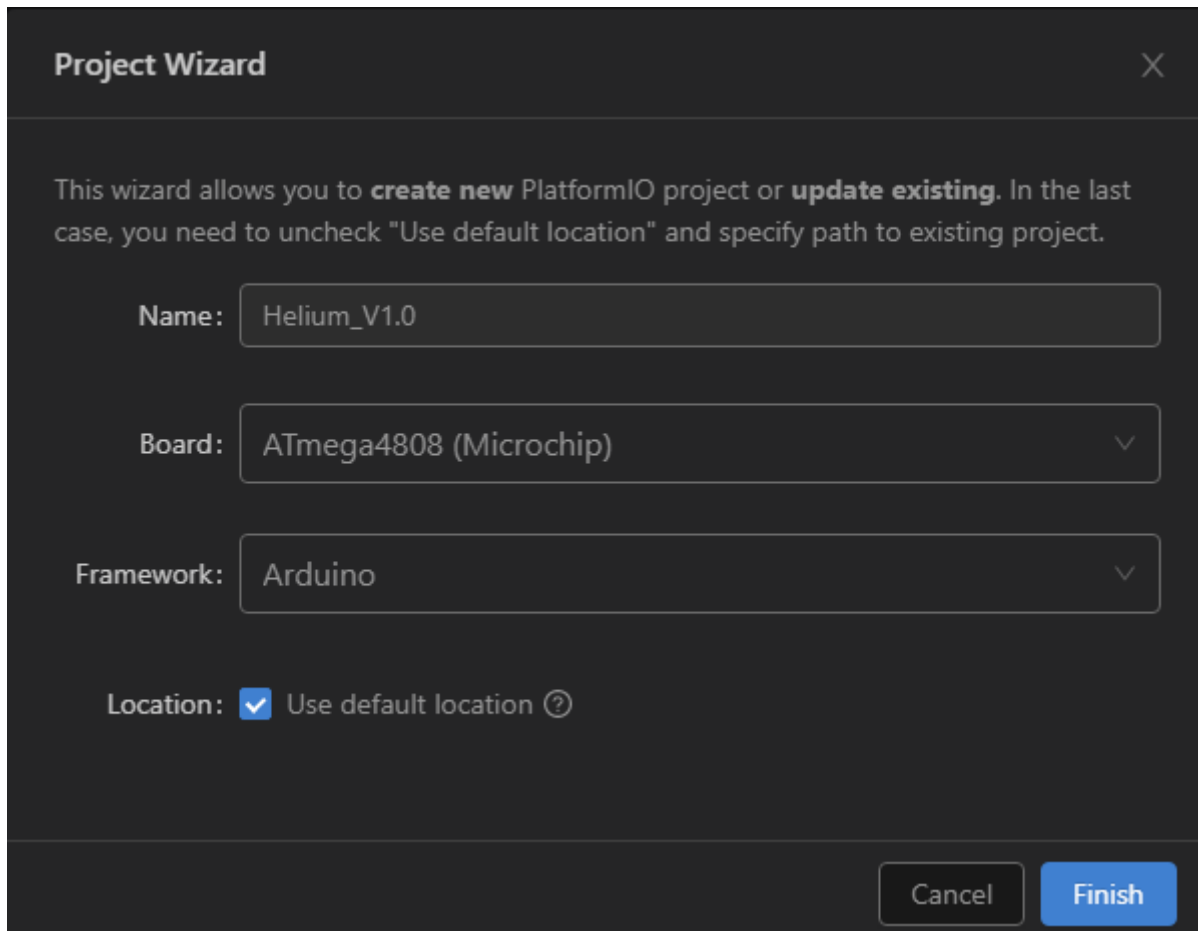


# Nuevo proyecto HELIUM.

Abrimos un nuevo proyecto seleccionando la placa ATmega4808 (Microchip)



The screenshot shows the 'Project Wizard' dialog box in PlatformIO. It has a dark theme. At the top, it says 'Project Wizard' with a close button (X). Below that, a message states: '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.' There are three input fields: 'Name:' with the text 'Helium\_V1.0', 'Board:' with a dropdown menu showing 'ATmega4808 (Microchip)', and 'Framework:' with a dropdown menu showing 'Arduino'. Below these is a 'Location:' section with a checked checkbox and the text 'Use default location' followed by a help icon (?). At the bottom right, there are two buttons: 'Cancel' and 'Finish'.

Como indica el propio nombre, lo que hemos seleccionado no es una placa en concreto sino placas que tengan el microchip ATmega4808.

Como se pueden montar varios modelos distintos necesitamos informar a PlatformIO las características específicas de la placa que vamos a utilizar. Para ello añadiremos las siguientes tres líneas al PlatformIO.ini

```
board_build.f_cpu = 8000000L
board_hardware.oscillator = internal
board_build.variant = 28pin-standard
```

Ahora vamos a cargar las librerías que nos hacen falta para el programa. La primera de ellas es la “lmic.h”. Como hay varias versiones de esta librería vamos a informar directamente en el PlatformIO.ini la versión a utilizar añadiendo la siguiente línea.

```
lib_deps = mcci-catena/MCCI LoRaWAN LMIC library@^4.1.1
```

En la documentación de la librería se informa que tenemos que añadir unos “flags” para el compilador, de forma que también añadiremos las siguientes líneas.

```
build_flags =
-D ARDUINO_LMIC_PROJECT_CONFIG_H_SUPPRESS
-D CFG_eu868=1
-D CFG_sx1276_radio=1
```

Con esto ya tenemos definida la placa y la librería “lmic”. Podemos añadir también la velocidad con que queremos que funcione el monitor serial y los parámetros necesarios para que el programa se suba a la placa con el pymcuprog.

```
upload_command = pymcuprog write -t uart -u COM4 -d atmega4808 -v info --erase --filename $SOURCE
```

```
monitor_speed = 115200
```

(En documentos anteriores se han explicado los parámetros)

Ya tenemos cargada la librería “lmic” y el PlatformIO.ini configurado correctamente para la placa y la compilación. Vamos a terminar de cargar librerías antes de empezar con el código.

Tenemos que descargar la librería “RocketScream\_LowPowerAVRZero.h”. Podemos buscarla en PlatformIO escribiendo “RocketScream” o podemos añadir en PlatformIO.ini a la variable lib\_deps la siguiente línea

```
rocketscream/Rocket Scream LowPowerAVRZero@^1.0.0
```

de forma que quede así.

```
lib_deps =
```

```
    mcci-catena/MCCI LoRaWAN LMIC library@^4.1.1
```

```
    rocketscream/Rocket Scream LowPowerAVRZero@^1.0.0
```

Ya tenemos descargadas todas las librerías que necesitamos, ahora empezamos a escribir el código.

Para que el código se lea mejor, vamos a sustituir los números de los pines por el nombre asignado al mismo en la placa. De esta forma en vez de escribir “digitalWrite(8, LOW)” escribiremos “digitalWrite(TPL5010\_DONE, LOW).

Para tener definidos los nombres de los pines incluiremos el archivo Pines.h con un “#include” y que previamente habremos copiado a la carpeta “include” del proyecto.

Añadimos la librerías necesarias y el apartado de los “includes” debe haber quedado así.

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

```
#include <pines.h>
```

```
#include <lmic.h>
```

```
#include <hal/hal.h>
```

```
#include <RocketScream_LowPowerAVRZero.h>
```

Si copiamos el código de Arduino y lo pegamos a continuación, al compilarlo saldrán varios errores.

Esto se debe a que el compilador de Arduino te permite llamar a una función antes de su declaración, pero en PlatformIO no. Simplemente tenemos que ver que funciones son las que nos devuelven error y declararla al principio del programa (solo hay que buscar la función, copiar el encabezado y finalizar con “;” (punto y coma)

En nuestro caso quedaría así.

```
void wakeUp(void);
```

```
void onEvent (ev_t ev);
```

```
void pulsador(void);
```

```
void do_send(osjob_t* j);
```

El último paso antes de compilar el programa es sustituir en los arrays “APPEUI”, “DEVEUI” y “APPKEY” los valores actuales por los que correspondan a vuestro dispositivo. Recordad que APPKEY se informa en formato “msb” y los otros dos en formato “lsb”.