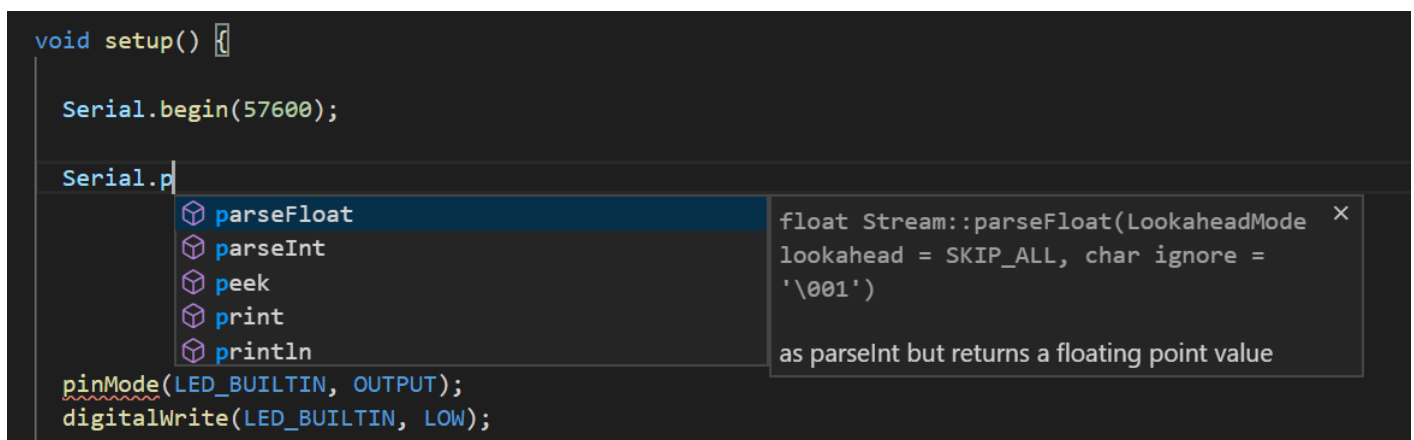


Cuando acumulas una variedad de tarjetas de desarrollo y/ microcontroladores, al final terminas teniendo el ordenador **lleno de mil programas**, que si el IDE para ST, que si el de Kinetis, el de NXP, Arduino,... Son miles de programas, que además de ocupar espacio en el disco duro, **ocupan mucho tiempo** si tienes que volver a reinstalar todo.

Si estáis acostumbrados a Arduino, igual no habéis tenido este problema, pero **aún así os interesa** lo que os voy a contar. Así que lo que os vengo a enseñar es el **PlatformIO**, un sistema o entorno preparado para poder trabajar sobre **muchos microcontroladores diferentes**, plataformas y frameworks, de la manera **más sencilla posible**.

Para los usuarios de Arduino, deciros que esta plataforma, tiene **bastantes mejoras** con respecto al IDE de Arduino original, entre ellas está el **autocompletado**, poder moverte por el código de **manera automática** o el mejor manejo para **proyectos** un poco más **complejos**.



```
void setup() {  
    Serial.begin(57600);  
    Serial.p  
    pinMode(LED_BUILTIN, OUTPUT);  
    digitalWrite(LED_BUILTIN, LOW);  
}
```

The screenshot shows an IDE with a code completion dropdown for `Serial.p`. The dropdown lists `parseFloat`, `parseInt`, `peek`, `print`, and `println`. A tooltip for `parseFloat` is visible, stating: `float Stream::parseFloat(LookaheadMode lookahead = SKIP_ALL, char ignore = '\001')` and `as parseInt but returns a floating point value`.

Ejemplo de autocompletado en Arduino

## ¿Qué es PlatformIO?

PlatformIO es un ecosistema para el desarrollo de **sistemas embarcados** e IoT, el cuál facilita y mucho la gestión de proyectos de software embarcado, gestión de dependencias y librerías, tests,... Es un poco, todo lo que se le puede pedir a un ecosistema así. De momento, **no le he encontrado ningún problema** en su plantemiento (aunque que sí algún fallo), y es que te permite de manera muy sencilla generar entornos para programar **casi cualquier microcontrolador** desde el mismo entorno.

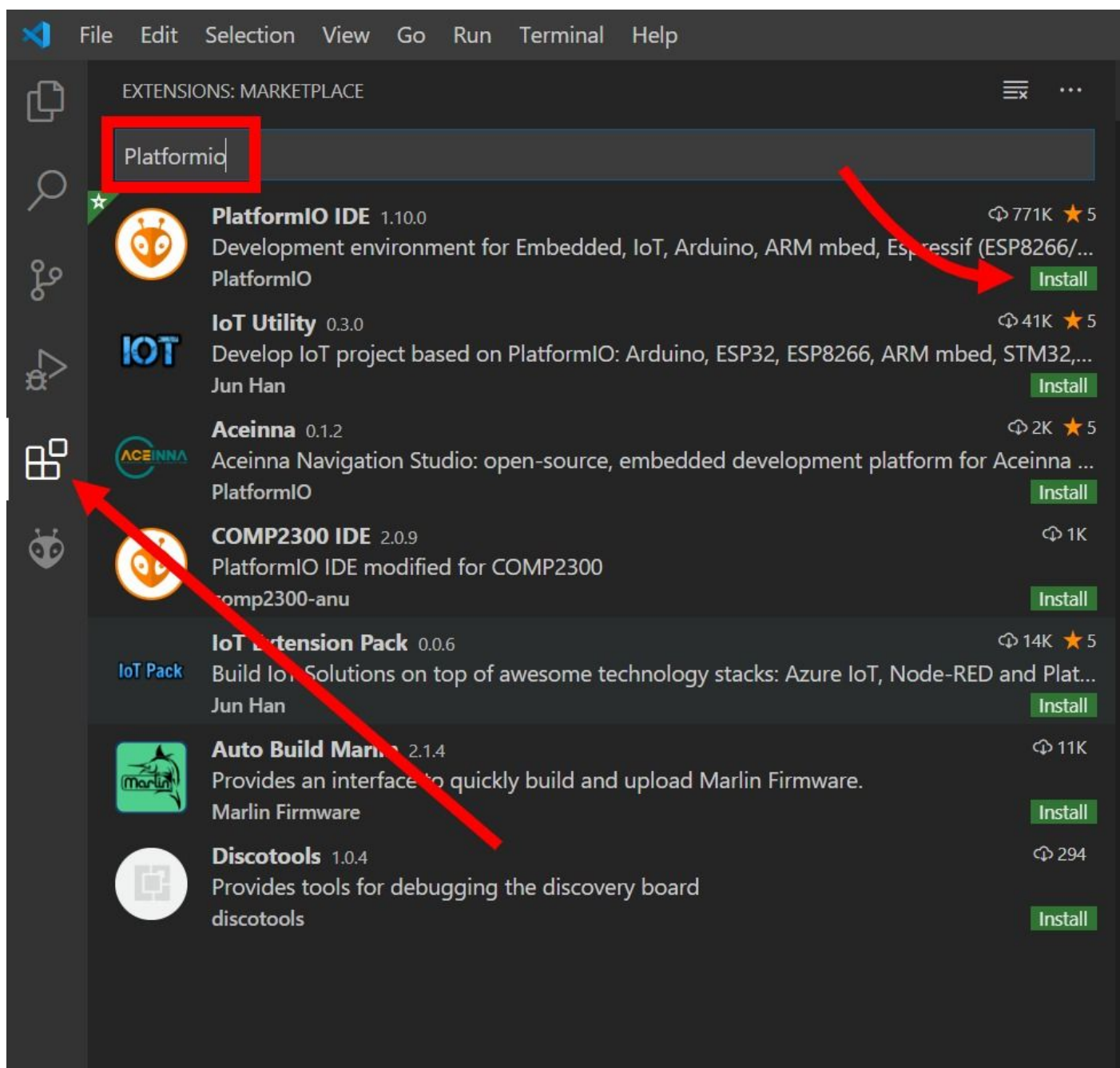
Además, te abstrae de toda la configuración que hay detrás de los *frameworks*. En algunos casos esto es malo, ya que pierdes el control de lo que se está haciendo por debajo. Aunque puedes **prácticamente configurar todos** los parámetros de compilación, flasheo y 'monitoreo', el funcionamiento básico **es muy asequible**. Por otra parte, te permite **depurar el software** mientras se ejecuta en la placa, siempre que esta lo permita.

En el momento de escribir esto soporta:

- 20 frameworks: Arduino, mbed, CMSIS, ESP-IDF, STM32Cube,....
- 785 placas de desarrollo: Casi todos los Arduinos, STM32F1, ESP32, ...
- 7163 librerías
- 179 ejemplos de código

## Instalación

La instalación de PlatformIO es muy sencilla, simplemente tenéis que descargar el **Visual Studio Code**, qué es el IDE de Microsoft, la versión gratuita. Una vez que la abráis teneis que ir a la pestaña de complementos, y buscar el “platformIO”, darle a Instalar y esperar. Os dejo una imagen con el proceso.



Instalación platformIO

# Crear un nuevo proyecto

---

Una vez que lo tengáis instalado, lo único que tenéis que hacer es crear un nuevo proyecto, para ello vais a lo que se llama el *PIO Home*, que es como la pantalla de inicio del PlatformIO. Desde esta parte podéis hacer todo lo referente a la gestión de proyectos del PlatformIO. Si no lo veis, en la parte izquierda os ha tenido que salir una **nueva pestaña** con la cabeza de una aveja.

Si abris esa pestaña, os saldra a la izquierda abajo un menu *Quick Access* con un elemento: *PIO Home*, le hacéis click y luego a *Projects & Configuracion* (debajo del PIO Home). Os saldrá una lista con los proyectos que tenéis, y un botón para añadir nuevo proyecto.

En la ventana que os tiene que salir, solo se piden 3 campos:

1. **Nombre del proyecto** (creo que es bastante autodescriptivo)
2. **Board** o placa de desarrollo. Aquí tenéis que buscar vuestra placa, según vais escribiendo os irá mostrando los resultados más parecidos. Algunos ejemplos (hay hasta casi 900 tarjetas):
  - Arduino Nano ATmega168
  - DOIT ESP32 DEVKIT V1
  - BluePill F103C8
3. **Framework**: Qué es el conjunto de librerías con el que queréis programar. Aquí podéis elegir Arduino (para programar como en arduino normal), o mejor, usar librerías más avanzadas si te lo permite como *mbed* o *ESP-IDF*...

## Project Wizard

This wizard allows you to **create new** PlatformIO project or **update existing**. In the last case, you need to uncheck "Use default location" and specify path to existing project.

Name:

Board:

Framework:

Location: ☒ Use default location ?

Ventana de nuevo proyecto

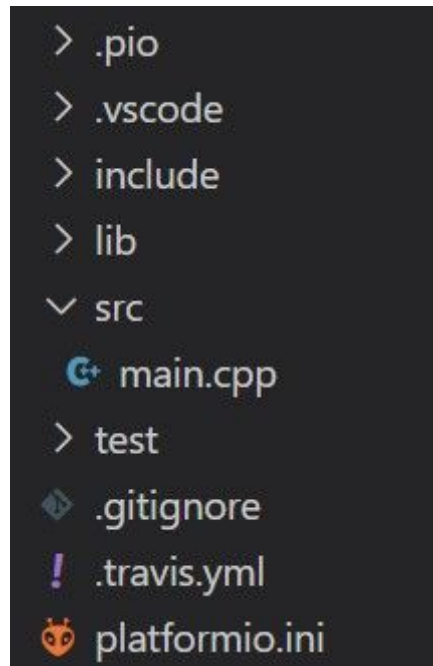
Una vez que aceptéis todo tardará un rato en generar el proyecto ya que la primera vez que generas un proyecto **se descarga todo lo necesario** es decir, la información de la placa que has elegido, y el framework.

## Compilando y ejecutando un proyecto.

Cuando se genera un proyecto, se crean las carpetas y ficheros básicos necesarios, en algunos casos no usaremos todos. En este caso para el platformIO se generarán:

- `.pio/` carpeta donde se generarán los ficheros intermedios de la compilación y temporales.
- `include/` Aquí se deberían guardar las cabeceras de tu ficheros, es decir los fichero `.h`
- `lib/` carpeta para generar librerías propias de tu proyecto, es decir librerías privadas
- `source/` para guardar el código fuente (`.c` y `.cpp`)
- `test/` aquí puedes definir tus tests para que se ejecuten y comprueben que el código funciona bien
- `platformio.ini` fichero de configuración del platformio, aquí se definen las librerías a usar por el programa (por ejemplo la librería SD de Arduino...), y muchas otras cosas. Generalmente

con el que viene **por defecto es suficiente**.



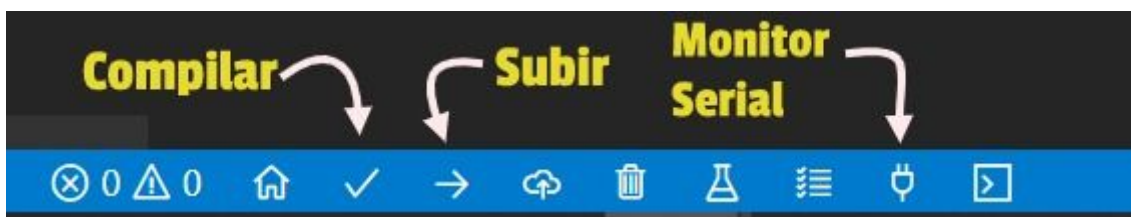
Proyecto base platformIO

De echo, el fichero **main.cpp** es el fichero principal de tu programa y si lo abres verás que ya está preparado para ser compilado con los *includes* necesarios. Si estas acostumbrado a los ficheros *\*.ino* verás que es igual solo que se le ha añadido una cabecera

```
#include <Arduino.h>
```

 el resto es exactamente igual.

Para compilar y subir el programa os habrán aparecido unos iconos en la **parte inferior izquierda** para poder trabajar de manera más cómoda. Son como los de el IDE de Arduino, además de funciones **más avanzadas**. Os dejo aquí las más comunes.



Nuevos botones platformIO

El único cambio que yo noto con respecto al Arduino es el puerto serie, hay que cerrarlo con **CTRL + T**, sino igual os da problemas!

## Configuración *platformio.ini*

Algunos valores de configuración que suelo cambiar yo son:

- **Velocidad del monitor serial:** Para adecuarlo a la velocidad que configuréis en vuestro micro. `monitor_speed = 57600`
- **Puerto monitor:** Podéis cambiar el puerto que se configura por defecto gracias a `monitor_port = ...`. Aquí podéis poner un puerto COM, o un socket, o lo que queráis.

## Conclusión

---

En esta entrada os quería dejar un primer vistazo al platformIO, lo he empezado a probar, y si sois como yo que tenéis mil placas de diferente fabricantes y mil IDEs, con esto se **os solucionan los problemas**.

Además al ser opensource, podemos ver y modificar el código, por ejemplo yo hoy necesitaba la placa FRDM-KL82Z y al ver que estaba fallando, **lo arregle yo** y no tuve que esperar a una nueva versión.