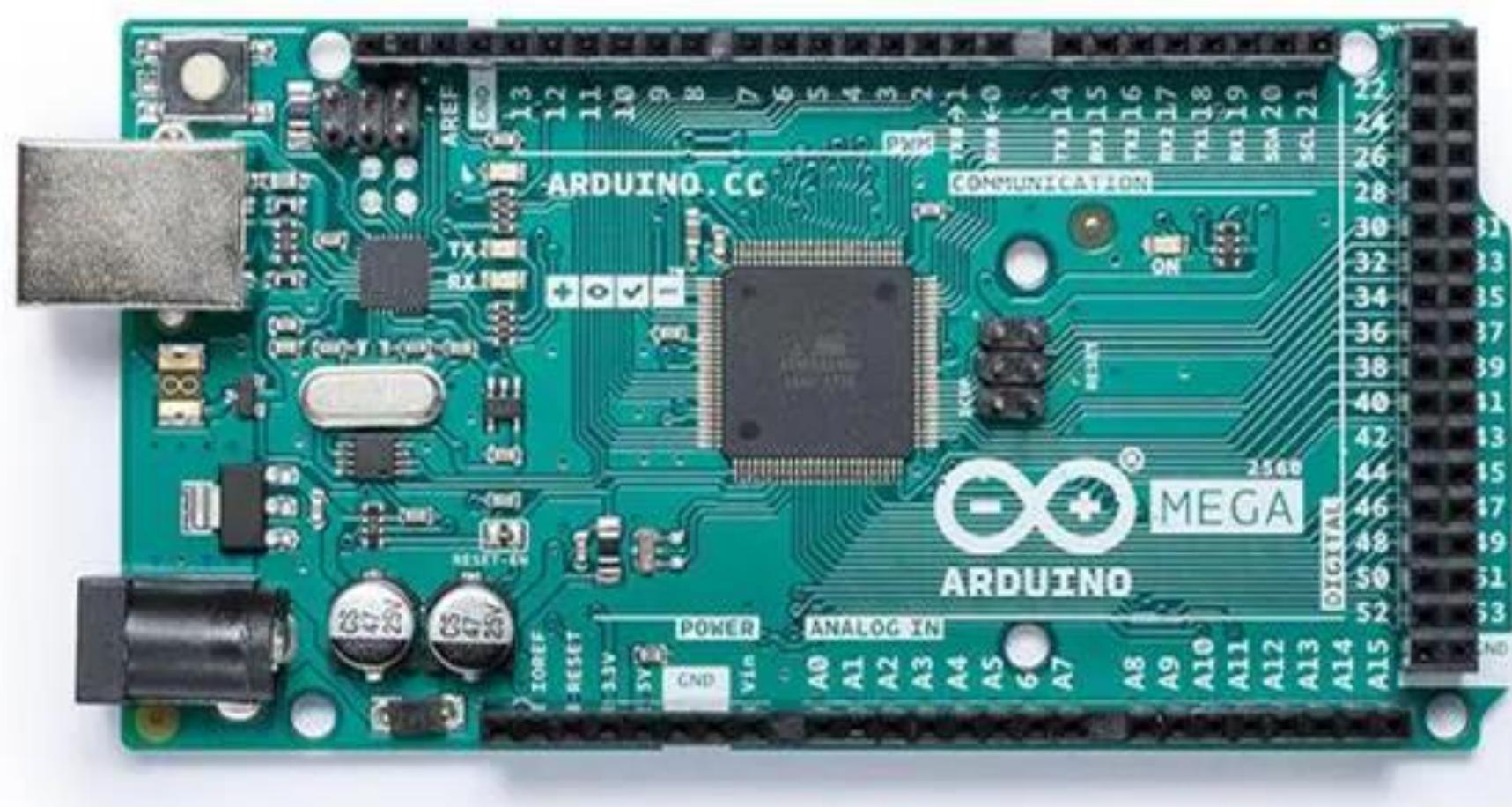


Microcontrolador ATMEGA328P

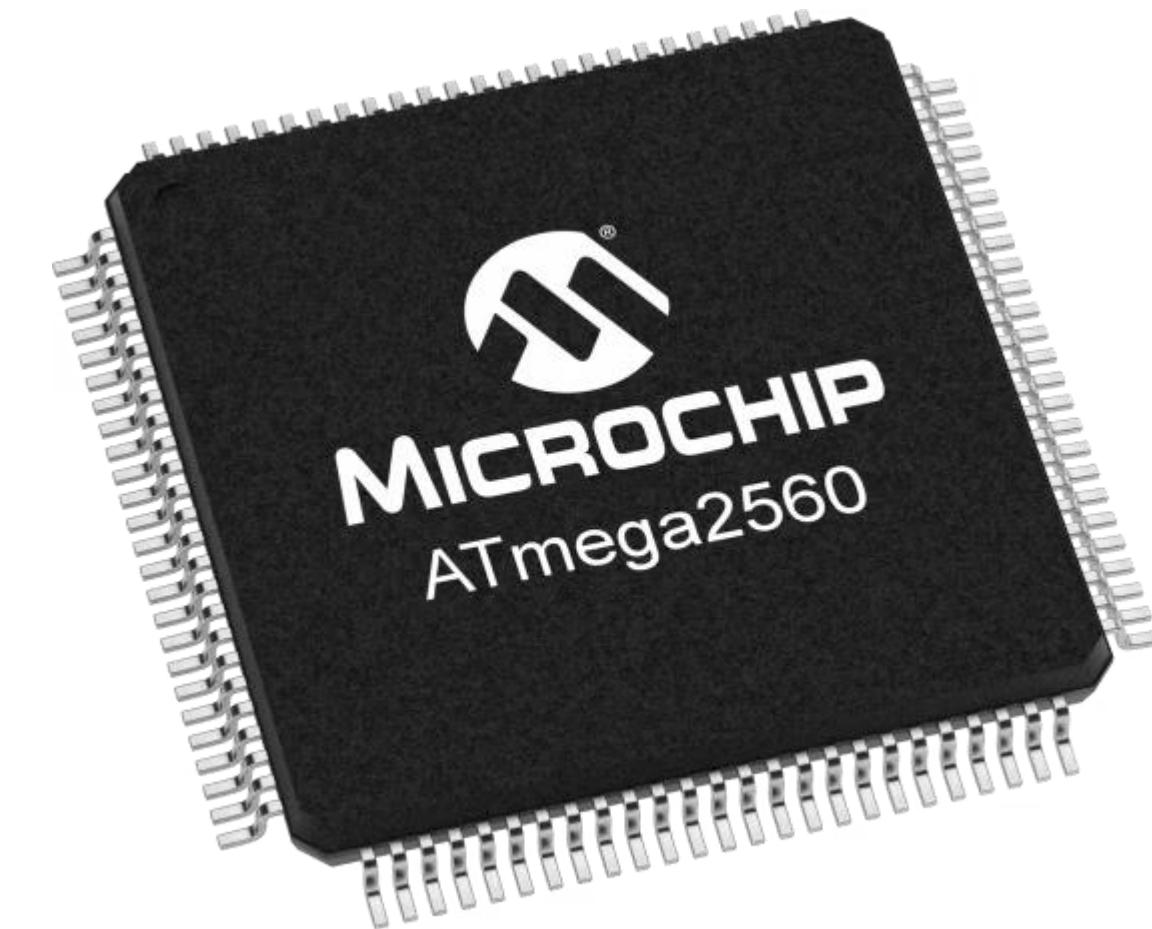


Introducción a Arduino

¿Qué es Arduino?



Arduino MEGA



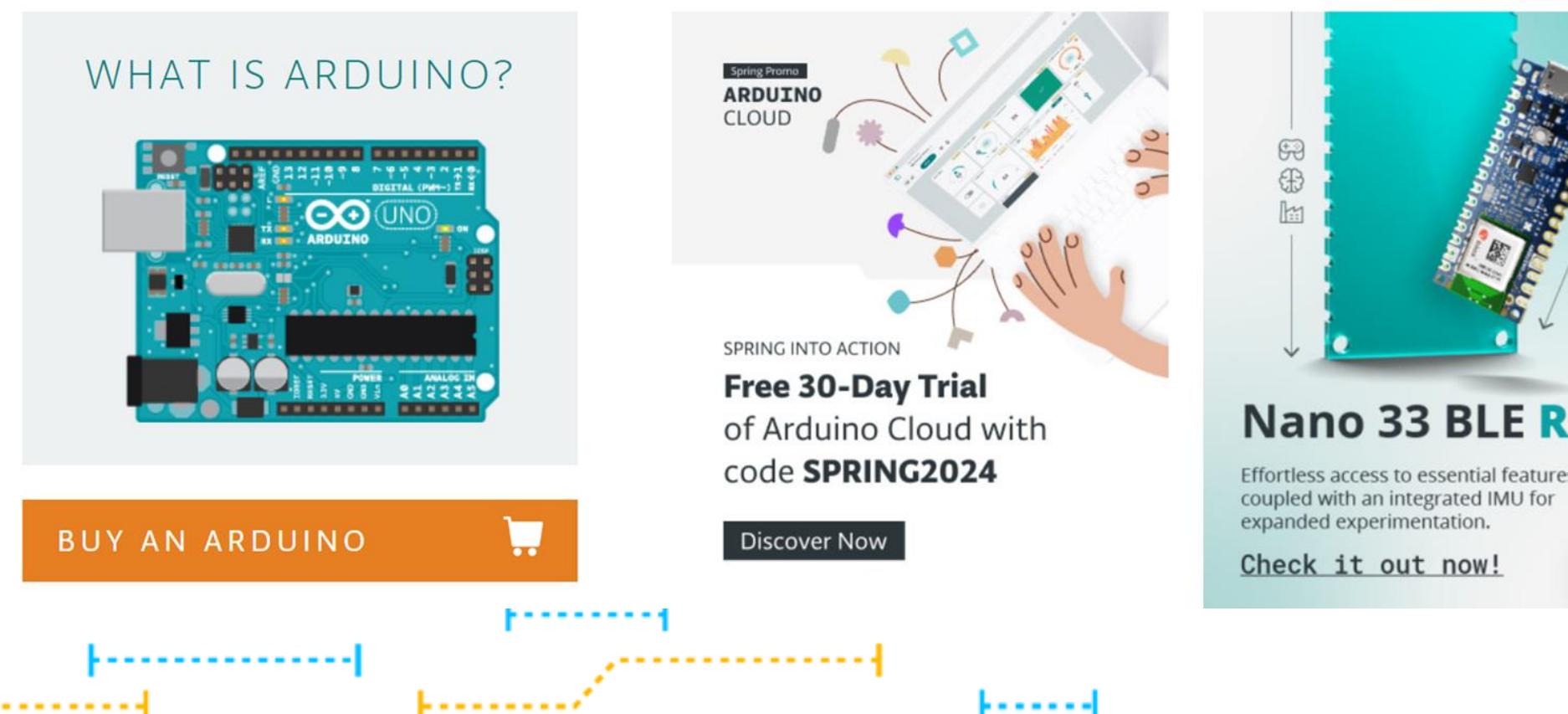
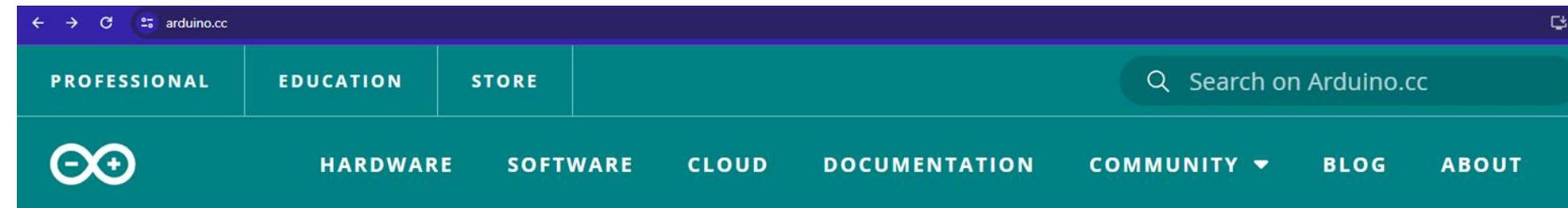
Microcontrolador ATMEGA2560



Introducción a Arduino

¿Qué es Arduino?

La página oficial de Arduino (www.arduino.cc) posee una gran cantidad de información tanto del hardware como del software, así mismo de tutoriales y librerías para una experiencia de desarrollo amigable.



The image displays three distinct promotional sections from the Arduino website:

- WHAT IS ARDUINO?**: Shows a close-up image of an Arduino Uno board. Below it is a large orange button labeled "BUY AN ARDUINO" with a shopping cart icon.
- ARDUINO CLOUD**: Features a hand interacting with a smartphone displaying the Arduino Cloud interface. Text includes "Spring Promo", "ARDUINO CLOUD", "SPRING INTO ACTION", "Free 30-Day Trial", "of Arduino Cloud with code SPRING2024", and a "Discover Now" button.
- Nano 33 BLE Rev2**: Shows a close-up of the Nano 33 BLE Rev2 board. Text includes "Effortless access to essential features coupled with an integrated IMU for expanded experimentation." and a "Check it out now!" button.

At the bottom of the page, there are decorative wavy lines in yellow and blue.

Introducción a Arduino

Comparativa de placas

FOR MAKERS

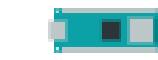
FOR PROFESSIONALS



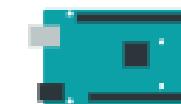
MKR Family



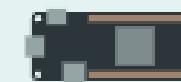
Classic



Nano Family



Mega



Portenta Family



Pro Solutions and
Kits



Nicla Family



Opta

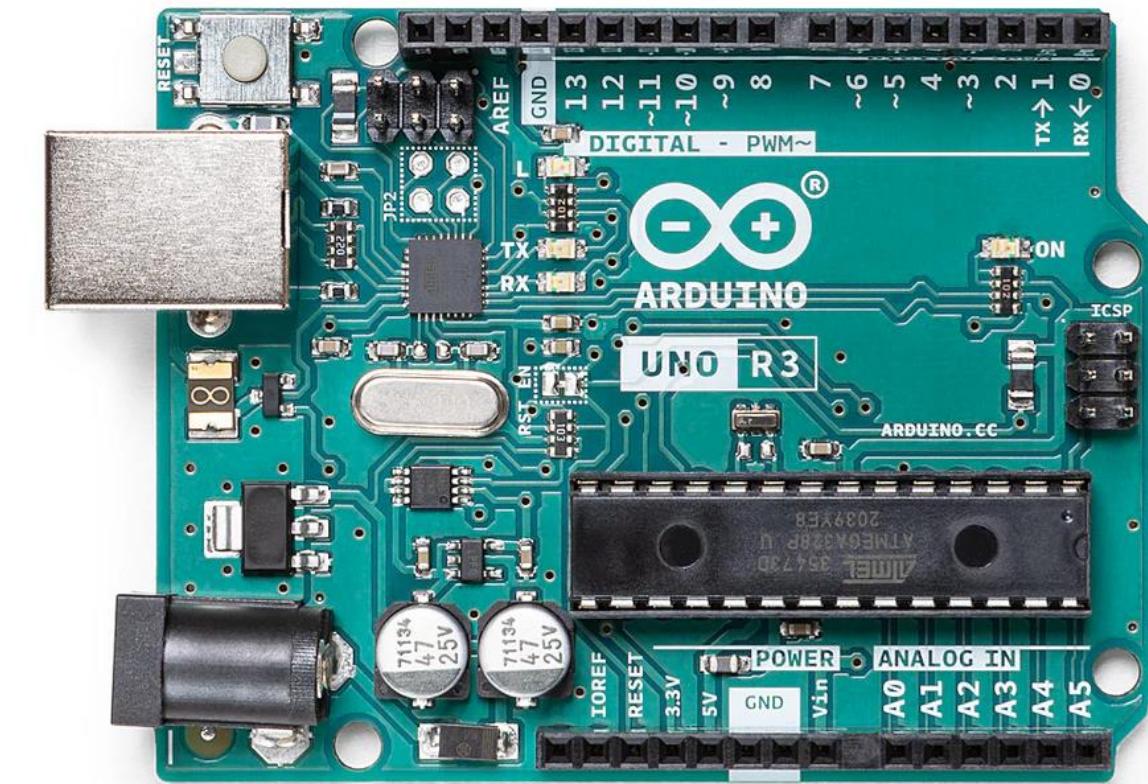


Arduino UNO R3

ATMEGA328P

El Arduino Uno R3 es una de las placas Arduino más populares y ampliamente utilizadas

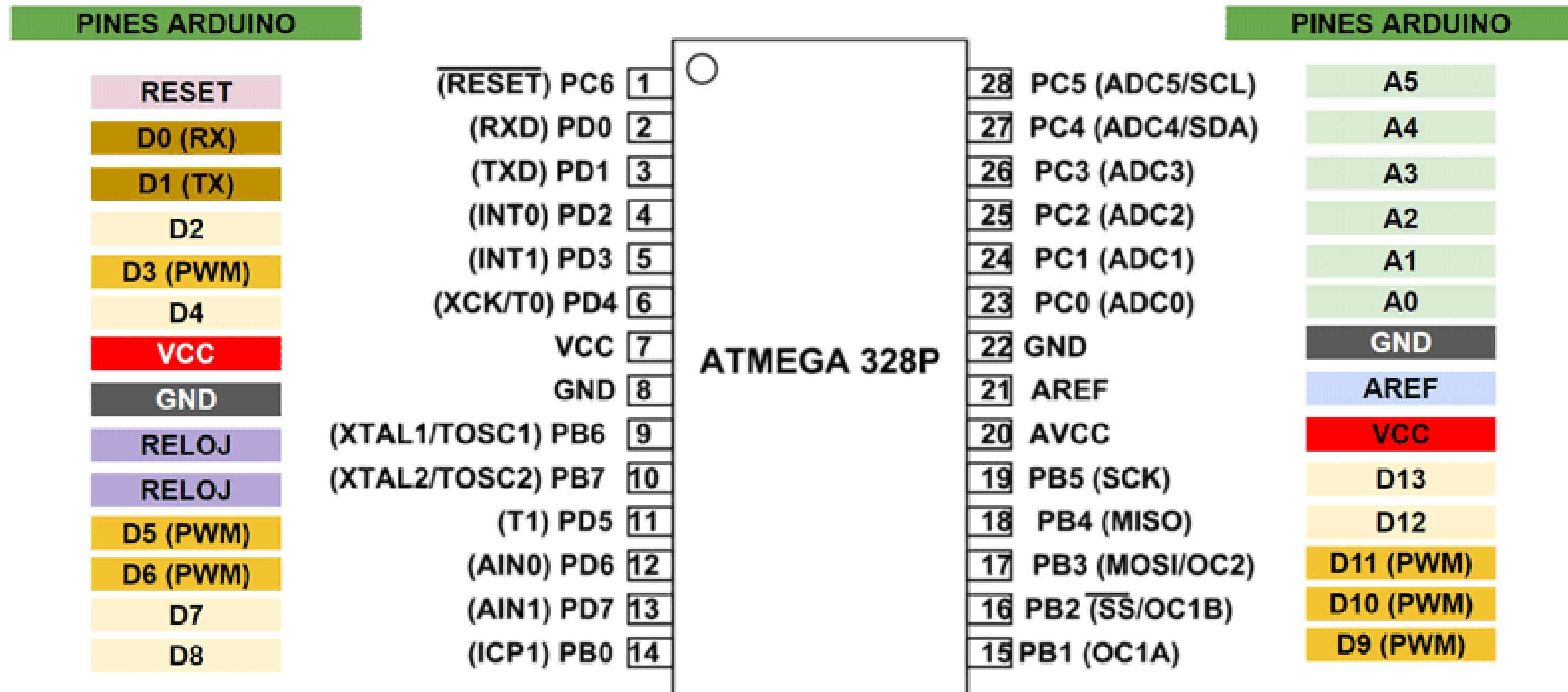
- Microcontrolador: Utiliza un microcontrolador ATmega328P.
- Pines de entrada/salida (E/S): Dispone de 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas PWM (Modulación por Ancho de Pulso) y 6 como entradas analógicas.
- Memoria: El ATmega328P tiene 32 KB de memoria flash (programa), 2 KB de SRAM y 1 KB de EEPROM.
- Velocidad de reloj: Opera a una velocidad de reloj de 16 MHz.
- Dimensiones: Aproximadamente 68.6 mm x 53.4 mm.



Arduino Uno Rev3

Arduino UNO R3

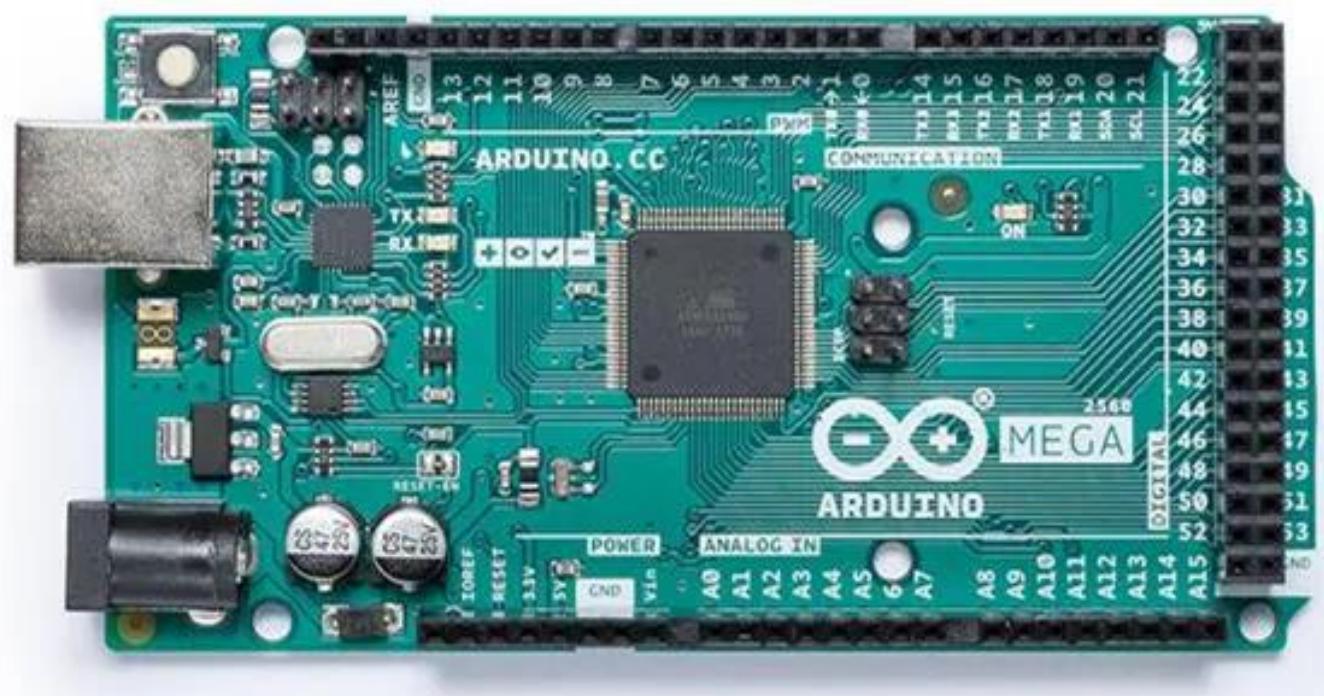
Pinout



Arduino MEGA

ATMEGA2560

- Microcontrolador: ATmega2560
- Velocidad de reloj: 16 MHz
- Voltaje de operación: 5 V
- Voltaje de entrada (límites): 6-20 V
- Pines digitales I/O: 54 (15 con salida PWM)
- Entradas analógicas: 16
- Memoria Flash: 256 KB (8 KB usados por el bootloader)
- SRAM: 8 KB
- EEPROM: 4 KB
- Interfaces de comunicación: 4 puertos UART, SPI, I2C
- Interrupciones externas: 6 pines
- Conversor A/D: de 10 bits

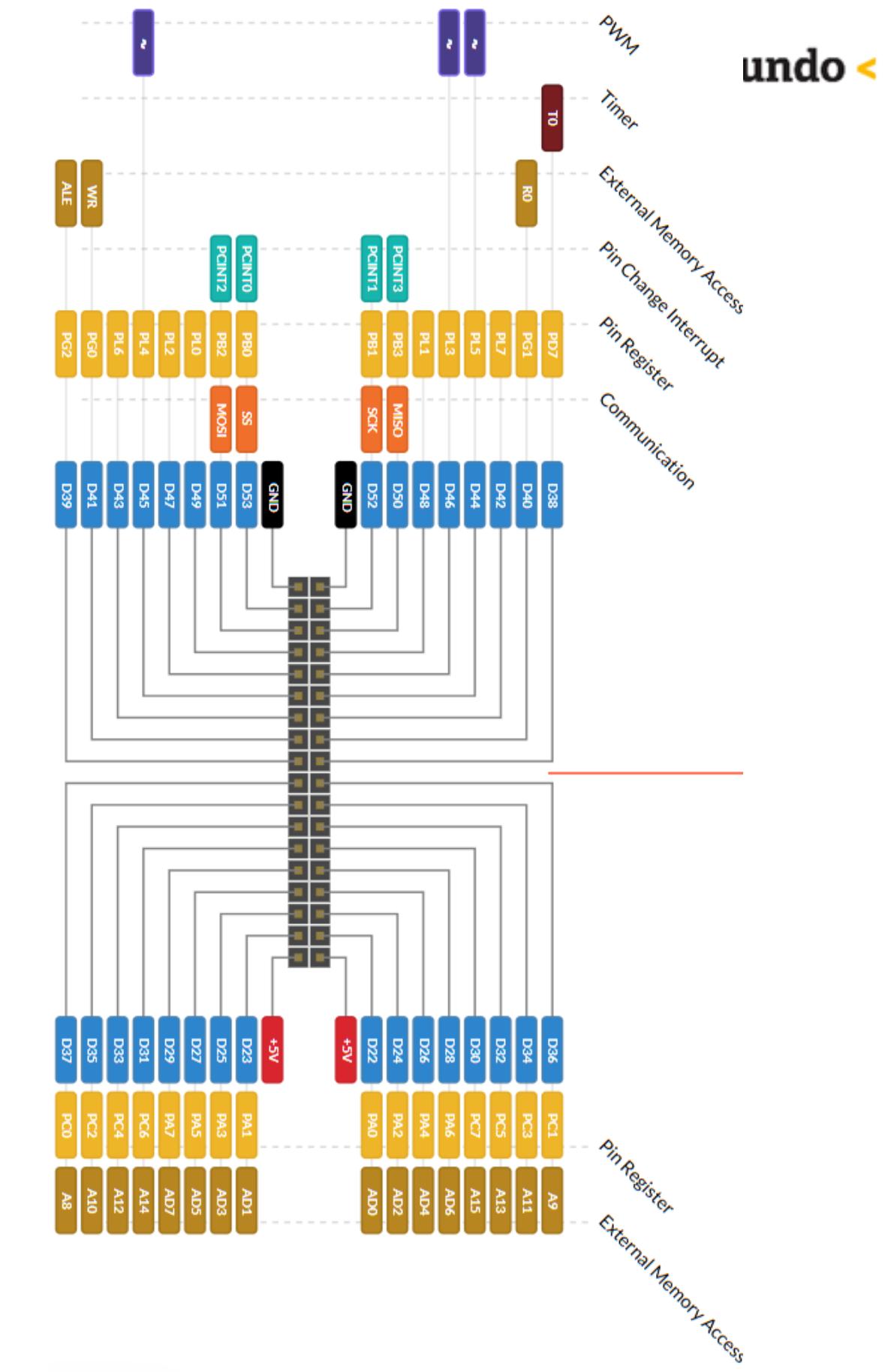
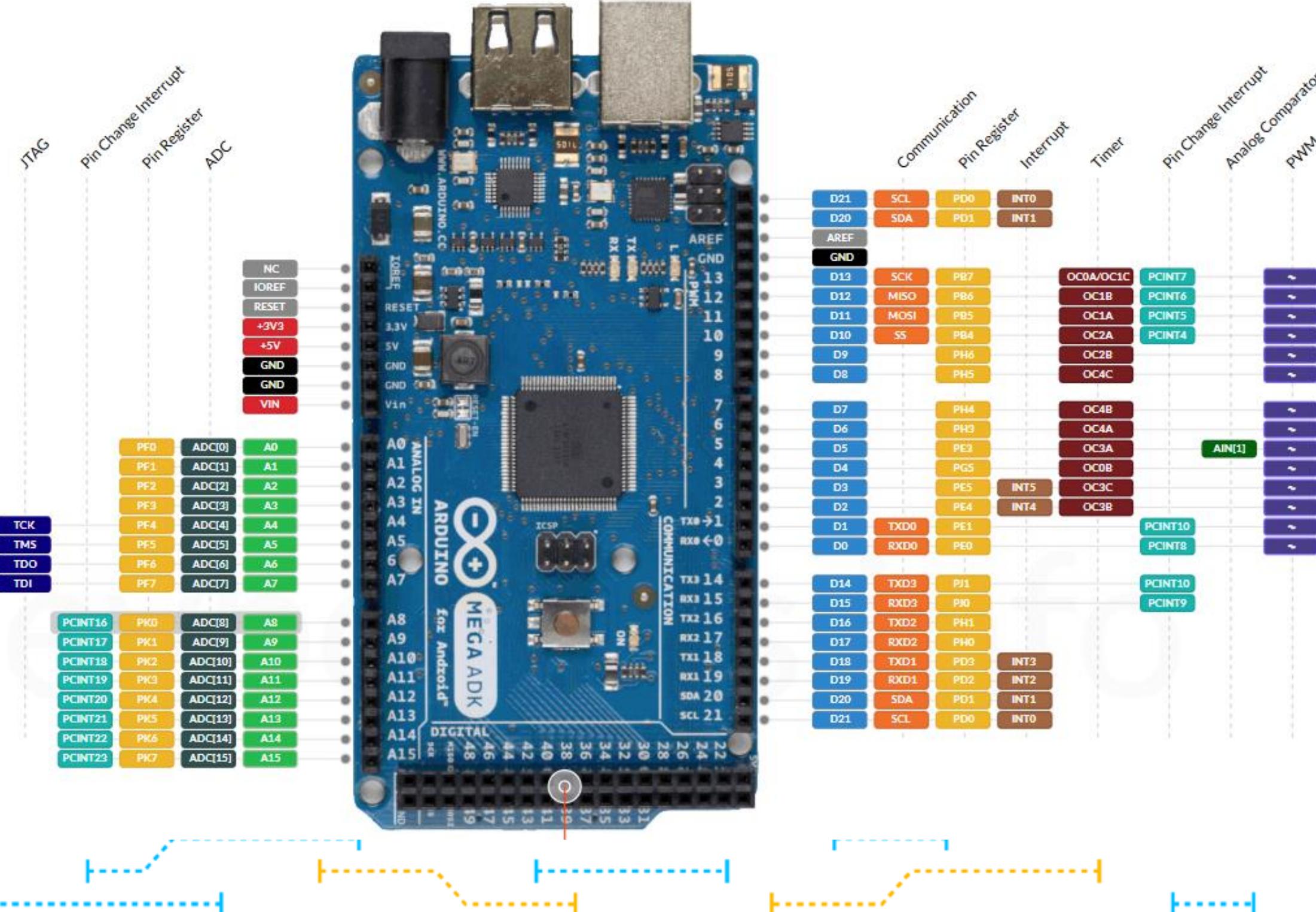


Arduino MEGA



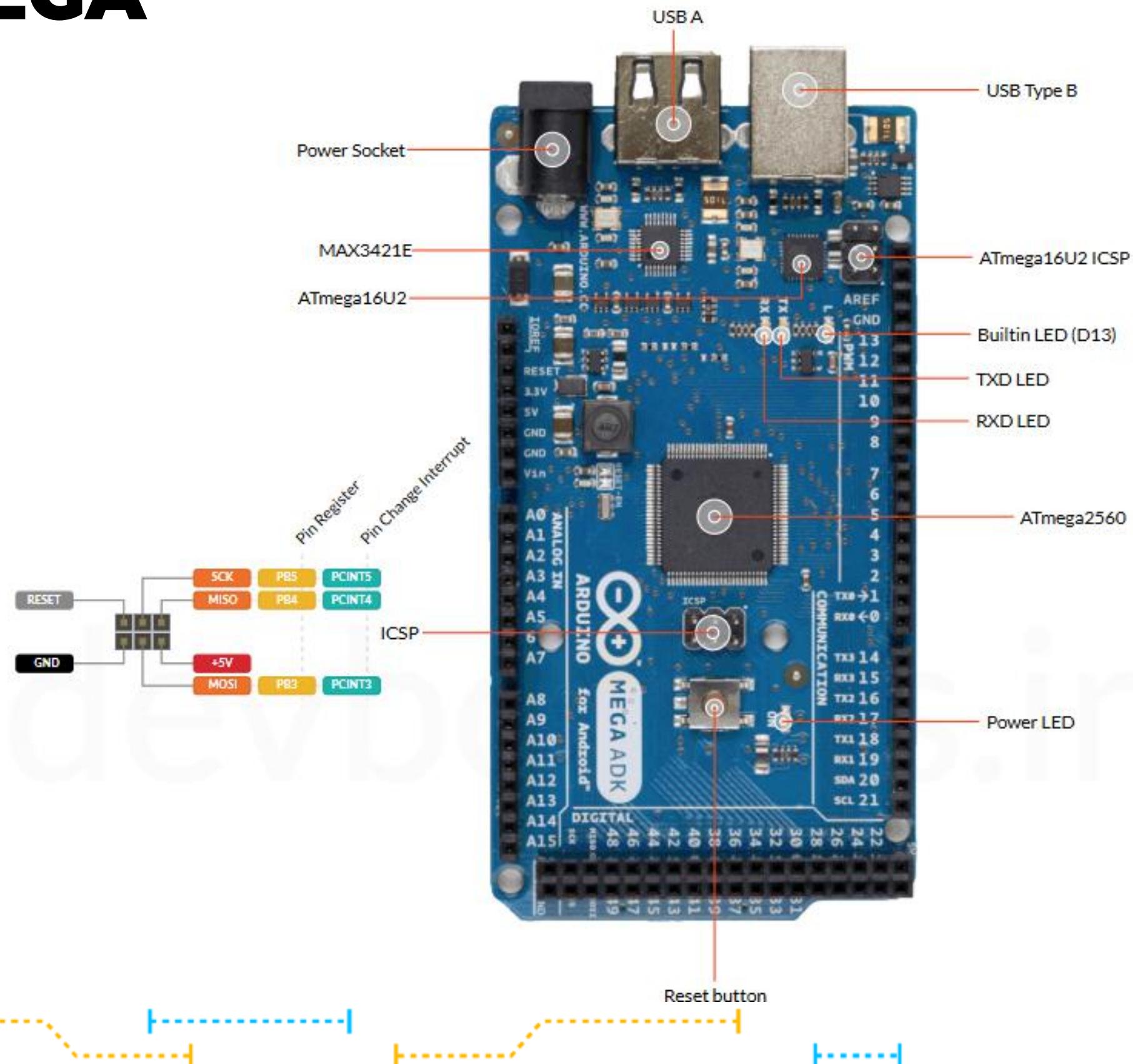
Arduino MEGA

Pinout



Arduino MEGA

Pinout



IDE de Arduino

Arduino IDE

El IDE de Arduino es una aplicación de software que permite escribir, compilar y cargar programas en placas Arduino.

Proporciona una interfaz simple para programar microcontroladores, ofreciendo herramientas como un editor de texto para el código, un compilador para traducir el código a instrucciones que el microcontrolador pueda ejecutar, y una interfaz para cargar ese código a través de un puerto USB.

El IDE de Arduino soporta varios lenguajes de programación, aunque comúnmente se utiliza un subconjunto de C/C++.



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

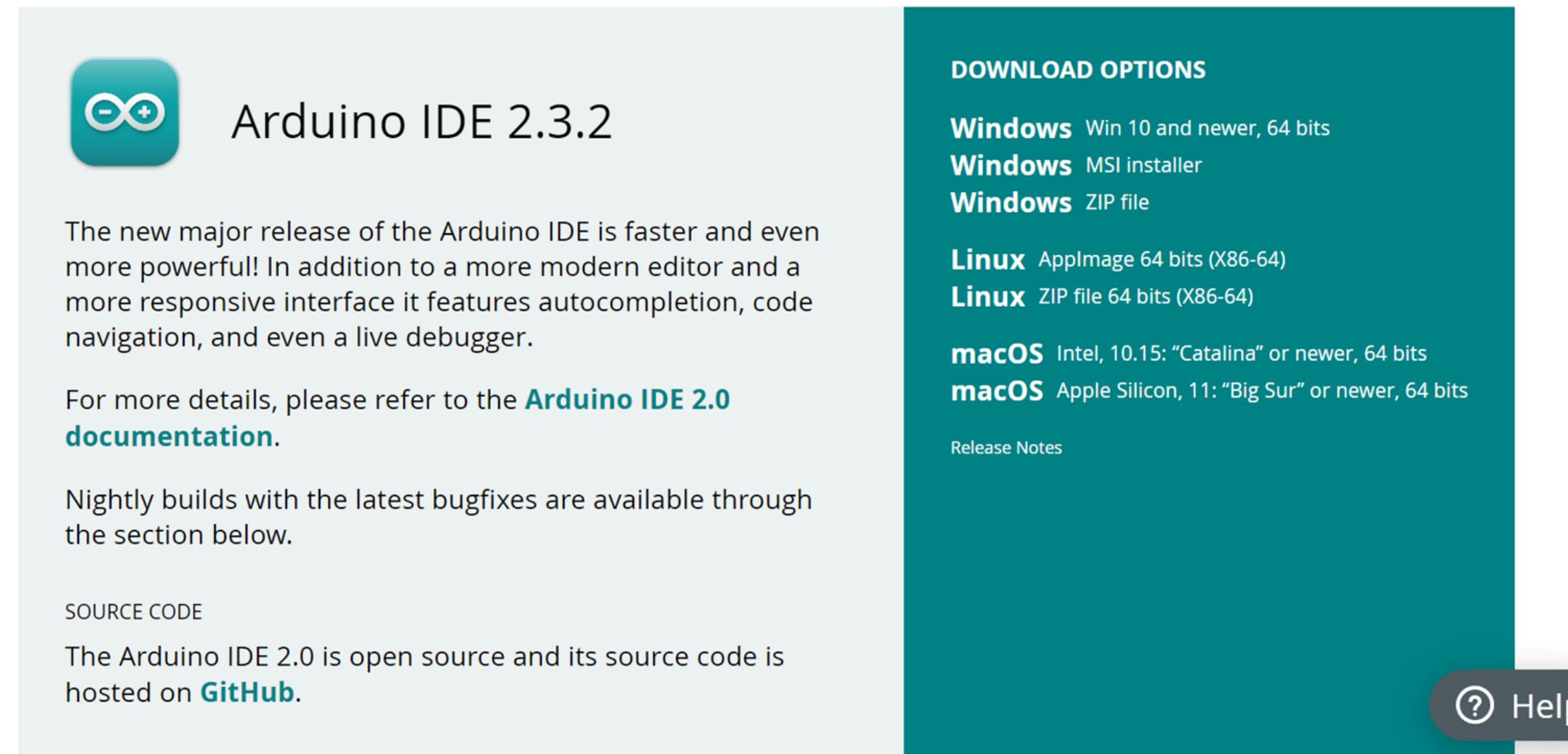
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
  delay(1000);                         // wait for a second
  digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW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

Captura de IDE de Arduino

IDE de Arduino

Instalación

Dirigirse al sitio web oficial de Arduino (<https://www.arduino.cc/en/software>) y descarga el IDE de Arduino. Podrás ver las múltiples versiones disponibles para Windows, Linux y Mac OS X.



The screenshot shows the download page for Arduino IDE 2.3.2. On the left, there's a teal square icon with a white 'E' and '+' symbol, followed by the text "Arduino IDE 2.3.2". Below it is a brief description of the new features: "The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger." It also links to the "Arduino IDE 2.0 documentation". A "SOURCE CODE" section mentions the code is open source and available on GitHub. On the right, a teal sidebar titled "DOWNLOAD OPTIONS" lists download links for Windows, Linux, and macOS, along with their respective system requirements. At the bottom of the sidebar is a link to "Release Notes". A "Help" button is located at the bottom right of the sidebar.

Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

?

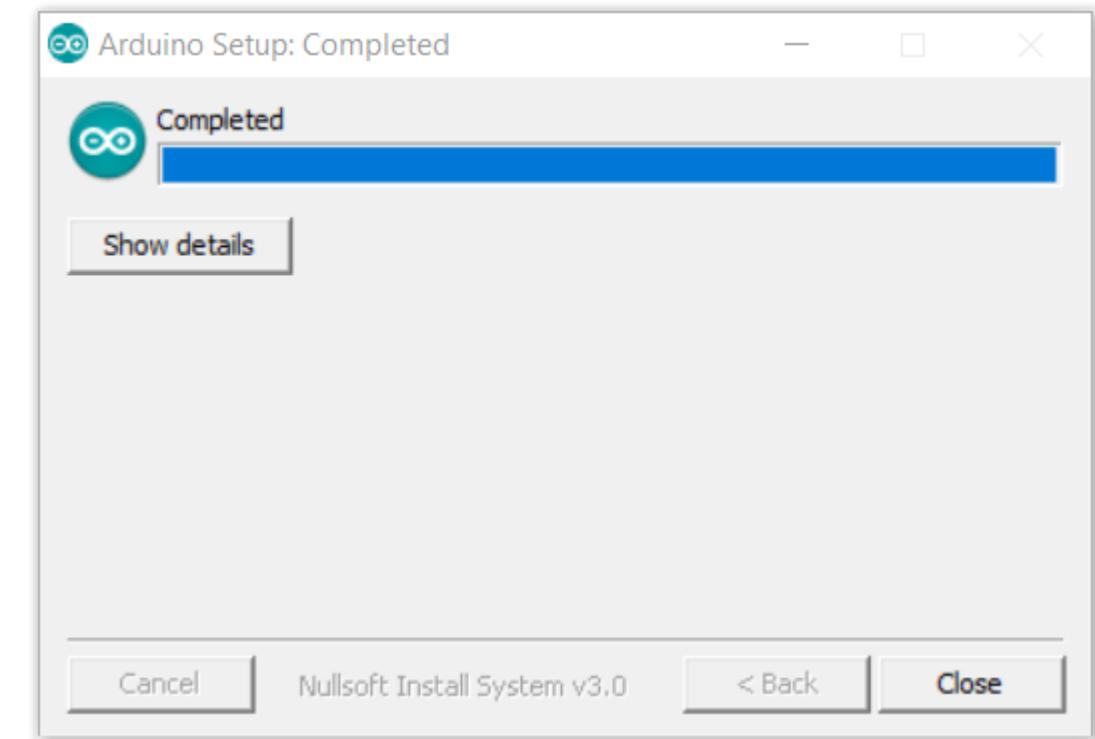
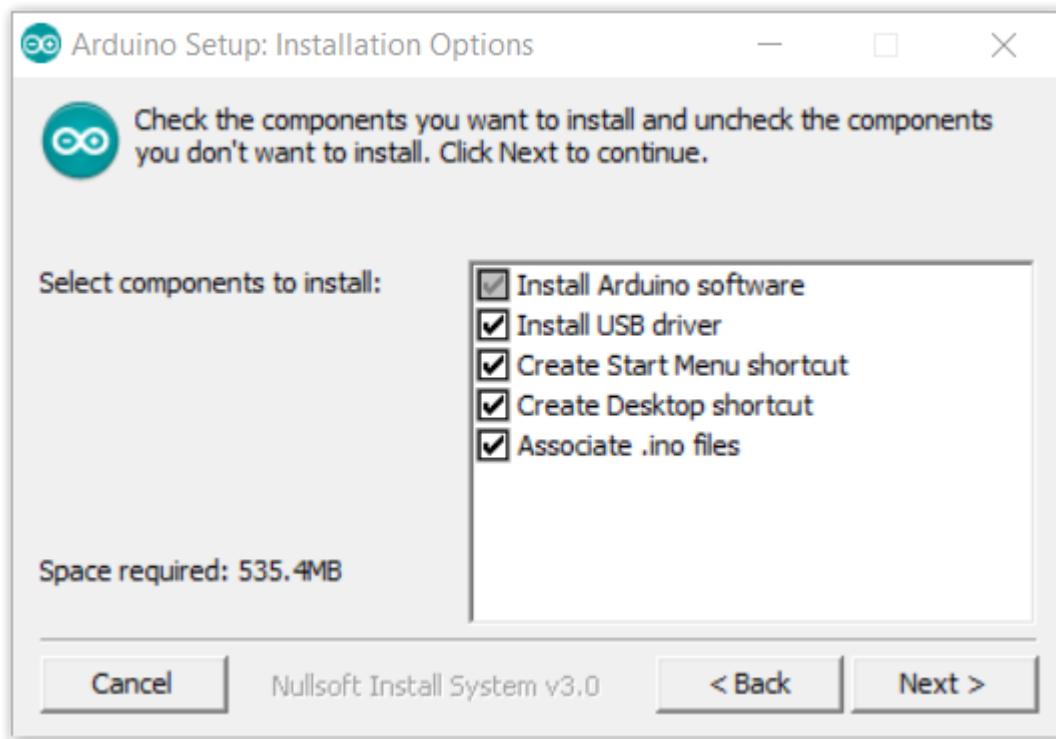
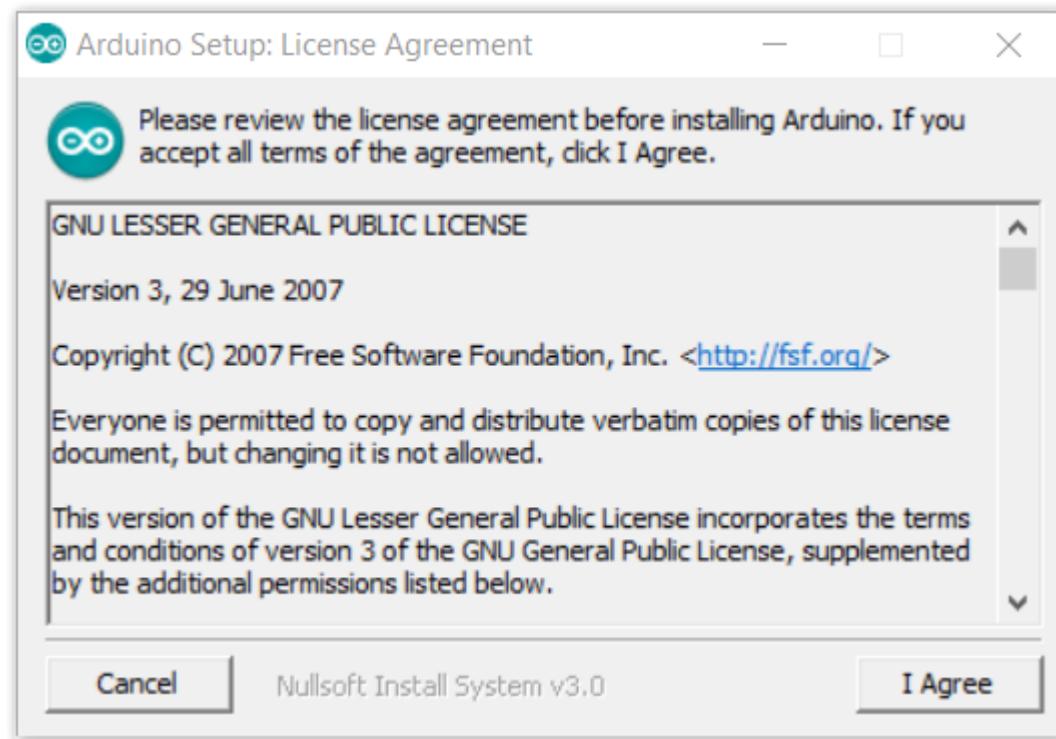
Help



IDE de Arduino

Instalación

Aceptar todos los permisos.



IDE de Arduino

Guía de la interfaz

Guía de la interfaz Arduino IDE

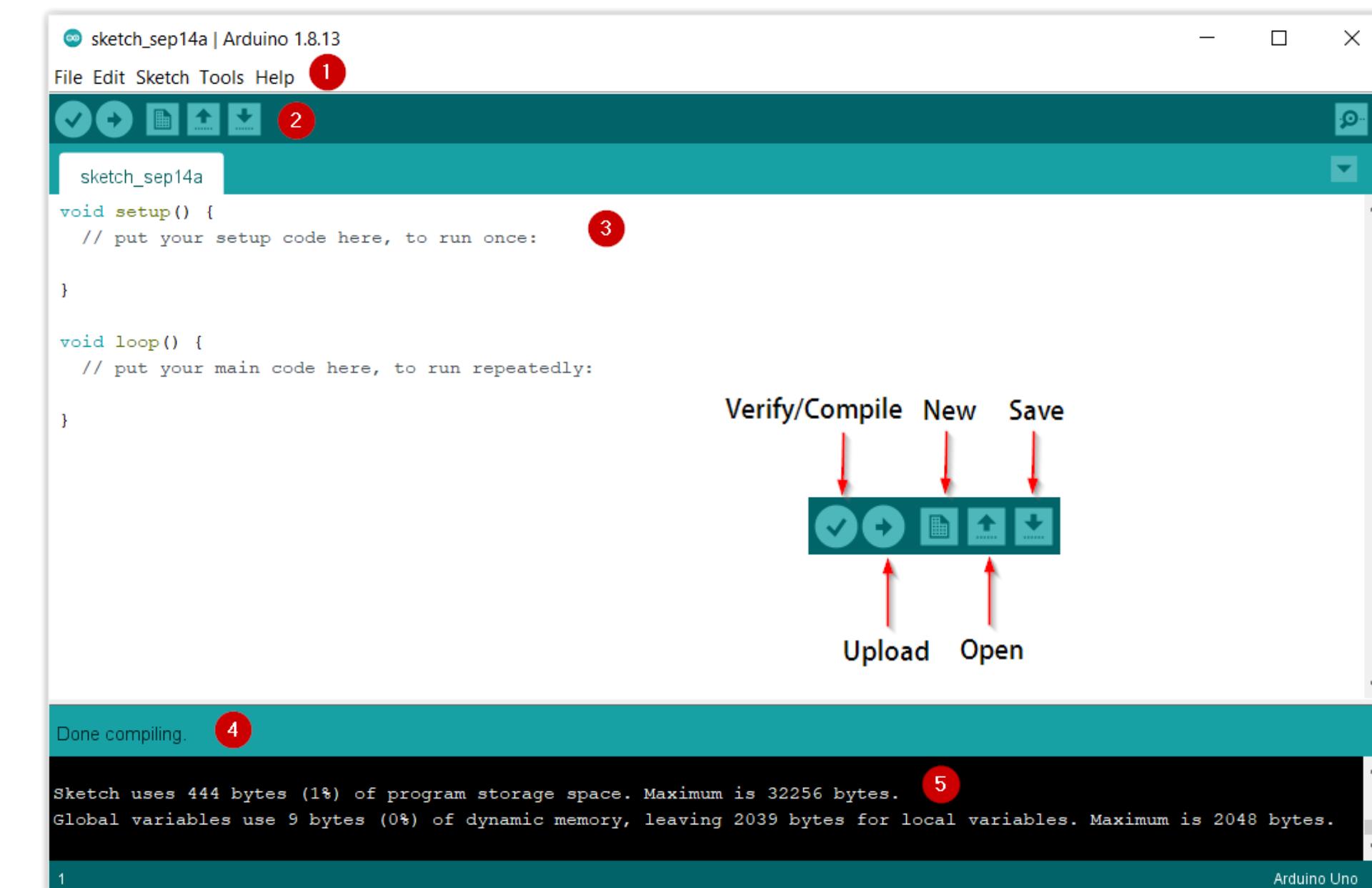
1. Menú de opciones del IDE

Puede configurar algunos parámetros generales como el puerto serie, la información de la placa, las bibliotecas, los parámetros de edición, etc.

1. Botones de operación

Los botones de operación tienen cinco operaciones:

- Verificar/Compilar el código fuente;
- Cargar el código compilado;
- Abrir una nueva ventana del IDE Arduino o una aplicación existente;
- Guardar la aplicación actual.



IDE de Arduino

Guía de la interfaz

3. Área de código

Puedes editar el código fuente, que será compilado y cargado en WisBlock más tarde en esta área.

4. Área de estado

5. Área de Mensaje de Salida

Puedes ver el mensaje de salida en esta área, ya sea información de fallo o de éxito.

The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with several icons. The main area contains the code for 'sketch_sep14a':

```
sketch_sep14a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_sep14a
void setup() {
  // put your setup code here, to run once:
}

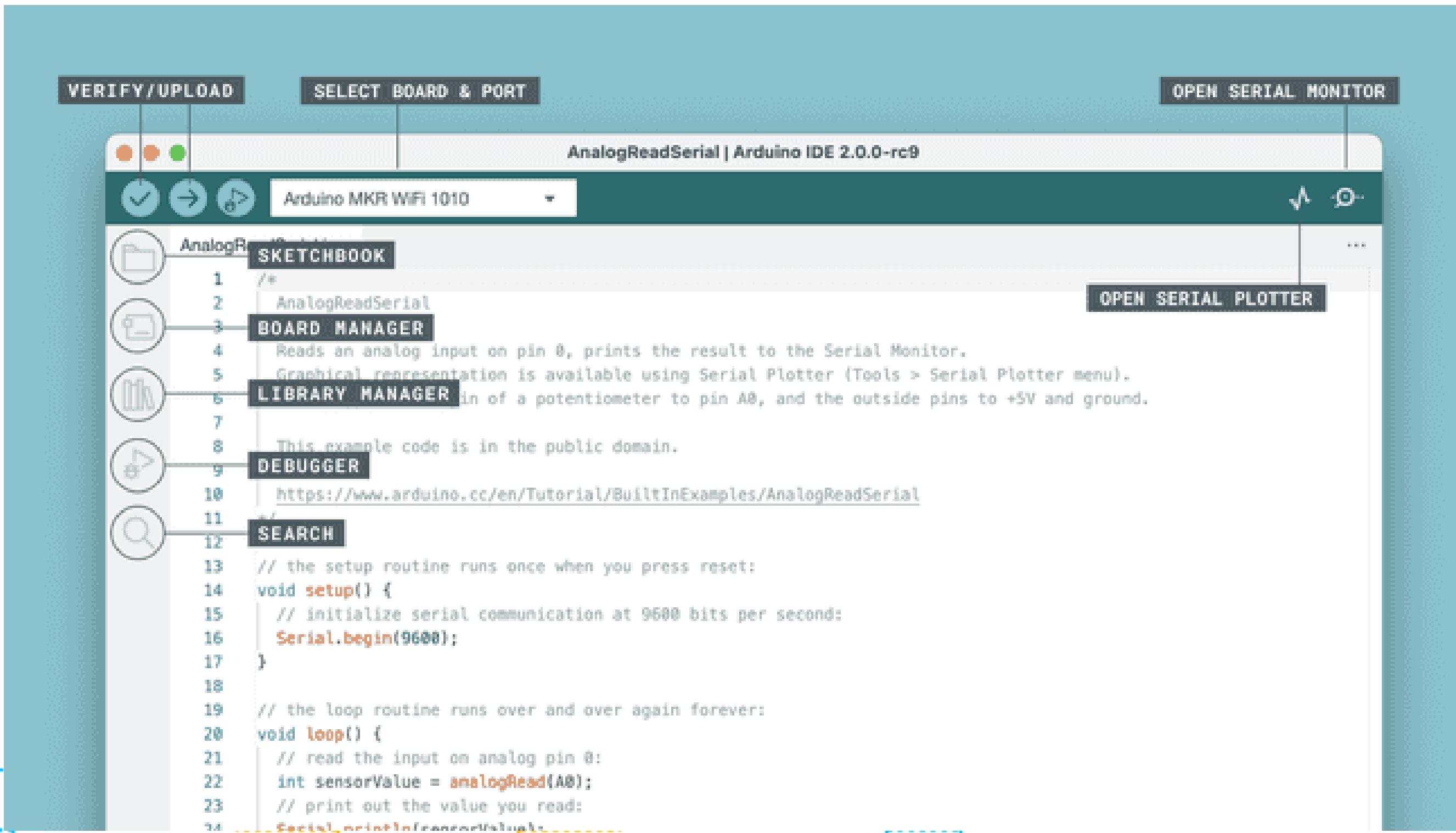
void loop() {
  // put your main code here, to run repeatedly:
}
```

The status bar at the bottom displays the message 'Done compiling.' and memory usage information: 'Sketch uses 444 bytes (1%) of program storage space. Maximum is 32256 bytes.' and 'Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.' The board selected is 'Arduino Uno'.



IDE de Arduino

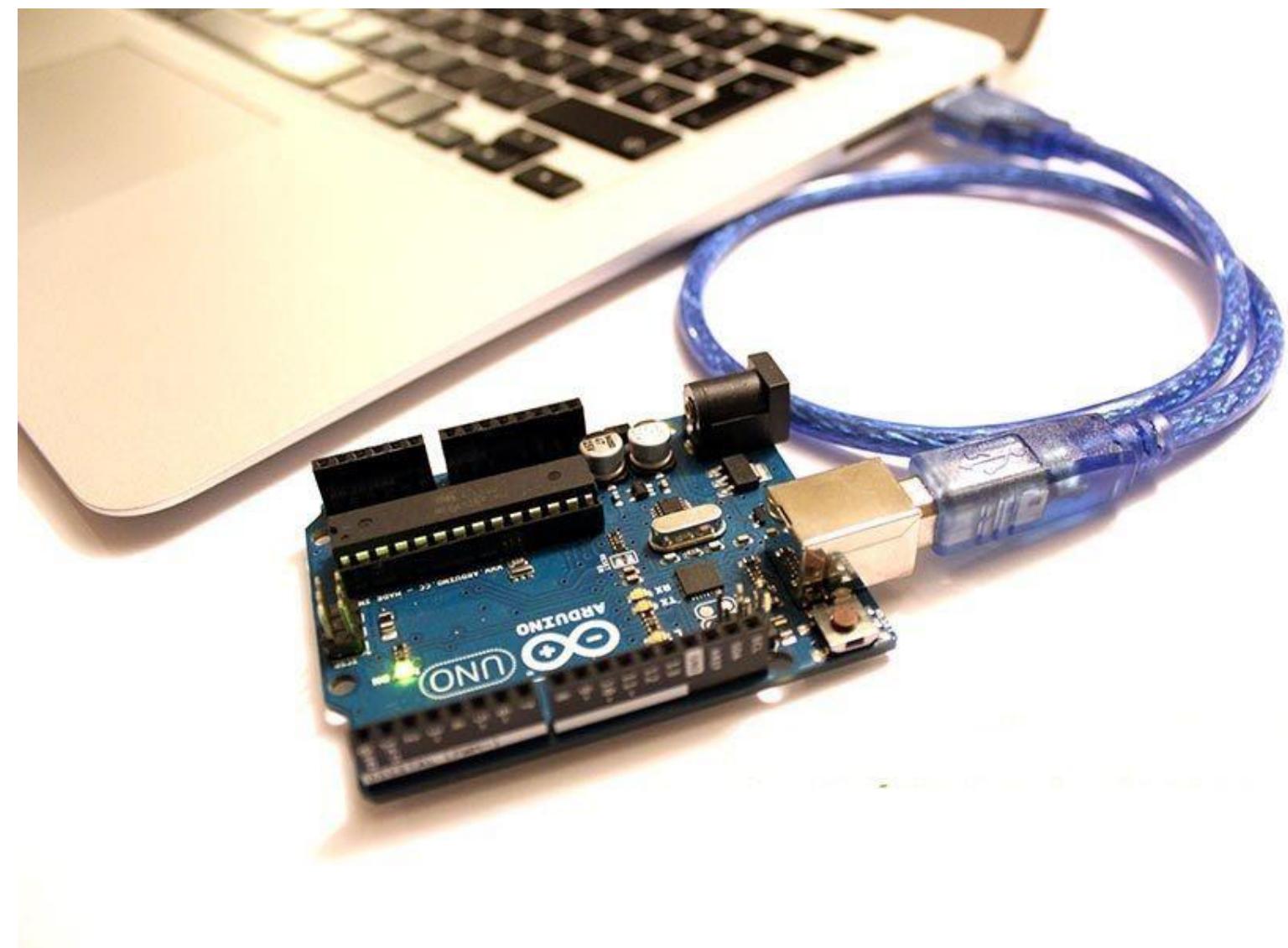
Guía de la interfaz



IDE de Arduino

Conexión con la placa

1. Conecta el Arduino a tu computadora: Utiliza el cable USB para conectar tu Arduino a uno de los puertos USB disponibles en tu computadora. Asegúrate de que la conexión esté segura y estable.



IDE de Arduino

Conexión con la placa

- 2. Abre el software de Arduino:** Una vez que hayas conectado tu Arduino, abre el software de Arduino en tu computadora. El software debería detectar automáticamente la conexión y seleccionar el puerto COM correcto.

- 3. Configura el tipo de placa:** En el menú «Herramientas» del software de Arduino, selecciona el tipo de placa que estás utilizando. Por ejemplo, si estás utilizando un Arduino Uno, elige «Arduino/Genuino Uno» en la opción «Placa».

- 4. Selecciona el puerto COM:** En el mismo menú «Herramientas», selecciona el puerto COM al que está conectado tu Arduino. Si no estás seguro de cuál es el puerto correcto, puedes verificarlo en el Administrador de dispositivos de tu computadora.



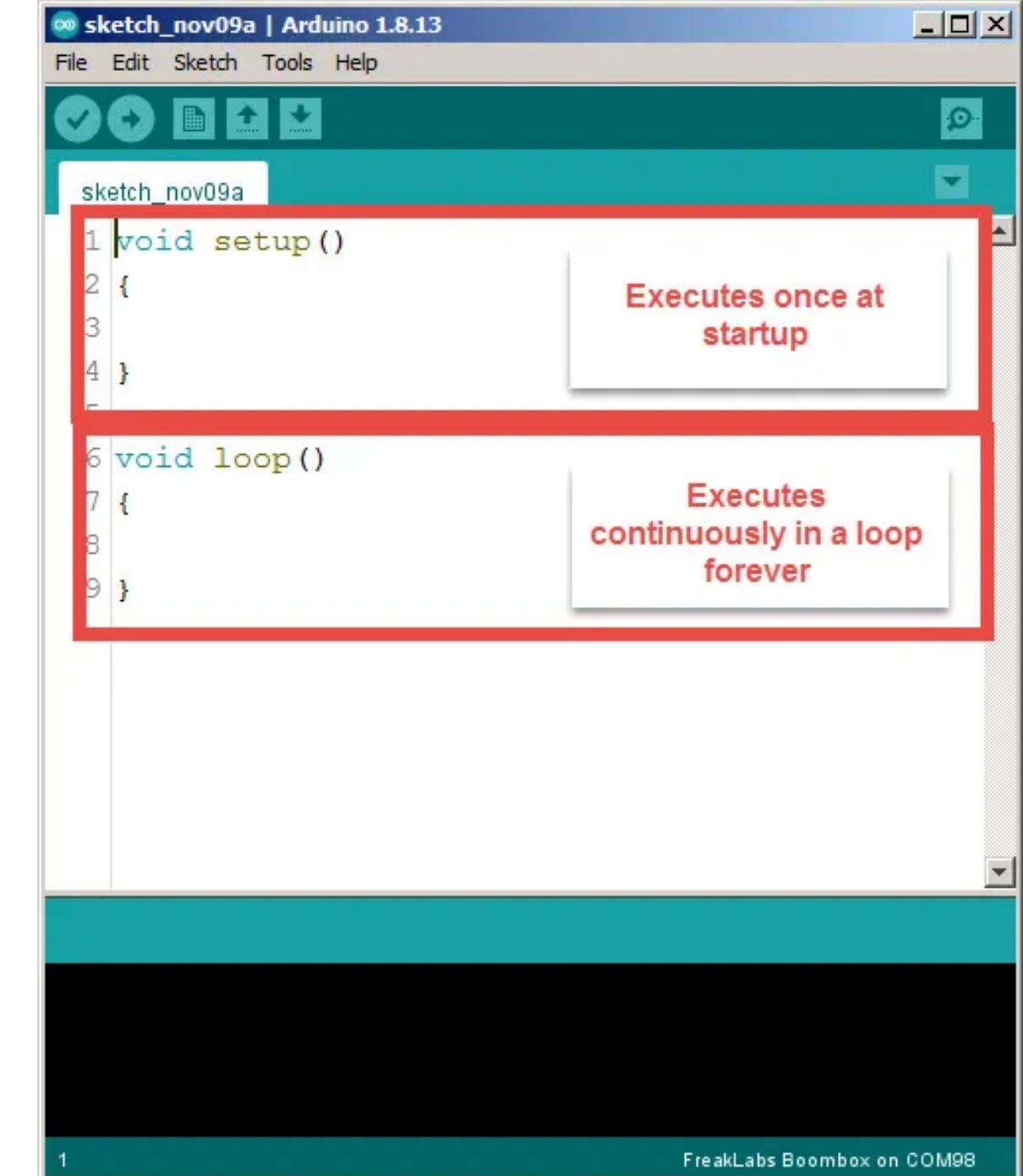
IDE de Arduino

Estructura

La estructura básica del código en Arduino IDE consta de dos partes principales:

- **setup()**: Función que se ejecuta una sola vez al iniciar el programa. Se usa para inicializar configuraciones, como definir pines o iniciar comunicaciones.
- **loop()**: Función que se ejecuta de manera continua después de setup(). Aquí se coloca el código que debe ejecutarse repetidamente, como la lectura de sensores o el control de actuadores.

Ambas funciones son obligatorias para que el programa funcione en Arduino.

```
sketch_nov09a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_nov09a
1 void setup()
2 {
3
4 }
5
6 void loop()
7 {
8
9 }
```

1 void setup()
2 {
3
4 }
5
6 void loop()
7 {
8
9 }

Executes once at startup

Executes continuously in a loop forever

FreakLabs Boombox on COM98

Estructura

4. Introducción a código C para Arduino

En la página de arduino, podemos encontrar diferentes referencias de códigos y programas pre-hechos para el desarrollo de diferentes soluciones. (<https://docs.arduino.cc/programming/>)

Programming

Learn all you need to know about the Arduino programming language as well as other compatible languages.

⌚ Language Reference

Serial
Wire
Print
attachInterrupt()
pinMode()
String()
[view all →](#)

ibraries

LiquidCrystal
Servo
LiquidCrystal I2C
WiFi
SD
Stepper
[view all →](#)

Built-in Examples

How to Wire and Program a Button
Blink
If Statement (Conditional Statement)
Arduino as ISP and Arduino Bootloaders
Analog Read Serial
String to Int Function
[view all →](#)

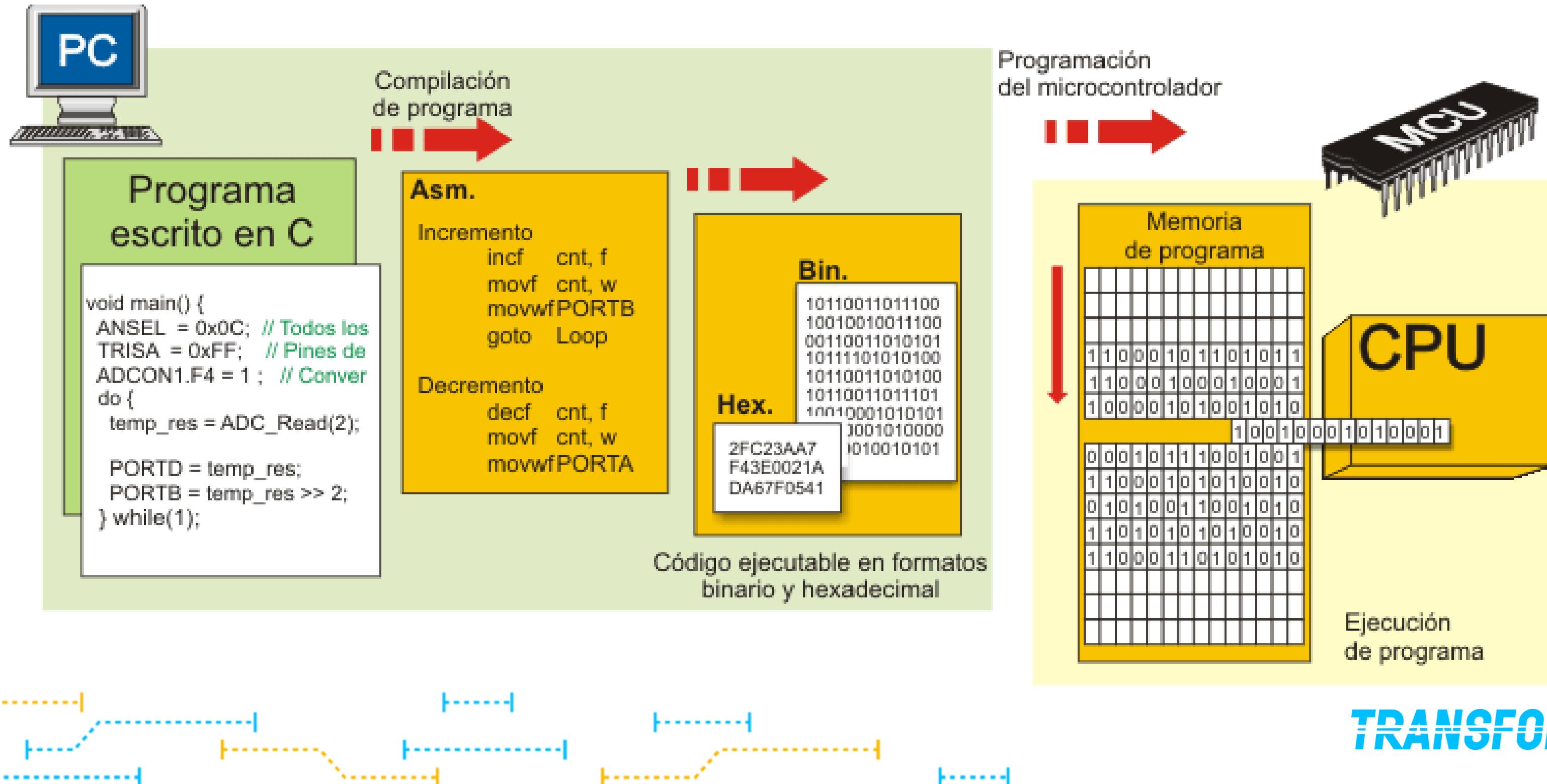


Lenguaje para Arduino

El lenguaje de programación de Arduino es C++. No es un C++ puro sino que es una adaptación que proveniente de avr-libc que provee de una librería de C de alta calidad para usar con GCC (compilador de C y C++) en los microcontroladores AVR de Atmel y muchas utilidades específicas para las MCU AVR de Atmel como avrdude.

Aunque se hable de que hay un lenguaje propio de programación de Arduino, no es cierto, la programación se hace en C++ pero Arduino ofrece una api o core que facilitan la programación de los pines de entrada y salida y de los puertos de comunicación, así como otras librerías para operaciones específicas.

Lenguaje para Arduino



a. Configuración Digital (GPIO)

GPIO significa entradas y salidas de propósito general (en inglés General Purpose Input Output). Estos pines se pueden definir dentro del código para dos tipos de función: lectura y escritura.

Es necesario conocer la siguiente sintaxis.

- `pinMode()`

Configura el pin especificado para que se comporte como entrada o como salida.

Syntax

```
pinMode(pin, mode)
```

Parameters

`pin`: the Arduino pin number to set the mode of.

`mode`: `INPUT`, `OUTPUT`, or `INPUT_PULLUP`. See the [Digital Pins](#) page for a more complete description of the functionality.

a. Configuración Digital (GPIO)

- `pinMode()`

Configura el pin especificado para que se comporte como entrada o como salida.

Example Code

The code makes the digital pin 13 `OUTPUT` and `Toggles` it `HIGH` and `LOW`

```
void setup() {  
    pinMode(13, OUTPUT);      // sets the digital pin 13 as output  
}  
  
void loop() {  
    digitalWrite(13, HIGH);   // sets the digital pin 13 on  
    delay(1000);             // waits for a second  
    digitalWrite(13, LOW);    // sets the digital pin 13 off  
    delay(1000);             // waits for a second  
}
```



a. Configuración Digital (GPIO)

- `digitalWrite()`

Escribe un valor ALTO o BAJO en un pin digital.

Si el pin se ha configurado como OUTPUT con `pinMode()`, su voltaje se ajustará al valor correspondiente: 5V (o 3,3V en placas de 3,3V) para HIGH, 0V (Tierra) para LOW.

Si el pin está configurado como INPUT, `digitalWrite()` activará (HIGH) o desactivará (LOW) el pullup interno del pin de entrada. Se recomienda ajustar `pinMode()` a INPUT_PULLUP para habilitar la resistencia interna de pull-up. *Vea el tutorial Pines Digitales para más información.*



a. Configuración Digital (GPIO)

● digitalWrite()

Escribe un valor ALTO o BAJO en un pin digital.

Example Code

Syntax

```
digitalWrite(pin, value)
```

The code makes the digital pin 13 an OUTPUT and toggles it by alternating between HIGH and LOW at one second pace.

Parameters

pin: the Arduino pin number.

value: HIGH OR LOW.

```
void setup() {
    pinMode(13, OUTPUT);      // sets the digital pin 13 as output
}

void loop() {
    digitalWrite(13, HIGH);   // sets the digital pin 13 on
    delay(1000);             // waits for a second
    digitalWrite(13, LOW);    // sets the digital pin 13 off
    delay(1000);             // waits for a second
}
```



a. Configuración Digital (GPIO)

● `digitalRead()`

Lee el valor de un pin digital especificado, ya sea HIGH o LOW.

Syntax

```
digitalRead(pin)
```

Parameters

pin: the Arduino pin number you want to read.

Example Code

Sets pin 13 to the same value as pin 7, declared as an input.

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7; // pushbutton connected to digital pin 7
int val = 0; // variable to store the read value

void setup() {
    pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output
    pinMode(inPin, INPUT); // sets the digital pin 7 as input
}

void loop() {
    val = digitalRead(inPin); // read the input pin
    digitalWrite(ledPin, val); // sets the LED to the button's value
}
```

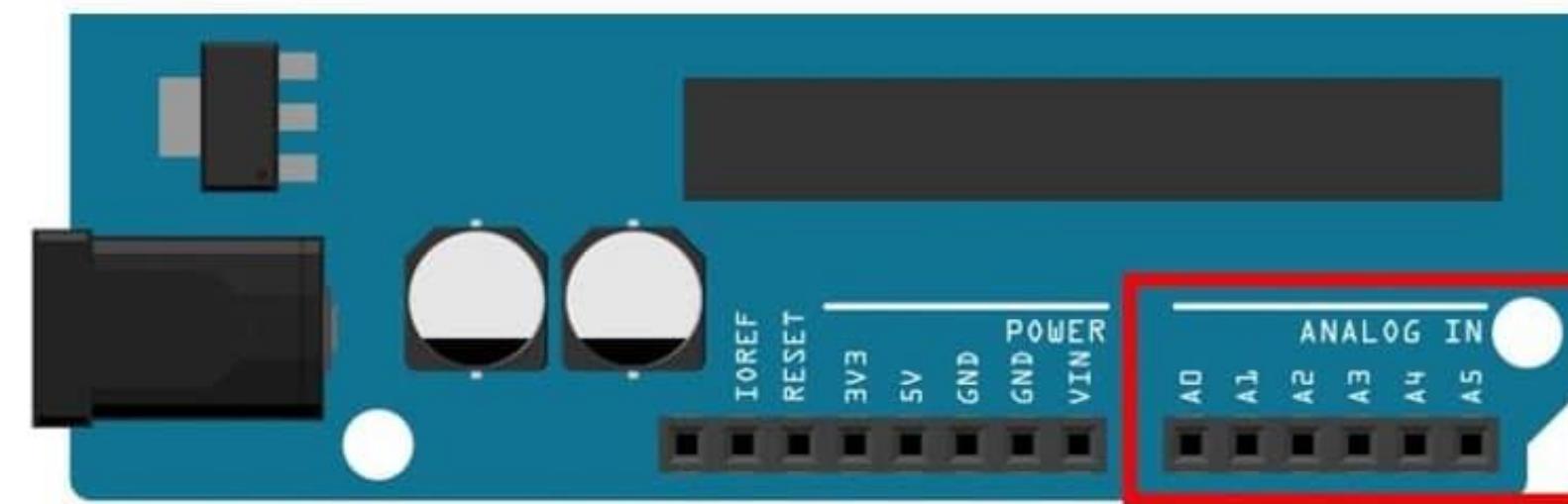


b. Configuración Analógica

Permite realizar una lectura de un valor analógico de voltaje en los pines del tipo Analog. El valor de voltaje medido no debe superar los 3.3V o 5V dependiendo de la placa. Emplea un ADC interno en la placa, cuya resolución depende del modelo.

- `analogRead()`

Lee el valor del pin analógico especificado. Las placas Arduino contienen un convertidor analógico-digital multicanal de 10 bits. Esto significa que mapeará los voltajes de entrada entre 0 y el voltaje de funcionamiento (5V o 3.3V) en valores enteros entre 0 y 1023.



Analog pins

INFORMATTEC

b. Configuración Analógica

Permite realizar una lectura de un valor analógico de voltaje en los pines del tipo Analog. El valor de voltaje medido no debe superar los 3.3V o 5V dependiendo de la placa. Emplea un ADC interno en la placa, cuya resolución depende del modelo.

- `analogRead()`

Lee el valor del pin analógico especificado. Las placas Arduino contienen un convertidor analógico-digital multicanal de 10 bits. Esto significa que mapeará los voltajes de entrada entre 0 y el voltaje de funcionamiento (5V o 3.3V) en valores enteros entre 0 y 1023.



b. Configuración Analógica

BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
UNO R3	5 Volts	A0 to A5	10 bits
UNO R4 (Minima, WiFi)	5 Volts	A0 to A5	14 bits**
Mini	5 Volts	A0 to A7	10 bits
Nano, Nano Every	5 Volts	A0 to A7	10 bits
Nano 33 (IoT, BLE, RP2040, ESP32)	3.3 Volts	A0 to A7	12 bits**
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
GIGA R1	3.3 Volts	A0 to A11	16 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

b. Configuración Analógica

Syntax

analogRead(pin)

a) Parameters

pin: the name of the analog input pin to read from.

Example Code

The code reads the voltage on analogPin and displays it.

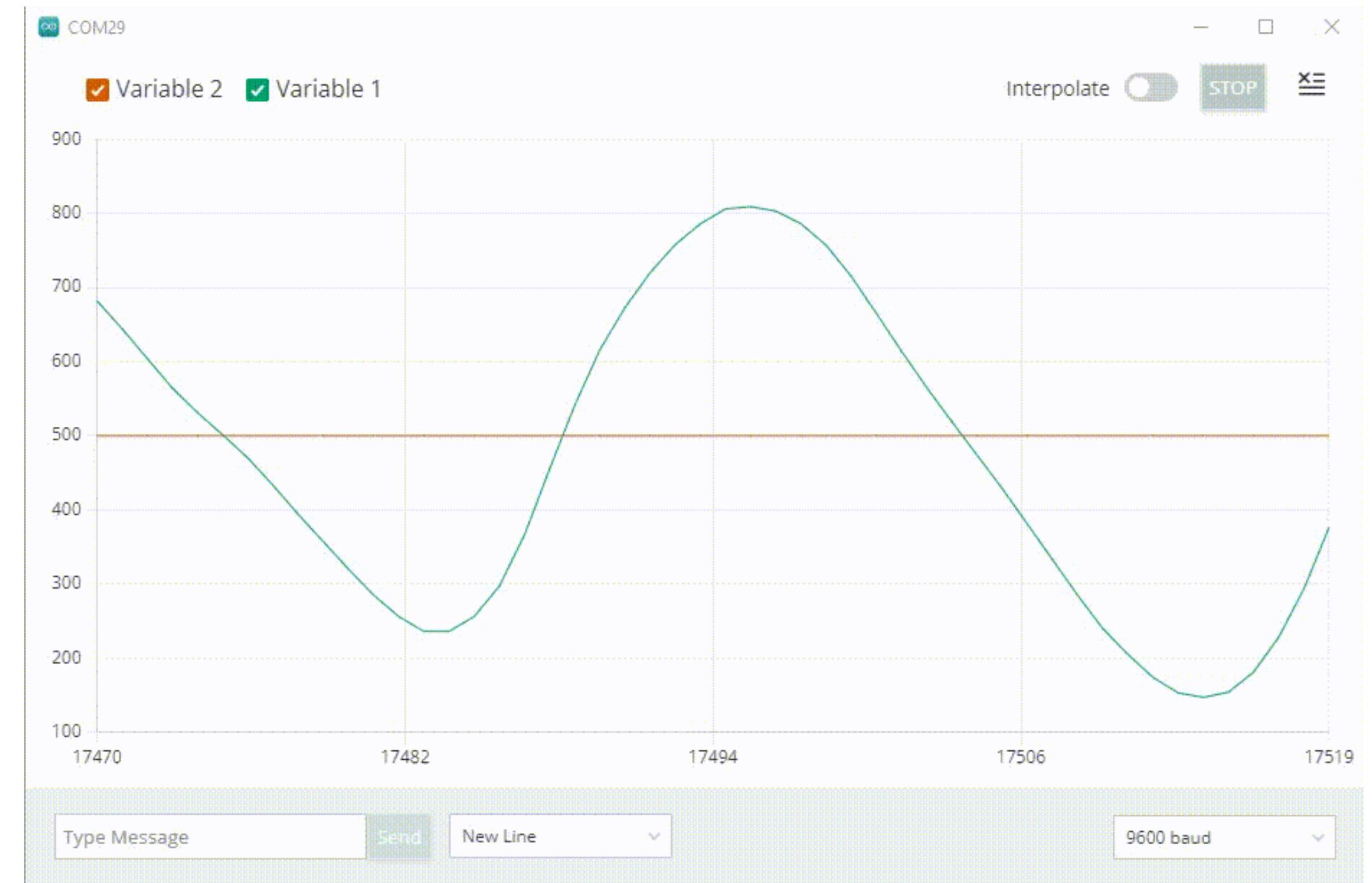
```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0; // variable to store the value read

void setup() {
    Serial.begin(9600); // setup serial
}

void loop() {
    val = analogRead(analogPin); // read the input pin
    Serial.println(val); // debug value
}
```



b. Configuración Analógica



TRANSFORMATEC

c. Comunicación Serial

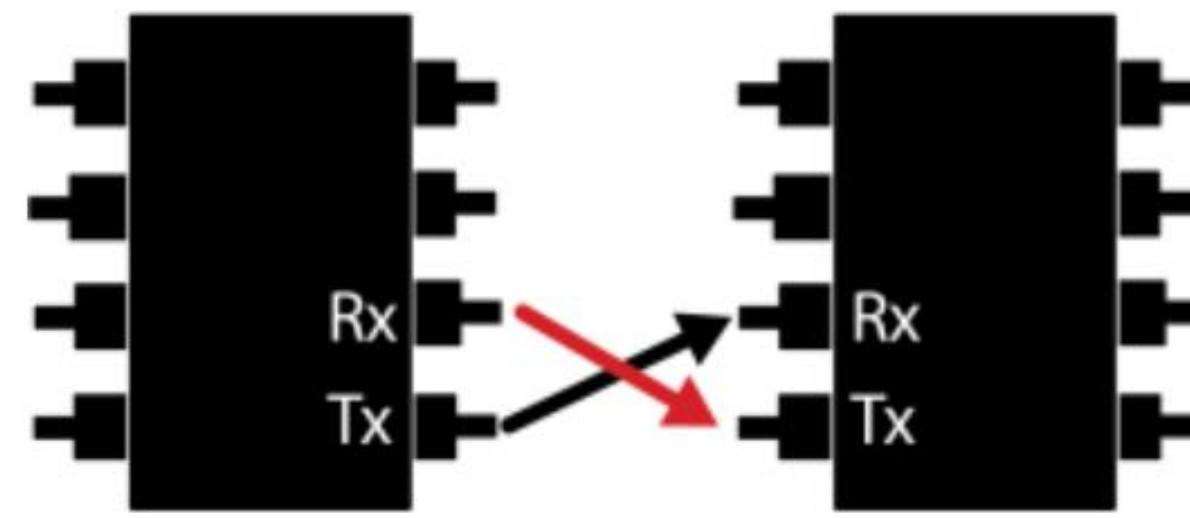
Se utiliza para la comunicación entre la placa Arduino y un ordenador u otros dispositivos (como algunos sensores). Todas las placas Arduino tienen al menos un puerto serie (también conocido como UART o USART), y algunas tienen varios. Puede utilizar el monitor serie incorporado en el entorno Arduino para comunicarse con una placa Arduino.

La comunicación serie en los pines TX/RX utiliza niveles lógicos TTL (5V o 3.3V dependiendo de la placa). No conecte estos pines directamente a un puerto serie RS232; funcionan a +/- 12V y pueden dañar su placa Arduino.



c. Comunicación Serial

Tarjeta Arduino		Serial	Serial1	Serial2	Serial3
UNO R3, Mini, Nano		RX TX 0 1	RX TX X X	RX TX X X	RX TX X X
MEGA R3		0 1	19 18	16 17	15 14

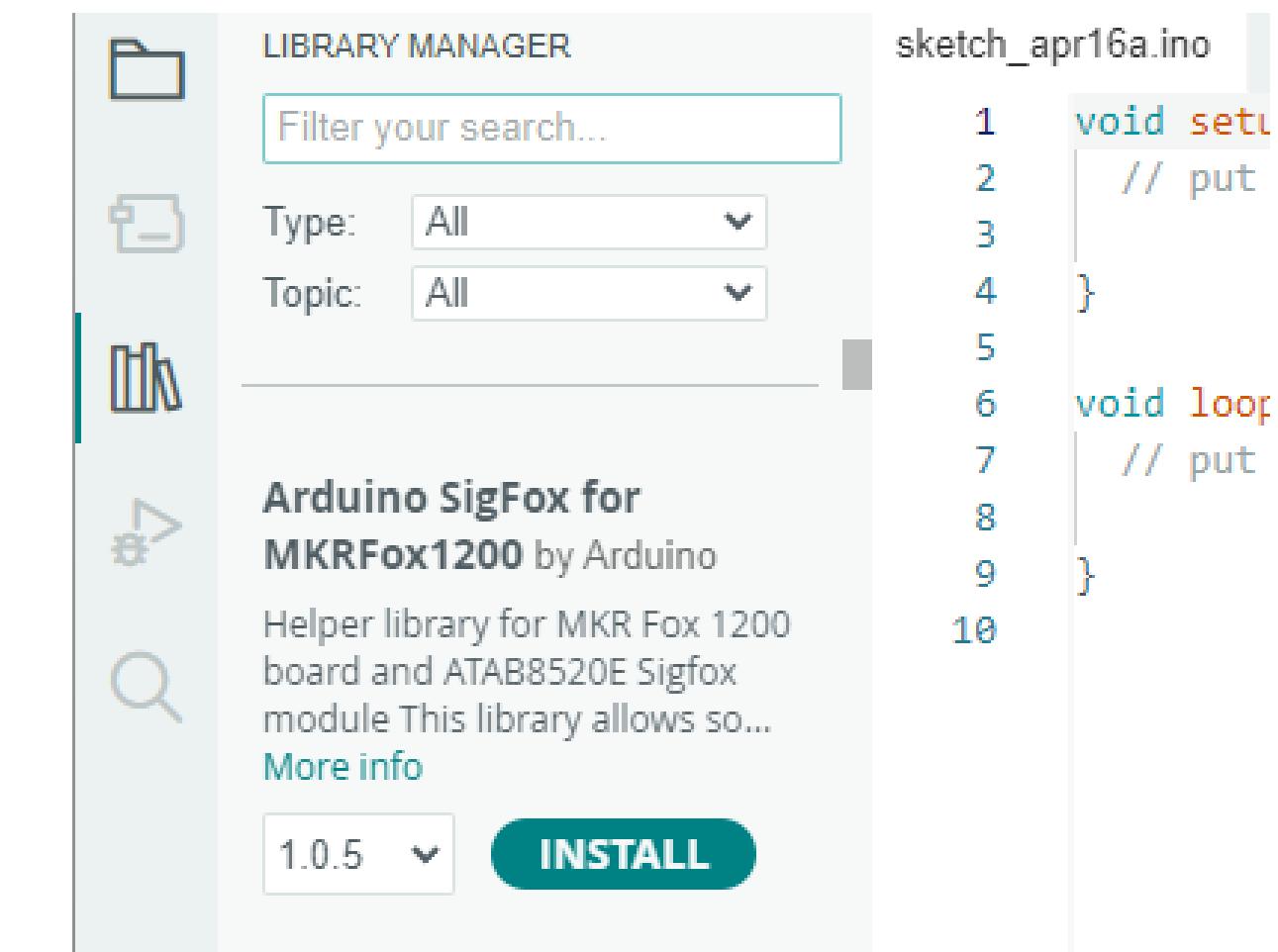


Conexión de puertos UART.



Las librerías son paquetes de código hechos por terceros que usamos en nuestro sketch. Esto nos facilita mucho la programación y hace que nuestro programa sea más sencillo de hacer y de entender.

- [Communication \(1458\)](#)
- [Data Processing \(418\)](#)
- [Data Storage \(177\)](#)
- [Device Control \(1191\)](#)
- [Display \(584\)](#)
- [Other \(597\)](#)
- [Sensors \(1364\)](#)
- [Signal Input/Output \(522\)](#)
- [Timing \(255\)](#)
- [Uncategorized \(253\)](#)



The screenshot shows the Arduino Library Manager interface. On the left, there are icons for folder, file, search, and refresh. In the center, there is a search bar with the placeholder "Filter your search..." and dropdown menus for "Type: All" and "Topic: All". Below these are two library entries:

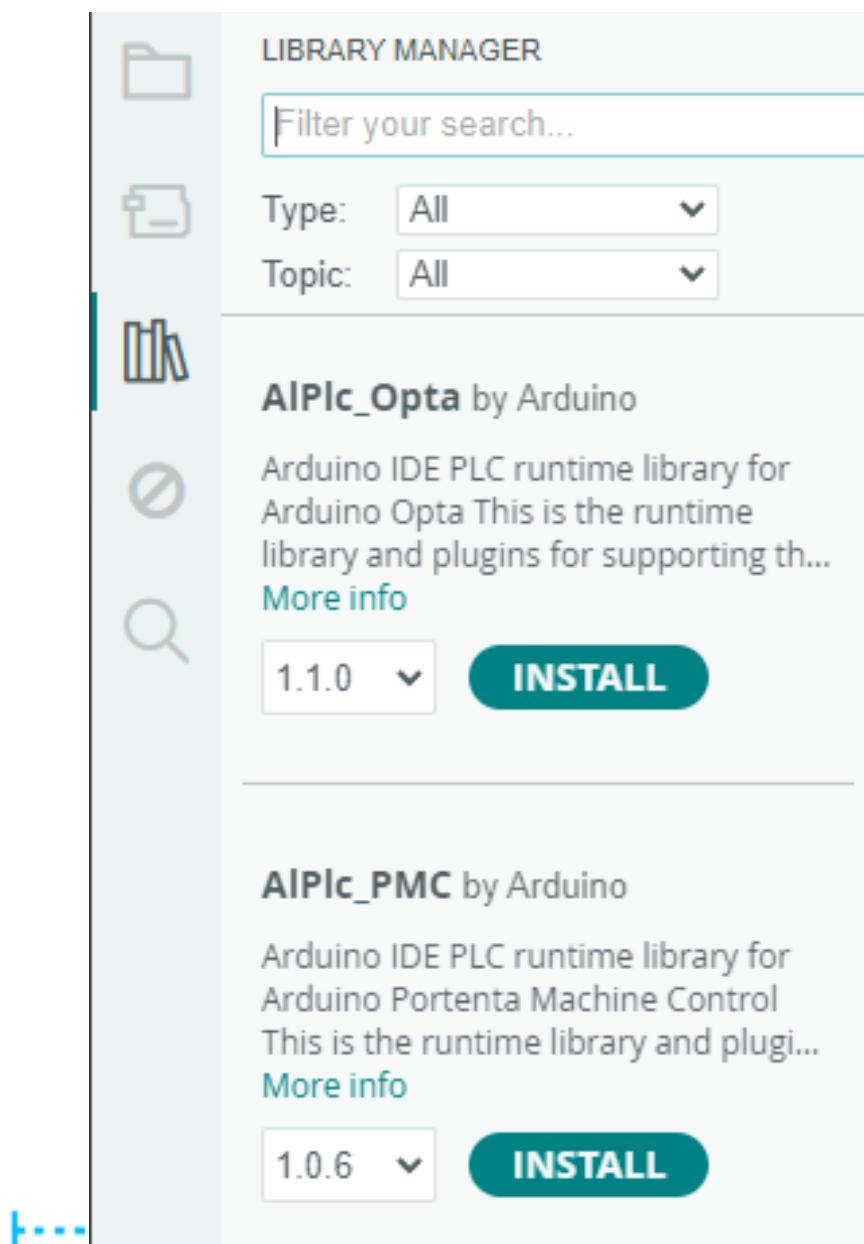
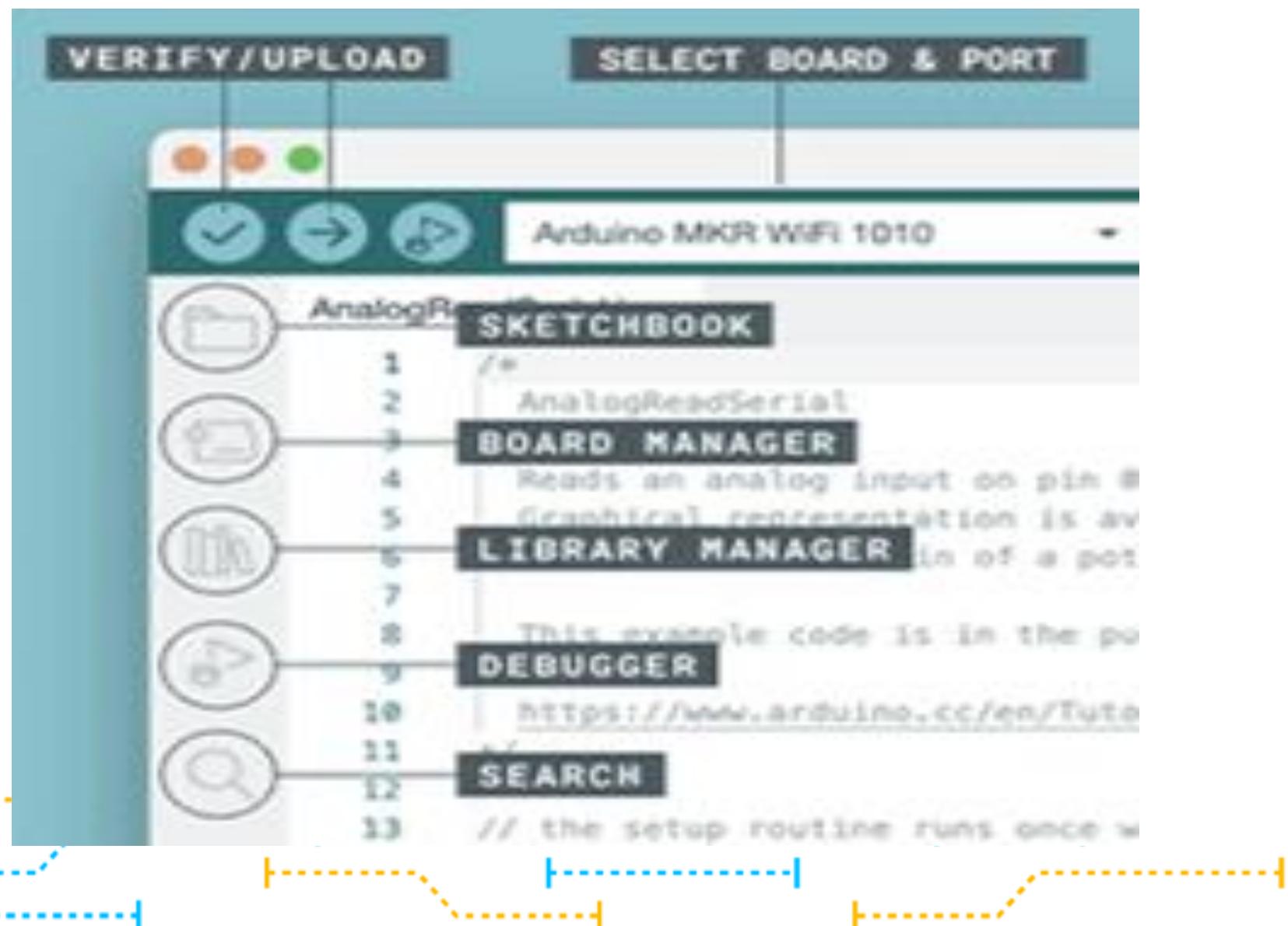
- Arduino SigFox for MKRFox1200 by Arduino**
Helper library for MKR Fox 1200 board and ATAB8520E Sigfox module This library allows so...
[More info](#)
- sketch_apr16a.ino**
1 void setup() // put
2 {
3 }
4 }
5 void loop() // put
6 {
7 }
8 }
9 }
10 }

At the bottom, there is a dropdown menu set to "1.0.5" and a teal "INSTALL" button.



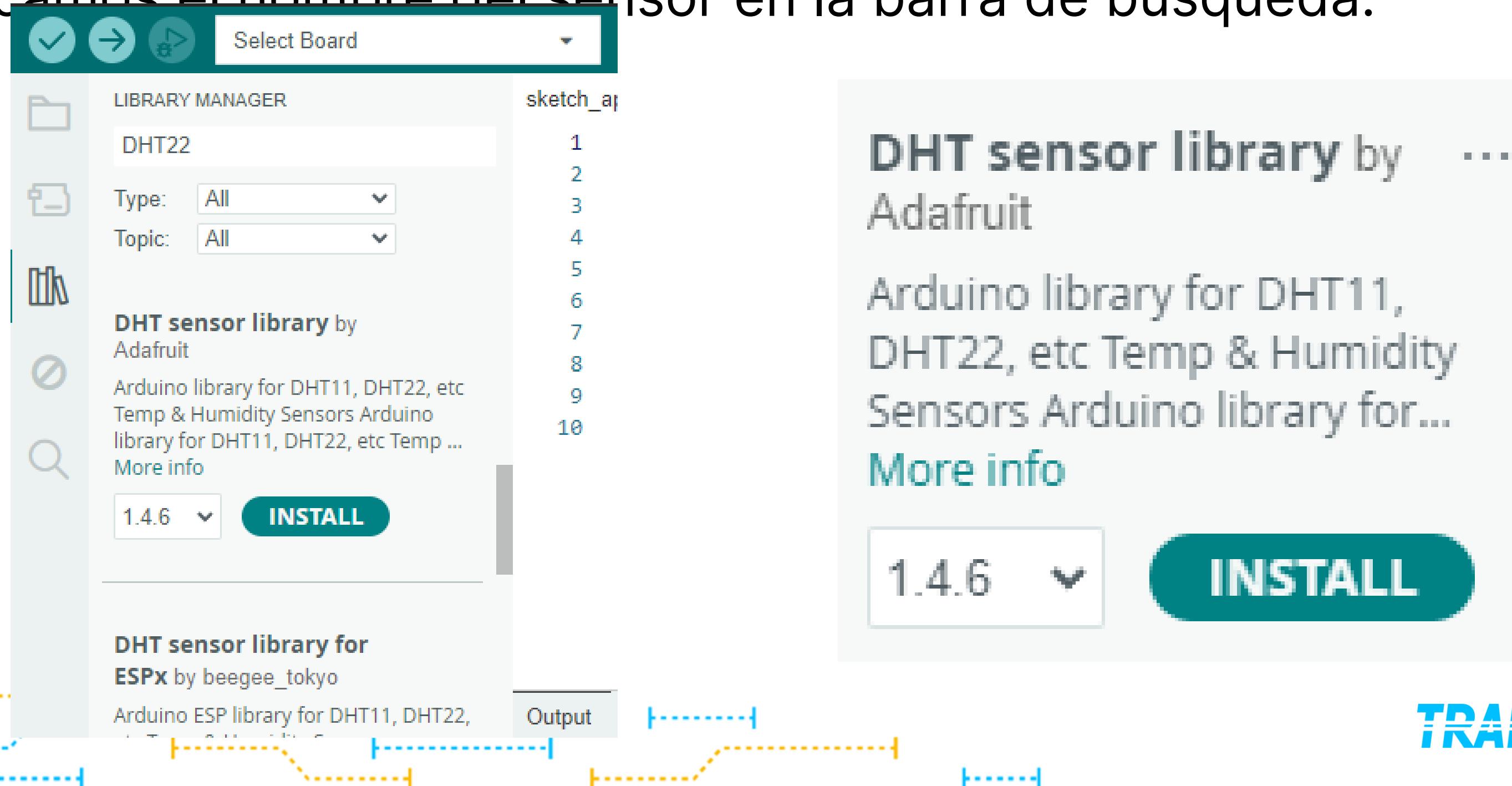
La instalación de librerías se puede hacer directamente desde el gestor de librerías o manualmente.

Así mismo ya existen librerías de diversos sensores en Arduino las cuales se pueden descargar en el **Administrador de Librerías**.



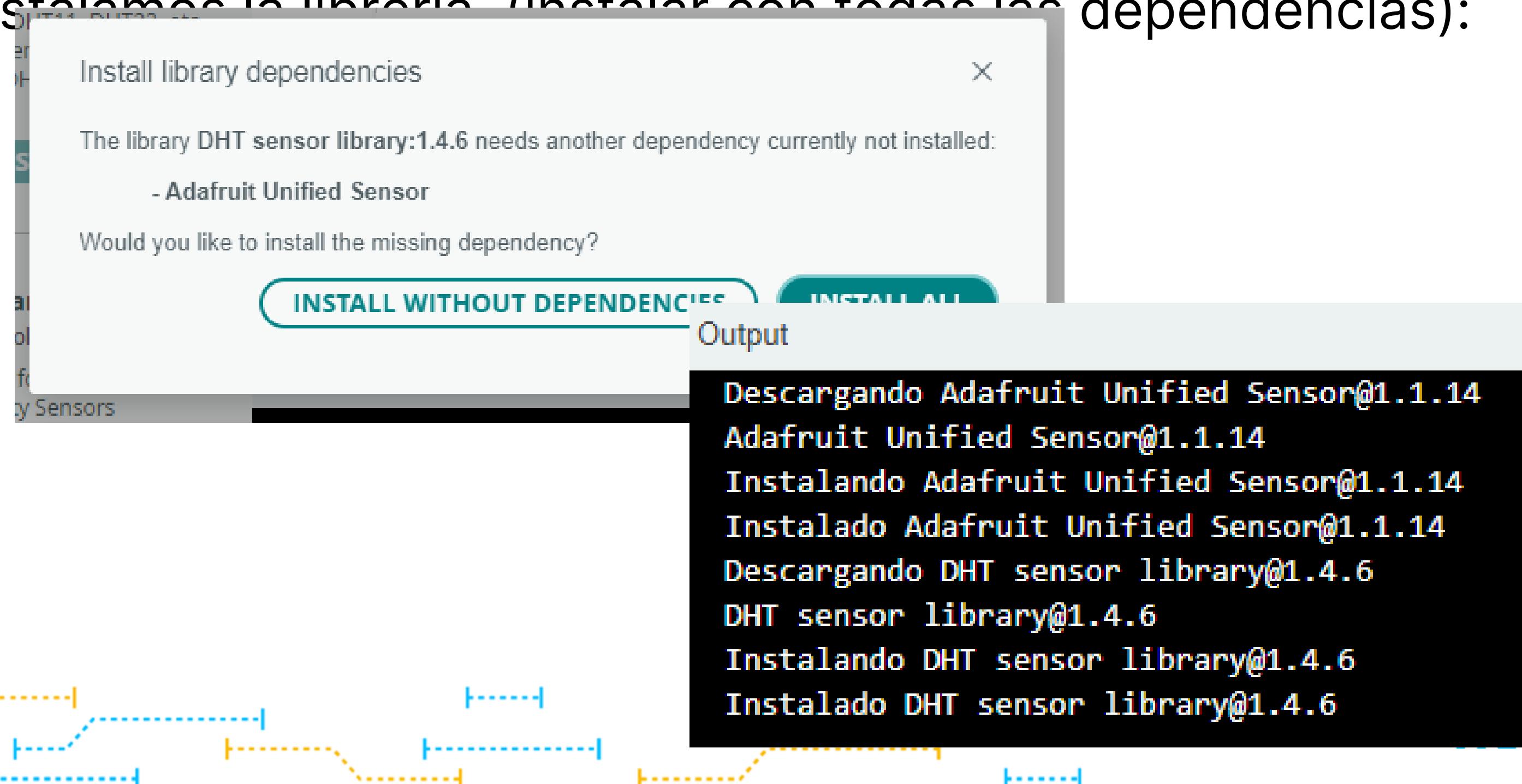
Por ejemplo para el sensor DHT22:

Buscamos el nombre del sensor en la barra de búsqueda:



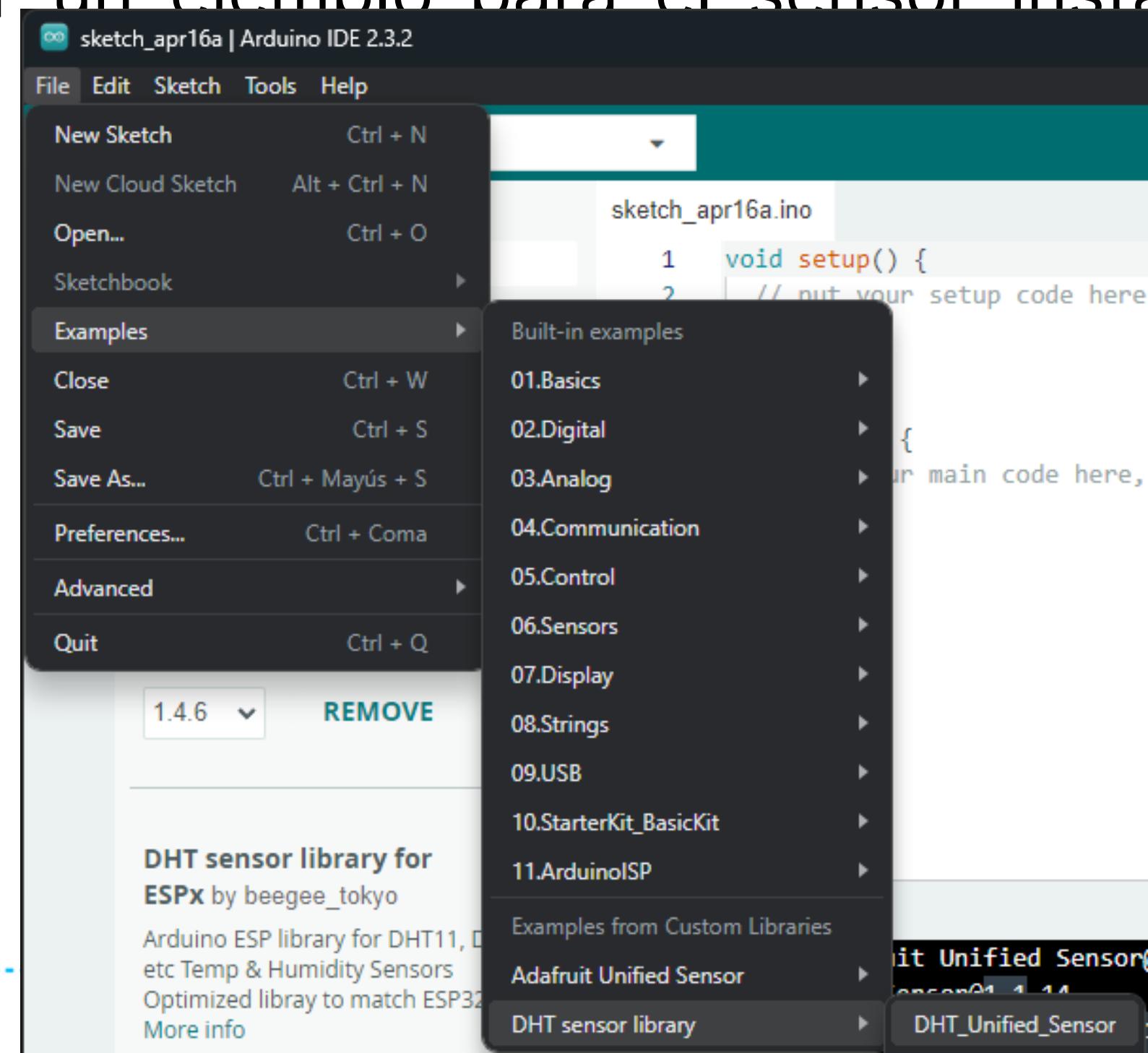
Por ejemplo para el sensor DHT22:

Instalamos la librería (instalar con todas las dependencias):



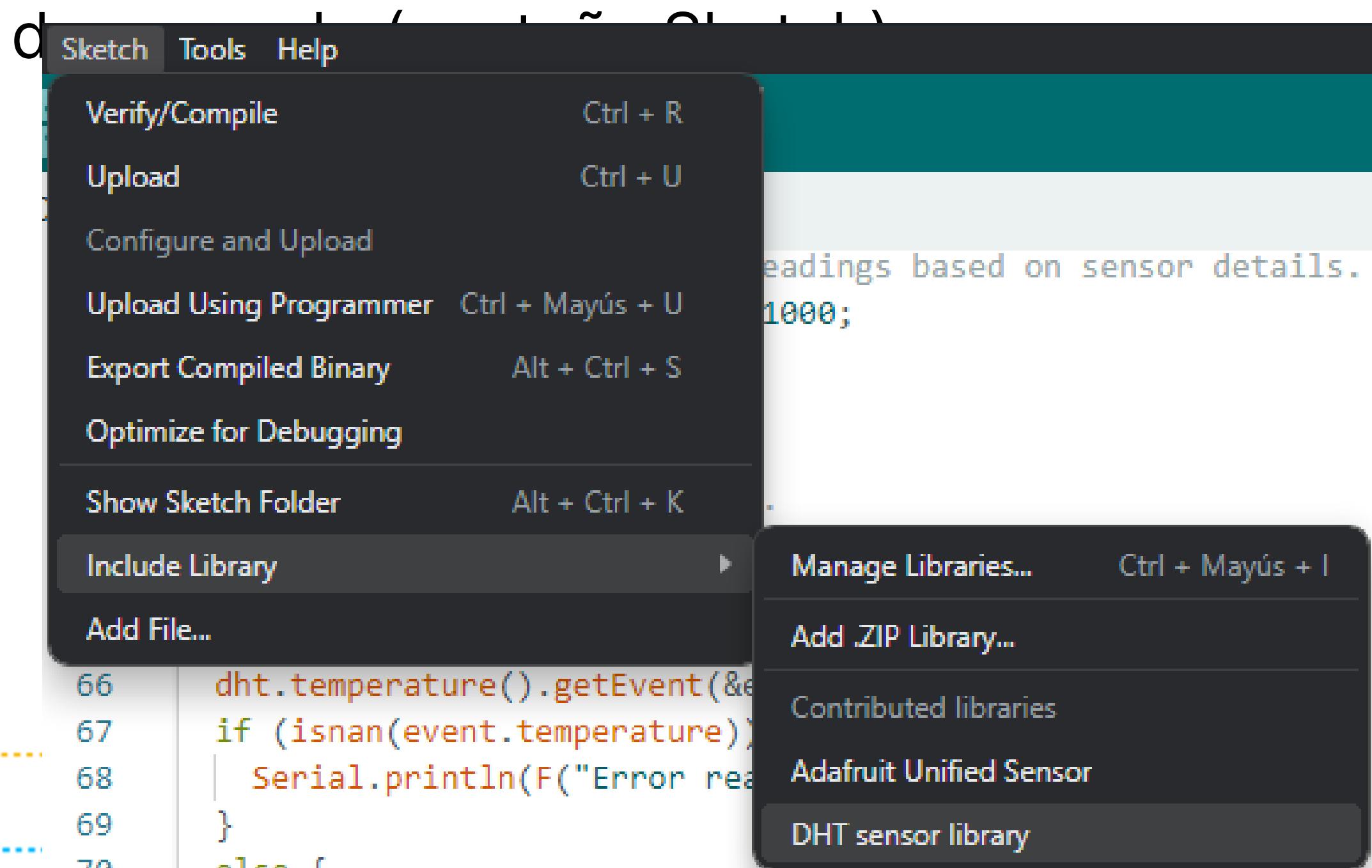
Por ejemplo para el sensor DHT22:

Luego podemos buscar un ejemplo para el sensor instalado (si la librería lo incluye)



Por ejemplo para el sensor DHT22:

También podemos buscar las librerías instaladas con el paquete



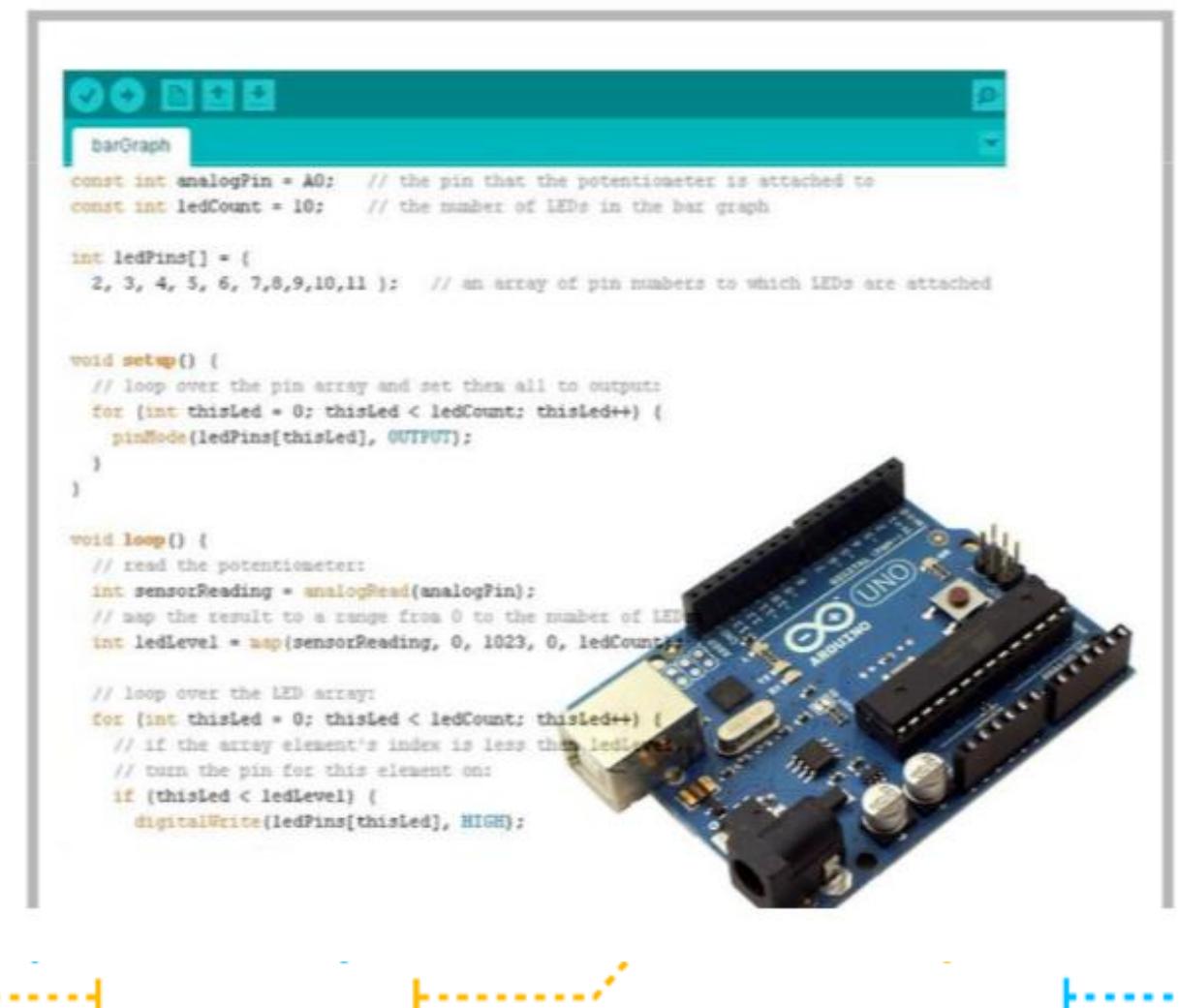
```
#include <DHT.h>  
  
#include <DHT_U.h>
```

Guía básica de Arduino.

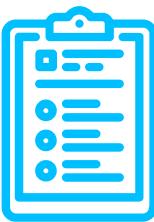
<https://www.calatman.com/board/001000012aa3ebf485e6e>

MANUAL DE ARDUINO

PROGRAMACIÓN Y CONCEPTOS BÁSICOS



3.

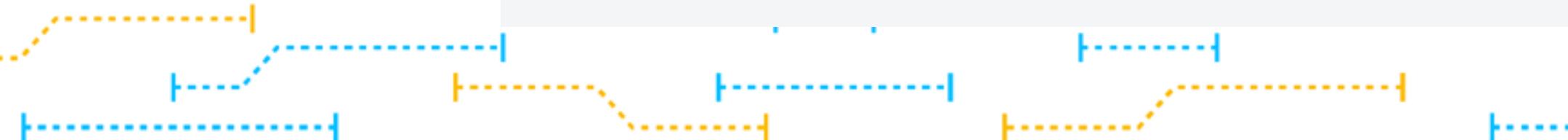
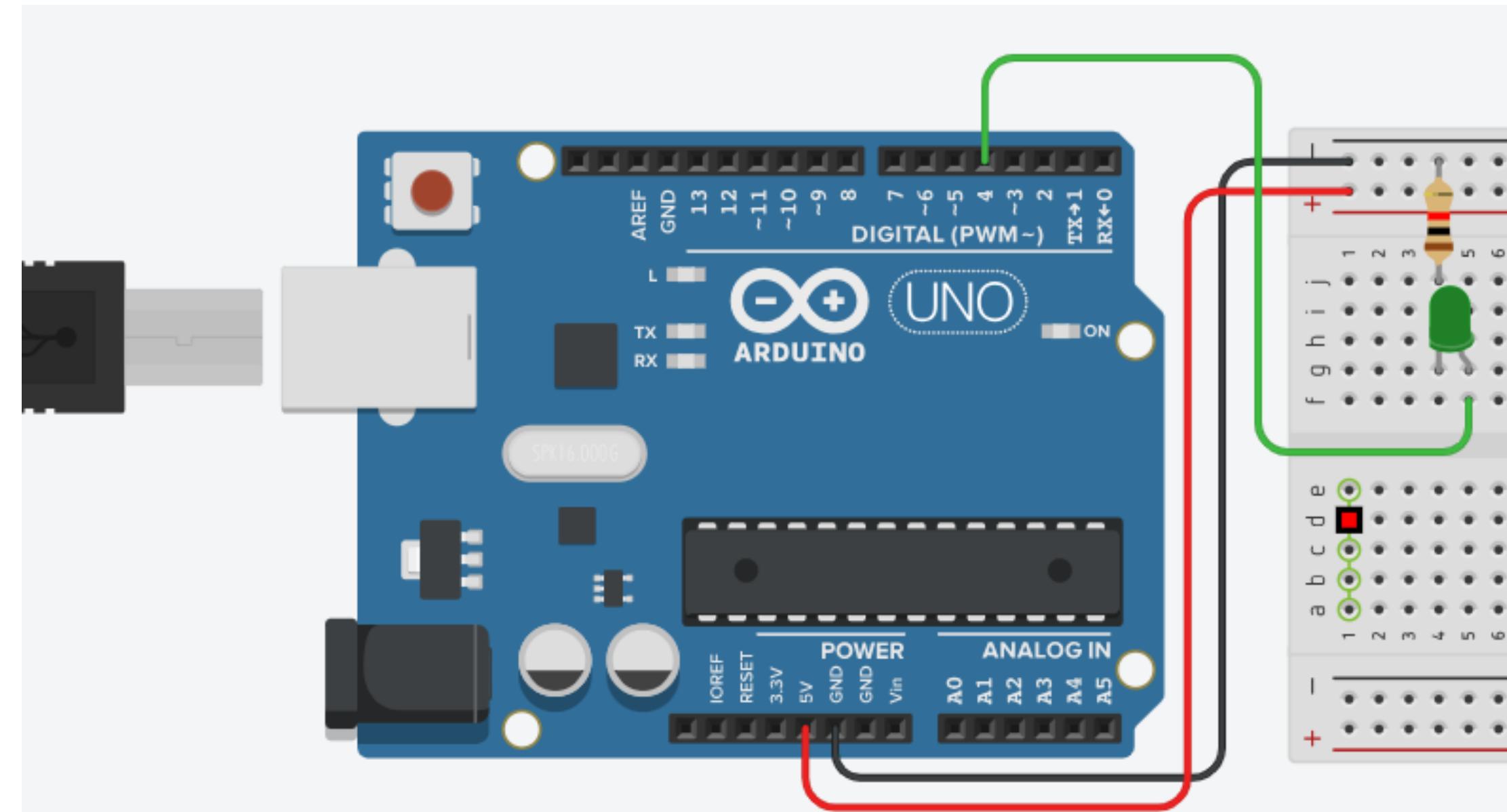


Ejemplos de Programación *Arduino*



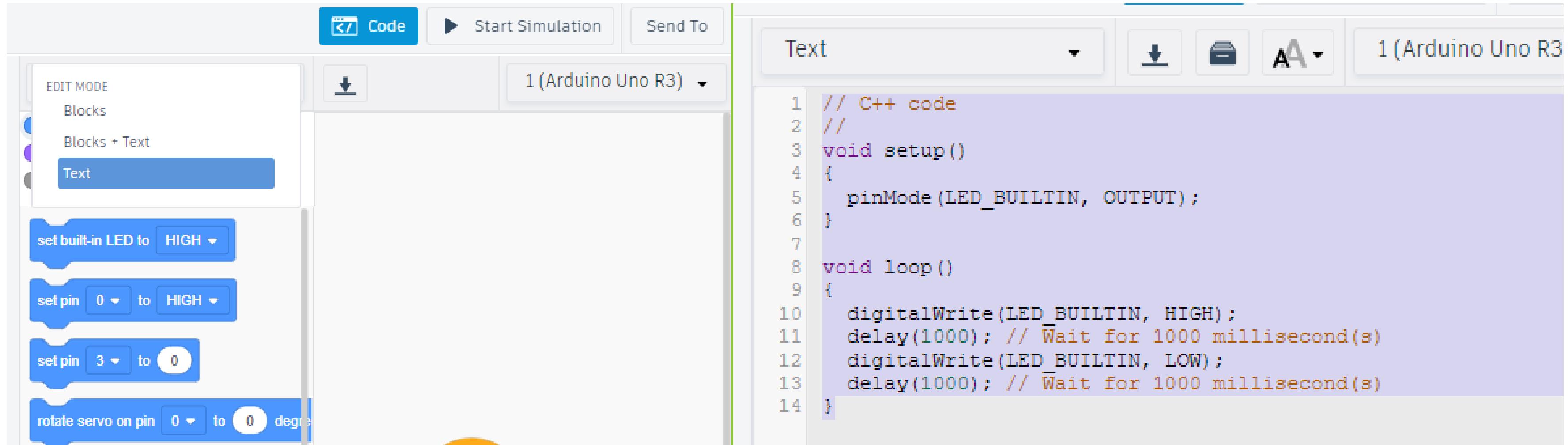
Ejemplo con LED

Ejemplo de encendido de LED con Arduino.



Ejemplo con LED

Ejemplo de encendido de LED con Arduino.



The image shows the Scratch IDE interface. On the left, the script editor displays four blue code blocks:

- set built-in LED to [HIGH v]
- set pin [0 v] to [HIGH v]
- set pin [3 v] to [0 v]
- rotate servo on pin [0 v] to [0 v] degree

On the right, the script editor displays the generated C++ code for an Arduino Uno R3:

```
1 // C++ code
2 //
3 void setup()
4 {
5     pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 void loop()
9 {
10    digitalWrite(LED_BUILTIN, HIGH);
11    delay(1000); // Wait for 1000 millisecond(s)
12    digitalWrite(LED_BUILTIN, LOW);
13    delay(1000); // Wait for 1000 millisecond(s)
14 }
```



Ejemplo con LED

Ejemplo de encendido de LED con Arduino.

```
void setup()
{
    pinMode(4, OUTPUT);
}

void loop()
{
    digitalWrite(4, HIGH);
    delay(1000); // Wait for 1000
    millisecond(s)
    digitalWrite(4, LOW);
    delay(1000); // Wait for 1000
    millisecond(s)
}
```



Ejemplo con LED

Ejemplo de encendido de LED con Arduino.

The screenshot shows the Tinkercad interface with a breadboard setup and a code editor.

Breadboard Setup: An Arduino Uno board is connected to a breadboard. Pin 4 (Digital PWM) is connected to a red LED through a 220 ohm resistor. Pin 4 is also connected to ground via a pushbutton. The pushbutton is connected between digital pin 5 and ground. The breadboard has a red power rail and a blue ground rail.

Code Editor: The code is written in C++ for an Arduino Uno R3.

```
// C++ code
//
3 int LEDAZUL = 4;
4 int button = 5;
5 void setup()
{
    pinMode(LEDAZUL, OUTPUT);
    pinMode(button, INPUT);
    digitalWrite(LEDAZUL, 0);
}
12 void loop()
{
    if(digitalRead(button)==1)
        digitalWrite(LEDAZUL, 1);
    else
        digitalWrite(LEDAZUL, 0);
    delay(100);
}
```

Scope Plot: At the bottom, there are two waveforms: a blue dashed line and an orange dashed line, representing the digital signals from the breadboard.

Ejemplo con LED

Ejemplo de encendido de LED con Arduino.

The screenshot shows the Tinkercad platform interface. At the top, there's a toolbar with various icons for file operations, simulation, and sharing. The main area displays a breadboard setup connected to an Arduino Uno. On the left, the Arduino board is shown with its pins labeled. A red wire connects the Arduino's 5V pin to the breadboard's power rail. A blue wire connects the Arduino's GND pin to the breadboard's ground rail. A purple wire connects the Arduino's digital pin 4 (labeled 'DIGITAL (PWM -)') to a red LED. A black wire connects the Arduino's digital pin 4 to a 220 ohm resistor. The other end of the resistor is connected to the positive power rail. A blue wire connects the Arduino's digital pin 5 to a blue LED. A black wire connects the Arduino's digital pin 5 to a 220 ohm resistor. The other end of the resistor is connected to the positive power rail. A black wire connects the Arduino's digital pin 5 to a push-button switch. The other end of the switch is connected to the negative power rail. Below the breadboard, two waveforms are displayed: a blue dashed line and a yellow dashed line, representing the digital signal from pin 4 and the button state respectively. To the right, the code editor shows the following C++ code:

```

1 // C++ code
2 //
3 int LEDAZUL = 4;
4 int button = 5;
5 void setup()
6 {
7     pinMode(LEDAZUL, OUTPUT);
8     pinMode(button, INPUT);
9     digitalWrite(LEDAZUL, 0);
10 }
11
12 void loop()
13 {
14     if(digitalRead(button)==1)
15         digitalWrite(LEDAZUL, 1);
16     else
17         digitalWrite(LEDAZUL, 0);
18     delay(100);
19 }

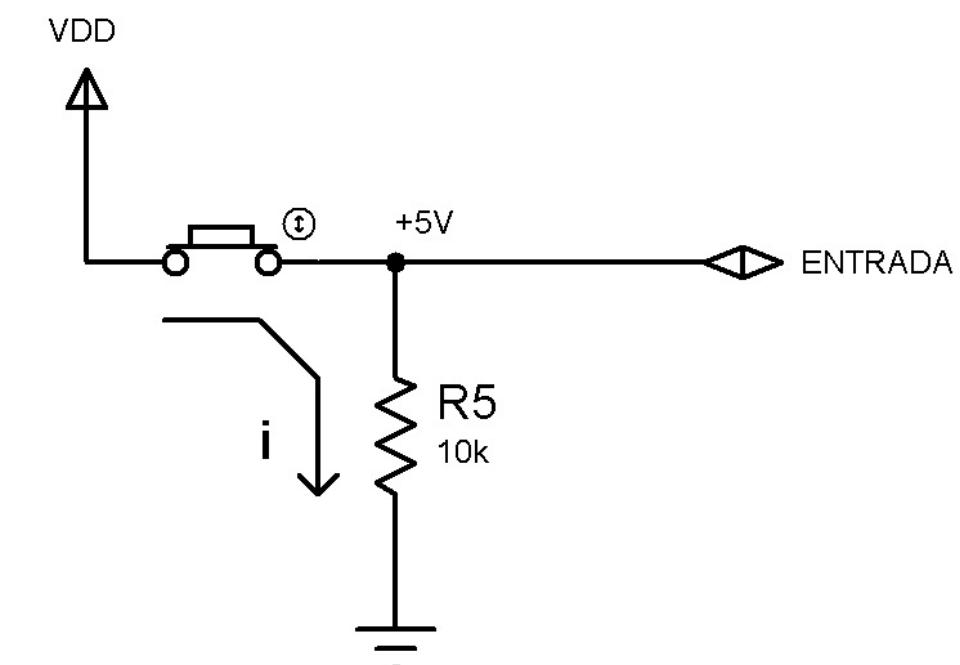
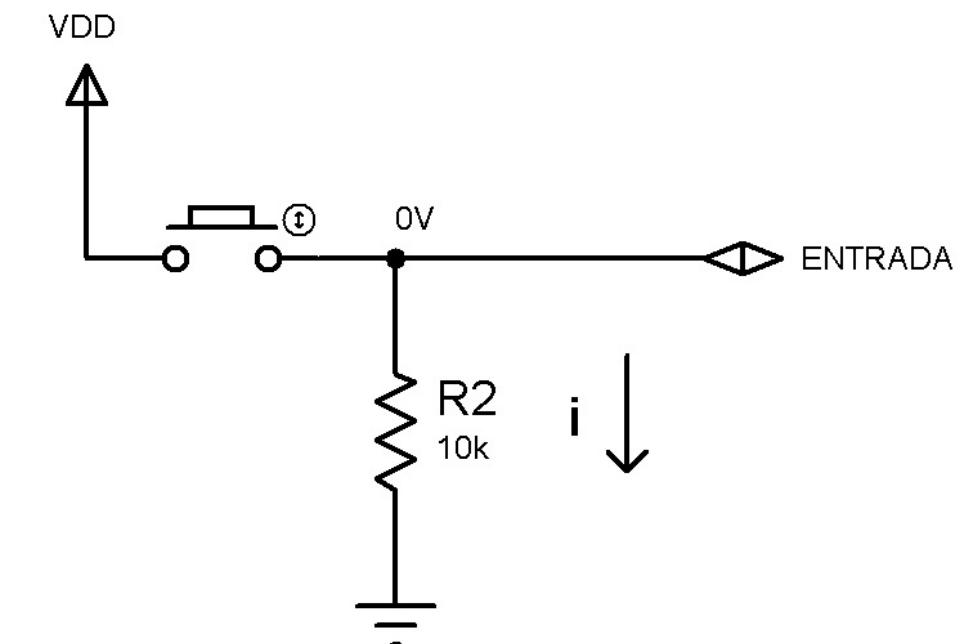
```

Ejemplo con LED + Botón

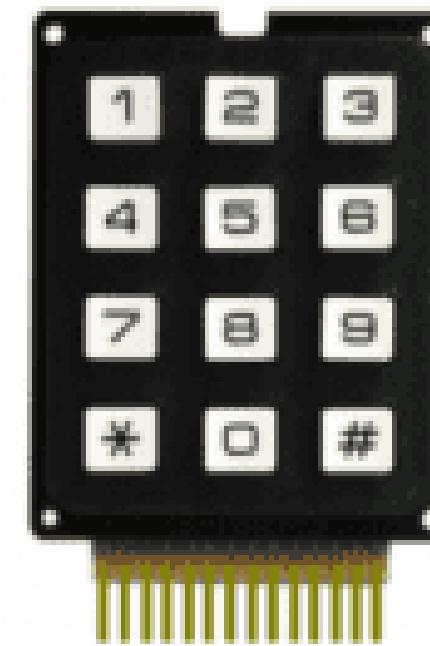
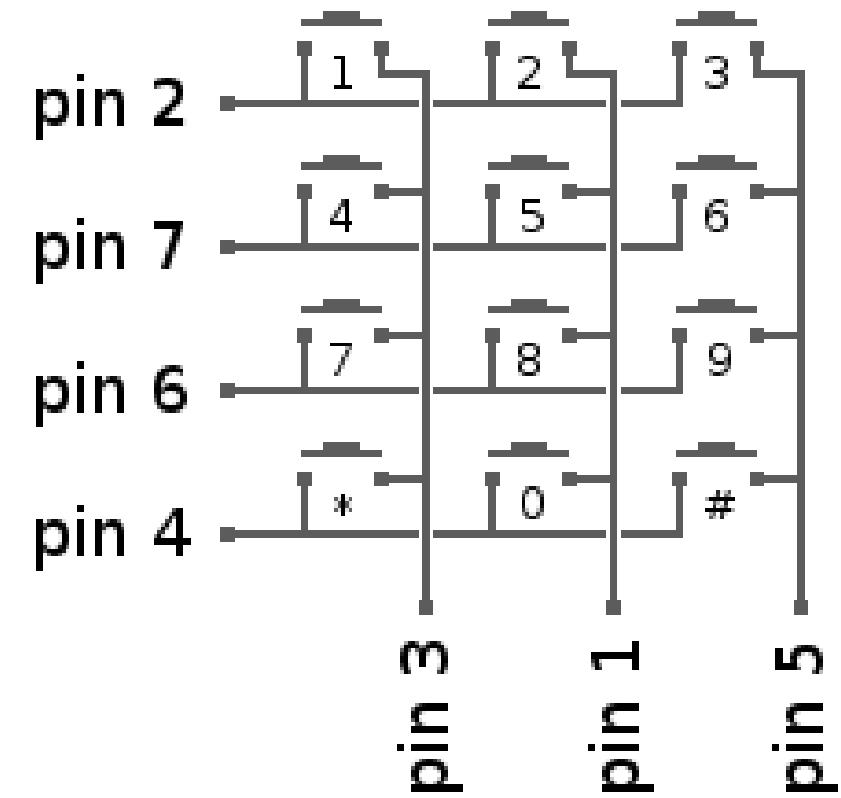
Uso de botón con Arduino.

```
// C++ code
//
int LEDAZUL = 4;
int button = 5;
void setup()
{
    pinMode(LEDAZUL, OUTPUT);
    pinMode(button, INPUT);
    digitalWrite(LEDAZUL, 0);
}

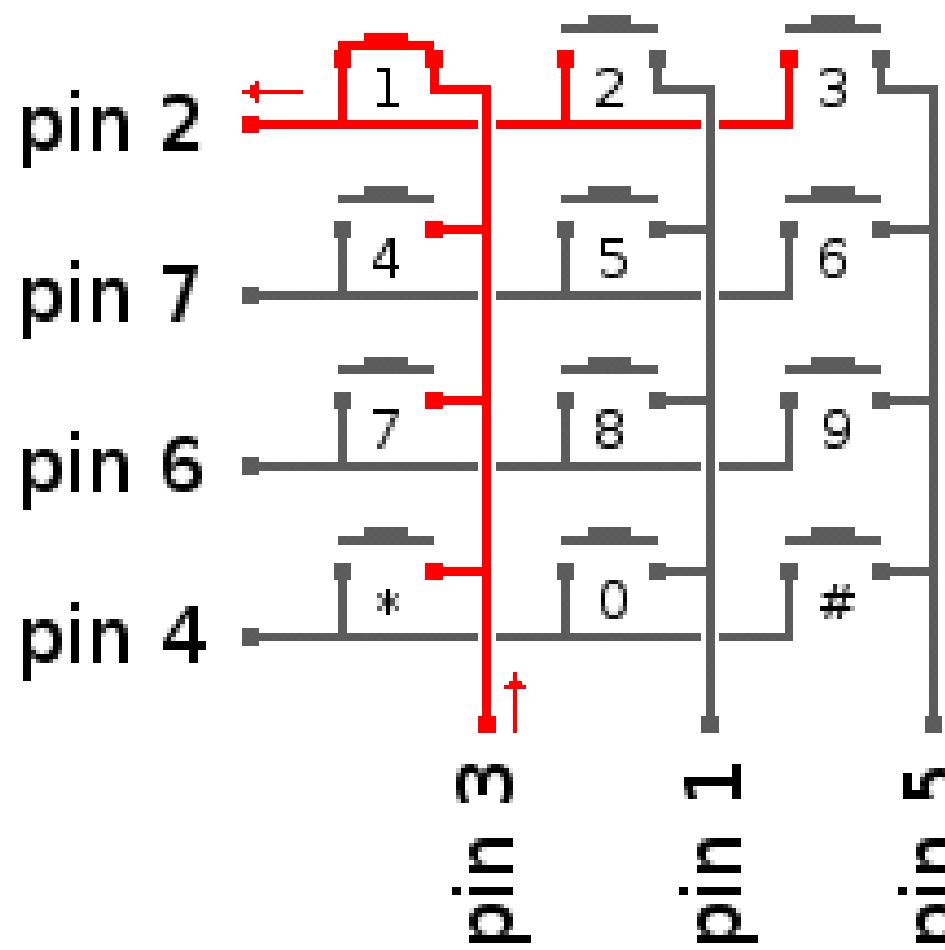
void loop()
{
    if(digitalRead(button)==1)
        digitalWrite(LEDAZUL, 1);
    else
        digitalWrite(LEDAZUL, 0);
    delay(100);
}
```



Uso de teclado matricial con Arduino.



Uso de teclado matricial con Arduino.



```
#include <Keypad.h>
const byte filas = 4;
const byte columnas = 4;
byte pinesFilas[] = {9, 8, 7, 6};
byte pinesColumnas[] = {5, 4, 3, 2};
char teclas[4][4] = {{'1','2','3','A'},
                     {'4','5','6','B'},
                     {'7','8','9','C'},
                     {'*','0','#','D'}};

Keypad teclado1 = Keypad( makeKeymap(teclas), pinesFilas,
                           pinesColumnas, filas, columnas);
void setup() {
  Serial.begin(9600);
  Serial.println("Teclado 4x4 con Biblioteca Keypad");
  Serial.println();
}
void loop() {
  //Verifica si alguna tecla fue presionada
  char tecla_presionada = teclado1.getKey();
  //Monitor Serial
  if (tecla_presionada)
  {
    Serial.print("Tecla: ");
    Serial.println(tecla_presionada);
  }
}
```

GRACIAS

Internet de las Cosas 2025-I