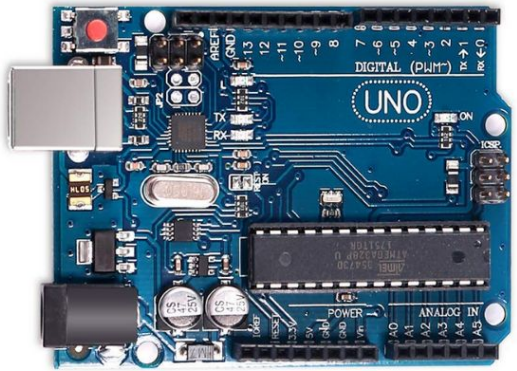
Taller de Arduino

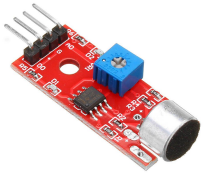
ING1004 - Desafíos de la Ingeniería

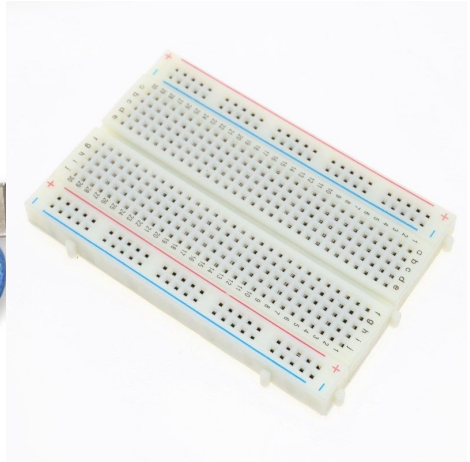
¿Qué veremos esta semana?

- Qué es Arduino
- Señales digitales y análogas
- Cómo se programa el microcontrolador
- Comunicación serial con la placa
- Interacción con sensores y actuadores
- Ley de ohm
- Uso de librerías
- Dónde encontrar más información

Específicamente...

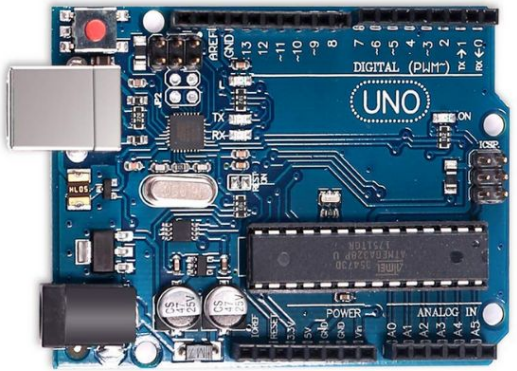






Hoy

- Detalles sobre Arduino
- Qué son los sensores y actuadores
- Señales digitales y análogas
- Comunicación Serial
- Cómo programar
- Cómo se programa un Arduino
- 3 Ejercicios de práctica usando la placa



This workshop is brought to you by…

La Resistencia

- Comunidad Maker en Ingeniería UC
- Hacemos proyectos y talleres de tecnología
- Tenemos oficina en el Centro de Alumnos de Ingeniería (CAi)
- Cualquiera es libre de unirse y participar

laresistenciamaker Seguir

36 publicaciones 1.025 seguidores 578 seguidos

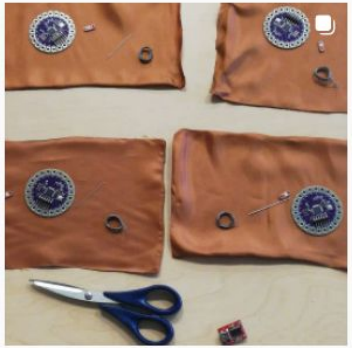
La Resistencia Maker

Comunidad

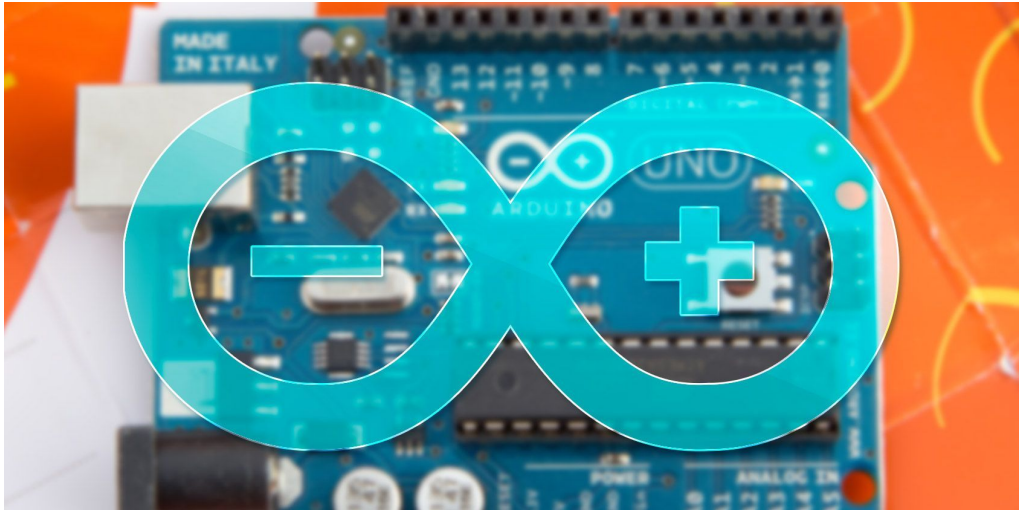
La comunidad Maker de Ingeniería UC 🛠⚡

Crear para saber

laresistencia.xyz



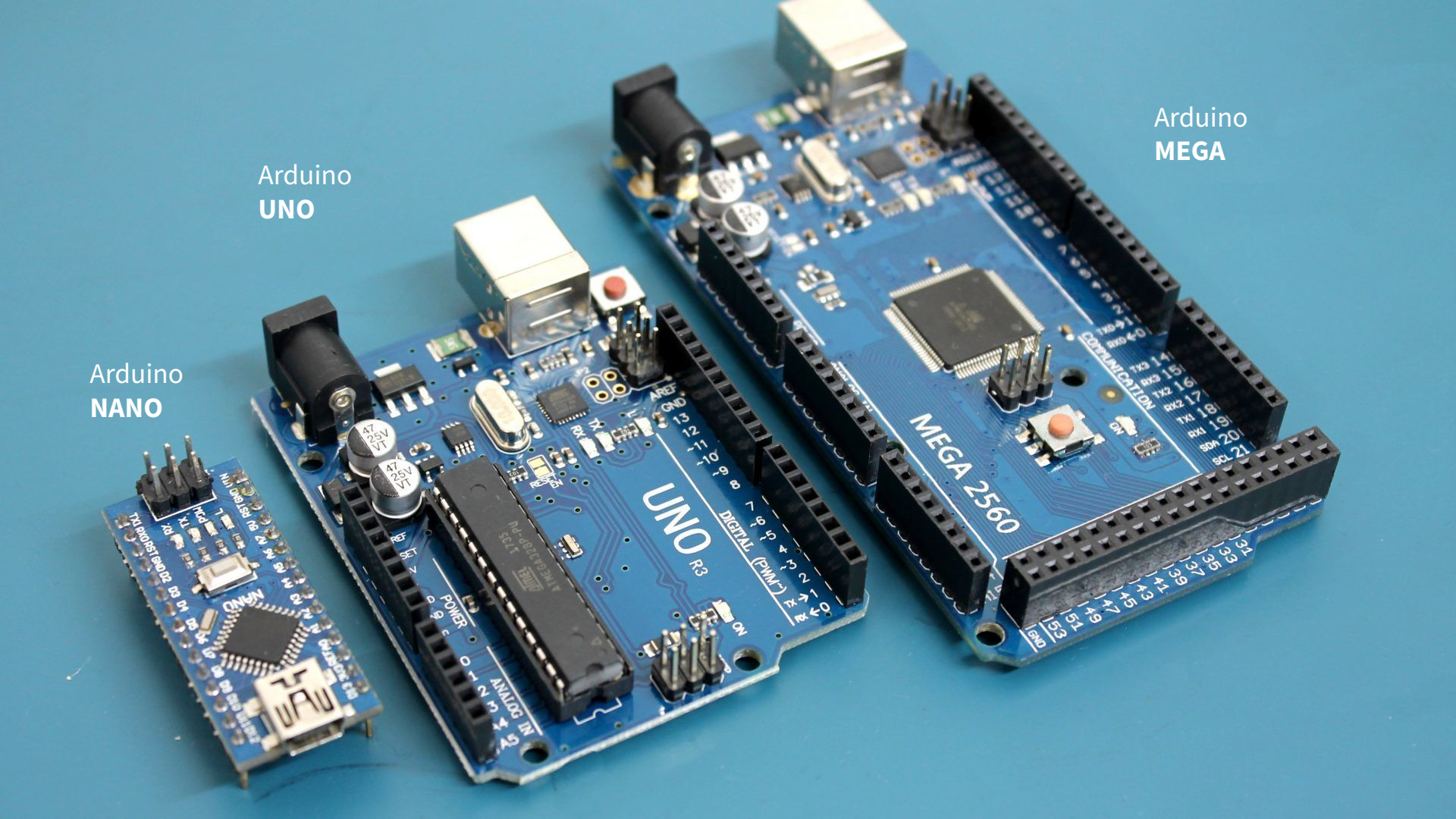
¿Qué es Arduino?



¿Qué es Arduino?

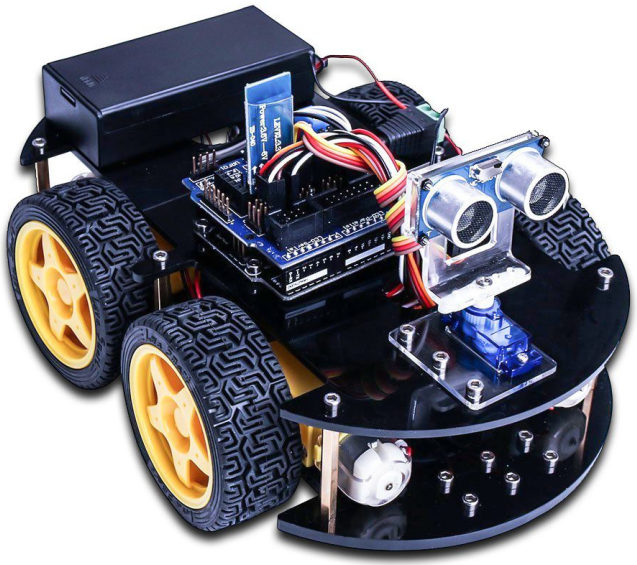
1. Empresa italiana de desarrollo de software y hardware **libre.**

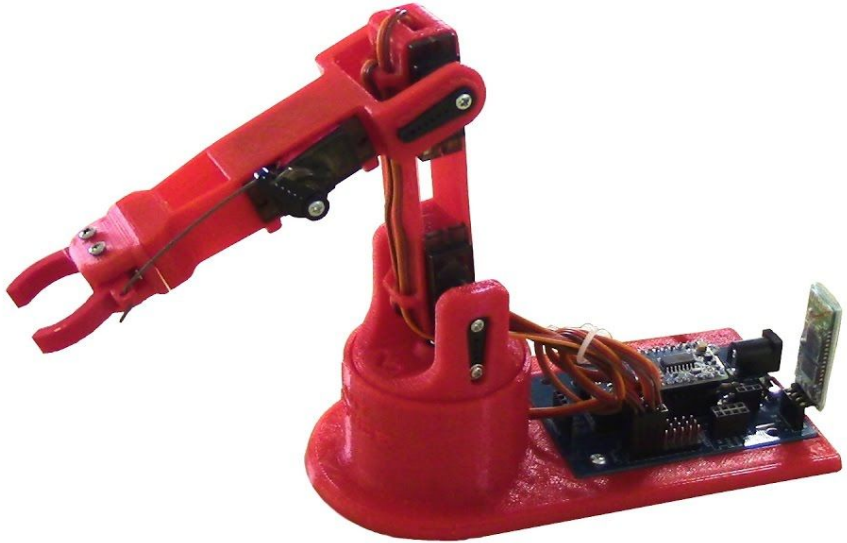
2. **Placas de desarrollo** de hardware

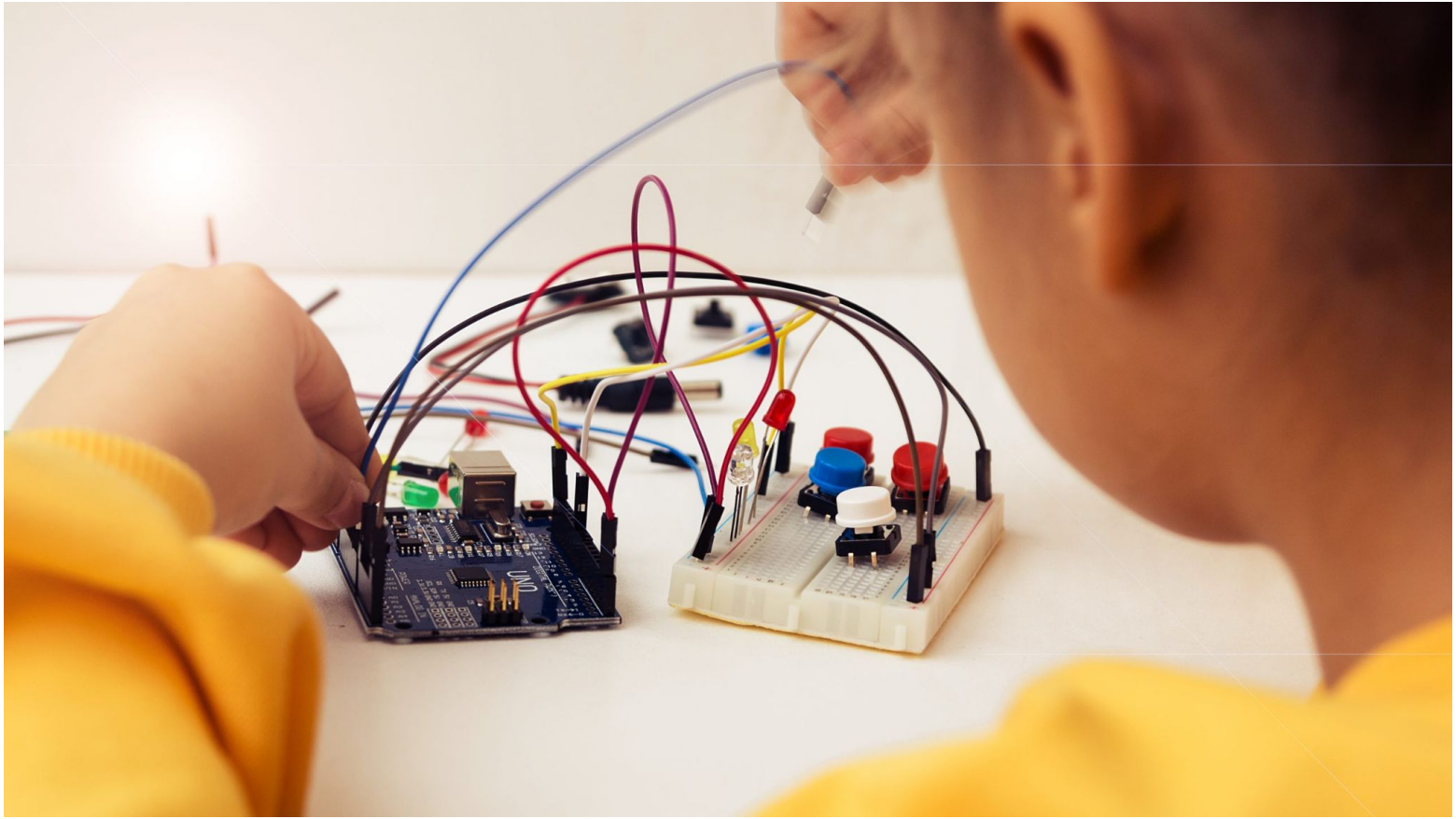

Arduino
NANO
Arduino
UNO
Arduino
MEGA
UNO R3
MEGA 2560

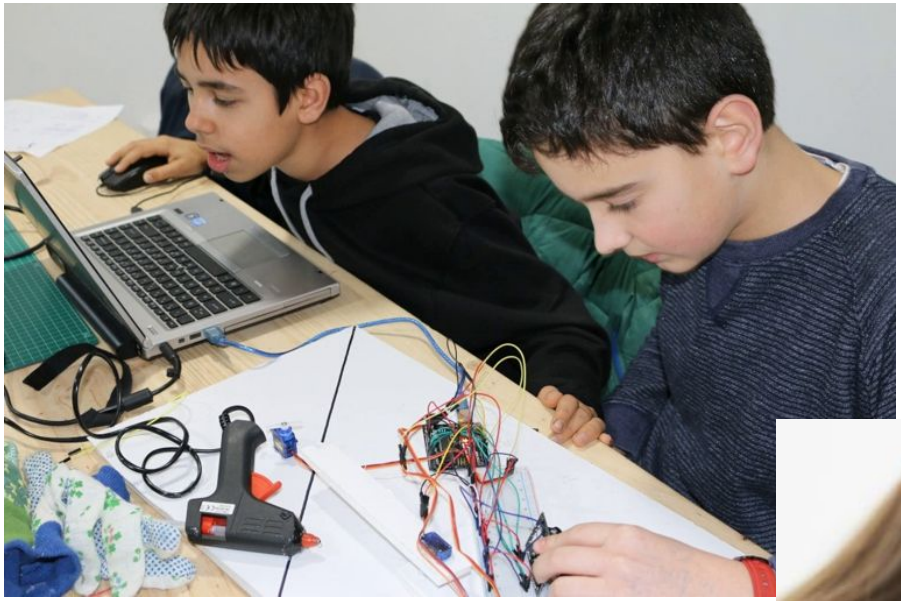
¿Para qué sirve Arduino?

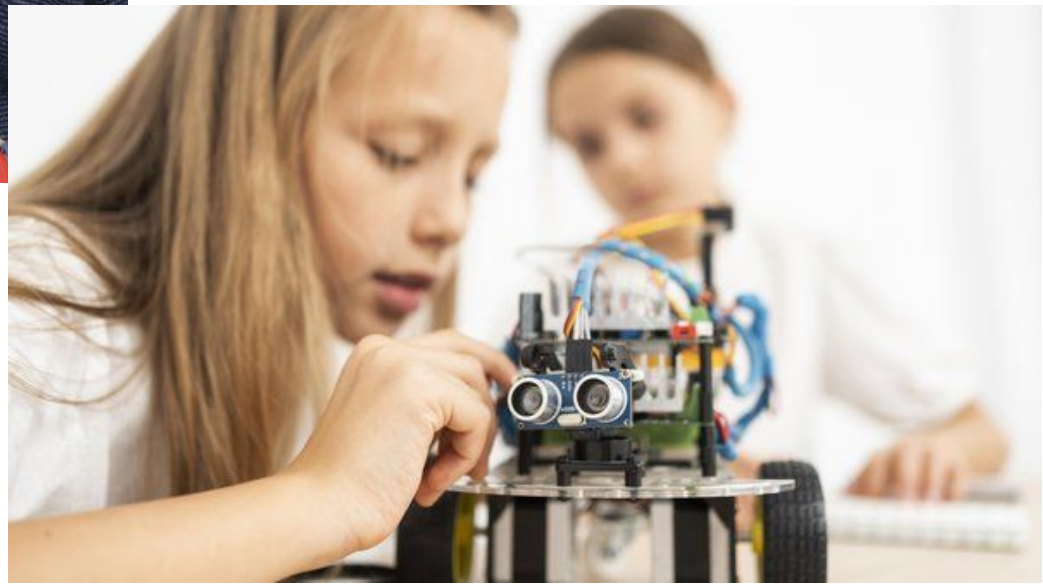
- Para **prototipar y crear dispositivos** que puedan detectar y/o controlar variables del mundo real de una forma sencilla.

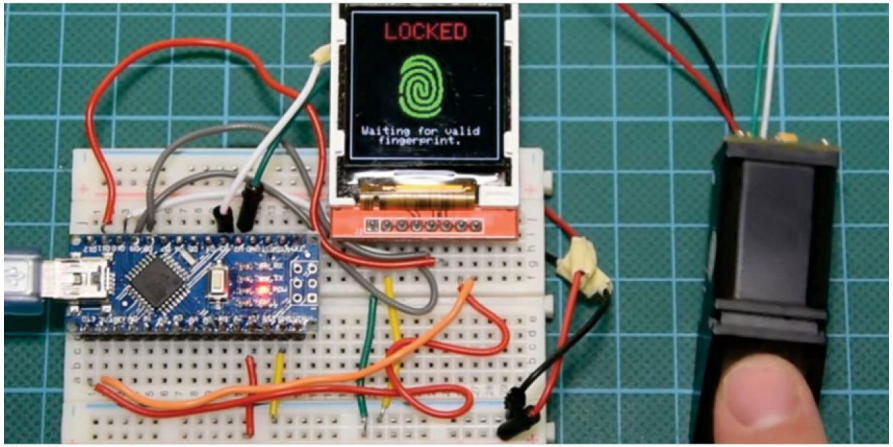


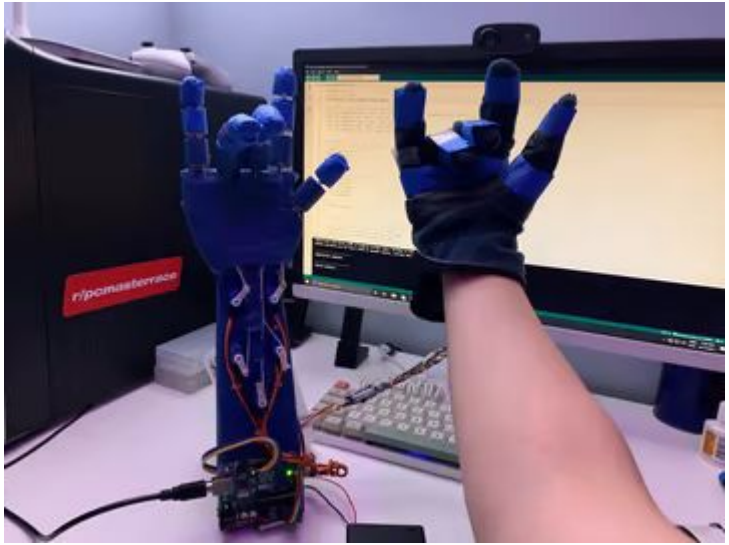


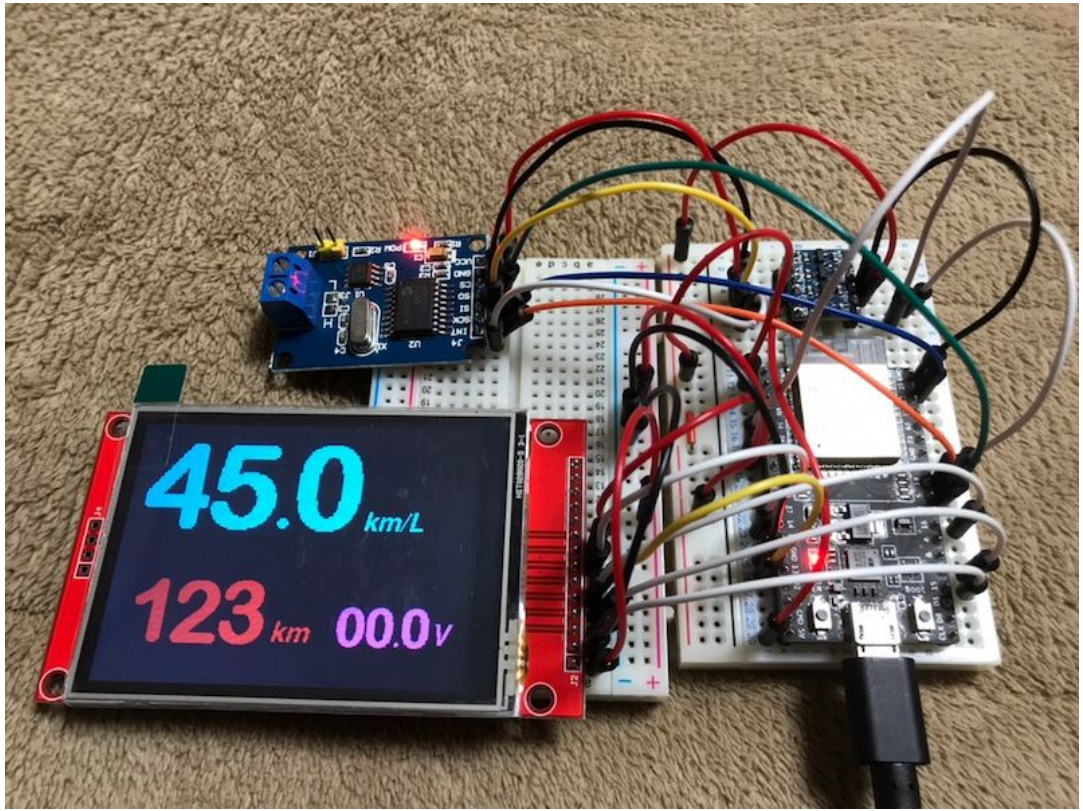


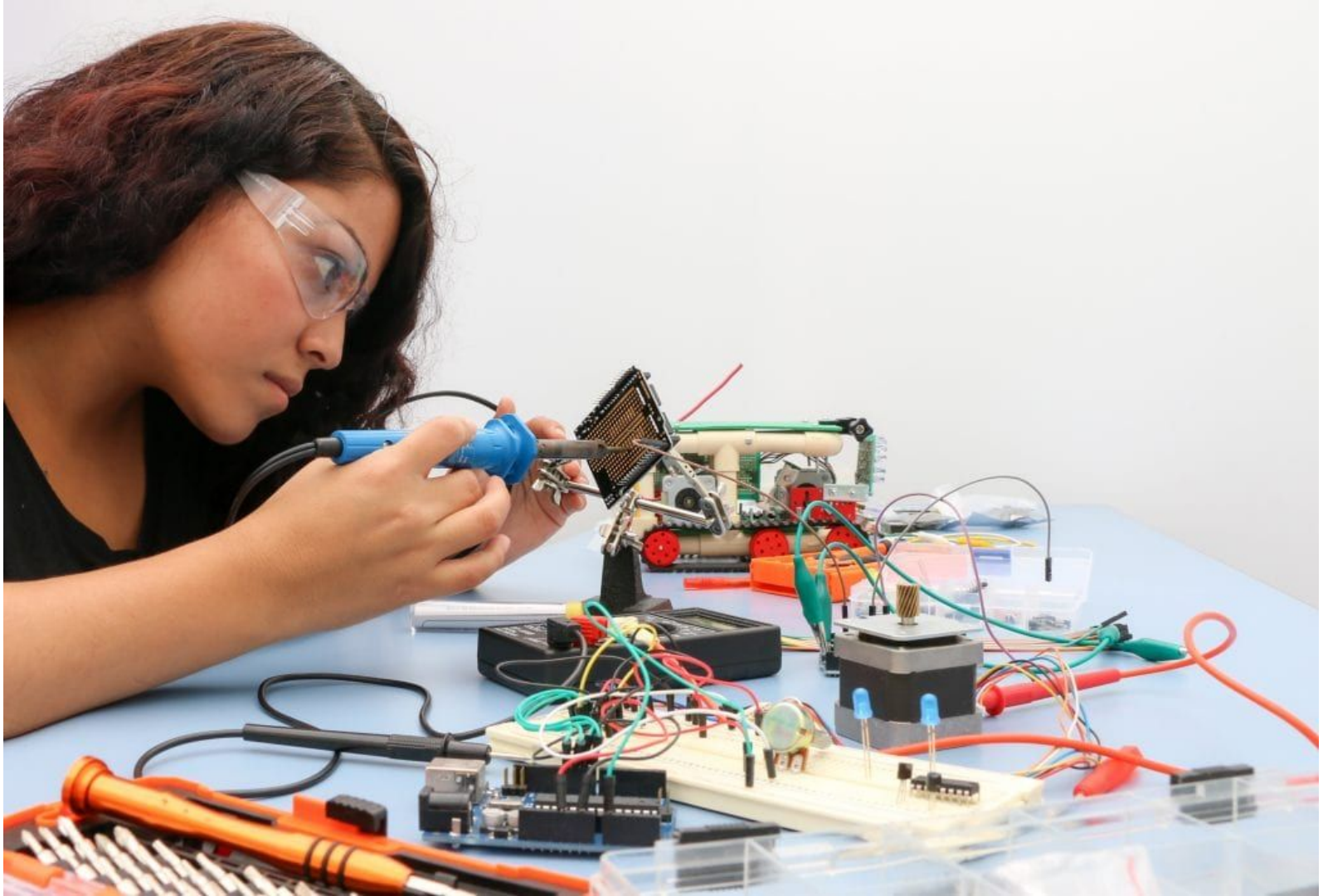


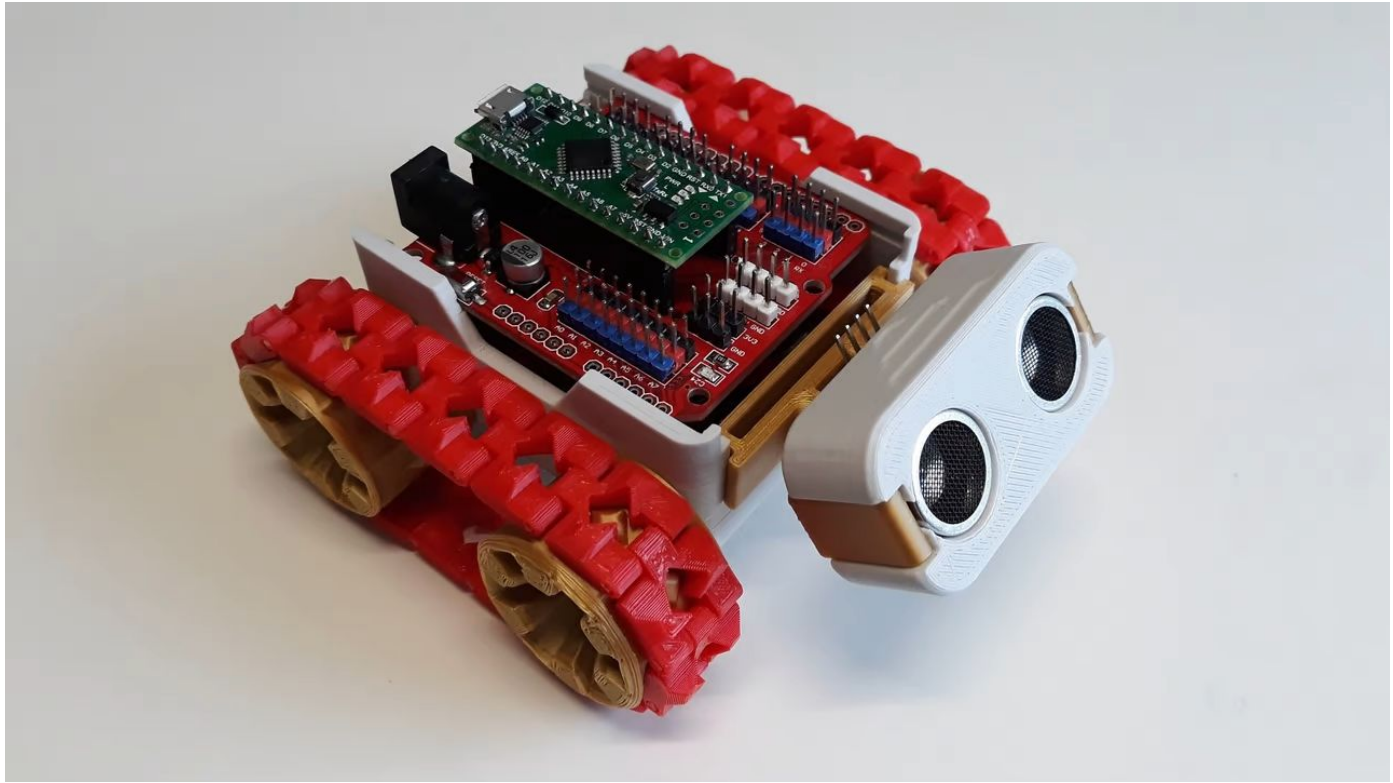



LOCKED
Waiting for valid fingerprint.


r/pcmasterrace


45.0 km/L
123 km 00.0 v

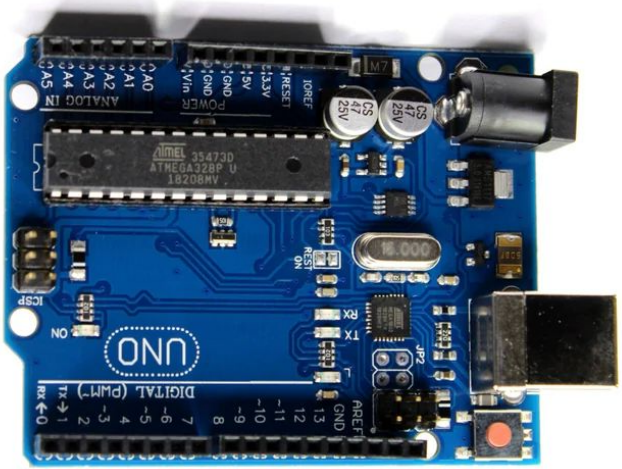


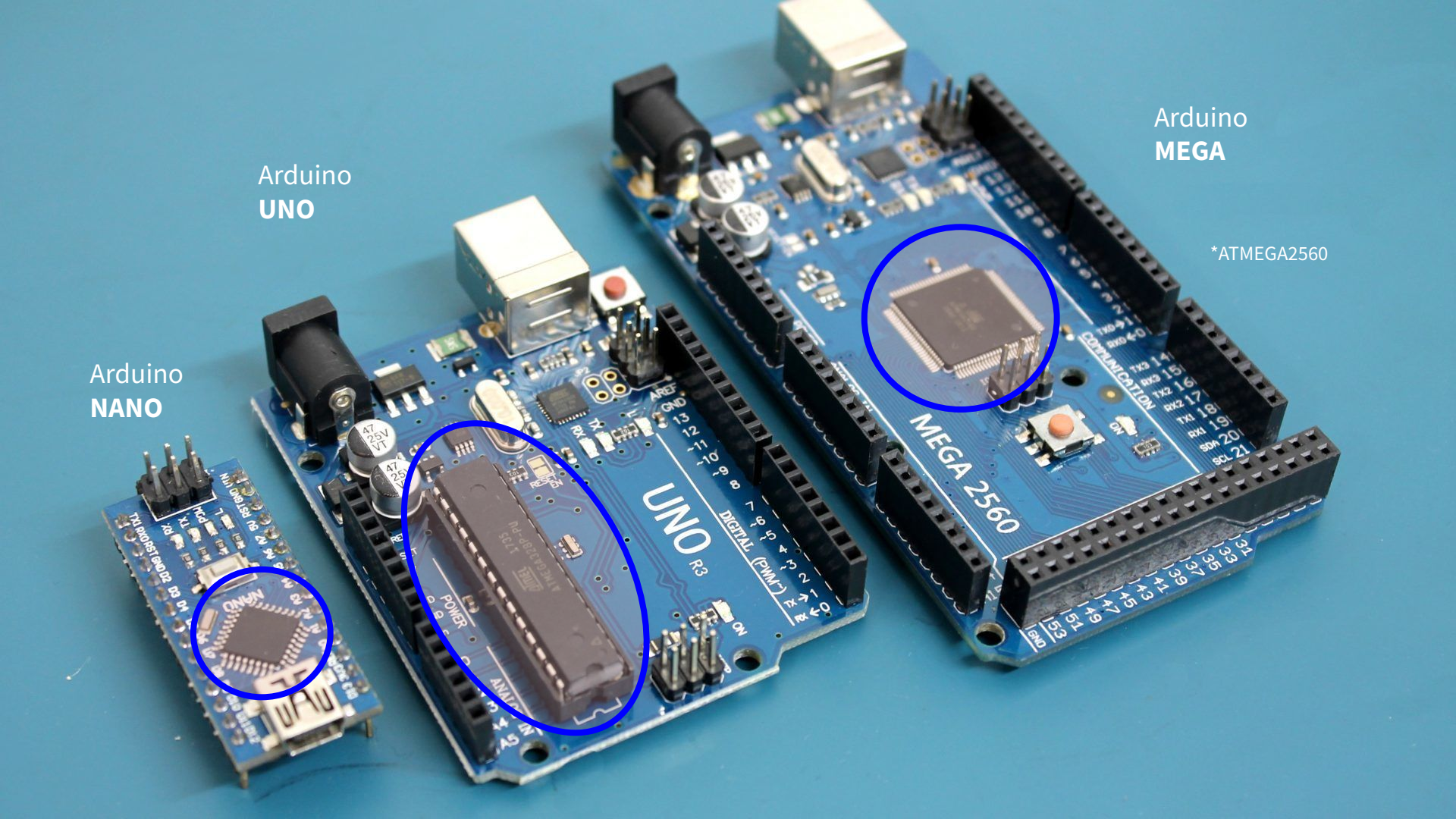


¿Cómo Arduino?

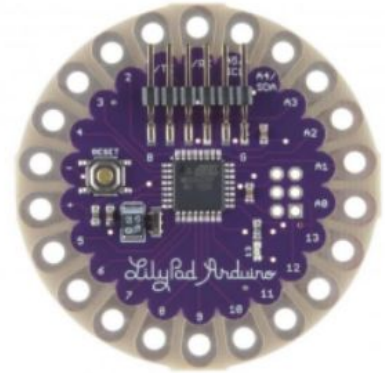
- Gracias a su microcontrolador re-programable (**MCU**).
- Gracias a su entorno de desarrollo integrado (**IDE**)

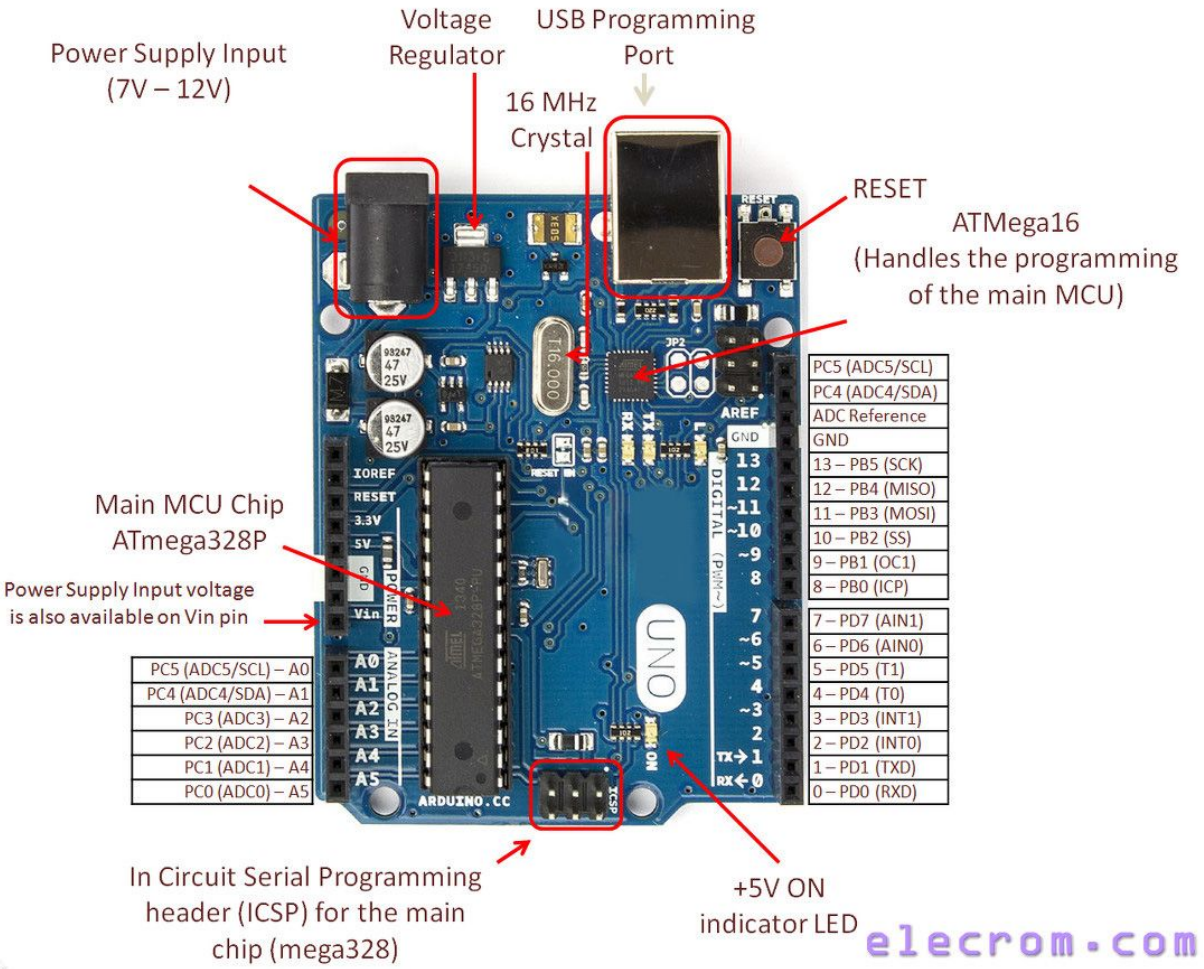
Microcontroller Unit (MCU)



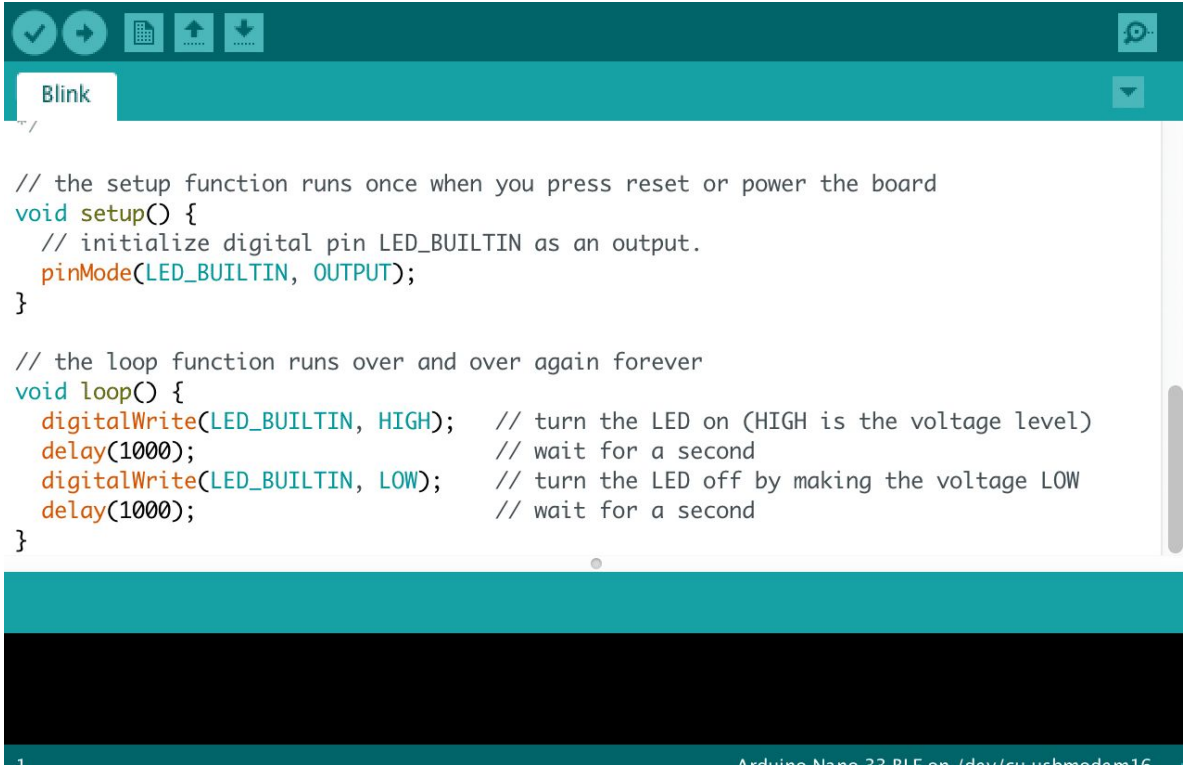

Arduino
NANO
Arduino
UNO
Arduino
MEGA
*ATMEGA2560
UNO R3
MEGA 2560

Otras variaciones



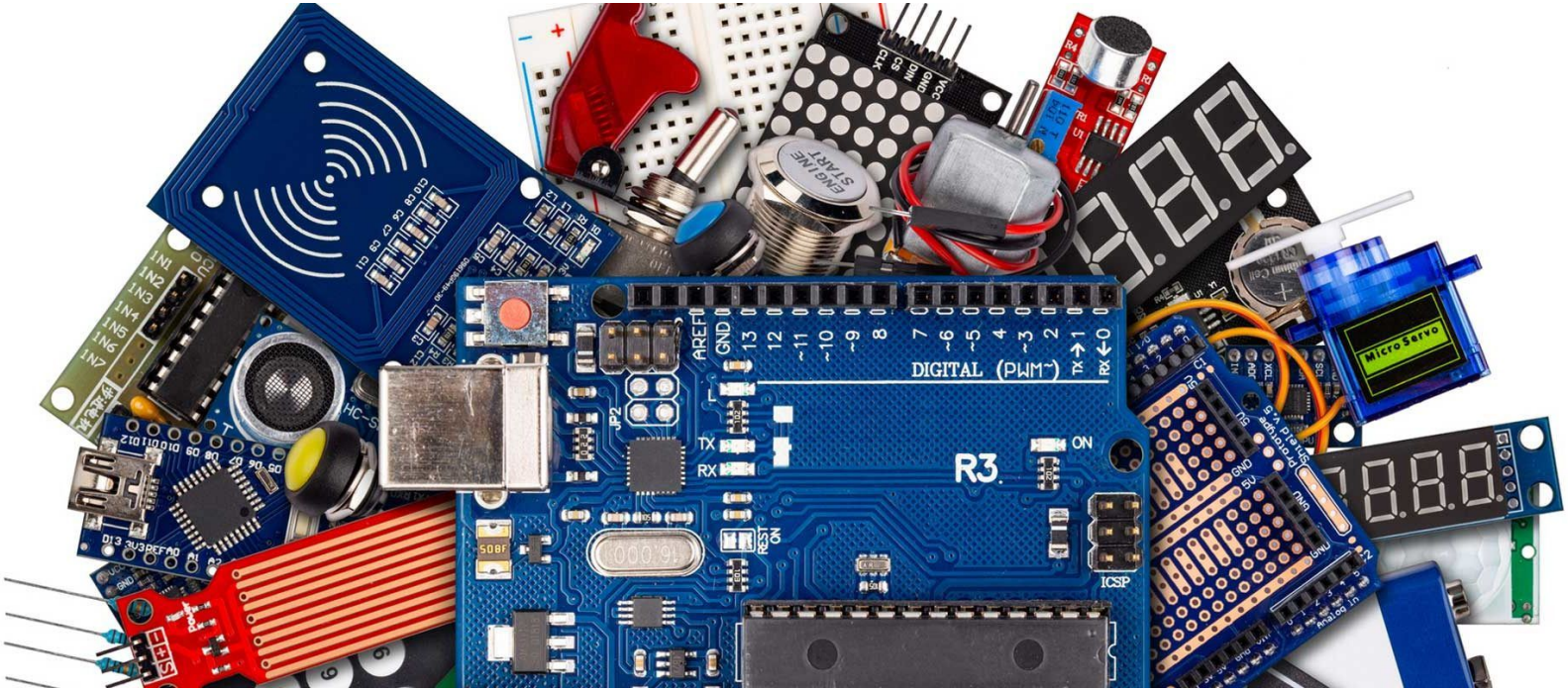

Power Supply Input
(7V – 12V)
Voltage
Regulator
USB Programming
Port
16 MHz
Crystal
RESET
ATMega16
(Handles the programming
of the main MCU)
Main MCU Chip
ATmega328P
Power Supply Input voltage
is also available on Vin pin
In Circuit Serial Programming
header (ICSP) for the main
chip (mega328)
+5V ON
indicator LED
PC5 (ADC5/SCL)
PC4 (ADC4/SDA)
ADC Reference
GND
13 – PB5 (SCK)
12 – PB4 (MISO)
11 – PB3 (MOSI)
10 – PB2 (SS)
9 – PB1 (OC1)
8 – PB0 (ICP)
7 – PD7 (AIN1)
6 – PD6 (AIN0)
5 – PD5 (T1)
4 – PD4 (T0)
3 – PD3 (INT1)
2 – PD2 (INT0)
1 – PD1 (TXD)
0 – PD0 (RXD)
PC5 (ADC5/SCL) – A0
PC4 (ADC4/SDA) – A1
PC3 (ADC3) – A2
PC2 (ADC2) – A3
PC1 (ADC1) – A4
PC0 (ADC0) – A5
ARDUINO.CC
UNO

Integrated development environment (IDE)



¿Por qué Arduino?

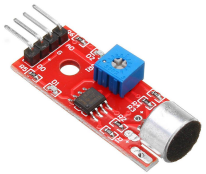
- Código y electrónica **libres**.
- Gran **comunidad**, mucha **documentación**.
- **Librerías** para casi todos los **módulos** que generalmente se diseñan para que se usen con Arduino.


DIGITAL (PWM~)
R3
ON
ICSP
MicroServo

Módulos

Sensores

- Permiten leer datos de nuestro entorno de forma digital o análoga
- Considerar:
  - Precisión
  - Exactitud
  - Consumo de energía
  - Tamaño
  - Otros



Actuadores

- Transforman energía en trabajo
- Más comunes: Luces y motores
- Actúan bajo la orden del microcontrolador (generalmente desencadenada por la lectura de un sensor)

Ultrasonic module	Human body sensor module	Tilt sensor	Photosensitive sensor
			
Smoke sensor	Infrared barrier sensor	Vibration sensor	Sound sensor
			
1 path search sensor	Flame sensor	Laser Head Sensor	Clock module
			
Super regenerative module	Temperature and humidity sensor	Raindrop sensor	Soil Sensors
			

Ultrasonic Sensor

Gas Sensor

Color Sensor

PIR Sensor

Accelerometer

Potentio-
meter

IR Sensor

Flex Sensor

LDR

Thermistor

Rain Sensor

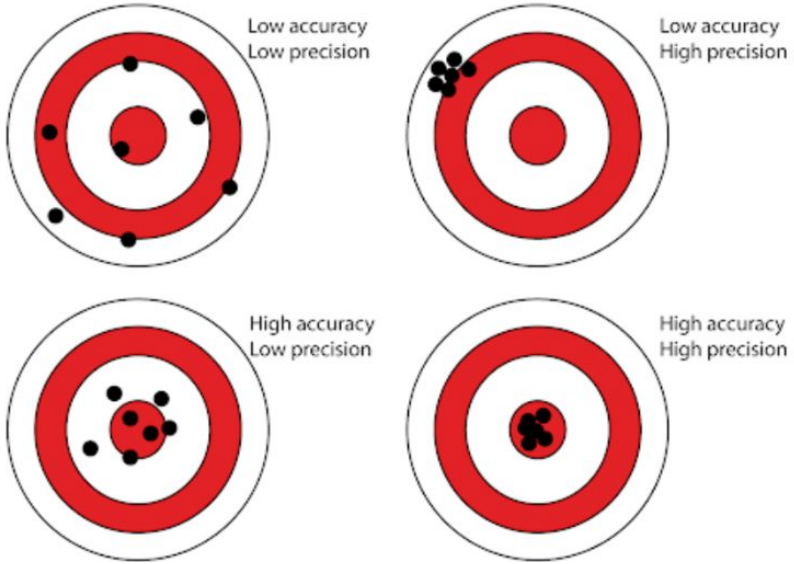
IR
Transmitter

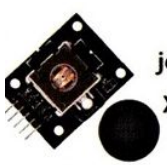
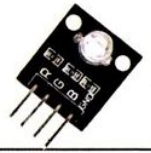
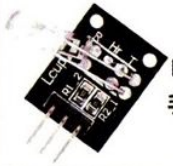
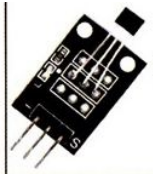
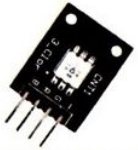
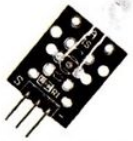
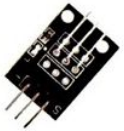
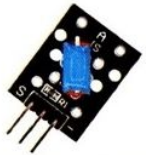
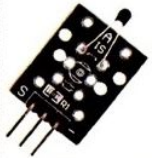
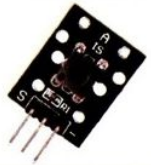
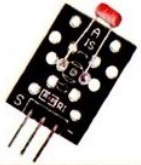
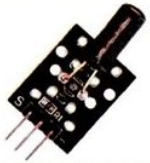
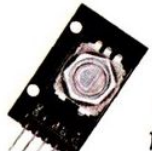
Photodiode
(IR Receiver)

LM35
(Temperature
Sensor)

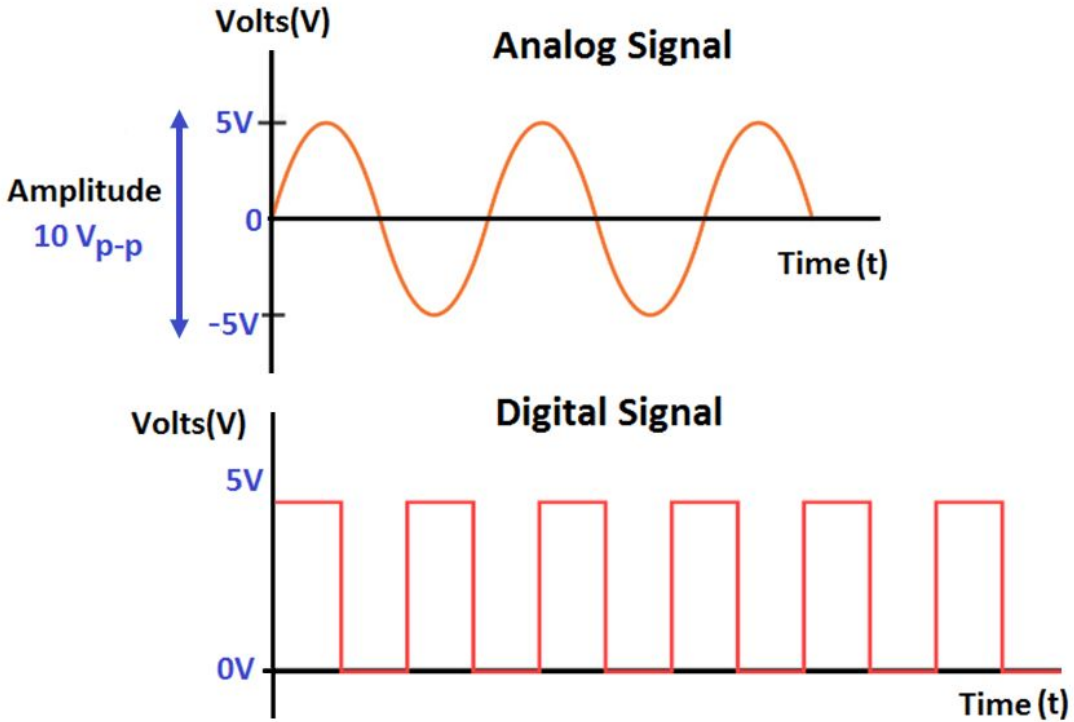
Micro-
phone

Hall
Sensor

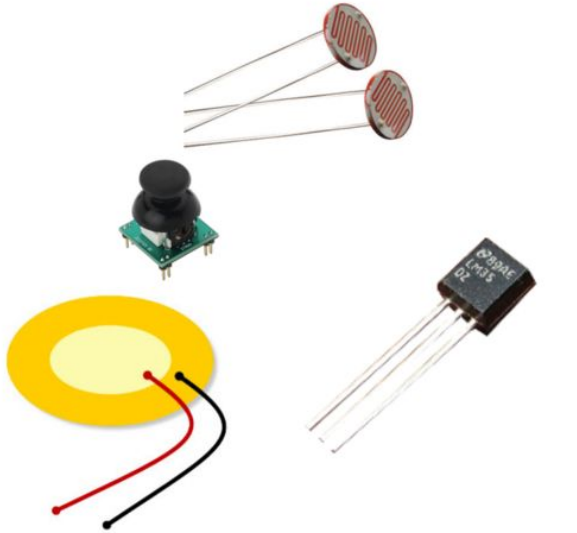

Low accuracy
Low precision
Low accuracy
High precision
High accuracy
Low precision
High accuracy
High precision

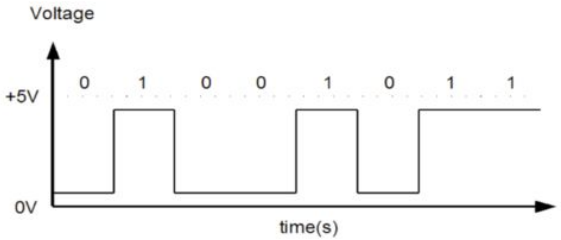
 joystick XY摇杆	 Flame 火焰传感器	 RGB LED 3色灯模块	 Heartbrat 手指测心跳	 Light Cup 魔术光杯	 Hall magnatic 霍尔磁力
 Relay 继电器	 Linear Hall 线性霍尔	 SMD RGB 3色RGB	 7 color flash 7彩闪烁	 Tilt switch 水银开关	 TEMP 18B20
 Big sound 高感度声音	 Touch 人体触摸	 Two-color LED 双色LED	 Laser emit 激光发射	 Ball switch 倾斜开关	 Analog temp 模拟温度
 Small sound 小麦克风	 Digital temp 数字温度	 Two-color 小双色	 Button 按键开关	 Photoresistor 光线	 IR emission 红外发射
 Tracking 循迹	 buzzer 有源蜂鸣器	 Reed switch 磁簧开关	 Shock 震动开关	 Temp and humldlty 温湿度	 IR receiver 红外接收
 Avoid 避障碍	 passive buzzer 无源蜂鸣器	 Mini Reed 迷你磁簧	 Rotary encoders 旋转编码器	 Analog Hall 模拟霍尔	 Tap module 敲击模块 Light blocking 光遮断

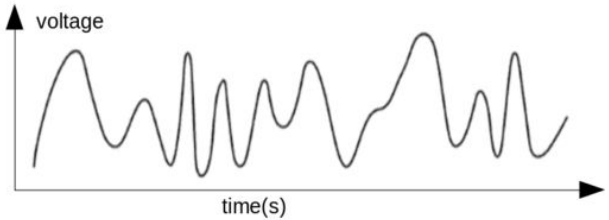
Señales digitales vs analógicas



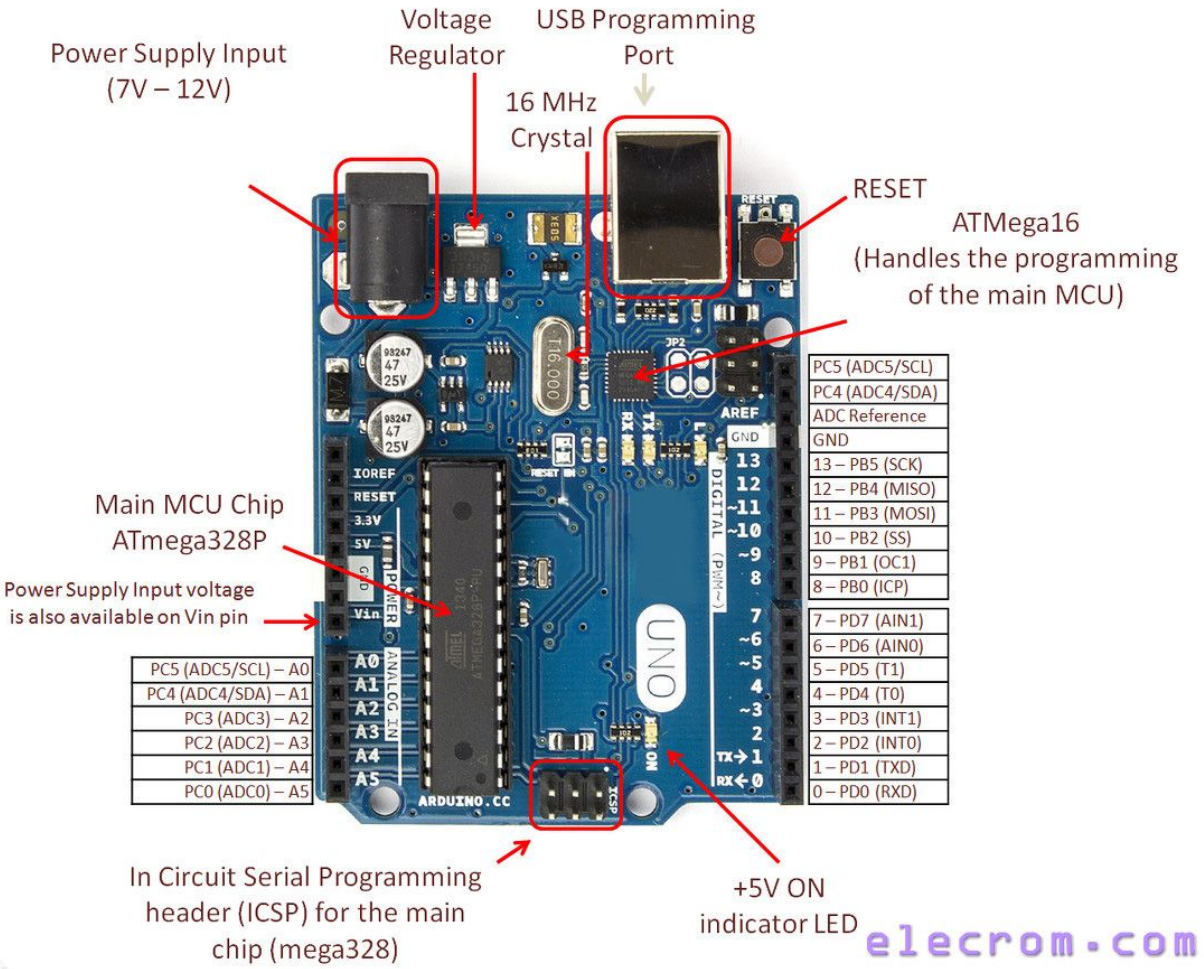
VS







https://www.linkedin.com/pulse/difference-between-digital-analog-sensors-ahmad-abdul-mageed


Power Supply Input
(7V – 12V)
Voltage
Regulator
USB Programming
Port
16 MHz
Crystal
RESET
ATMega16
(Handles the programming
of the main MCU)
Main MCU Chip
ATmega328P
Power Supply Input voltage
is also available on Vin pin
In Circuit Serial Programming
header (ICSP) for the main
chip (mega328)
+5V ON
indicator LED
PC5 (ADC5/SCL)
PC4 (ADC4/SDA)
ADC Reference
GND
13 – PB5 (SCK)
12 – PB4 (MISO)
11 – PB3 (MOSI)
10 – PB2 (SS)
9 – PB1 (OC1)
8 – PB0 (ICP)
7 – PD7 (AIN1)
6 – PD6 (AIN0)
5 – PD5 (T1)
4 – PD4 (T0)
3 – PD3 (INT1)
2 – PD2 (INT0)
1 – PD1 (TXD)
0 – PD0 (RXD)
PC5 (ADC5/SCL) – A0
PC4 (ADC4/SDA) – A1
PC3 (ADC3) – A2
PC2 (ADC2) – A3
PC1 (ADC1) – A4
PC0 (ADC0) – A5
ARDUINO.CC
UNO

Actividades de hoy

- Comunicación serial

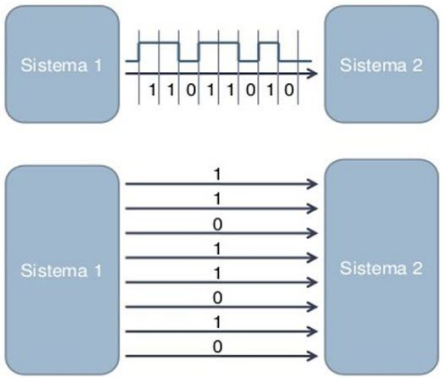
- Aprender a programar la placa (sin módulos)

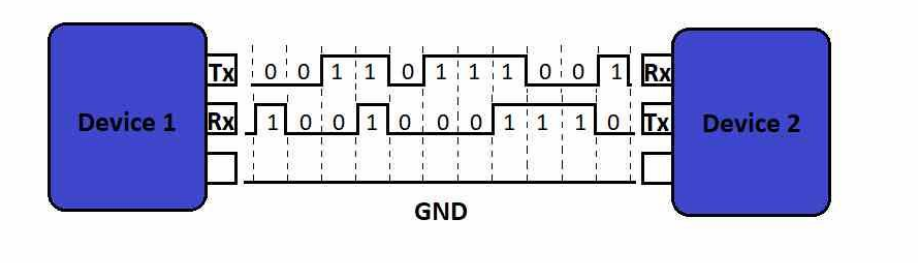
- Aprender a comunicarse con la placa (usando comunicación serial)

Comunicación serial

Es un protocolo de comunicación muy común y **estándar** para dispositivos electrónicos y computadores.

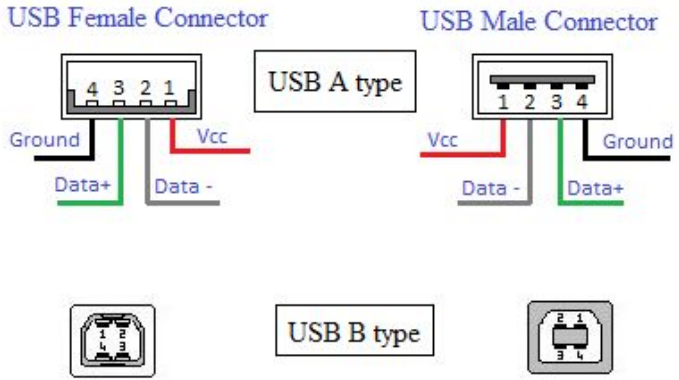
Permite enviar y recibir muchos **datos en serie** (uno tras otro) y evita que se tengan que usar muchos cables para comunicar algo (paralelo)





Comunicación serial

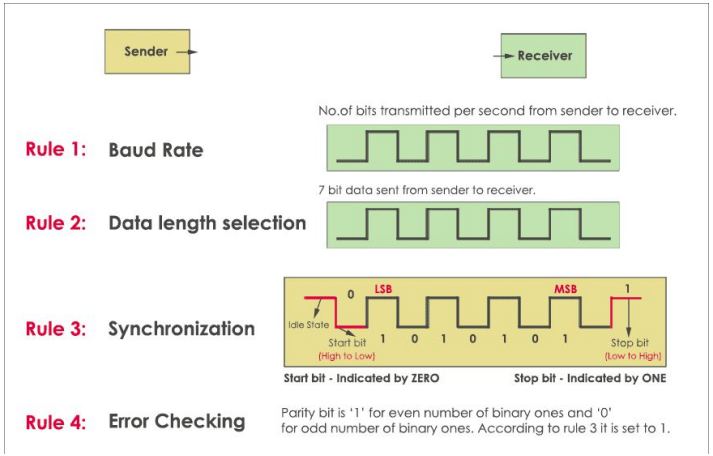
Es muy confiable y se ocupa en muchos lados, básicamente todo lo que se conecte vía **USB** es comunicación serial de hecho USB significa *Universal Serial Bus.* Por ejemplo tu mouse envía datos a tu pc vía este protocolo usando solamente **4 entradas** (de transmisión (Tx) de recibimiento (Rx) de tierra (GND) y de energía (VCC)).



Comunicación serial

Arduino UNO se conecta via un **USB tipo B** (el mismo que las impresoras) y por lo tanto también se puede comunicar por este protocolo, de hecho es así como se programa la placa.

Para establecer una conección es necesario indicar un ***baud rate*** que para el monitor serial de Arduino UNO es **9600**



Comunicación serial

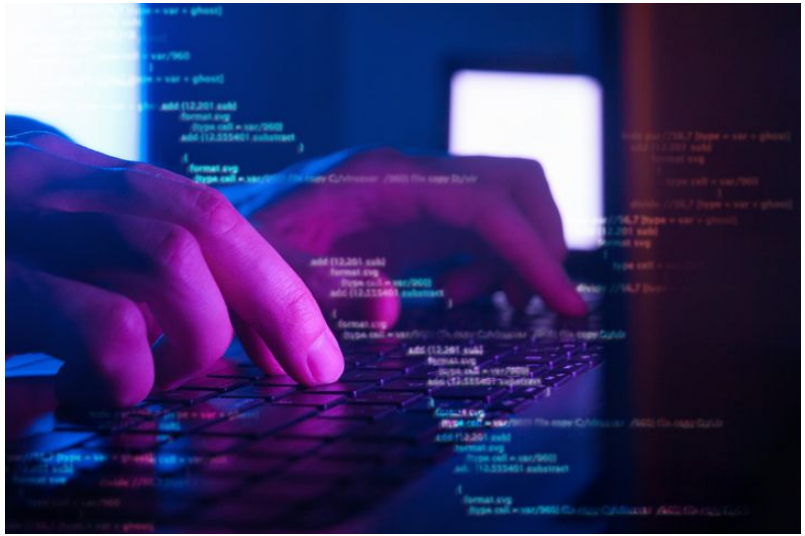
En resumen, toda esta explicación fue solo para entender esta línea que usaremos más adelante. (Y para que aprendieran algo nuevo sobre aparatos que usamos todos los días 😁)

```
Serial.begin(9600)
```

¿Qué es programar?

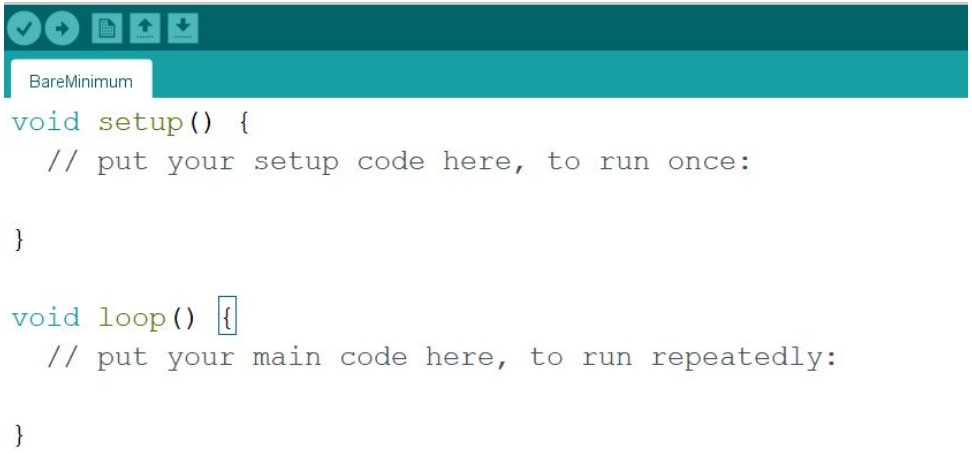
- Dar las **instrucciones** necesarias a una máquina para que realice su función de manera automática

- Un programa es una serie de instrucciones escritas de forma **secuencial** (una tras otra)



¿Cómo programar un Arduino?

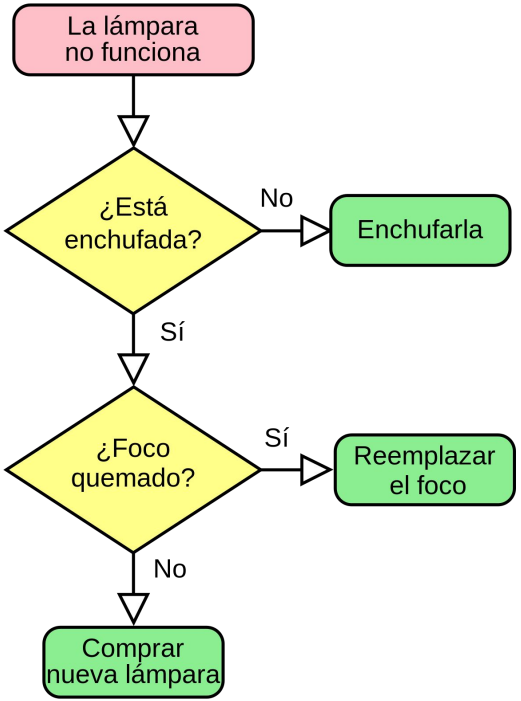
- Escribiendo código en **C++** (apoyándose de funciones pre-hechas) preferentemente a través de su **IDE**



¿Cómo describo lo que tiene que hacer sin programar todavía?

Usando **algoritmos**

Un algoritmo es un conjunto ordenado y finito de **instrucciones** que conducen a la solución de un problema



¿Qué tipo de instrucciones?

- De asignación
- De comparación
- De control (Condicionales y bucles)
- etc…

¿Cómo sé que estoy asignando y comparando?

Con **variables**

Una variable es un **dato** que puede cambiar con el tiempo. Se puede almacenar en memoria con un nombre para ser referenciado más adelante

```
/*Variable que guarda días sin accidentes*/

dias_sin_accidentes = 11
```

¿Qué tipos de datos?

Existen varios

Los más comunes que vamos a utilizar son:

- Números enteros (Integer) (**int**)
- Números decimales (Floating point) (**float**)
- Valores booleanos (Boolean) (**bool**)

```
/*Variable que guarda días sin accidentes*/

int dias_sin_accidentes = 11;
```

```
/*Variable decimal que aún no sé cuanto vale*/

float numero_misterioso;
```

Más adelante se entrega una lista más completa de tipos de datos que se usan en C++

¿Cómo se ven estas instrucciones?

Instrucciones en secuencia

```
Instrucción 1;
Instrucción 2;
Instrucción 3;
Instrucción 4
     Parte de la Instrucción 4;
Instrucción 5;
```

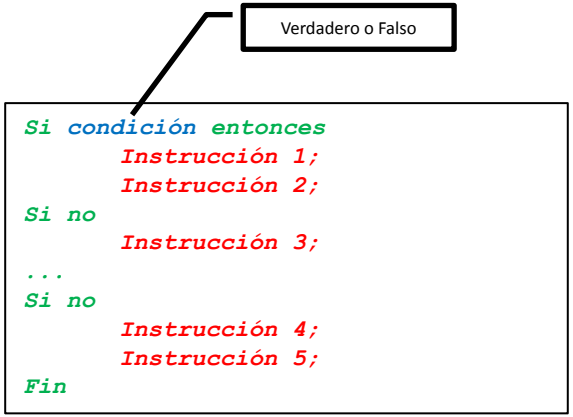
Asignaciones

```
int Variable1 = 100;
int Variable2 = 50;
bool Variable3 = false;
```

Comparaciones

```
Variable1 > Variable2;
Variable1 == Variable2;
Variable1 != Variable2;
```

Estructuras de control



Condicionales

```
Mientras condición hacer
      Instrucción 1;
      Instrucción 2;
Fin Mientras
```

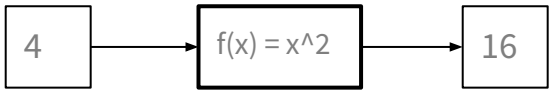
Loops

¿Qué más nos falta saber?

Funciones

Una función tanto en código como en matemáticas son una estructura que reciben uno o más argumentos y hacen algo con ellos. En código se definen como un conjunto de instrucciones bajo un nombre, estas instrucciones pueden ser ejecutadas en cualquier momento del programa llamando a su nombre seguido de paréntesis. Opcionalmente (si fue programada así), en estos paréntesis se le puede entregar uno o más argumentos a la función.

ejemplo




```
int cuadrado(int x){
  return x^2;
}

cuadrado(4);
```

Funciones pre-hechas

Arduino trae funciones pre-hechas que nos serán útiles para programar la placa, estas funciones se destacan en naranjo al escribirlas en el IDE.

Por ejemplo para asignar si un pin será usado para leer datos de un sensor, se escribe

pinMode(*número del pin*, INPUT)

Y para asignar un pin para entregar señales a un actuador

pinMode(*número del pin*, OUTPUT)

Otras función útil que trae Arduino y que usaremos es la función delay(*milisegundos a esperar*) Que hace que el programa “se congele” en ese punto durante los milisegundos que escribamos en su argumento.

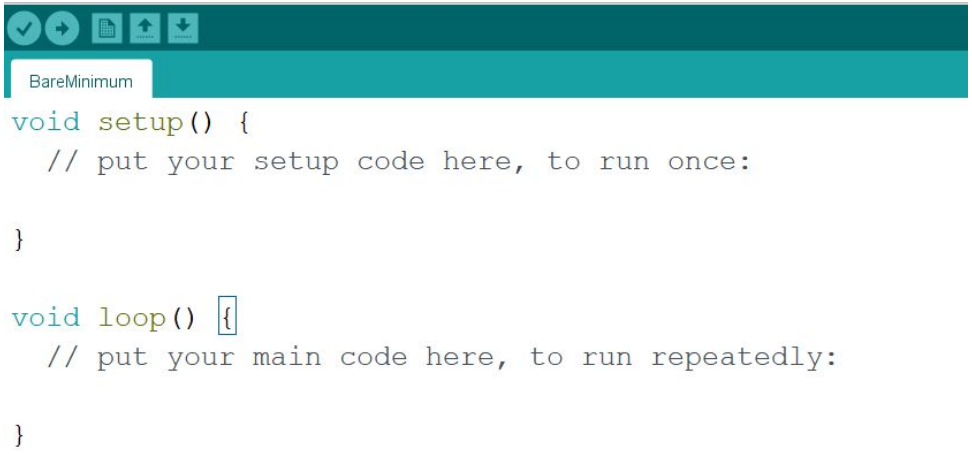
¿Cómo luce un programa real?

```
  bool terminado = false;
  long tiempos[] = {millis() + espera_por_grado[0], millis() + espera_por_grado[1]};
  while (terminado == false) {
    for (int k = 0; k < 2; k++) {
      if (millis() > tiempos[k]) {
        if (cords_servo[k] > ejes[k]) {
          cords_servo[k]--;
        }
        else if (cords_servo[k] < ejes[k]) {
          cords_servo[k]++;
        }
        actualizarServos();
        tiempos[k] = millis() + espera_por_grado[k];
      }

      if (cords_servo[0] == x && cords_servo[1] == y) {
        terminado = true;
      }
      delay(10);
    }
  }
}
else {
  int dif = abs(cords_servo[2] - y);
  int espera = tiempo / dif;
```

¿Cómo programar un Arduino?

- Escribiendo código en **C++** (apoyándose de funciones pre-hechas) preferentemente a través de su **IDE**



SINTAXIS BÁSICA	
Delimitadores:	**;, {}**
Comentarios:	**//, /* */**
Cabeceras:	#define, #include
Operadores aritméticos:	**+, -, *, /, %**
Asignación:	**=**
Operadores de comparación:	**==, !=, <, >, <=, >=**
Operadores Booleanos:	**&&, ||, !**
Operadores de acceso a punteros:	*, &
Operadores de bits:	&, |, ^, ~, <<, >>
Incremento y decremento de variables:	++, --
Asignación y operación:	+=, -=, *=, /=, &=, |=

ESTRUCTURAS DE CONTROL	
Condicionales:	if, if...else, switch case
Bucles:	for, while, do. while
Bifurcaciones y saltos:	break, continue, return, goto

CONSTANTES	
HIGH/LOW:	Representan los niveles alto y bajo de las señales de entrada y salida. Los niveles altos son aquellos de 3 voltios o más.
INPUT/OUTPUT:	entrada o salida.
false	Señal que representa al cero lógico.
true	Cualquier número entero diferente de cero es "verdadero“.

TIPOS DE DATOS	
Booleano: true, false (8 bit)	**boolean**
Caracteres (1byte)	**char**
Entero de 8 bit sin signo (0 a 255)	byte
Entero de 16 bit con signo (-32,768 a 32,767):	**int**
Entero de 16 bit sin signo (0 a 65,535):	word
Entero de 32 bit con signo (-2,147,483,648 a 2,147,483,647):	long
Decimal de 32 bit con signo (-3.4028235E+38 a 3.4028235E+38):	float
Arreglo de caracteres (Se define siempre entre comillas “ “)	string
Arreglo (datos secuenciales)	array

Declarar pin:	**pinMode(pin, modo)**

I/O DIGITAL	
Escritura digital (0V o 5V):	**digitalWrite(pin, valor)**
Lectura digital (1 o 0):	**int digitalRead(pin)**

I/O ANALÓGICO	
Setear el valor más alto leído	analogReference(tipo)
Escritura análogica (PWM de 0 a 255):	**analogWrite(pin, valor)**
Lectura analógica (resolución de 1024):	**int analogRead(pin)**

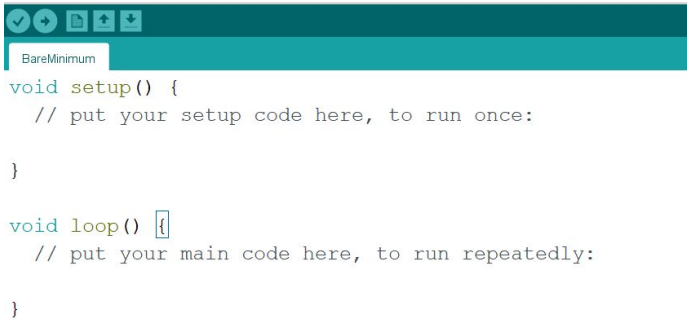
Cosas a considerar de C++

- (Casi) todas las líneas terminan con ;
- No olvidar poner el tipo de dato cuando sea necesario
- Leer bien los errores que salen, a veces son muy descriptivos

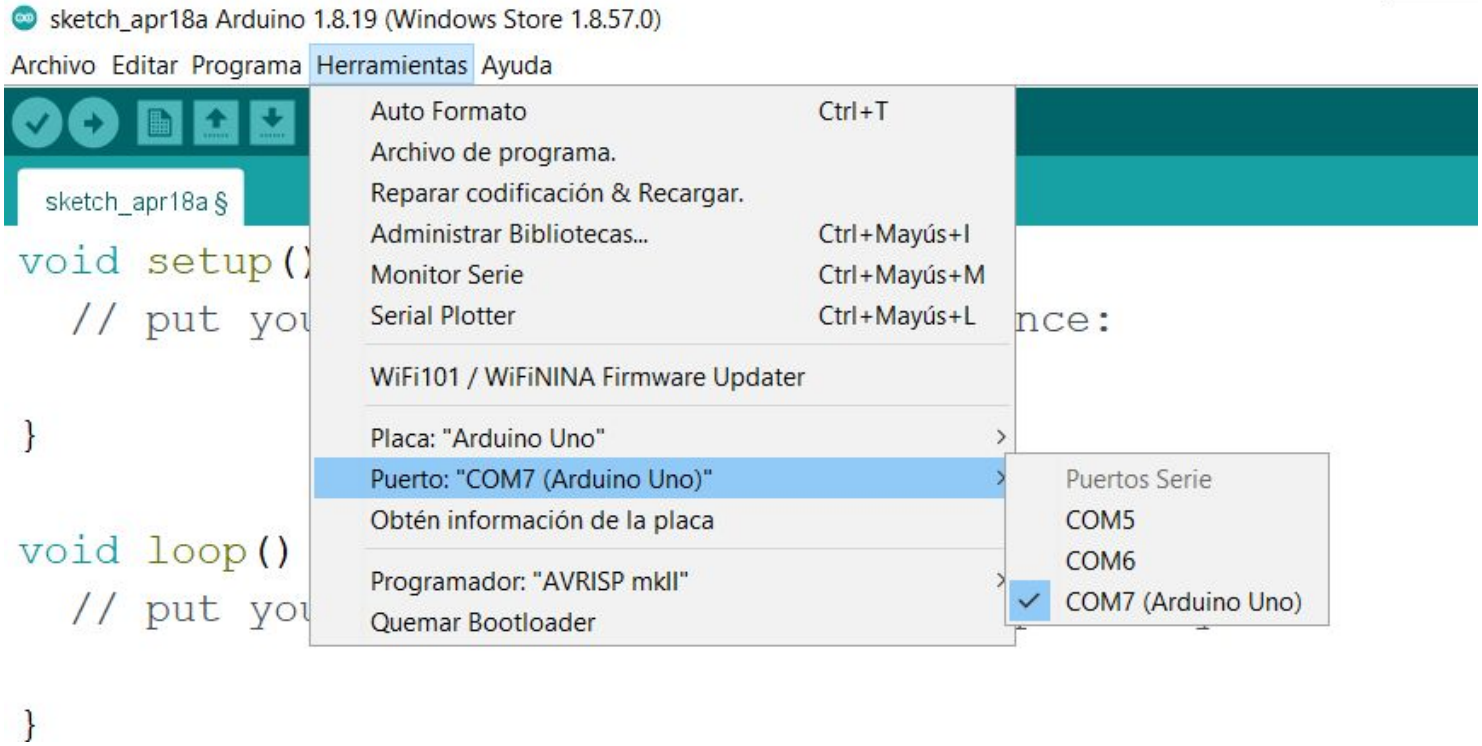
Descargar e instalar el IDE

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

- O directamente desde **Microsoft Store** para windows



Conectar Arduino y elegir el puerto correcto



¿Cómo programar un Arduino?

- Todo lo que está dentro de la función **setup()** corre **una vez**

  Se utiliza para asignar pines, entre otros

- Todo lo que está dentro de la función **loop()** corre **infinitas veces** (de forma **muy** rápida)

  Se utiliza para lo que la placa va a hacer, si nada lo detiene este código correrá por siempre mientras tenga energía.

- Se pueden declarar variables antes de la función **setup()**, estas serán **variables globales**

¿Cómo programar un Arduino?

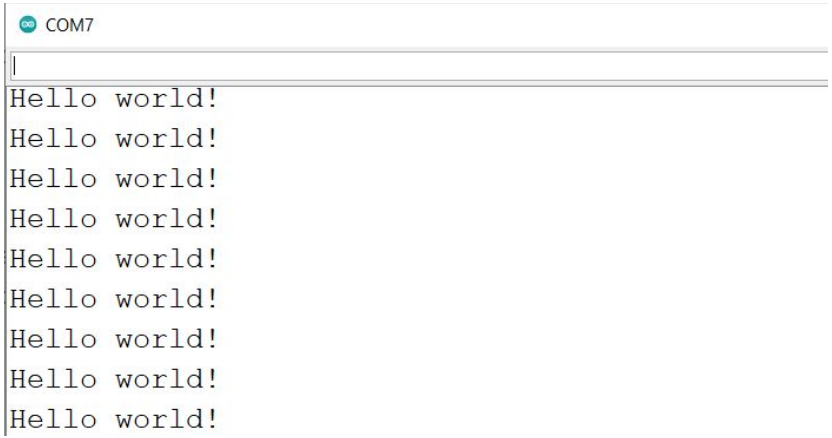
- El botón verificar **compila** el código y revisa que funcione pero no lo sube a la placa

- El botón subir, **verifica** y **sube** el código a la placa

Primer ejercicio: Hello World!


```
void setup() {
 Serial.begin(9600);
}

void loop() {
  Serial.println("Hello world!");
  delay(500);
}
```



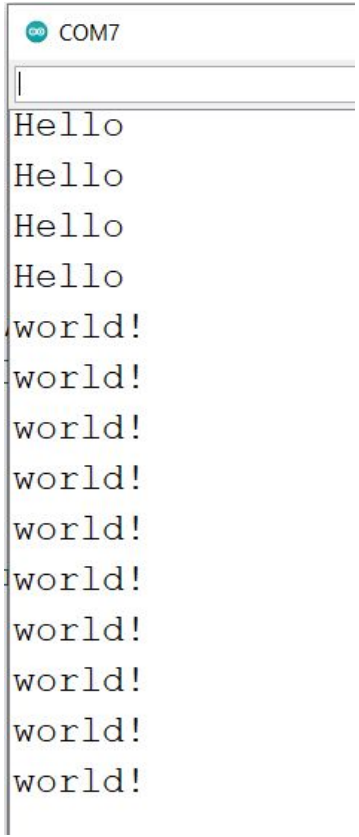
Hello World, desafío:

Imprimir 5 veces “Hello” y luego 10 veces “world!” infinitamente

hello_world_desafio

```
int y = 0;
void setup() {
 Serial.begin(9600);
 }

void loop() {
  for(int i = 0; i < 5; i++){
    Serial.println("Hello");
  }
  delay(500);
  while(y < 10){
    Serial.println("world!");
    y += 1;
  }
  y = 0;
  delay(500);
}
```



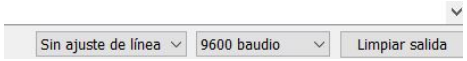
Segundo ejercicio, lectura COM serial:

serial_read

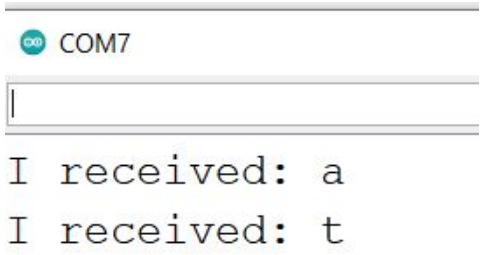
```
/*
 code adapted from https://www.arduino.cc/en/serial/read
 and https://hetpro-store.com/TUTORIALES/arduino-serial-read/
*/
char incomingChar;   // for incoming serial data
void setup() {
  Serial.begin(9600);
}

void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0) {
          // read the incoming char:
          incomingChar = Serial.read();

          // say what you got:
          Serial.print("I received: ");
          Serial.println(incomingChar);
  }
}
```



- Poner monitor serial en modo sin ajuste de línea
- ¿Qué pasa si enviamos más de un caracter?



Lectura COM serial, desafío:

Responder con “Hola que tal” al recibir “a” y responder con “Adiós” al recibir “b”, responder “No entiendo” si recibe otra cosa

dato: no olvidar poner el monitor serial en modo “sin ajuste de línea”

```
char incomingChar;
void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
          incomingChar = Serial.read();
          if(incomingChar == 'a'){
            Serial.println("Hola que tal");
          }
          else if(incomingChar == 'b'){
            Serial.println("Adiós");
          }
          else{
            Serial.println("No entiendo");
          }
  }
}
```

Tercer ejercicio: Blink LED

Blink

```
// 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
}
```

ALternativamente: cambiar LED_BUILTIN por 13

Blink LED + serial, desafío:

Que la luz se prenda cuando reciba “prendete” desde el monitor serial y se apague al recibir “apagate”

dato: usar Serial.readString()

https://www.arduino.cc/reference/en/language/functions/communication/serial/readstring/

Blink_desafio

```
String incomingString;
void setup() {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  if (Serial.available() > 0) {
          incomingString = Serial.readString();
          Serial.println(incomingString);
          if(incomingString == "prendete"){
            digitalWrite(LED_BUILTIN, HIGH);
          }
          else if(incomingString == "apagate"){
            digitalWrite(LED_BUILTIN, LOW);
          }
  }
}
```

Miércoles:

- Lectura análoga y digital de sensores

- Código más complejo para hacer *cosas interesantes* con actuadores y estas lecturas de sensores