Entornos de Desarrollo.

PlatformIO

¿Qué es PlatformIO?

PlatformIO es un entorno de desarrollo integrado (IDE) para el desarrollo de software embebido. Está diseñado para facilitar la creación, compilación y carga de código en microcontroladores y dispositivos embebidos de diferentes arquitecturas, como Arduino, ESP8266, ESP32, STM32 y muchas otras.

Una de las características principales de PlatformIO es su capacidad para trabajar con múltiples plataformas de hardware y frameworks de desarrollo, lo que permite a los desarrolladores escribir código una vez y luego desplegarlo en una variedad de dispositivos sin tener que preocuparse demasiado por las diferencias en el hardware subyacente.

PlaformIO está desarrollada sobre VSCode (Visual Studio Code) otro IDE desarrollado por Microsoft.

Por su parte **PlatformIO** presenta características adicionales:

Depurador: ideal para examinar elementos particulares de tu código, cuando las cosas salen mal.

Detección de puerto automático.

Gestor de librerías privadas: permite añadir librerías a tus proyectos de forma particular, en lugar de global como sucede en Arduino IDE. También puedes controlar qué versión usarás de cada librería.

Integración con un repositorio de código: como GitHub

Permite **desarrollo remoto.**

Asistencia inteligente de escritura: funciones como el autocompletado y la comprobación de sintaxis en tiempo real, ayudan a cualquiera a escribir código de forma profesional con menos errores.

Referencias integradas: permiten una mejor comprensión del código. Gracias a esto, puede que aprendas algo nuevo a medida que escribes tus líneas.

Archivo platformio.ini: contiene la configuración de tu proyecto y permite que lo muevas fácilmente de una computadora a otra.

Arduino IDE

¿Qué es Arduino IDE?

La IDE Arduino, o Entorno de Desarrollo Integrado Arduino, es una plataforma de software libre y de código abierto para programar y desarrollar proyectos utilizando placas Arduino. Dispone de una interfaz fácil de usar para crear, compilar y cargar código en los microcontroladores Arduino.

Características principales del IDE de Arduino

El entorno de desarrollo Arduino incluye una serie de características que lo convierten en una herramienta imprescindible para los aficionados a la electrónica.

1. Resaltado de sintaxis en el editor de código.

Arduino IDE incluye un editor de código con resaltado de sintaxis para ayudarte a leer y escribir código más fácilmente. Utiliza diferentes colores para resaltar diferentes aspectos del código, como variables, funciones y comentarios, mejorando la legibilidad del código y minimizando los errores.

2. Director de biblioteca.

Con el IDE Arduino dispones de un Director de Bibliotecas que te permite gestionar e instalar bibliotecas con facilidad. El código pre-escrito en las bibliotecas amplía la funcionalidad de las placas Arduino. Puedes ahorrar tiempo y esfuerzo utilizando el Director de Bibliotecas para buscar e instalar rápidamente las bibliotecas que necesites para tu proyecto.

3. Monitor Serial.

El Monitor Serial es una herramienta muy útil para depurar y comunicarse con su placa Arduino. Te permite enviar y recibir datos entre tu placa Arduino y tu ordenador, facilitando la monitorización y resolución de problemas de funcionamiento del programa.

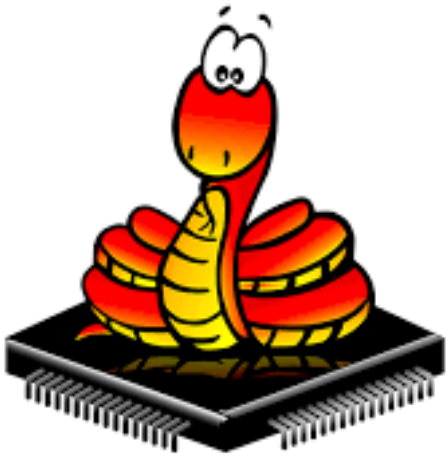
4. Gestor de placas.

El Administrador de placas es una herramienta del IDE Arduino que te permite añadir nuevas placas. Las placas Arduino están disponibles en una variedad de configuraciones, cada una con su propio conjunto de características y funciones. El Gestor de Placas te permite elegir la placa específica que estás utilizando, asegurando que el IDE genera el código para esa placa de forma apropiada.

5. Ilustraciones integradas.

Viene con una gran cantidad de ejemplos integrados que cubren un amplio espectro de temas de programación y electrónica. Estos ejemplos son excelentes para principiantes y sirven de inspiración para proyectos más sofisticados. Puede aprender a usar varios sensores, actuadores y protocolos de comunicación con su placa Arduino examinando los ejemplos.

MicroPython



¿Qué es MicroPython?

Para empezar, a hablar de Micropython primero tenemos que comenzar a hablar del lenguaje del que deriva. **Python.**

Phython es un lenguaje de alto nivel y sencillo de aprender, recuerda que de cuanto más alto nivel es un lenguaje más fácil resulta la programación ya que el lenguaje de programación hace cosas por nosotros que en un lenguaje de bajo nivel tendríamos que definir hasta el más último detalle, simplifica el número de líneas de código, también usan un lenguaje más natural, más cercano al nuestro resultándonos más amigable.

Micropython como ya habrás podido intuir es una versión adaptada de Python para microcontroladores. Es un lenguaje interpretado es decir no se compila para ejecutarse, sino que se interpreta durante su ejecución.

Como entenderás un microcontrolador tiene unos recursos más limitados que un ordenador por lo que micropython no tiene todas las instrucciones, funciones y librerías de las que dispone Python, pero no por eso deja de ser un lenguaje muy potente en su campo.

Con MicroPython, se pueden hacer muchísimos proyectos controlando las entradas/salidas de un microcontrolador, tomando lecturas de señales analógicas y digitales, generando señales PWM, control de motores y servomotores, pantallas LCD, OLED, comunicarnos mediante I2C, SPI, etc.

MICROPYTHON VS C/C++ ARDUINO

La diferencia importante es que el lenguaje MicroPython se **interpreta** en lugar de compilarse como con el lenguaje de programación Arduino lo que significa que cuando se ejecuta el código MicroPython, tiene que hacer un poco más de trabajo para convertir el código MicroPython en instrucciones que el microcontrolador entienda.
El código interpretado es mucho más limpio y simple en comparación con los lenguajes que se compilan. Incluso puede escribir y ejecutar código interpretado como MicroPython directamente en una placa sin compilar ni cargar, cosa que no se puede hacer con Arduino

Una desventaja del código interpretado y MicroPython frente a Arduino es que a veces hay menos rendimiento y más uso de memoria al interpretar el código. Un programa escrito en Arduino se ejecutará lo más rápido posible en el microcontrolador, mientras que MicroPython será un poco más lento porque tiene que interpretar cada instrucción y convertirla en código de máquina. En la práctica el tipo de proyectos que acometen personas que suelen usar arduino no se van a resentir o prácticamente serán imperceptibles, pero si estuviéramos condicionados por tiempo o memoria, MicroPython le permite escribir código en C/C++ para obtener el máximo rendimiento.

En resumen, no hay unas aplastantes razones para usar C/C++ o micropython pero en resumen diríamos si queremos mayor rapidez de ejecución usaríamos C/C++ y si queremos sencillez usaríamos Micropython, además que sería el aperitivo perfecto para adentrarnos en la programación de su hermano mayor el Python para hacer proyectos bastante más completos y complejos, usando por ejemplo una Raspberry pi.

PLACAS QUE SOPORTAN MICROPYTHON

→ ESP 8266

→ ESP 32

→Raspberry Pi Pico

→ BBC Microbit

→ Teensy 3.x

→ Pyboard

Profesor: Gonzalo Vera.

Alumno: Nicolás Barrionuevo.