Introducción a Arduino

Lucas Martire - Santiago Rodríguez - Germán Scillone - Jorge Anderson - Sebastián Millán - Facundo Aparicio - Juan C. Scattuerchio

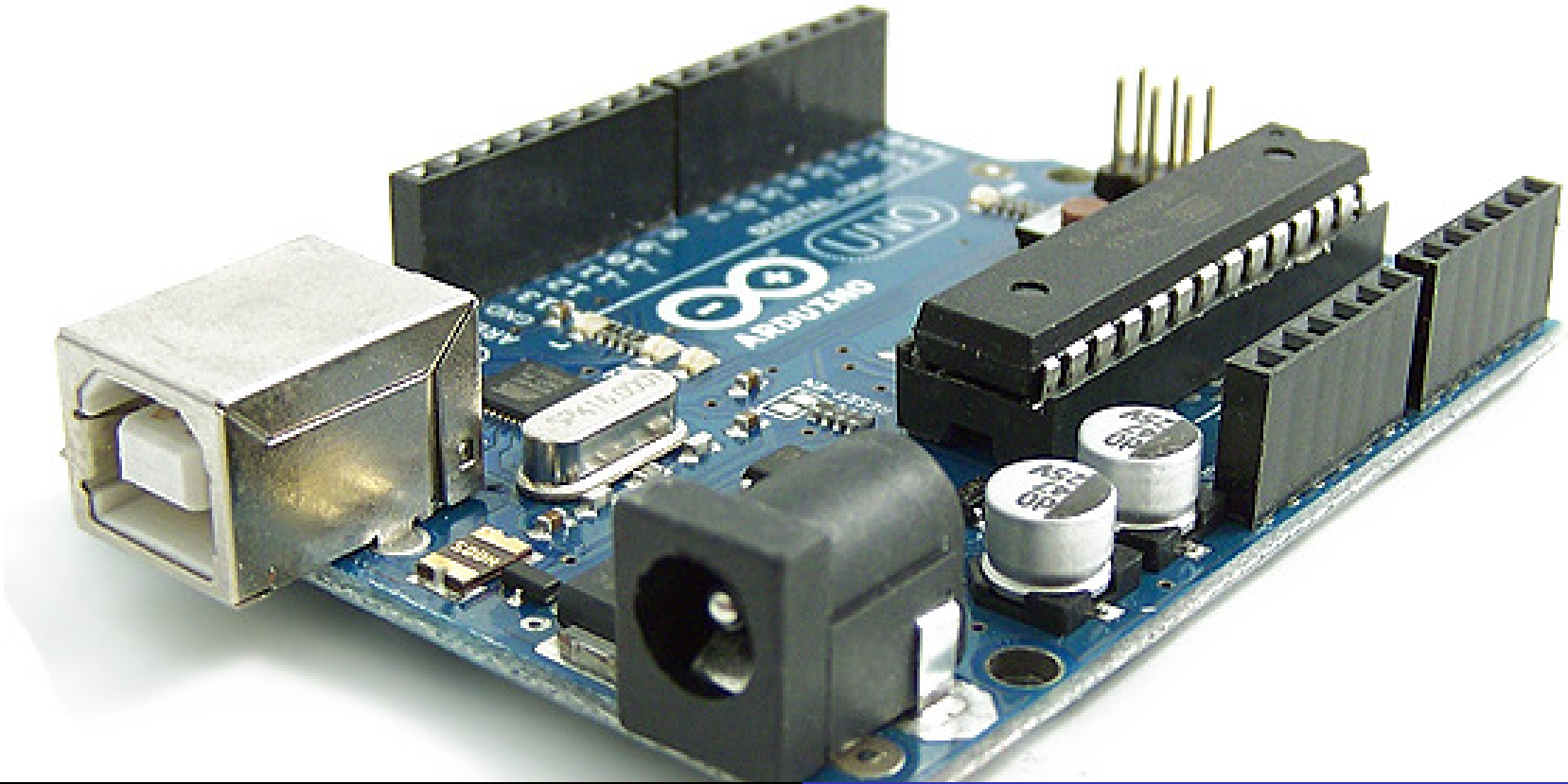
Depto. ELECTROTECNIA - FI - UNLP

Índice

Índice

¿Por qué Arduino?

El Arduino es una plataforma de desarrollo que nos brindará una rápida y fácil introducción al mundo de los microcontroladores, permitiendo tanto la creación como implementación de una gran variedad de proyectos sin un conocimiento demasiado extenso sobre la materia.

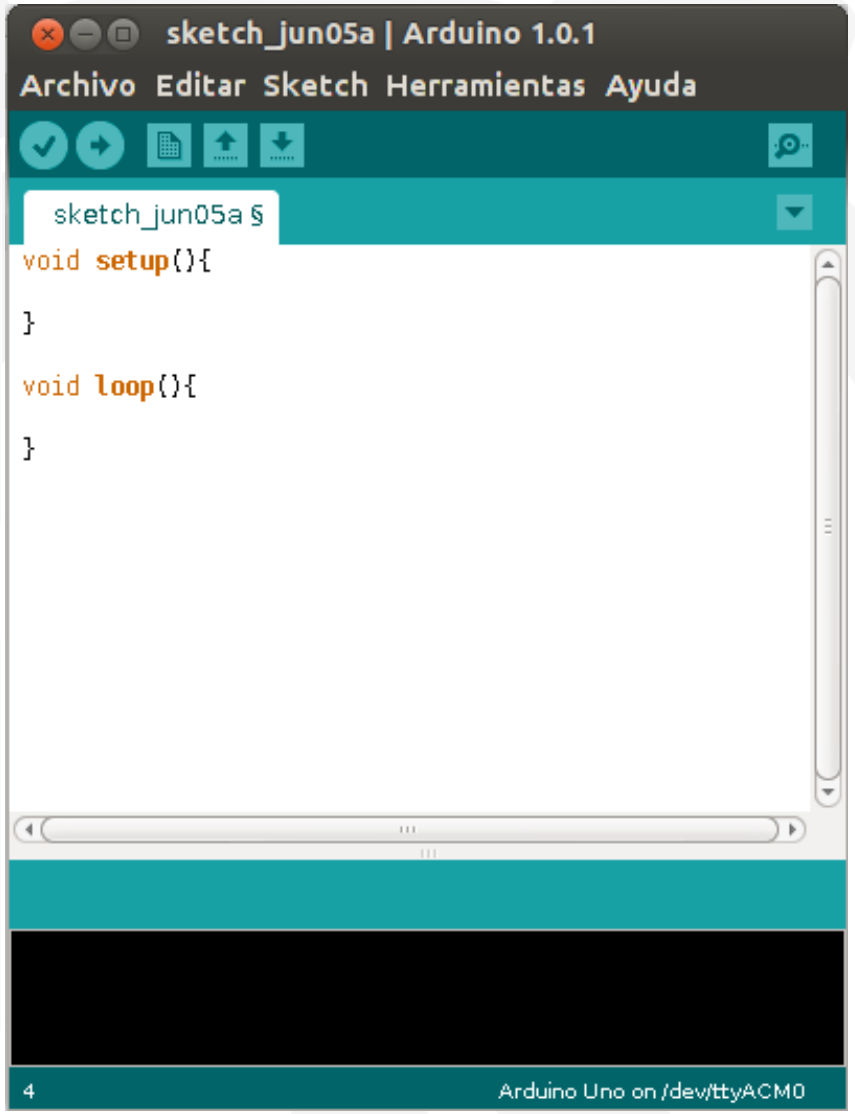


Entorno de Desarrollo Integrado (IDE)

La IDE es el lugar donde trabajaremos nuestra programación, para luego poder cargarla al Arduino. Se comunica a través del puerto UART del microcontrolador, creando una interfaz USB.

¿En que lenguaje programaremos?

El Arduino tiene la facilidad de adaptación para un sin fin de lenguajes de programación posibles, nosotros nos dedicaremos a la programación en el lenguaje *C*.



Un poco de programación

Poniendo manos a la obra

Para hacer uso del microcontrolador, habrá que especificar las instrucciones que deberá seguir para realizar las tareas deseadas. Para ello, recurrimos a la programación usando el lenguaje *C* tal como se anticipó.

Fundamentos de *C* para Arduino

La IDE de Arduino hace uso varias librerías (Las cuales son conjuntos de funciones predefinidas) para facilitar la tarea de programación. En tanto, este entorno consta de dos funciones fundamentales. Estas son las funciones *setup*() y *loop*() y las destacamos por ser indispensables para cualquier proyecto.

Algunas funciones básicas

Las siguientes funciones están relacionadas con el módulo de puertos Entrada/salida:

- pinMode(*pin*,*mode*)

Algunas funciones básicas

Las siguientes funciones están relacionadas con el módulo de puertos Entrada/salida:

- pinMode(*pin*,*mode*)
- digitalRead(*pin*)

Algunas funciones básicas

Las siguientes funciones están relacionadas con el módulo de puertos Entrada/salida:

- pinMode(*pin*,*mode*)
- digitalRead(*pin*)
- digitalWrite(*pin*,*value*)

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

- analogWrite(*pin*,*value*)

Las siguientes funciones están relacionadas con el módulo de comunicación:

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

- analogWrite(*pin*,*value*)

Las siguientes funciones están relacionadas con el módulo de comunicación:

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

- analogWrite(*pin*,*value*)

Las siguientes funciones están relacionadas con el módulo de comunicación:

- Serial.begin(*speed*)

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

- analogWrite(*pin*,*value*)

Las siguientes funciones están relacionadas con el módulo de comunicación:

- Serial.begin(*speed*)
- Serial.write(*val*)

Algunas funciones básicas

La siguiente función está relacionada con el módulo del conversor Analógico/Digital:

- analogRead(*pin*)

La siguiente función está relacionada con el módulo del temporizador:

- analogWrite(*pin*,*value*)

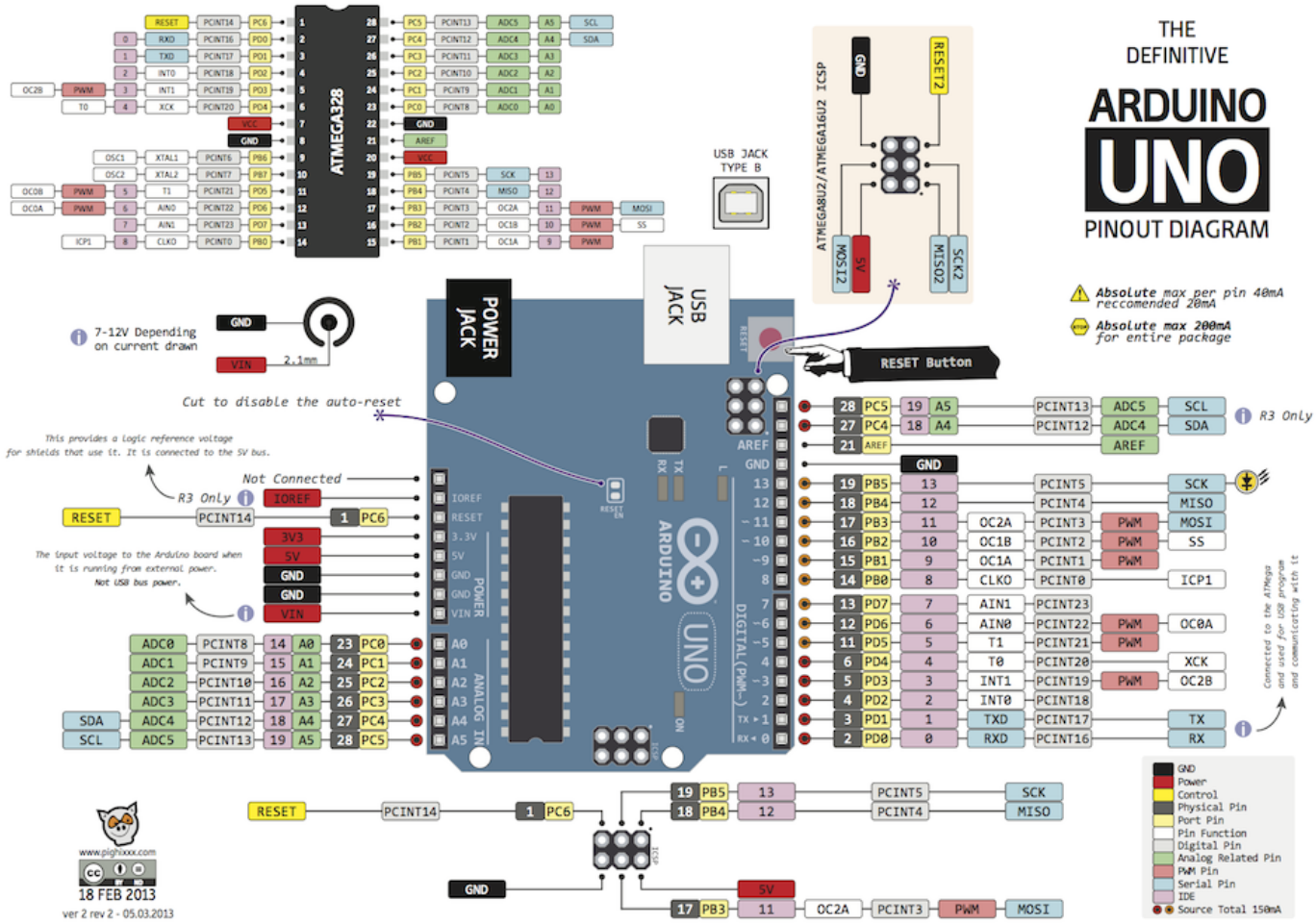
Las siguientes funciones están relacionadas con el módulo de comunicación:

- Serial.begin(*speed*)
- Serial.write(*val*)
- Serial.read()

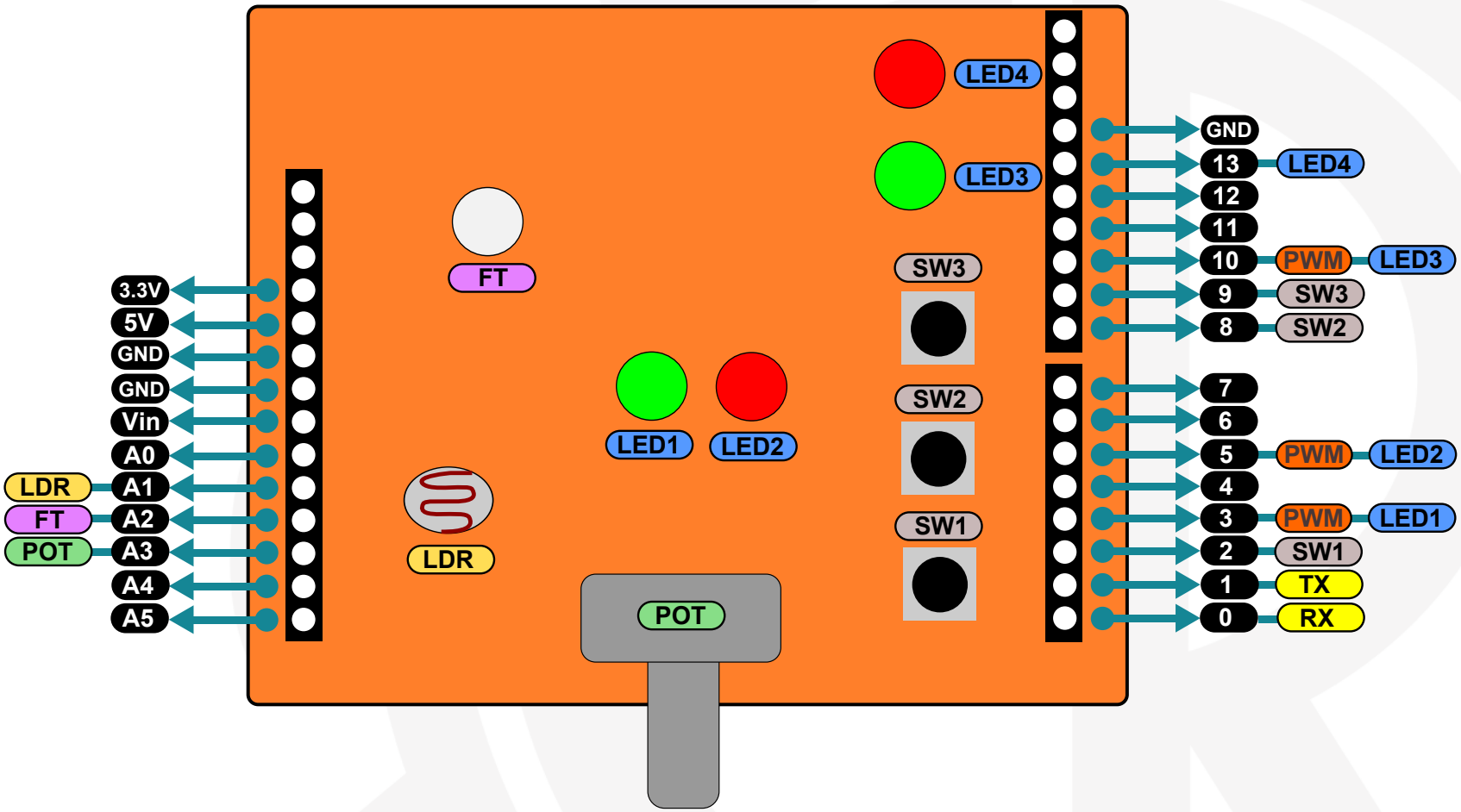
Esquema de pines del Arduino

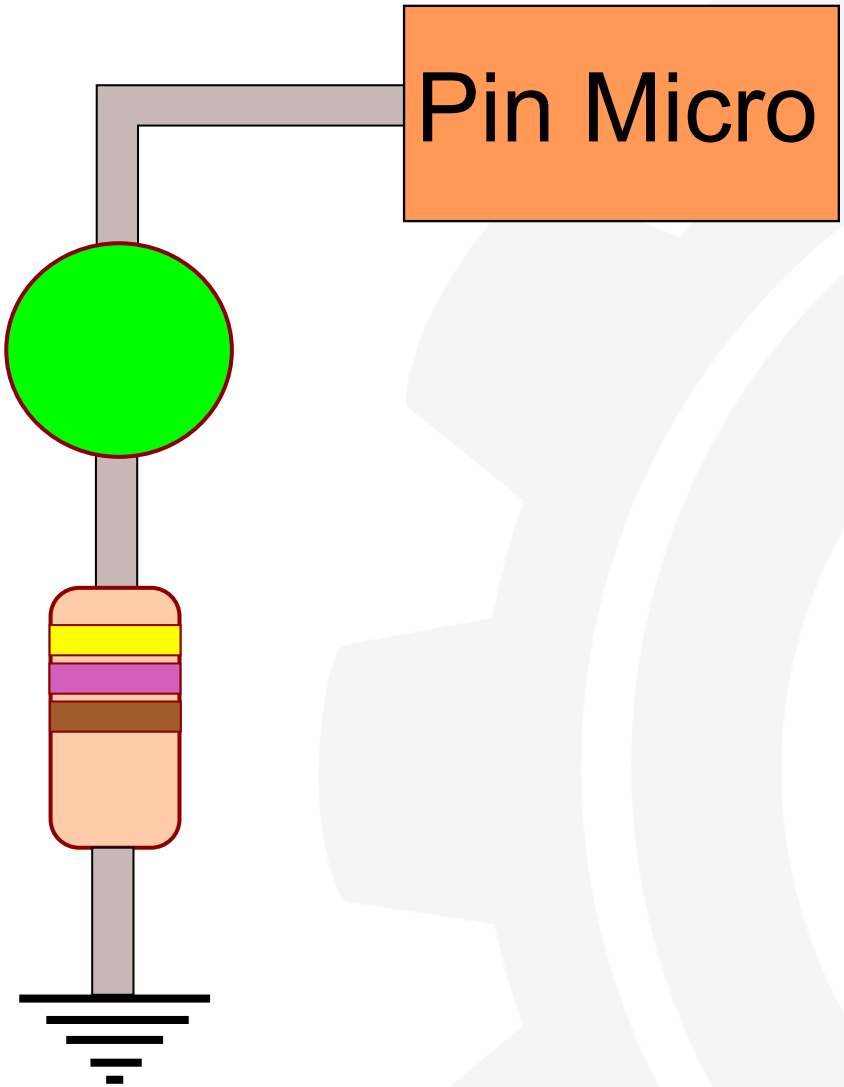
Poniendo manos a la obra

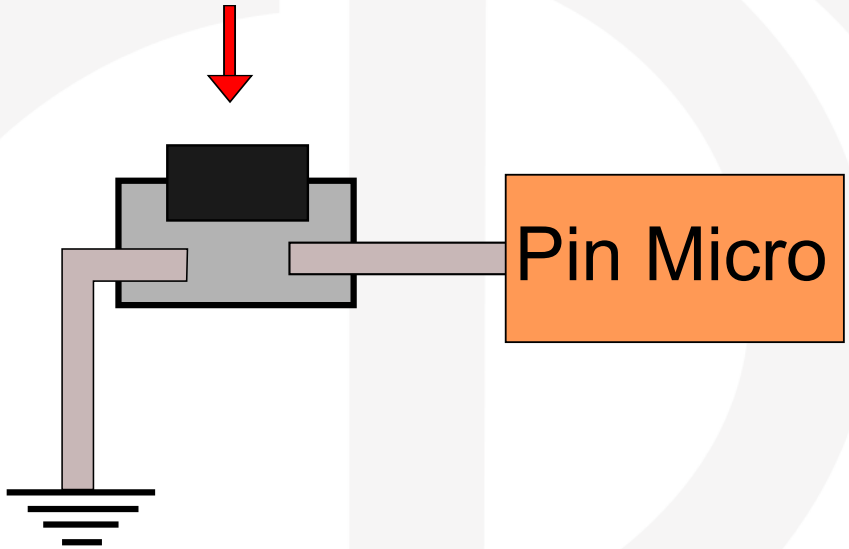
El Arduino posee su propia enumeración para los puertos del microcontrolador, la cual tendremos que tomar en cuenta a la hora de crear nuestras aplicaciones.

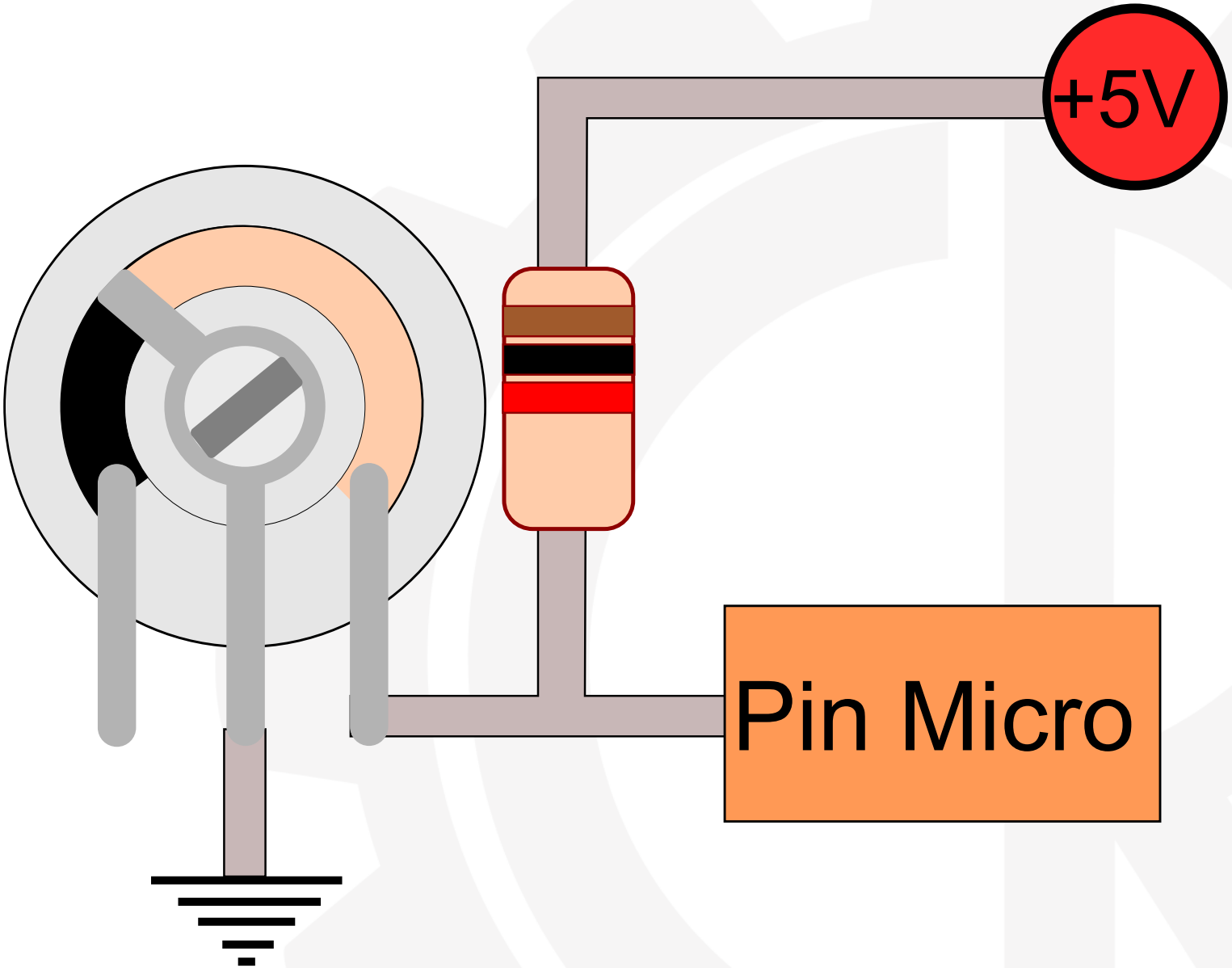


Placa de prueba




Pin Micro


Pin Micro


+5V
Pin Micro

¡A trabajar!

Para concluir, se mostrará un pequeño ejemplo, clásico para la introducción en el manejo de microcontroladores. El código del proyecto siguiente servirá para mostrar un led resplandeciente con una frecuencia fija y en dicho código se utilizarán algunas de las funciones presentadas anteriormente.

Utilizando los puertos digitales

Para nuestro primer programa en Arduino comenzaremos por probar los puertos digitales. Prueben encender y apagar uno de los leds con una espera de 1 segundo, recuerden que para poder utilizar los puertos digitales deben ser configurados previamente.

Pistas

Utilice las funciones **pinMode()** para configurar los puertos como salida, **digitalWrite()** para encender o apagar los leds y la función **delay()** pare generar los retardos. ¿Hay una forma más sencillo de hacerlo?

¡Hola mundo!

```
#include <DEF_PP.h>

/* La rutina de inicialización se ejecuta una vez que se
presione "reset" */
void setup() {
  // Se inicializa el pin digital como salida.
  pinMode(LED1, OUTPUT);
}

/* La rutina de "loop" se ejecuta indefinidamente */

void loop() {
  digitalWrite(LED1, HIGH);   // encender el LED
  delay(1000);                // esperar un segundo
  digitalWrite(LED2, LOW);    // apagar el LED
  delay(1000);                // esperar un segundo
}
```

Utilizando los puertos digitales

Continuando con los puertos digitales, intente utilizar alguno de los pulsadores para encender uno de los leds cuando sea presionado.

Pistas

Recuerde que para poder leer los puertos digitales utilizaremos la función **digitalRead()**. La función para configurar los puertos digitales era **pinMode()**. ¿Con que opción tendremos que configurar el puerto de entrada?

Para finalizar

Ahora intentaremos utilizar los puertos de lectura analógicos, trate de leer el valor de tensión en el potenciómetro asignandole intervalos de valores para el encendido de diferentes leds.

Pistas

Para leer los puertos analógicos podremos utilizar la función **analogRead()**. Recuerde que la función devuelve valores entre 0 y 1023 unicamente, asi que puede dividir los intervalos de 0 a 511 y 512 a 1023.

¡ Gracias por la atención !