Universidad Autónoma de TAMAULIPAS


# Guía rápida para el uso de PlatformIO en VS Code

## Programación de Interfaces y Puertos / Diseño Electrónico basado en Sistemas Embebidos

**Dr. Daniel Lopez Piña**

**Dra. Azucena Contreras Villanueva**

**M.D. Felipe Silva Hernández**

**2026-1**

**Dependencia Académica:** **Unidad Académica Multidisciplinaria Mante Centro**

**Programa Educativo:** **Ingeniero en Sistemas Computacionales**

https://github.com/dnllp/Material_Didactico_ISC

## Guía Rápida: Transición a PlatformIO en VS Code

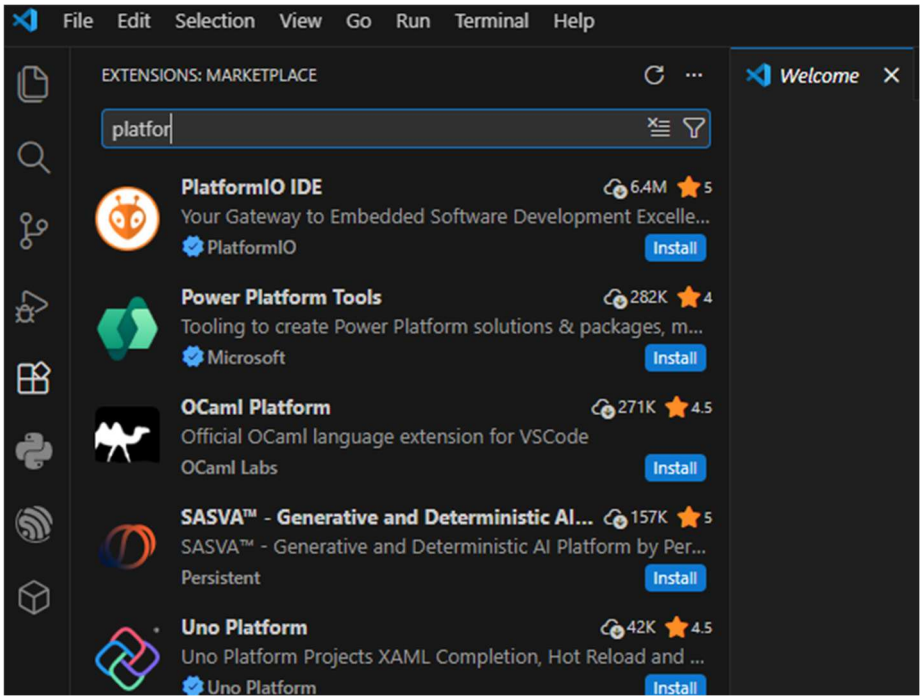
El desarrollo de firmware a nivel industrial rara vez se hace en un solo archivo .ino. PlatformIO (PIO) es un ecosistema profesional que transforma VS Code en un potente entorno de desarrollo integrado (IDE) para sistemas embebidos.

**Paso 1: Preparar el Entorno Base**

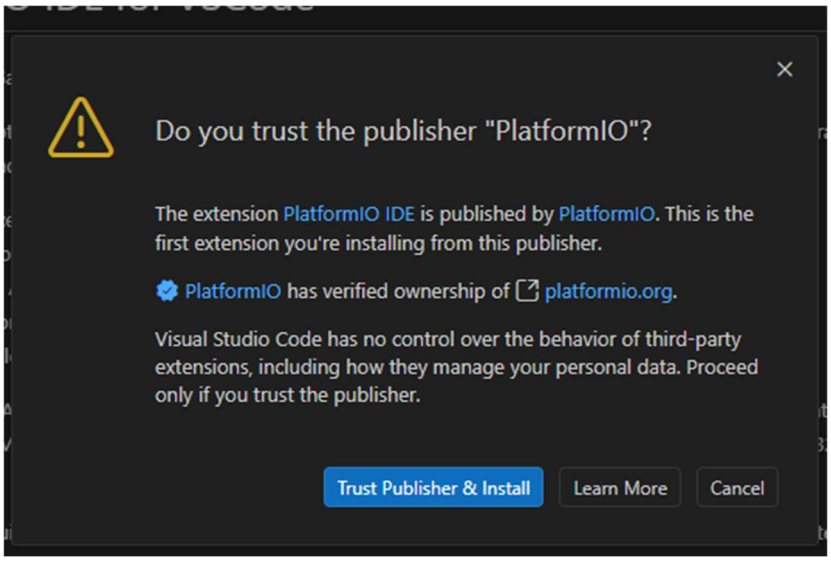
1. **Descargar VS Code:** Ingresa a code.visualstudio.com y descarga la versión correspondiente a tu sistema operativo (Windows, macOS o Linux).
2. **Instalar:** Sigue el asistente de instalación estándar. No requiere configuraciones especiales.

**Paso 2: Instalar la Extensión PlatformIO**

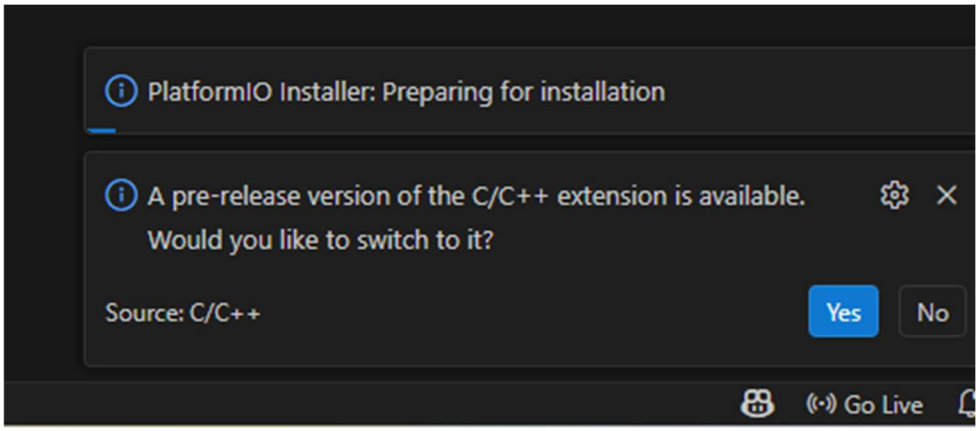
1. Abre VS Code.
2. En la barra lateral izquierda, haz clic en el icono de **Extensiones** (o presiona Ctrl+Shift+X).
3. En el buscador, teclea **PlatformIO IDE**.

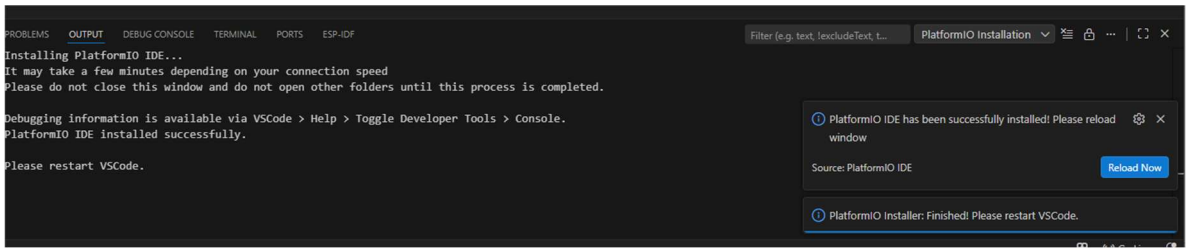



4. Haz clic en **Instalar** en el resultado oficial (desarrollado por PlatformIO).



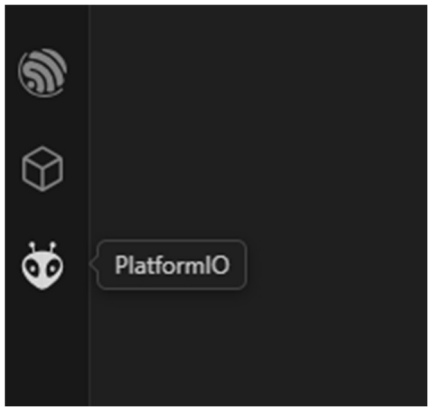

5. **Importante:** Una vez instalada, PlatformIO comenzará a descargar su núcleo (*Core*) en segundo plano. Verás una notificación en la esquina inferior derecha. **Debes esperar a que termine y reiniciar VS Code** antes de continuar.



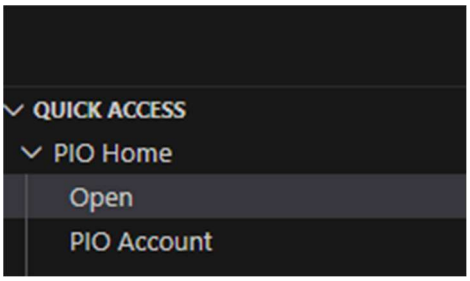




**Paso 3: Creando el Primer Proyecto (Ejemplo con ESP32)**

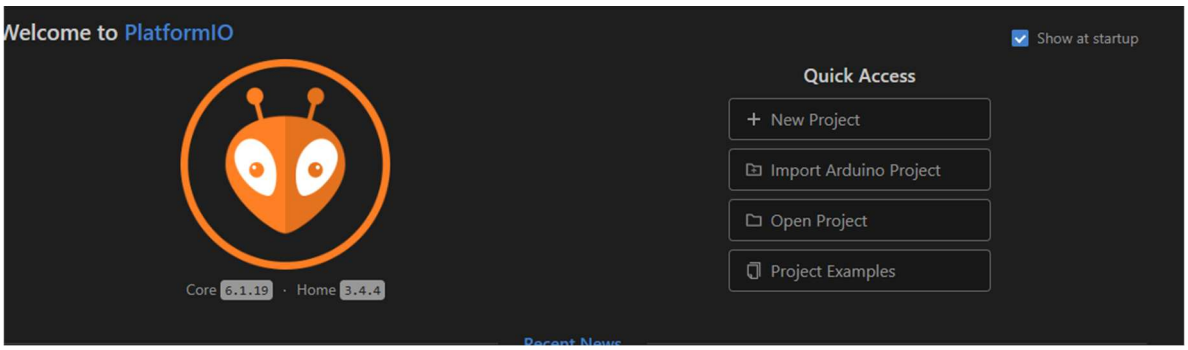
Una vez reiniciado, verás un nuevo ícono de una **cabeza de extraterrestre (Alien)** en la barra lateral izquierda. Ese es el menú de PlatformIO.



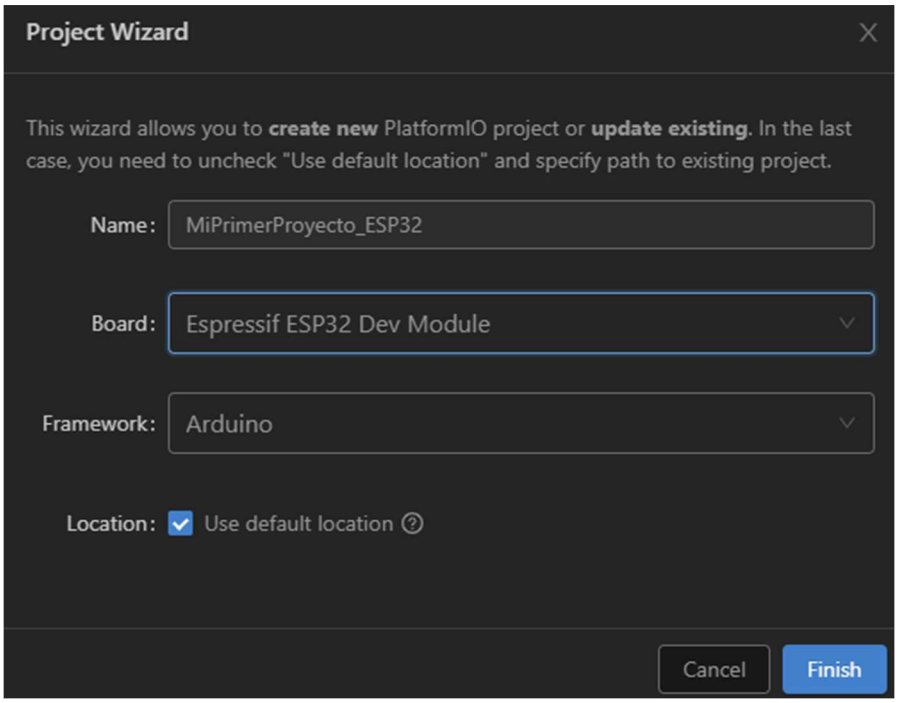

1. Haz clic en el ícono de PlatformIO y selecciona **PIO Home -> Open**.



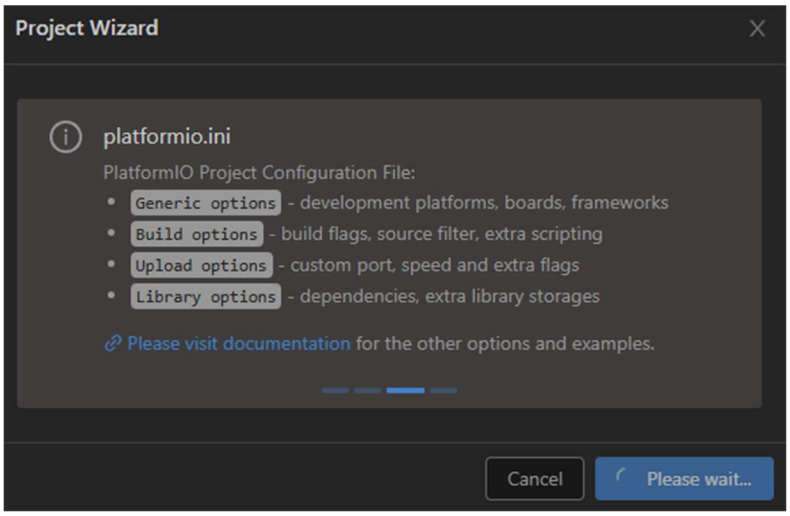

2. Haz clic en el botón **+ New Project**.

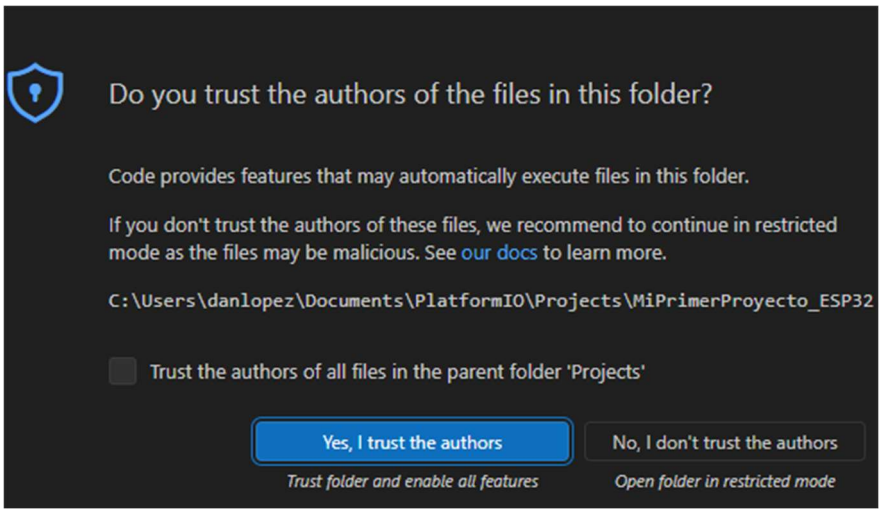



3. Llena el formulario del proyecto:
    - **Name:** MiPrimerProyecto_ESP32 (Evita espacios y caracteres especiales).
    - **Board:** Busca y selecciona tu tarjeta física. Por ejemplo: Espressif ESP32 Dev Module.
    - **Framework:** Selecciona Arduino (esto nos permite usar la sintaxis de C++ y funciones como digitalWrite() que ya conocemos).



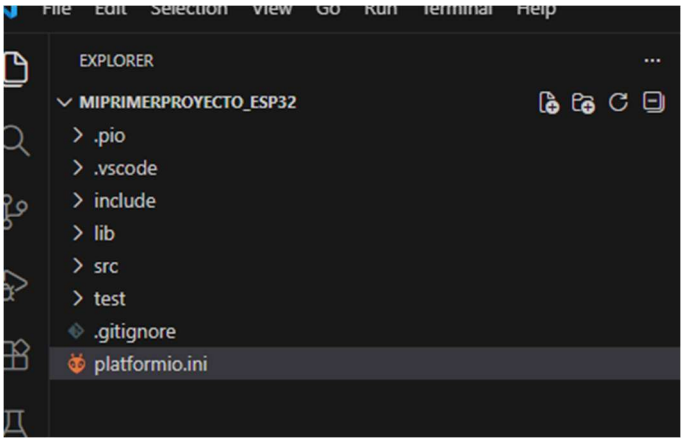

4. Haz clic en **Finish**. PlatformIO creará la estructura de carpetas y descargará las herramientas de compilación necesarias para esa placa específica (esto puede tardar un par de minutos la primera vez, PACIENCIAAA).





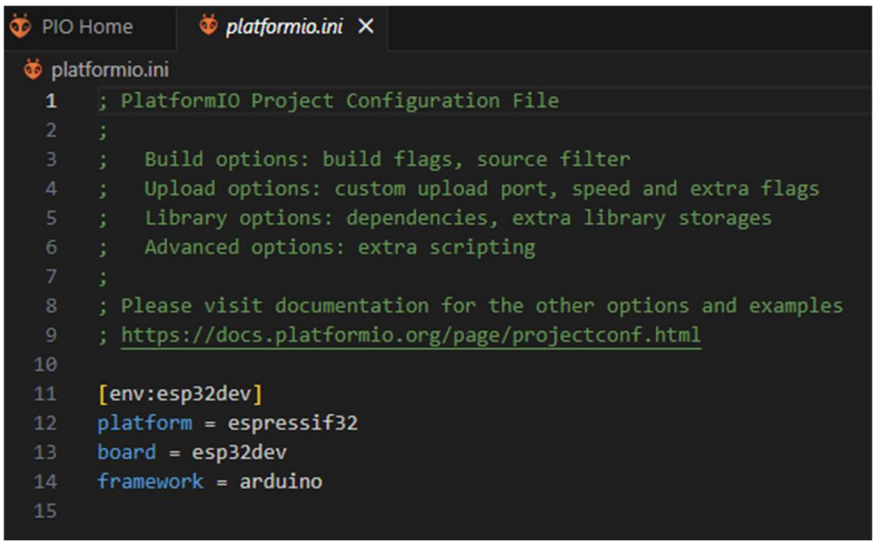

**Paso 4: Anatomía de un Proyecto Profesional**

A diferencia del IDE de Arduino, PlatformIO organiza el código bajo estándares de ingeniería de software. En el explorador de archivos (izquierda) verás varias carpetas.




Las dos más importantes son:

- **src/ (Source):** Aquí vive tu código. El archivo principal ya no es un .ino, sino un archivo C++ real llamado **main.cpp**.
    - *Nota vital:* En .cpp, es obligatorio incluir la librería principal del framework al inicio del archivo: #include <Arduino.h>.
- **platformio.ini:** Es el archivo de configuración general del proyecto. Aquí se definen la velocidad del monitor serial, las librerías externas a descargar y los entornos de compilación.




Ejemplo del archivo platformio.ini

[env:esp32dev]

platform = espressif32

board = esp32dev

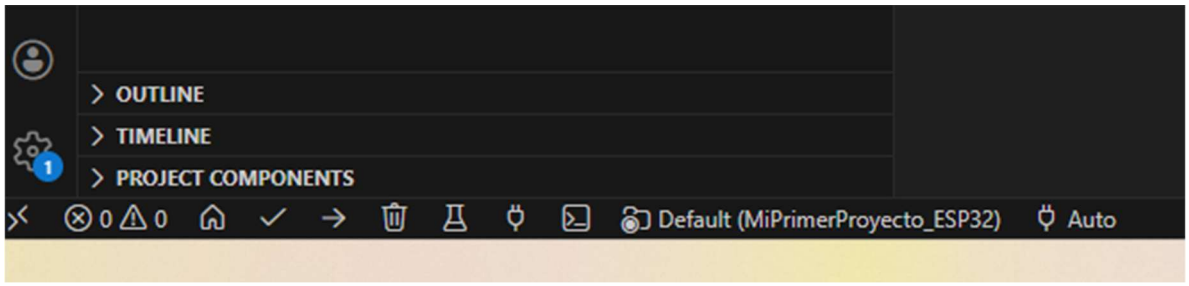
framework = arduino

monitor_speed = 115200 ; Velocidad del monitor serial (agregue esta línea)

**Paso 5: Compilar y Subir el Código al Hardware**

PlatformIO agrega una **barra de herramientas azul** en la parte inferior (Status Bar) de VS Code. Los botones principales de izquierda a derecha son:

1. **Home (Casa):** Abre la pantalla principal de PIO.
2. **Build (Palomita / Checkmark):** Compila el código para verificar que no haya errores de sintaxis, pero no lo sube a la placa.
3. **Upload (Flecha hacia la derecha):** Compila el código y lo transfiere a la memoria flash del microcontrolador conectado por USB. PIO detecta el puerto COM automáticamente.
4. **Serial Monitor (Enchufe):** Abre la terminal para ver los mensajes enviados por Serial.print().




Paso 6: Código de prueba

Accede a la carpeta SRC y realiza lo siguiente:

1. En la carpeta encontrarás el archivo main.cpp, modifícalo con el siguiente código:

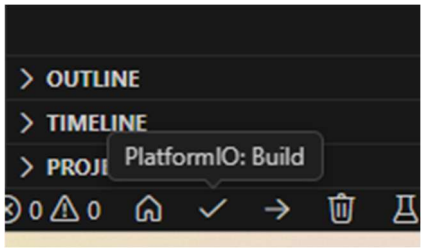
```
#include <Arduino.h>

int led = 2;

void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT);
}

void loop() {
  digitalWrite(led,HIGH);
  delay(1000);
  digitalWrite(led,LOW);
  delay(1000);
}
```

2. Compila el código (Build) para verificar que no haya errores de sintaxis con la palomita.




3. Si todo esta correcto te debe aparecer lo siguiente:

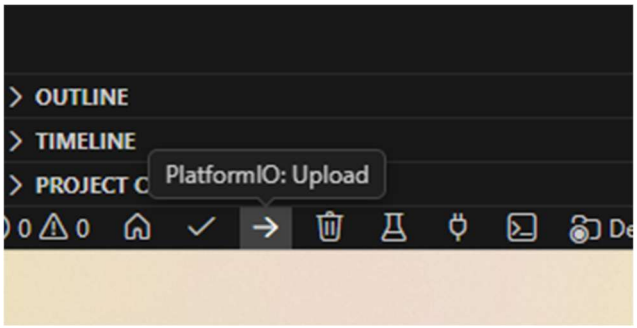
```
*  Terminal will be reused by tasks, press any key to close it.

*  Executing task: C:\Users\danlopez\.platformio\penv\Scripts\platformio.exe run

RAM:   [=         ]   6.4% (used 21112 bytes from 327680 bytes)
Flash: [==        ]  18.3% (used 239869 bytes from 1310720 bytes)
Building .pio\build\esp32dev\firmware.bin
esptool.py v4.11.0
Creating esp32 image...
Merged 2 ELF sections
Successfully created esp32 image.
========================================================================= [SUCCESS] Took 7.61 seconds ==
*  Terminal will be reused by tasks, press any key to close it.
```

Navigate to Command
Scroll to Previous Command (
Scroll to Next Command (Ctrl-

4. Asegúrate de tener conectada la tarjeta ESP32 a la computadora y transfiérelo con Upload (flecha a la derecha)




5. Si todo se realiza correctamente el Led de la ESP32 debe parpadear y te debe aparecer lo siguiente en VS Code:

```
*  Executing task: C:\Users\danlopez\.platformio\penv\Scripts\platformio.exe run --target upload
Writing at 0x00037462... (66 %)
Writing at 0x0003f47e... (77 %)
Writing at 0x00044b05... (88 %)
Writing at 0x0004a452... (100 %)
Wrote 240240 bytes (132158 compressed) at 0x00010000 in 3.3 seconds (effective 587.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
========================================================== [SUCCESS] Took 18.64 seconds ==========
*  Terminal will be reused by tasks, press any key to close it.
```

**Consideraciones**: Algunas placas ESP32 requieren la instalación de un driver CP2102 o CH340 para que las reconozca el sistema operativo. Consulta: https://docs.sunfounder.com/projects/esp32-starter-kit/es/latest/faq/install_driver.html

**Gestión Profesional de Librerías**

En el IDE de Arduino, instalar librerías implica descargar archivos .zip o usar un gestor visual que muchas veces rompe la compatibilidad si movemos el proyecto a otra computadora.

PlatformIO resuelve esto de manera elegante mediante la **declaración de dependencias**. Cuando escalamos nuestros proyectos de un simple parpadeo de LED a sistemas completos, invariablemente necesitaremos librerías externas para interactuar con hardware complejo.

En PlatformIO, el archivo platformio.ini es el corazón del proyecto. Actúa como una "receta": le dice al compilador exactamente qué placa estamos usando, qué framework y, lo más importante, qué librerías descargar automáticamente de internet antes de compilar. El comando para esto es **lib_deps** (Library Dependencies).

**Caso de Estudio 1: Navegación Robótica con IMU**

Imaginemos que están desarrollando el firmware para un robot diferencial que requiere navegación precisa basada en un IMU (Unidad de Medición Inercial, como el MPU6050 o BNO055). En lugar de buscar archivos por toda la web, solo necesitamos decirle a PlatformIO el nombre del autor y la librería.

Abrimos el archivo platformio.ini y agregamos:

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200

; Aquí declaramos nuestras dependencias
lib_deps =
   adafruit/Adafruit MPU6050 @ ^2.2.4
   adafruit/Adafruit Unified Sensor @ ^1.1.9
```

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200

; Aquí declaramos nuestras dependencias
lib_deps =
    adafruit/Adafruit MPU6050 @ ^2.2.4
    adafruit/Adafruit Unified Sensor @ ^1.1.9

```

**¿Qué significa esto?**

- adafruit/: Es el fabricante o desarrollador de la librería.
- Adafruit MPU6050: Es el nombre exacto de la librería en el registro de PlatformIO.
- @ ^2.2.4: Es el control de versiones. El símbolo ^ le dice a PlatformIO: *"Usa la versión 2.2.4, y si hay actualizaciones menores que no rompan el código, descárgalas, pero NO cambies a la versión 3.0"*. Esto garantiza que su código funcione igual hoy que dentro de cinco años.
- Al compilar (build) observamos que descarga las librerías y sus dependencias.

```
* Terminal will be reused by tasks, press any key to close it.

* Executing task: C:\Users\danlopez\.platformio\penv\Scripts\platformio.exe run

Processing esp32dev (platform: espressif32; board: esp32dev; framework: arduino)
--------------------------------------------------------------------------------
Library Manager: Installing adafruit/Adafruit MPU6050 @ ^2.2.4
Downloading  [####################################]  100%
Unpacking  [####################################]  100%
Library Manager: Adafruit MPU6050@2.2.9 has been installed!
Library Manager: Resolving dependencies...
Library Manager: Installing Adafruit BusIO
```

**Caso de Estudio 2: Nodos IoT y Monitoreo Hidrológico (LoRa)**

Otro escenario clásico de ingeniería de interfaces es el despliegue de telemetría a larga distancia. Supongamos que están construyendo un sistema de monitoreo hidrológico remoto basado en módulos LoRa (como el SX1276) para sistemas de alerta temprana o control de inundaciones.

Para interactuar con el transceptor por SPI, simplemente agregamos la librería correspondiente a nuestra lista en el mismo proyecto:

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200

lib_deps =
   adafruit/Adafruit MPU6050 @ ^2.2.4
   adafruit/Adafruit Unified Sensor @ ^1.1.9
   sandeepmistry/LoRa @ ^0.8.0
```

Una vez que guardan el archivo platformio.ini y presionan el botón de **Build** (la palomita azul en la barra inferior), PlatformIO se conectará a sus servidores, buscará las librerías Adafruit MPU6050 y LoRa, las descargará, las vinculará al proyecto local y compilará todo.

Si le envían esta carpeta a un compañero de equipo a través de GitHub, no tienen que enviarle las librerías. Al abrir él el proyecto en su propia computadora, PlatformIO leerá el platformio.ini y descargará exactamente las mismas versiones.

¡Se acabaron los errores de "Archivo o directorio no encontrado"!

**Laboratorio Rápido: Búsqueda e Integración**

Diríjanse a la pestaña principal de PlatformIO (PIO Home) y abran la sección **"Libraries"**. Busquen una librería para manejar un sensor de temperatura **DHT22** o una pantalla **OLED SSD1306** vía I2C. Encuentren el código de instalación bajo la pestaña "Installation" y agréguenlo correctamente a su archivo platformio.ini. Compilen el proyecto vacío para verificar que la descarga se realice con éxito.