Arduino

Índice

Introducción a Arduino

Arduino como tal, es una plataforma de desarrollo de hardware de código abierto que se basa en una placa con un microcontrolador y un entorno de desarrollo.
El microcontrolador es el cerebro de la placa y es el que se encarga de ejecutar las instrucciones que le damos.
El entorno de desarrollo es el software que utilizamos para programar la placa.

¿Qué es un microcontrolador?

Un microcontrolador es un circuito integrado que incluye los elementos necesarios para que un sistema electrónico pueda funcionar. Es decir, un microcontrolador es una especie de mini computadora que se encarga de ejecutar las instrucciones que le damos.

¿Qué es Arduino IDE?

Arduino IDE es un entorno de desarrollo que nos permite programar el Arduino. Es un software que se instala en nuestra computadora y que nos permite escribir el código que queremos que ejecute la placa.

Lenguaje .ino

El lenguaje de programación de Arduino es muy parecido a C y C++, pero tiene algunas diferencias. Por ejemplo, no es necesario declarar las variables al principio del programa, como en C. Además, tiene algunas funciones

predefinidas que nos facilitan la programación.

Estructura de un programa

Un programa de Arduino tiene dos partes: la función `setup()` y la función `loop()`, estas funciones vienen predefinidas en la librería de Arduino. La función `setup()` se ejecuta una sola vez al principio del programa, mientras que la función `loop()` se ejecuta continuamente en un bucle.

```
void setup() {
  // Inicializaciones
}

void loop() {
  // Código que se ejecuta continuamente
}
```

Funciones propias de Arduino

pinMode()

La función `pinMode()` se utiliza para configurar un pin como entrada o salida. Por ejemplo, si queremos configurar el pin 13 como salida, podemos hacer:

```
pinMode(13, OUTPUT);
```

digitalWrite()

La función `digitalWrite()` se utiliza para escribir un valor en un pin digital. Por ejemplo, si queremos encender un LED conectado al pin 13, podemos hacer:

```
digitalWrite(13, HIGH);
```

delay()

La función `delay()` se utiliza para hacer una pausa en el programa, toma como parametro el tiempo en milisegundos. Por ejemplo, si queremos esperar un segundo, podemos hacer:

```
delay(1000);
```

digitalRead()

La función `digitalRead()` se utiliza para leer el valor de un pin digital. Por ejemplo, si queremos leer el valor del pin 2, podemos hacer:

```
int valor = digitalRead(2);
```

analogRead()

La función `analogRead()` se utiliza para leer el valor de un pin analógico. Por ejemplo, si queremos leer el valor del pin A0, podemos hacer:

```
int valor = analogRead(A0);
```

analogWrite()

La función `analogWrite()` se utiliza para escribir un valor analógico en un pin PWM. Por ejemplo, si queremos escribir un valor de 128 en el pin 9, podemos hacer:

```
analogWrite(9, 128);
```

Programación en Arduino

Variables

En Arduino podemos declarar variables de la misma forma que en C o C++. Por ejemplo, para declarar una variable entera, podemos hacer:

```
int numero = 5;
```

o para declarar una variable de tipo `string`:

```
String nombre = "Juan";
```

Y lo mismo aplica para otros tipos de variables, como `float`, `char`, `boolean`, etc.

Condicionales

En Arduino podemos utilizar condicionales para tomar decisiones en nuestro programa. Por ejemplo, si queremos encender un LED si un botón está presionado, podemos hacer:

```
int boton = 2;
int led = 13;

void setup() {
  pinMode(boton, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {
  if (digitalRead(boton) == HIGH) {
    digitalWrite(led, HIGH);
  } else {
    digitalWrite(led, LOW);
  }
}
```

Ademas de `if` podemos utilizar `else if` y `else` para tomar decisiones más complejas.

```
int valor = 5;

if (valor > 10) {
  // Código que se ejecuta si valor es mayor a 10
} else if (valor < 5) {
  // Código que se ejecuta si valor es menor a 5
} else {
  // Código que se ejecuta si valor no es mayor a 10 ni menor a 5
}
```

Además de la estructura `if`, podemos utilizar la estructura `switch`.

```
int opcion = 2;

switch (opcion) {
  case 1:
    // Código que se ejecuta si opcion es 1
    break;
  case 2:
    // Código que se ejecuta si opcion es 2
    break;
  default:
    // Código que se ejecuta si opcion no es ni 1 ni 2
    break;
}
```

Operadores lógicos

En Arduino podemos utilizar operadores lógicos para comparar valores. Por ejemplo, si queremos comprobar si un valor es mayor que otro, podemos hacer:

```
int a = 5;
int b = 3;

if (a > b) {
  // Código que se ejecuta si a es mayor que b
}
```

Además de `>`, podemos utilizar otros operadores lógicos como `<`, `≥`, `≤`, `==`, `≠`, `&&`, `||`, etc.

```
int a = 5;
int b = 3;

if (a > b && a < 10) { // AND
  // Código que se ejecuta si a es mayor que b y menor que 10
}

if (a == 5 || b == 3) { // OR
  // Código que se ejecuta si a es igual a 5 o b es igual a 3
}

if (!(a == 5)) {  // Sería lo mismo que if (a ≠ 5)
  // Código que se ejecuta si a no es igual a 5
}
```

Operadores aritméticos

En Arduino podemos utilizar operadores aritméticos para realizar operaciones matemáticas. Por ejemplo, si queremos sumar dos números, podemos hacer:

```
int a = 5;
int b = 3;

int suma = a + b;
```

Además de `+`, podemos utilizar otros operadores aritméticos como `-`, `*`, `/`, `%`, etc.

Loops while y for

En Arduino podemos utilizar loops `while` y `for` repetir una parte de codigo determinadas veces.

While

```
int contador = 0;

void loop() {
  while (contador < 10) {
    // Código que se ejecuta mientras contador sea menor a 10
    contador++;
  }
}
```

For

```
void loop() {
  for (int i = 0; i < 10; i++) {
    // Código que se ejecuta 10 veces
  }
}
```

Arrays

En Arduino podemos utilizar arrays para almacenar varios valores en una sola variable. Por ejemplo, si queremos almacenar los valores de un sensor en un array, podemos hacer:

```
int sensor[5]; // Declaramos un array de 5 elementos

void setup() {
  for (int i = 0; i < 5; i++) {
    sensor[i] = analogRead(A0); // Guardamos el valor del sensor en el array
  }
}
```

Y después podemos acceder a los valores del array de la siguiente forma:

```
int valor = sensor[2]; // Accedemos al tercer elemento del array
```

Funciones

En Arduino podemos definir nuestras propias funciones. Por ejemplo, para definir una función que sume dos números, podemos hacer:

```
int suma(int a, int b) { // Escribimos int porque la función devuelve un entero y tomamos dos parámetros enteros
  return a + b; // Devolvemos la suma de los dos números
}
```

Y después podemos llamar a esta función desde la función `loop()`:

```
void loop() {
  int resultado = suma(5, 3); // Llamamos a la función suma con los parámetros 5 y 3
}
```

Ejemplo de programa

A continuación, un ejemplo de un programa que enciende y apaga un LED cada segundo:

```
int led = 13; // Declaramos la variable led y le asignamos el valor 13

void setup() {
  pinMode(led, OUTPUT); // Configuramos el pin 13 como salida
}

void loop() {
  digitalWrite(led, HIGH); // Encendemos el LED
  delay(1000); // Esperamos un segundo
  digitalWrite(led, LOW); // Apagamos el LED
  delay(1000); // Esperamos un segundo
}
```

En este programa, encendemos y apagamos el LED conectado al pin 13 de la placa cada segundo.

Librerías

En Arduino podemos utilizar librerías para facilitar la programación. Por ejemplo, si queremos utilizar la librería `Servo`, podemos incluirla al principio del programa de la siguiente forma:

```
#include <Servo.h> // o la libreria que se desea utilizar
```

Gracias a que incluimos la librería, podemos utilizar las funciones y variables que vienen predefinidas en ella. En este caso, la librería `Servo` nos permite controlar un servo motor de una forma más sencilla.

Comunicación Serial

En Arduino podemos comunicarnos con la computadora a través de la comunicación serial. Por ejemplo, si queremos enviar un mensaje a la computadora, podemos hacer:

```
void setup() {
  Serial.begin(9600); // Inicializamos la comunicación serial a 9600 baudios
}

void loop() {
  Serial.println("Hola mundo!"); // Enviamos el mensaje "Hola, mundo!" a la computadora
  delay(1000); // Esperamos un segundo
}
```

Y después podemos ver el mensaje en el monitor serial del Arduino IDE, esto es útil para debuggear el programa y ver los valores de las variables en tiempo real. Para abrir el monitor serial, podemos hacer clic en el icono de la lupa en la esquina superior derecha del Arduino IDE.

Para recibir datos de la computadora, podemos hacer:

```
void setup() {
  Serial.begin(9600); // Inicializamos la comunicación serial a 9600 baudios
}

// si el dato es una letra
void loop() {
  if (Serial.available() > 0) { // Si hay datos disponibles en el puerto serial
    char dato = Serial.read(); // Leemos el dato
    Serial.print("Dato recibido: "); // Imprimimos un mensaje
    Serial.println(dato); // Imprimimos el dato
  }
}

// si el dato es un número
void loop() {
  if (Serial.available() > 0) { // Si hay datos disponibles en el puerto serial
    int dato = Serial.parseInt(); // Leemos el dato
    Serial.print("Dato recibido: "); // Imprimimos un mensaje
    Serial.println(dato); // Imprimimos el dato
  }
}
```

Creación de librerías

Para crear una librería en Arduino, debemos seguir los siguientes pasos:

Crear una carpeta con el nombre de la librería en la carpeta `libraries` de la carpeta de instalación de Arduino.
Crear un archivo `.h` y un archivo `.cpp` con el mismo nombre que la carpeta.
En el archivo `.h` debemos declarar las funciones y variables de la librería.
En el archivo `.cpp` debemos definir las funciones de la librería.
Incluir el archivo `.h` en el programa de Arduino.

```
// MiLibreria.h

#ifndef MiLibreria_h // Si la librería no está definida
#define MiLibreria_h // Definimos la librería

#include "Arduino.h" // Incluimos la librería de Arduino

class MiLibreria { // Definimos la clase MiLibreria
  public:
    MiLibreria(); // Constructor
    void funcion(); // Declaración de la función
};

#endif
```

```
// MiLibreria.cpp

#include "MiLibreria.h"

MiLibreria::MiLibreria() {
  // Constructor
}

void MiLibreria::funcion() { // Definición de la función indicando a qué clase pertenece
  // Código de la función
}
```

```
// Programa de Arduino

#include <MiLibreria.h> // Incluimos la librería

MiLibreria miLibreria; // Creamos un objeto de la clase MiLibreria

void setup() {
  miLibreria.funcion(); // Llamamos a la función de la librería
}
```

Crear una librería propia en Arduino nos permite reutilizar código y hacer nuestros programas más modulares y fáciles de mantener.

Referencias

- Arduino
- Arduino IDE
- Arduino Reference
- Stack Overflow
- Foro de Arduino