TECHNISCHE UNIVERSITÄT

Arduino Tutorial

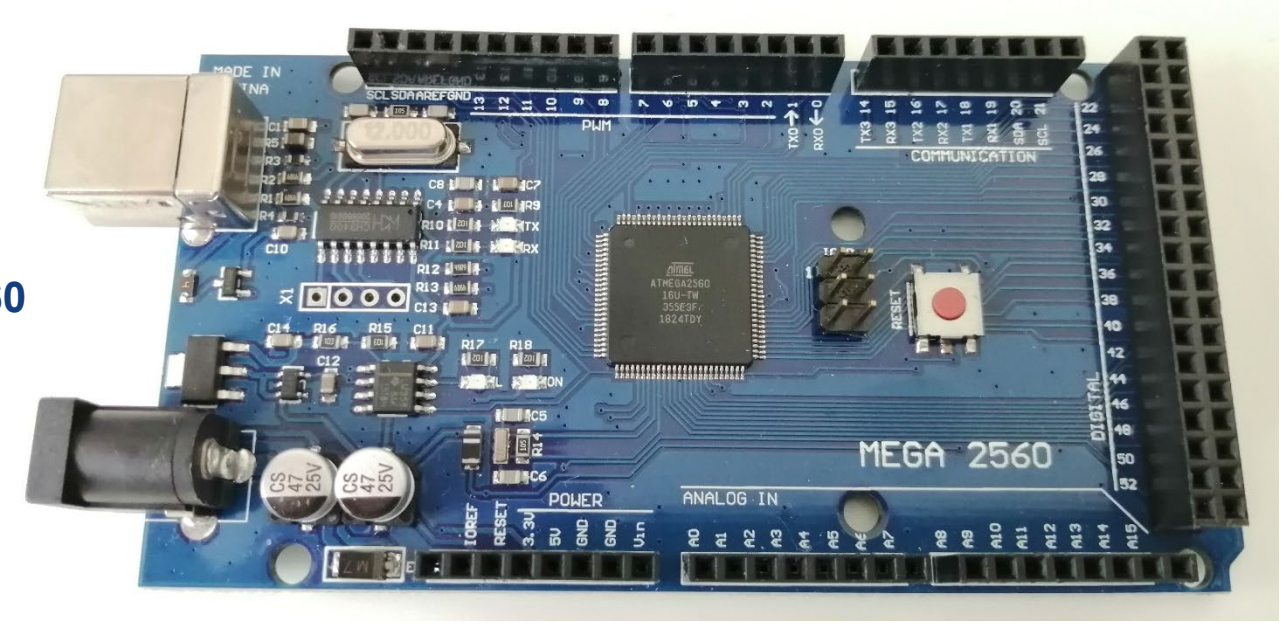
AUTOPISTA

Arduino MEGA 2560

TECHNISCHE UNIVERSITÄT

Características

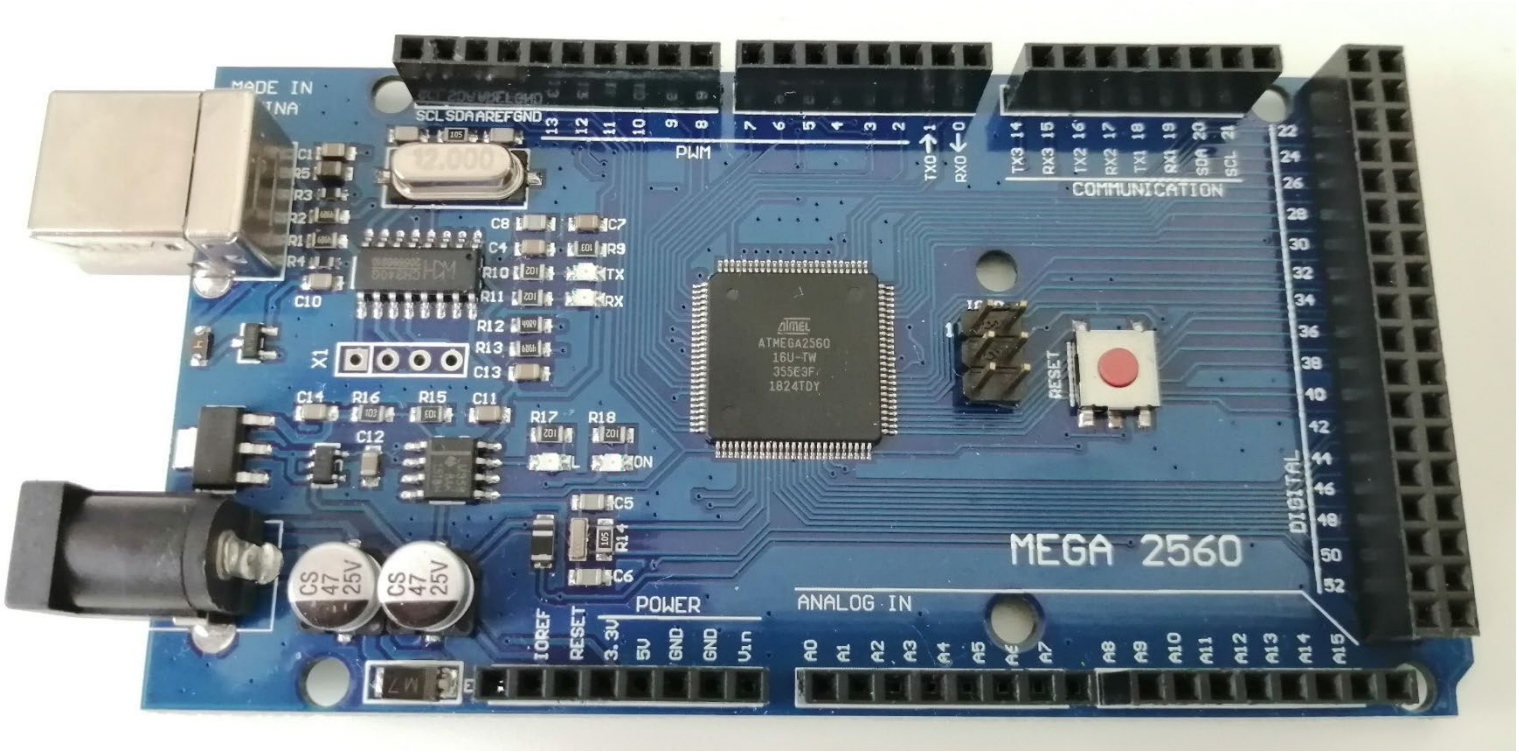
- Procesor Atmega2560
- 8-bit CPU
- 16 MHz velocidad
- 8 kB SRAM
- 256 kB almacén
- 54 pins digitales
- 16 pins analógicos
- Medidas: 101,52 mm x 53,3 mm



Arduino MEGA 2560

TECHNISCHE UNIVERSITÄT

El controlador MEGA 2560 es un micro-controlador basado en ATmega2560.
El Mega 2560 es compatible con muchos sistemas de Arduino.



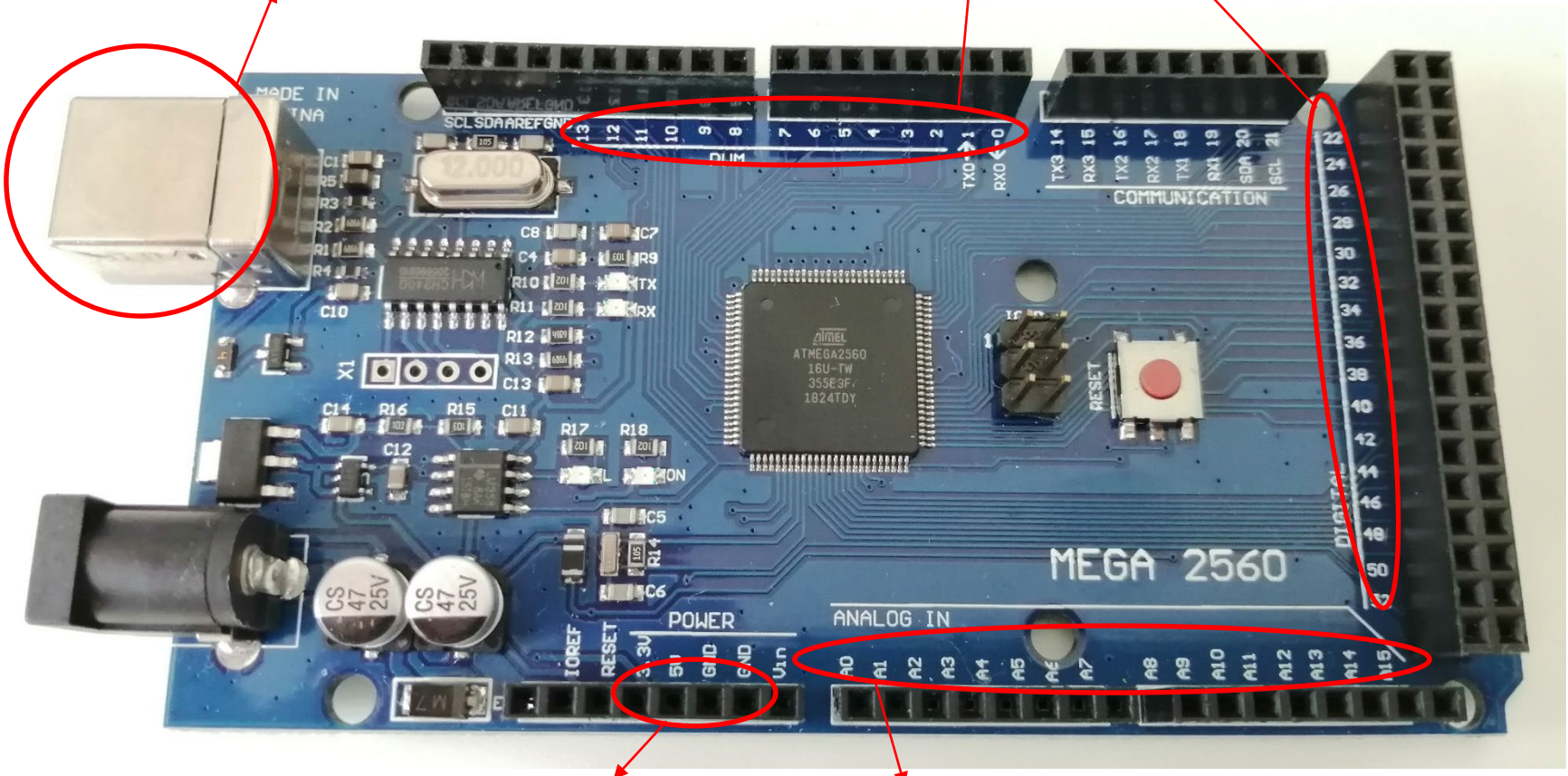
El uso de este micro-controlador es para proyectos grandes donde se requieren muchas entradas y salidas.

Arduino MEGA 2560

TECHNISCHE UNIVERSITÄT

Entrada conector USB:
Voltaje (5V) y comunicación con la computadora

Señales digitales

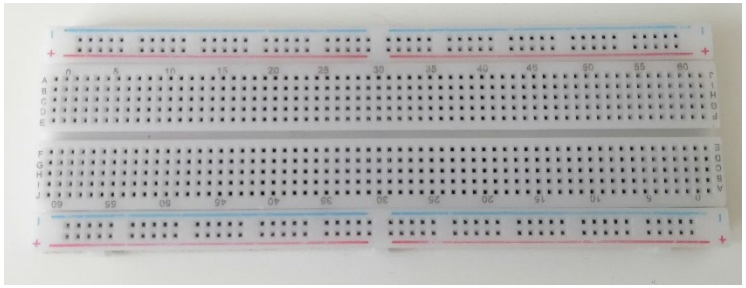


Voltaje y tierra (GND)

Señales análogas

Breadboard – tabla de pan

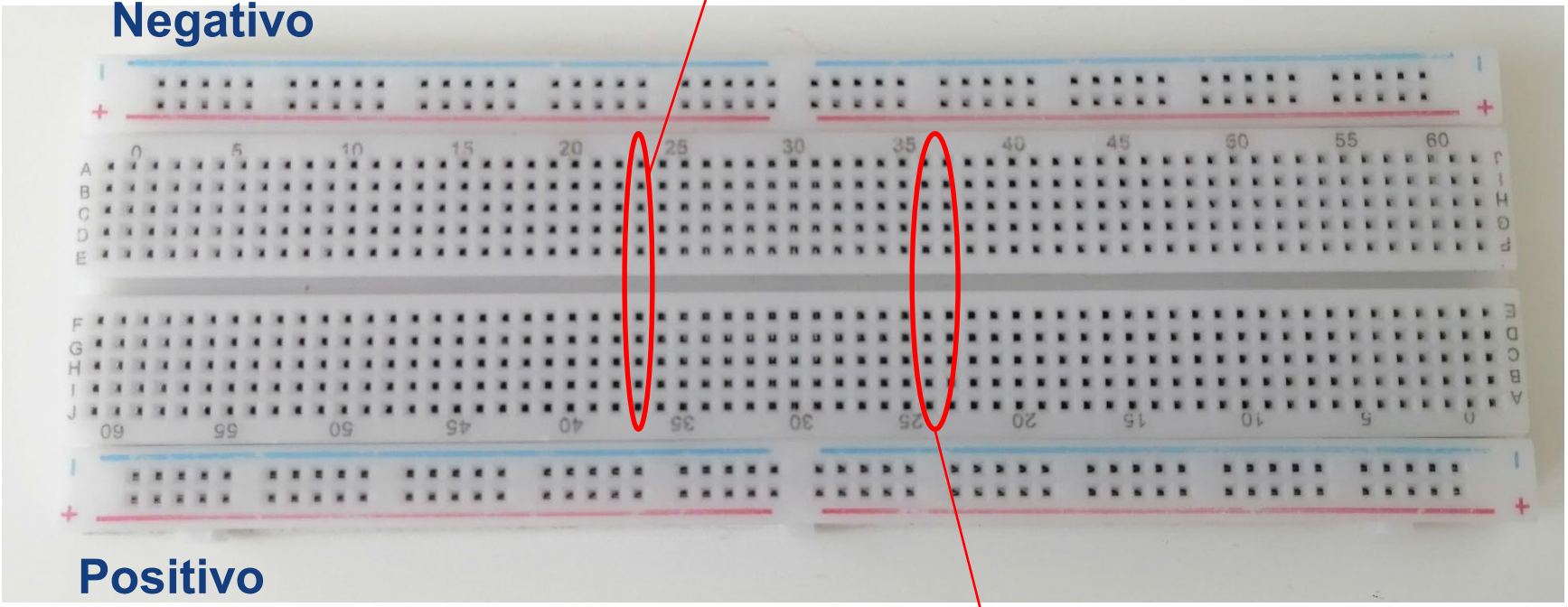
- Se le conoce Breadboard porque anteriormente los primeros proyectos electrónicos la gente usaba tablas de pan reales. Es decir, los „tuppers“ de sus mamás en aquellos tiempos.
- Hasta la fecha se sigue usando la palabra “Breadboard” aunque sean ahora de plástico.
- Aplicación: Conectar tu equipo o sistema de manera rápida y estar evitando soldadura. Normalmente se usa siempre un breadboard en la etapa de prueba y desarrollo.



Breadboard – tabla de pan

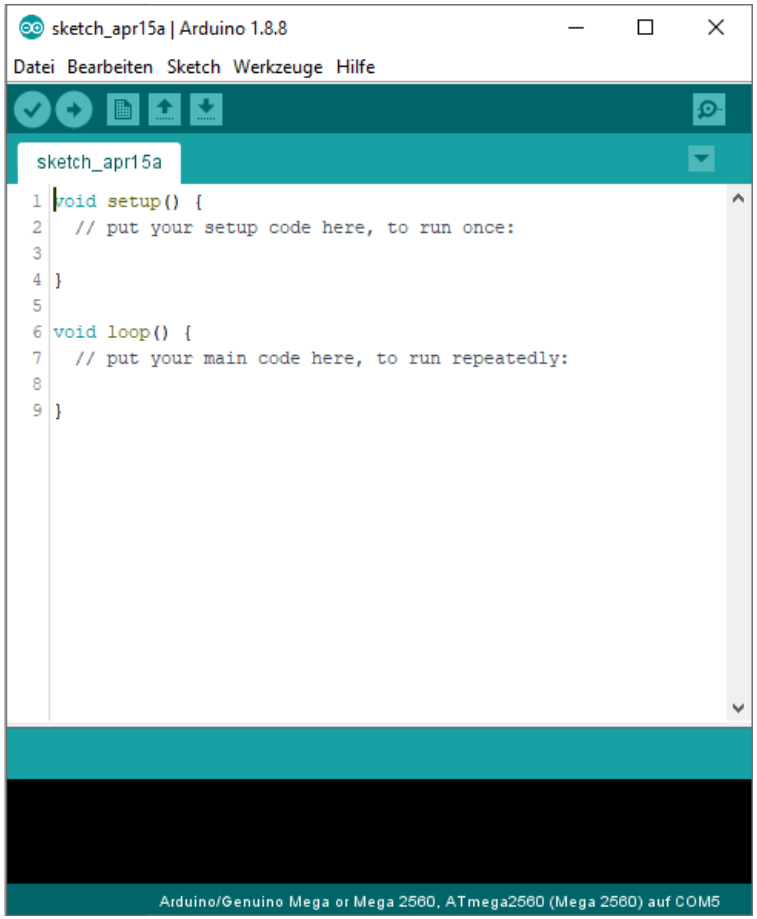
TECHNISCHE UNIVERSITÄT

Misma línea (23), es decir todo está en contacto (serie)



36 y 37 no están conectadas

Estructura básica



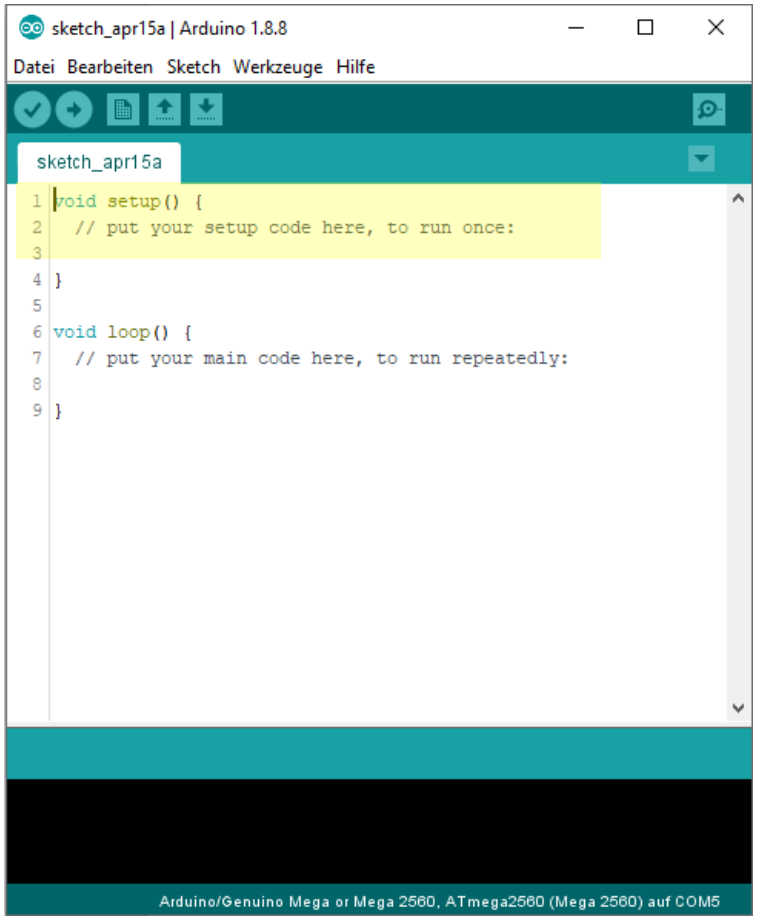
La estructura básica de la programación del Arduino se divide en dos partes:

1. Setup
2. Loop

El código que uno escribe se le conoce como “sketches” y es escrito en C++.

Cada “sketch” necesita dos funciones “void”: setup() y loop()

Estructura básica – setup()

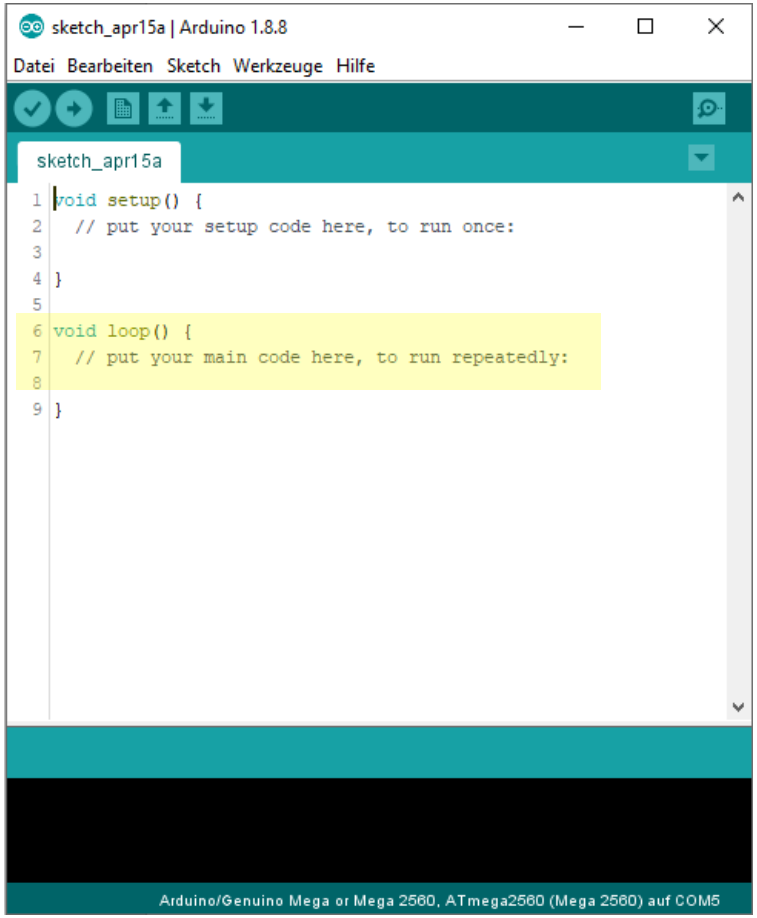


El setup() corre una sola vez.

Éste se acciona cuando el Arduino es prendido.

El setup() es la parte donde se indican los pasos de la inicialización de tu programa.

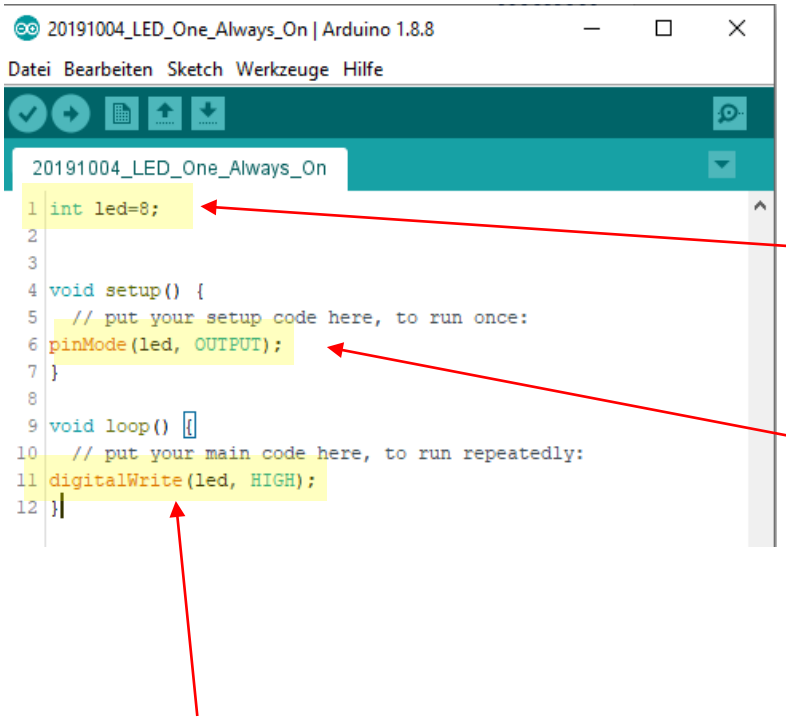
Estructura básica – loop()



El loop() corre una y otra vez, de manera contínua.

El loop() incluye el código que deseas correr una y otra vez.

Ejemplo – Prendiendo un LED



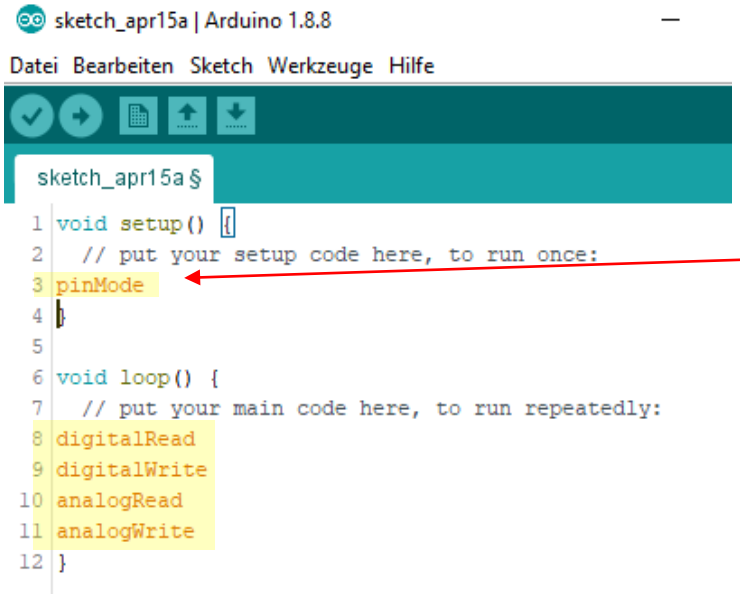
Este pequeño programa mantiene prendido un LED.

Antes del setup() definimos nuestros pins. Decimos que el pin 8 se llama “led”.

En el setup() definimos que es “led”. Le decimos que es un “pinMode” donde se tiene o una entrada o una salida. El “pinMode” lo veremos más adelante. Y le decimos que es una salida con “OUTPUT”.

En el loop() decimos que queremos una salida “HIGH” o “LOW”. Esta indicación lo hacemos por medio de “digitalWrite”. El “digitalWrite” al igual que “pinMode” son comandos para entradas y salidas. Más adelante se definirán estos comandos.

Comandos para Entrada y Salida



Arduino cuenta con algunos comandos para definir entradas y salidas.

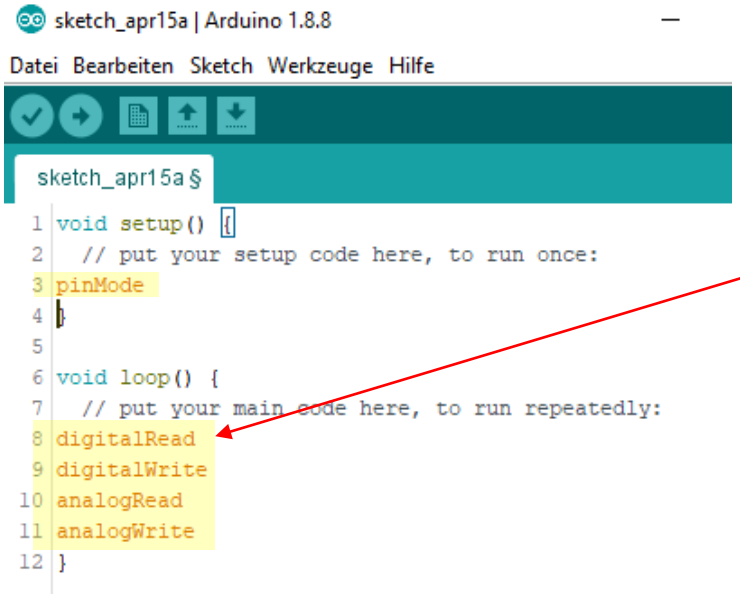
El pinMode es un comando usado para definir un pin como entrada o salida dentro del setup()

En el Arduino, los pins digitales siempre son entendidos como entrada por configuración.

Si usas un pin de entrada, no es necesario volver a definirlo. Sin embargo, lo puedes hacer. Aquí un ejemplo:

```
1 void setup() {
2   // put your setup code here, to run once:
3 pinMode(1, INPUT); // El pin1 es entrada
4 pinMode(2, OUTPUT); // El pin2 es salida
5 }
```

Comandos para Entrada y Salida



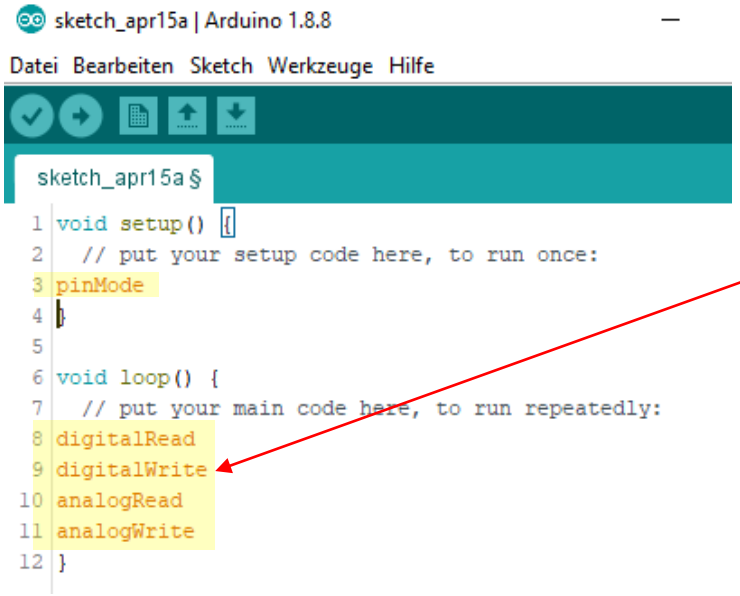
Arduino cuenta con algunos comandos para definir entradas y salidas.

El digitalRead es una función usada para leer un valor digital de un pin. El resultado sólo puede ser HIGH o LOW.

Ejemplo:

```
1  void setup() {
2    // put your setup code here, to run once:
3  pinMode(1, INPUT); // El pin1 es entrada
4  pinMode(2, OUTPUT); // El pin2 es salida
5  }
6
7  void loop() {
8    // put your main code here, to run repeatedly:
9  int valor = digitalRead(1); // La variable valor
10                             // es igual a la entrada
11                             // del pin 1
12 }
```

Comandos para Entrada y Salida



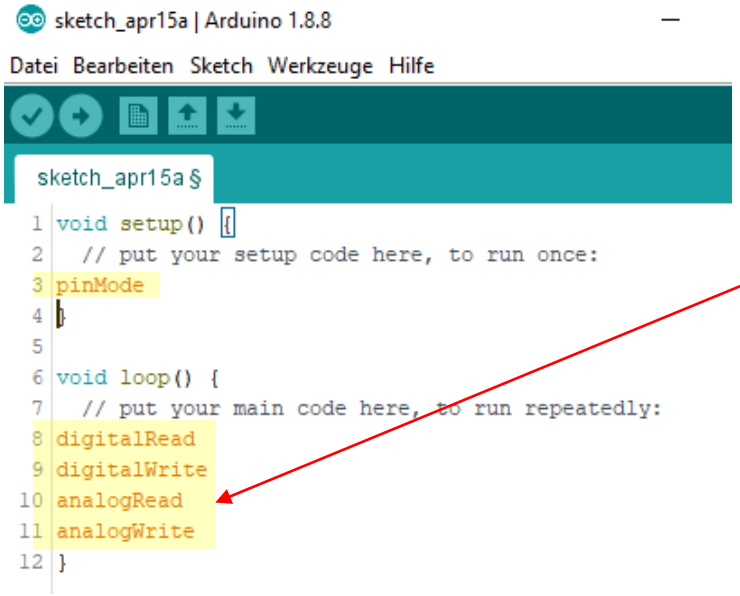
Arduino cuenta con algunos comandos para definir entradas y salidas.

El digitalWrite establece el pin como prendido o apagado (ON / OFF) escribiendo la variable HIGH o LOW en su salida.

Ejemplo:

```
1  void setup() {
2    // put your setup code here, to run once:
3  pinMode(1, INPUT); // El pin1 es entrada
4  pinMode(2, OUTPUT); // El pin2 es salida
5  }
6
7  void loop() {
8    // put your main code here, to run repeatedly:
9  digitalWrite(2, HIGH); // El valor del pin 2 se establece
10                        // como HIGH (encendido)
11 }
```

Comandos para Entrada y Salida



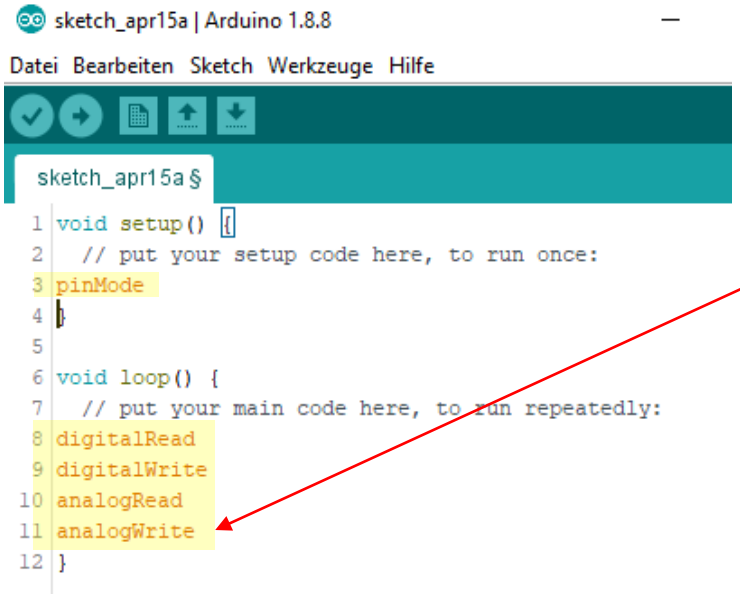
Arduino cuenta con algunos comandos para definir entradas y salidas.

El analogRead es una función que puede leer un valor de un pin análogo. El resultado oscila entre 0 y 1023. Esta función sólo funciona con los pins análogos del 0 al 5.

Ejemplo:

```
1  void setup() {
2    // put your setup code here, to run once:
3  pinMode(3, INPUT); // El pin3 es entrada
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8  analogRead(3, HIGH); // El valor del pin análogo 3 es leído
9                       // La respuesta será entre 0 y 1023
10 }
```

Comandos para Entrada y Salida



Arduino cuenta con algunos comandos para definir entradas y salidas.

El analogWrite es una función para dar una salida análoga. Revisar el manual de tu Arduino ya que sólo algunos pins análogos soportan el analogWrite. Por ejemplo en Arduino Uno son los pines 3,5,6,9,10 y 11. El valor que se puede dar es entre 0 y 255.

Ejemplo:

```
1  void setup() {
2    // put your setup code here, to run once:
3  pinMode(6, OUTPUT); // El pin6 es salida
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8  analogWrite(6, x); // El pin6 adquiere el valor de x que
9                     // uno haya decidido entre 0 y 255.
10 }
```