

# Numeric dimmer

- [01 - Install of Visual Studio Code](#)
- [02 - Copy or update repository sources](#)
- [03 - USB code upload](#)
- [04 - Remote code upload](#)
- [05 - Wi-Fi Setup](#)

# 01 - Install of Visual Studio Code

To transfer the code to the microcontroller (ESP8266 or Robotdyn) it is necessary to install [Visual Studio Code](#).

Once installed, you must install the [PlatformIO](#) package which will be used later for all your projects and not only for the Dimmer or the Pv router.

image not found or type unknown



# 02 - Copy or update repository sources

The sources are available on the Github (a code repository web server)

once your Visual Studio is launched, go to your Terminal and type

```
git clone https://github.com/xlyric/PV-discharge-Dimmer-AC-Dimmer-KIT-Robotdyn.git
```

it will then clone the repository on your machine and you can adapt the code to your needs and upload it.

```
PS C:\Users\c_lyr\Documents\PlatformIO\Projects\1> git clone https://github.com/xlyric/PV-
discharge-Dimmer-AC-Dimmer-KIT-Robotdyn.git
Cloning into 'PV-discharge-Dimmer-AC-Dimmer-KIT-Robotdyn'...
remote: Enumerating objects: 179, done.
remote: Counting objects: 100% (179/179), done.
remote: Compressing objects: 100% (136/136), done.
Recote: Total 179 (delta 90), reused 119 (delta 38), pack-reused 0 eceiving objects: 75%
(135/179)
Receiving objects: 100% (179/179), 630.36 KiB | 2.58 MiB/s, done.
Resolving deltas: 100% (90/90), done.
```

you can then go to the directory created during the command

```
Resolving deltas: 100% (90/90), done.
PS C:\Users\c_lyr\Documents\PlatformIO\Projects\1> ls

Directory: C:\Users\c_lyr\Documents\PlatformIO\Projects\1

Mode                LastWriteTime         Length Name
----                -
d---              10/03/2022   16:53                pv-router-esp32
```

In the case of an update, you can update your code again with the following command

```
git pull
```

```
PS C:\Users\c_lyr\Documents\PlatformIO\pv-router-esp32> git pull
Already up to date.
PS C:\Users\c_lyr\Documents\PlatformIO\pv-router-esp32> |
```

## Default configuration

There is nothing more to configure in the dimmer,  
you just have to upload the corresponding version to the dimmer.

Wifi configuration is done in Wifi autoconnect.

He creates a WiFi "dimmer" with a web interface accessible at 192.168.4.1 which allows you to  
configure your personal WiFi.

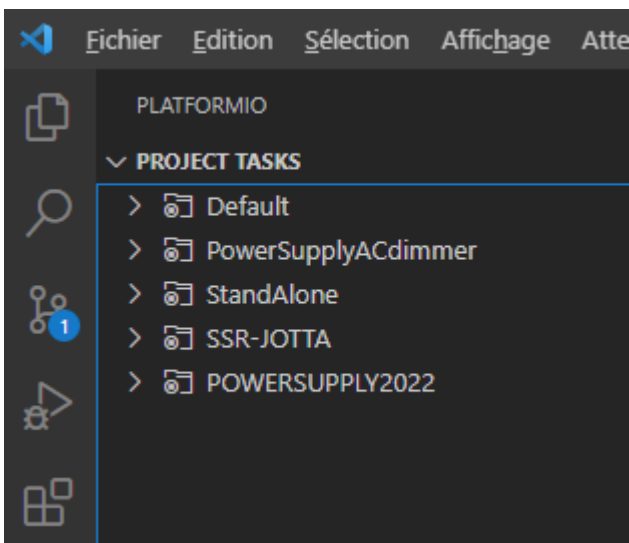
# 03 - USB code upload

Uploading is done with Visual Studio Code (VS) using the PlatformIO tab



During your 1st Upload, you must plug in your ESP, wemos or TTL/USB adapter for programming

There are different versions available depending on what you are using as a dimmer.



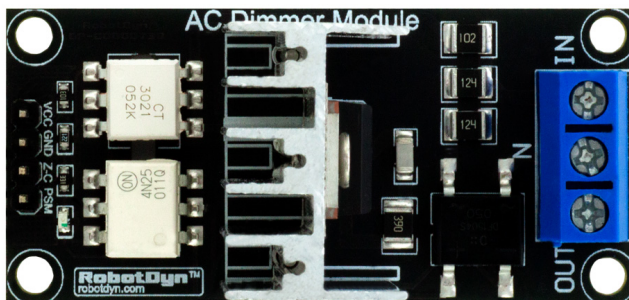
1. **PowerSupplyACdimmer** is used for old version of Robotdyn dimmer,  
this is the easiest version to install, it was compatible with the daughterboards supplied  
with the PV router V1.2 TTGO  
the recommended maximum power is 5A (and not 8A as indicated by the manufacturer)  
( D0 and D1(zc) are used )  
the Dallas probe uses D2

RobotDyn



2. **StandAlone** is used to later add a Robotdyn dimmer to your ESP.  
there are different versions supporting more or less power,  
on the smallest pod, the maximum recommended power is 5A (and not 8A as indicated by the manufacturer)  
on the 16A model, I think you should not exceed 12A -> ~2500W (to be tested)  
( D5 and D6(zc) are used )  
The Dallas probe uses D7

RobotDyn



**28.5mm(1.12")x57.0mm(2.24")**

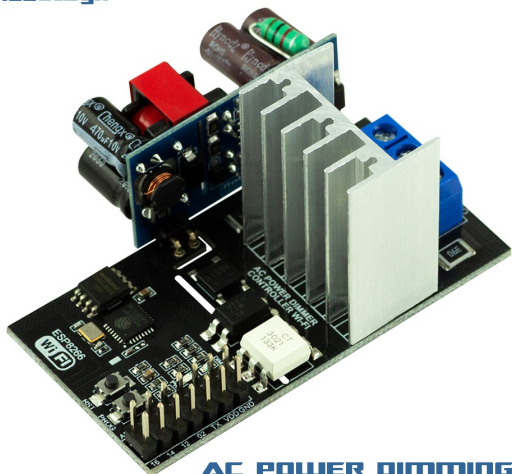
3. SSR-JOTTA is used to live control an SSR The - connects to the GND and D4 to the + of the Jotta  
The Dallas probe uses D2



4. **POWERSUPPLY2022** is for the 2022 version of the Robotdyn dimmer.  
it requires a TTL adapter for the 1st programming.  
the jumper between vdd and 3.3V must be removed during TTL programming  
then put back when connecting the assembly to the 220V

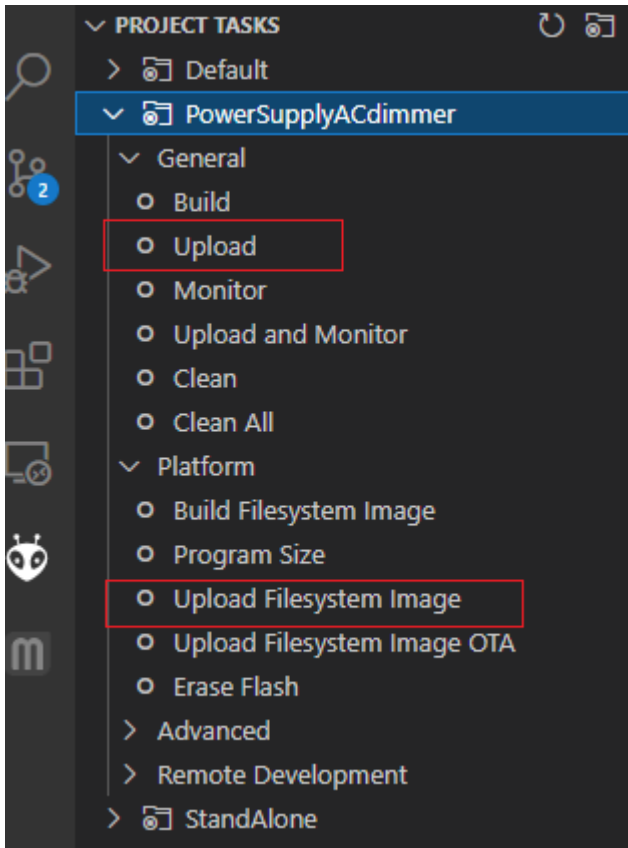
The Dallas probe uses pin 14 - GND at 16 and 3.3v at 12

RobotDyn



**AC POWER DIMMING  
WI-FI SMART CONTROLLER**

Once the version has been chosen, thanks to VS you will load the firmware and the HTML pages of the router into the microcontroller



Then you can directly upload the code remotely with the /update page of the router



# 04 - Remote code upload

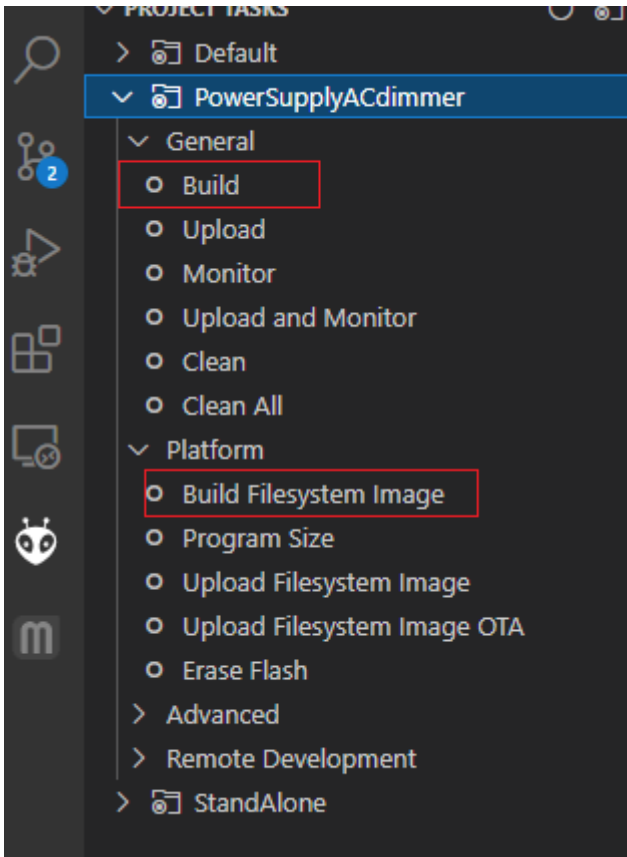
Uploading is done with Visual Studio Code (VS) using the PlatformIO tab



your code being already present on the router, you can now directly generate the binary files to be sent.

In general, only the General Build is to be done.

The Build Filesystem Image is only there to update the HTML pages when functionalities evolve.



once the build is done:

```

> Executing task: C:\Users\c_lyr\.platformio\penv\Scripts\platformio.exe run --environment PowerSupplyACdimmer <

Processing PowerSupplyACdimmer (platform: espressif8266; board: d1_mini; framework: arduino)
-----
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif8266/d1_mini.html
PLATFORM: Espressif 8266 (3.2.0) > WeMos D1 R2 and mini
HARDWARE: ESP8266 80MHz, 80KB RAM, 4MB Flash
PACKAGES:
- framework-arduinoespressif8266 3.30002.0 (3.0.2)
- tool-esptool 1.413.0 (4.13)
- tool-esptoolpy 1.30000.201119 (3.0.0)
- toolchain-xtensa 2.100300.210717 (10.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 44 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <ESP Async WebServer> 1.2.3
|   |-- <ESPAsyncTCP> 1.2.2
|   |-- <Hash> 1.0
|   |-- <ESP8266WiFi> 1.0
|   |-- <ArduinoJson> 6.19.3
|-- <ArduinoJson> 6.19.3
|-- <PubSubClient> 2.8.0
Building in release mode
Retrieving maximum program size .pio\build\PowerSupplyACdimmer\firmware.elf
Checking size .pio\build\PowerSupplyACdimmer\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [===== ] 55.4% (used 45348 bytes from 81920 bytes)
Flash: [===== ] 62.1% (used 648289 bytes from 1044464 bytes)
===== [SUCCESS] Took 2.32 seconds =====

Environment      Status      Duration
-----
PowerSupplyACdimmer  SUCCESS    00:00:02.325
===== 1 succeeded in 00:00:02.325 =====

```

it shows where the firmware is.

all you have to do is connect with the internet browser on your pv router and go to the /update page



☒ Firmware ☐ Filesystem

Choisir un fichier

Aucun fichier choisi

66F23A08 - ESP32

and upload the firmware

## Case of a Filesystem update

In the case of updating the Filesystem (HTML file), it's the same procedure, you just have to take the Filesystem binary and select Filesystem.

# 05 - Wi-Fi Setup

The dimmer code uses the Wifi autoconnect library

at the 1st firmware upload, it will create a "dimmer" wifi

once the user is connected to this Wifi, the browser will redirect to the page <http://192.168.4.1>

to ask you to configure your Wifi

Sign in to network  
192.168.4.1

<u>MEO-WiFi</u>	100%
<u>MEO-620B4B</u>	94%
<u>MEO-AB4E8B</u>	16%

SSID

password

save

Scan

it will memorize your Wi-Fi connection confirmations even after a firmware update.

For more details: <https://www.raspberryme.com/wifimanager-avec-esp8266-connexion-automatique-parametre-personnalise-et-gestion-de-votre-ssid-et-mot-de-passe/>