PUESTA EN MARCHA DE RELES WIFI (SSR o mecánicos) con PVControl+

1 INSTALACION SW

Actualmente para poder utilizar relés vía WiFI existen dos posibilidades en PVControl+:

- Utilizar Tasmota
- Utilizar un SW desarrollado específicamente para PVControl+ en Micropython

Esta parte del manual de instalación SW cubre el uso de TASMOTA para su uso con el control de excedentes de PVControl+, no trata posible integración de sensores desde Tasmota a PVControl+ aunque se ha editado un manual específico para Tasmota que es posible que se integre en versiones posteriores.

1.1 OPCION con TASMOTA

TASMOTA es una plataforma bastante activa que permite añadir mucha funcionalidad (integrar sensores, pagina web, etc), por este motivo en PVControl+ se puede usar TASMOTA en los ESP32/ESP01 que se quieran controlar por Wifi

1.1.1 Instalación TASMOTA

TASMOTA realmente ha simplificado mucho el proceso de su instalación

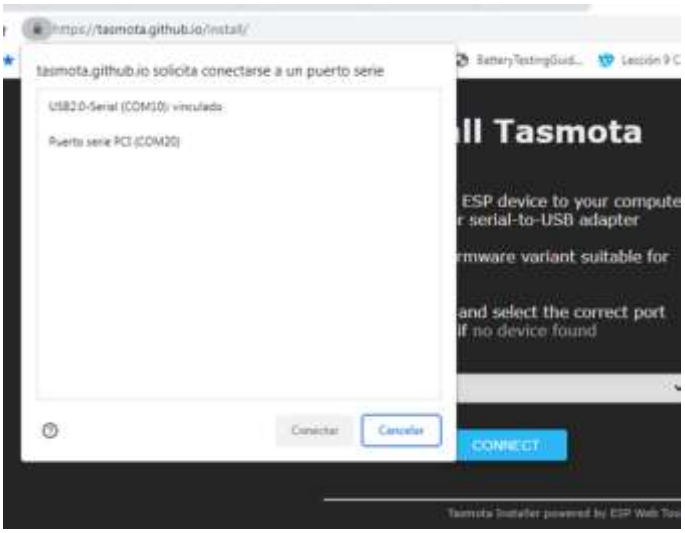
Seguidamente se indican los pasos a seguir para la configuración de un ESP32/ESP01 (NodeMCU) para su uso en PVControl+

A) Carga del Firmware TASMOTA

Conectamos el microntrolador ESP32/ESP01 a un puerto USB de un PC que tenga acceso a internet y tecleamos:

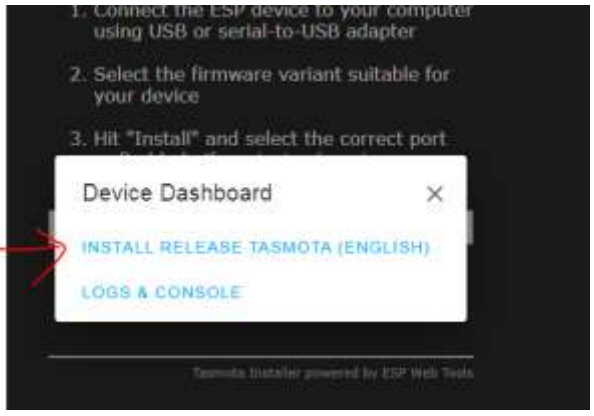
https://tasmota.github.io/install/

Pulsamos CONNECT



Seleccionamos el puerto COM donde reconoce el PC al micro y conectamos

INSTALL RELEASE…



Marcamos opción de

Erase device….

NEXT…INSTALL

Empezara a instalar TASMOTA

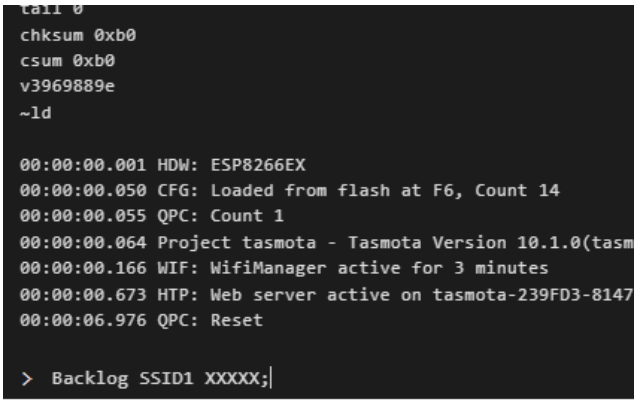
Tras un rato la instalación se habrá completado

Pulsamos NEXT

Ahora se configura el Wifi, MQTT, GPIO etc

Esta configuración se puede realizar desde la Web que crea Tasmota, pero es más rápido hacerlo por consola

Desde la consola podemos darle los comandos necesarios para su configuración



Por ejemplo con Backlog podemos mandar varios comandos separados por ;

Para la configuración completa de un SSR de angulo de fase para PVControl+ el comando sería (cambiar lo marcado en amarillo según su wifi, etc

- **Backlog SSID1 XXXXX; Password1 YYYYYY; MqttHost IP_Raspberry; Topic PVControl/Reles/61; MqttUser rpi; MqttPassword fv; Module 18; SetOption15 0**

- Para configurar el pin GPIO mandaremos el siguiente comando:
  - **GPIO12 416** …… para configurar como PWM el GPIO12 del ESP32
  - **GPIO04 416** ….. para configurar como PWM el GPIO 04 del ESP01/NodeMCU

Si se quiere se puede poner la IP fija con el comando….. IPAddress1 192.168.0.61

También se puede cambiar el nombre con ….. DeviceName Termo1

La frecuencia PWM por defecto de TASMOTA es 977Hz, si se quiere disminuir algo el parpadeo en SSR por semiciclos se puede cambiar a 73Hz con el comando …. PWMFrequency 73

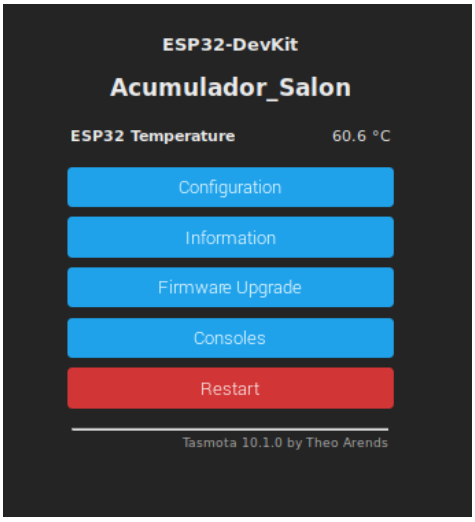
Para usar SSR por semiciclos se produce algo menos de parpadeo si se utiliza el programa en micrpython según el apartado siguiente

Una vez se reinicia el ESP32/ESP01 ya se puede acceder a la web que crea en la dirección IP que le haya asignado el router o la que se haya puesto en la configuración

Desde la web se podrán cambiar básicamente lo mismo que se puede realizar por comandos

Asi, por ejemplo desde la web se pueden configurar parámetros (nombre, puerto GPIO, etc)

Recomiendo manejar un poco dicha web para familiarizarse con las posibilidades que ofrece



Para ver si funcionamiento desde PVControl+, si por ejemplo le hemos asignado el topic PVControl/Reles/61 tendremos que dar de alta el rele 611 en PVControl+ y ya se podría controlar

Para el uso de TASMOTA los códigos de reles son:

- **5XX : Relés tipo ON/OFF (sonoff, etc)** ….configurar el GPIO como Rele
- **6XX Relés SSR regulación por ángulo de fase o semiciclos**… .configurar el GPIO como PWM

1.2 OPCION con Micropython

Lo primero es tener instalado el entorno de programacion Thonny u otro entorno (uPyCraft, ESPplorer,…)

Thonny para Python **ya viene preinstalado en la imagen de la Rpi**, si se quiere hacer desde un PC Windows u otro entorno debemos instalarlo… https://thonny.org/

1.2.1 Descarga de firmware (ya vemos si se pone en sourceforge para descargar)

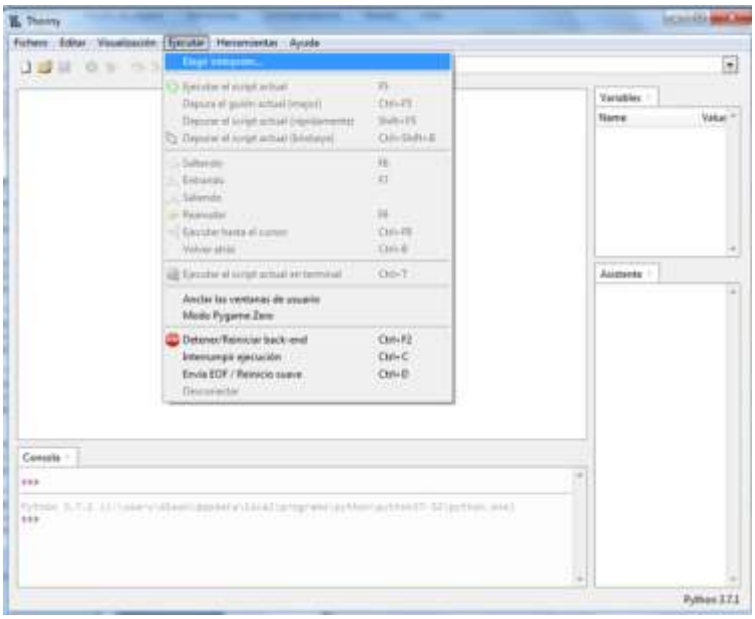
Si es la primera vez que vamos a instalar el firmware de micropython, debemos primero tener descargado el archivo “bin” con el firmware de micropython de nuestro micro

https://micropython.org/download/esp8266/

https://micropython.org/download/esp32/

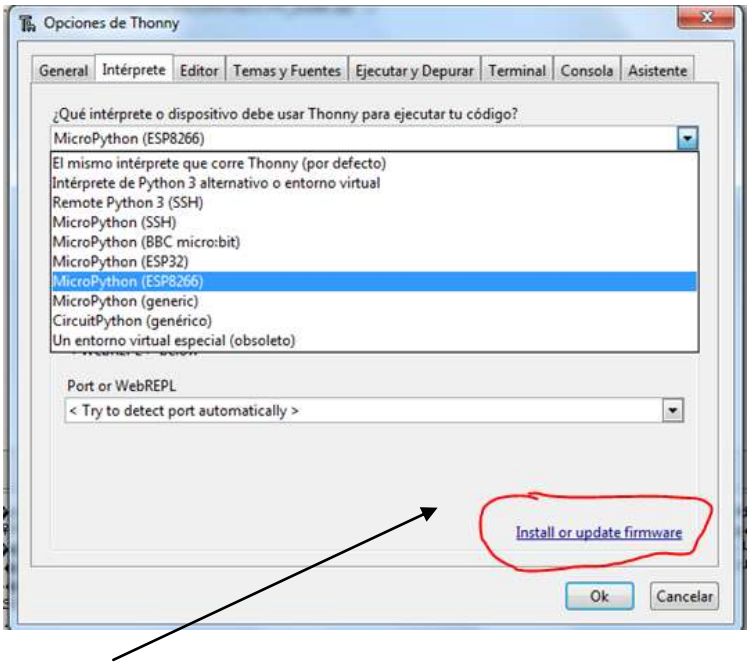
1.2.2 Conexión con el ESP32/ESP8266 (NodeMCU o ESP01)

Conectamos el micro al USB , y desde la pantalla principal de Thonny seleccionamos



- **Ejecutar – Elegir Interprete**

- **Seleccionamos el micro que tengamos (ESP32 o ESP8266)**



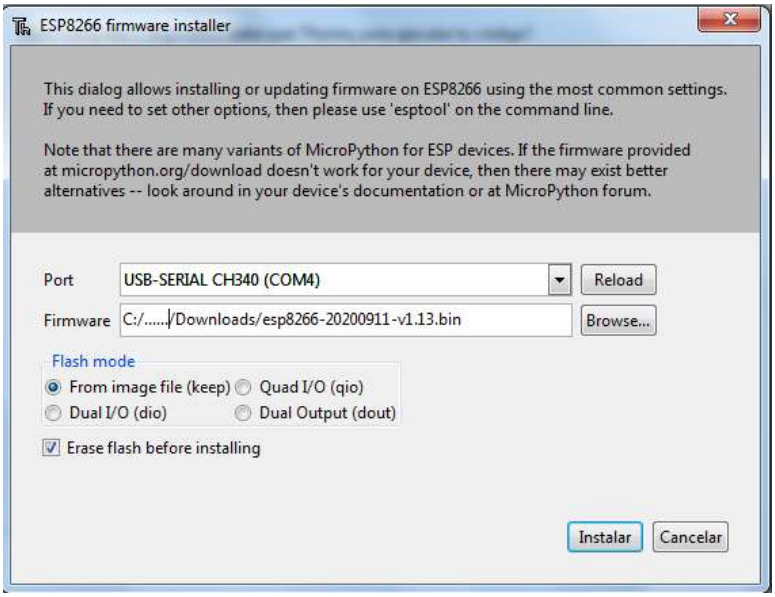
- **Si es la primera vez y queremos cargarle el firmware de micropyhon pulsaremos en Install or update firmware**

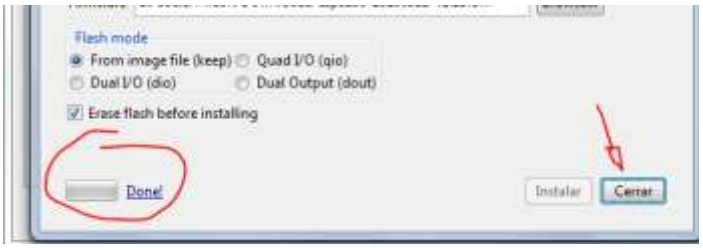
- **Se introduce el puerto donde esta el ESP32/ESP8266 (sale un desplegable mostrado los puertos asignados)**

- **Se indica la ubicación donde esta el fichero ”bin” que nos hemos descargado con el firmware de micropython**

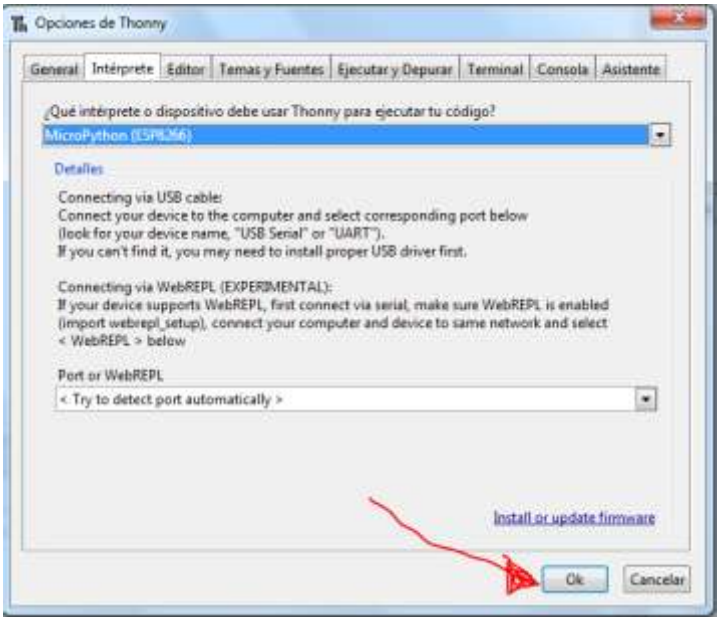
- **Flash mode según se muestra**

- **Se pulsa Instalar y comienza la grabación del firmware**



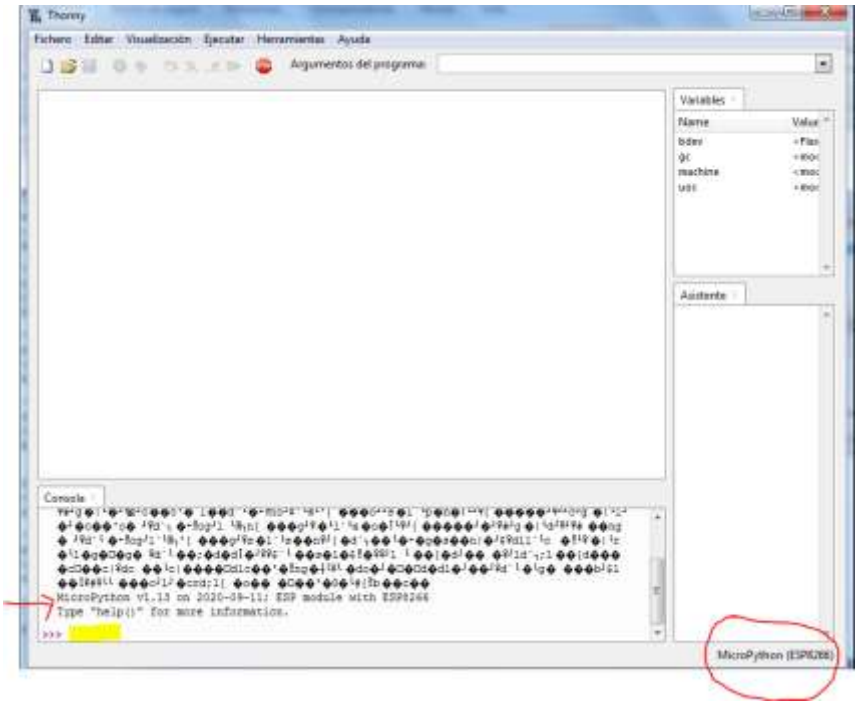


- **Una vez cargado el firmware ya podemos acceder al micro pulsando OK)**

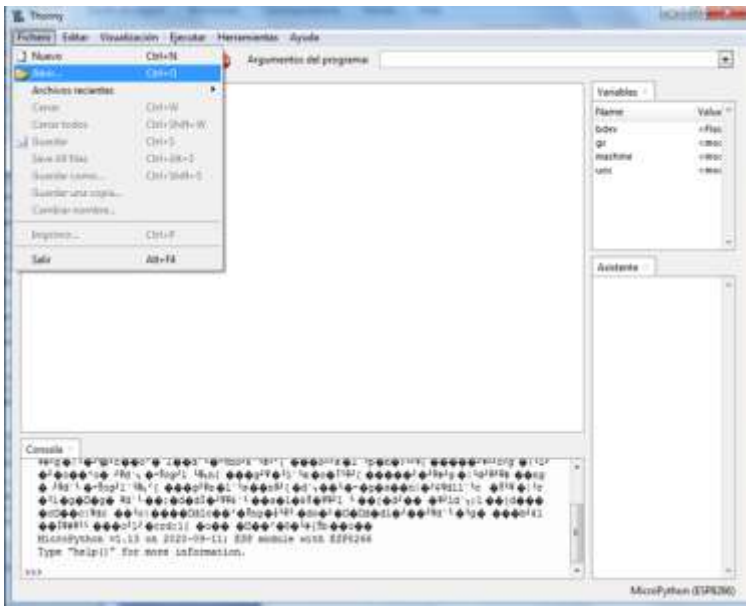


1.2.3 Carga de los programas de PVControl+

Aquí ya se ve que Thonny se ha conectado al micro y por tanto ya se puede cargar, ejecutar etc los programas



Para cargar un programa en Thonny simplemente seleccionamos Fichero/abrir



En este caso nos pide que digamos desde donde queremos cargar el fichero

Seleccionamos “Este Computador” y buscamos el fichero que queramos cargar:

Excedentes_XXXX.py

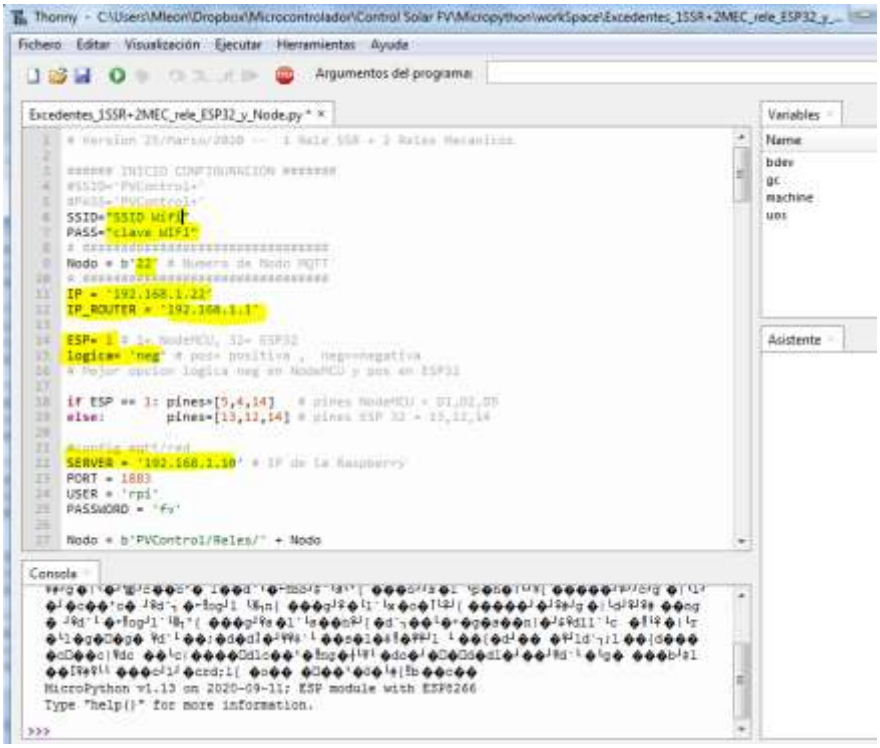
Se debe actualizar lo marcado de acuerdo a nuestra wifi y el numero de rele que se quiere asignar

Si se usan SSR de regulación por semiciclos (SC), se recomienda:

- **Lógica negativa para ESP01**
- **Lógica positiva para ESP32**

Para SSR por ángulo de fase siempre se usara lógica positiva

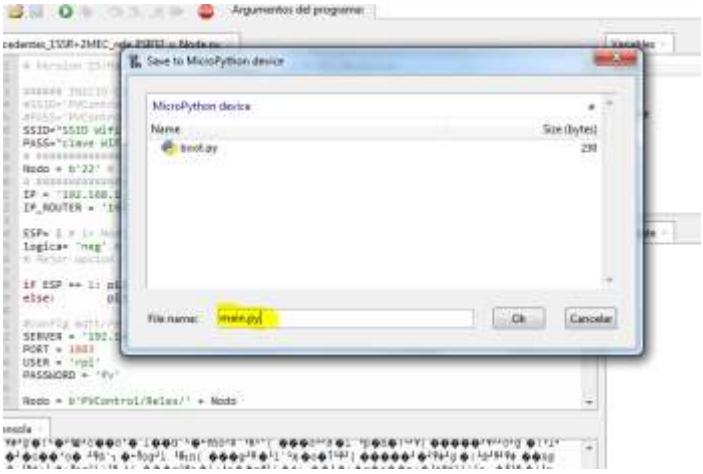
Una vez modificado simplemente hay que guardarlo en el micro con la opción **de "Fichero/guardar como" y eligiendo destino el Microcontrolador**



Podemos poner cualquier nombre al archivo, hay dos opciones principalmente:

A) Si nombramos el fichero como **main.py** en el microcontrolador, ya estaría todo, dado que el micro al arrancar busca un fichero llamado main.py y lo ejecuta

B) Le ponemos otro nombre a nuestro gusto, pero después debemos crear el archivo main.py que direccione a ese archivo

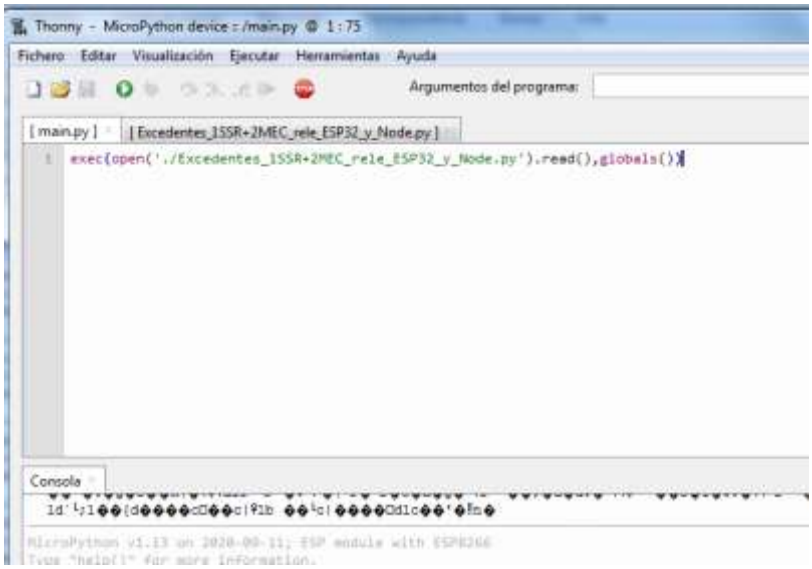


Yo prefiero la opción B) dado que así podemos tener guardados varios archivos en el micro con distintos nombres y poder ejecutarlos a nuestro antojo poniendo en main.py el que queramos que se ejecute al reiniciarse el micro

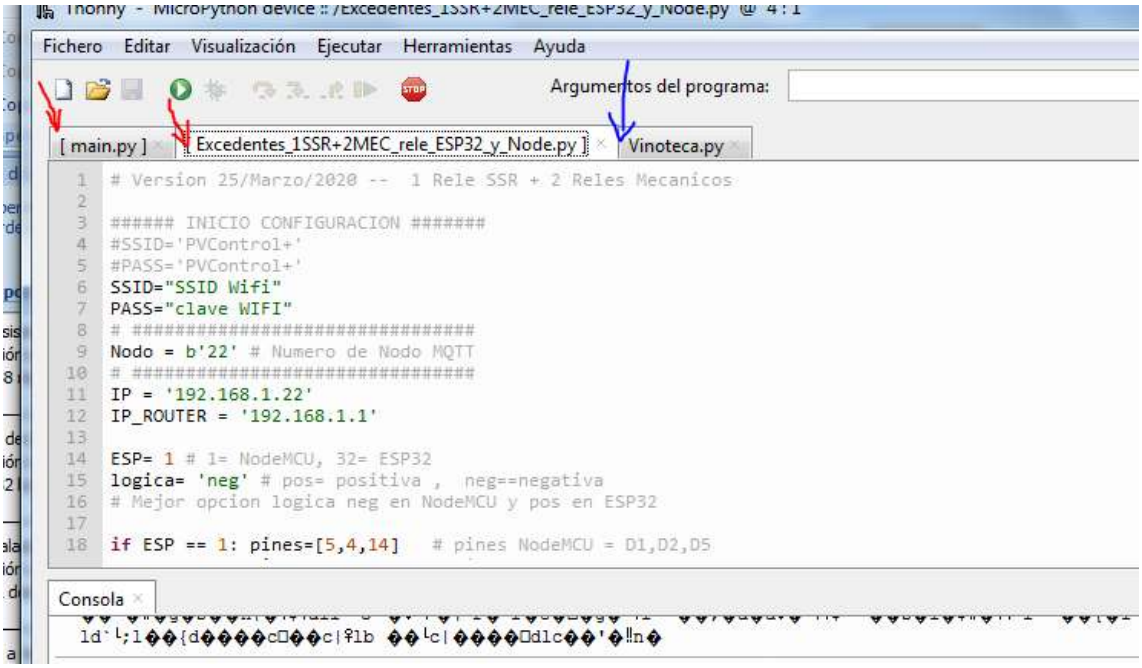
El archivo main.py a crear simplemente debe tener esta línea:

exec(open('./Excedentes_XXXX.py').read(),globals())

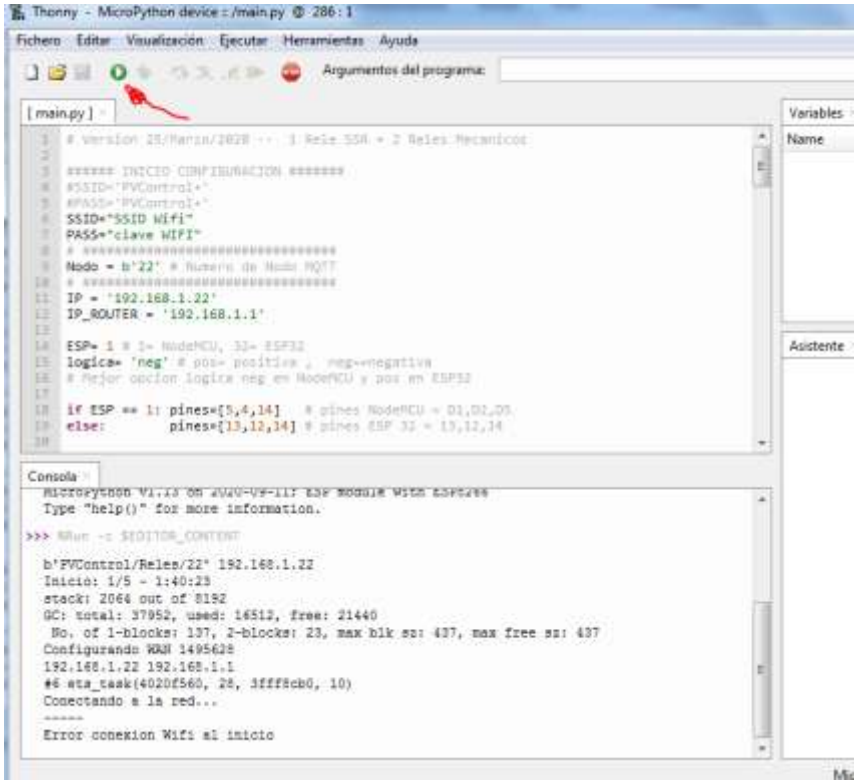
cambiando el nombre del archivo por el que hayamos puesto



Thonny es bastante flexible y te permite editar archivos tanto en el micro como el un PC, por ejemplo aqui vemos que hay cargados 3 archivos… si el nombre esta entre corchetes[] significa que esta ubicado en el micro



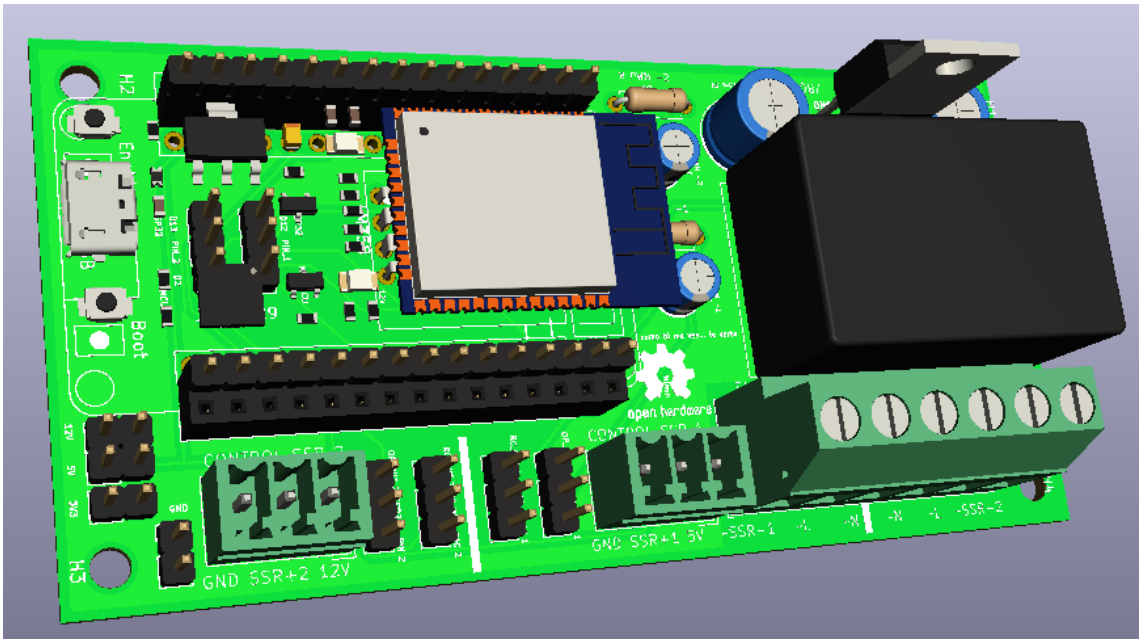
Ya podemos ejecutar desde Thonny el archivo guardado en el micro pulsando el boton “Ejecutar”



2 Instalación HW

Una vez que ya tenemos cargado el SW en el Micro, toca instalarlo donde queramos que actué

Aunque no es necesario, se ha diseñado una PCB para facilitar la instalación HW **con capacidad hasta dos reles SSR**



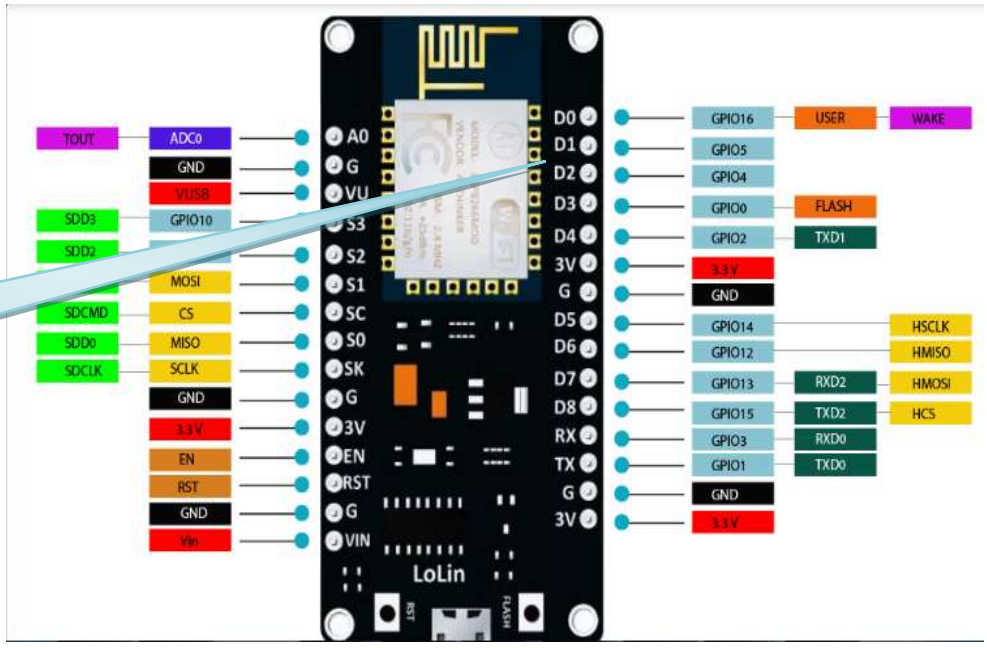
2.1 Microcontrolador a usar

En PVControl+ se han integrado dos tipos de microcontroladores (ESP01 y ESP32)

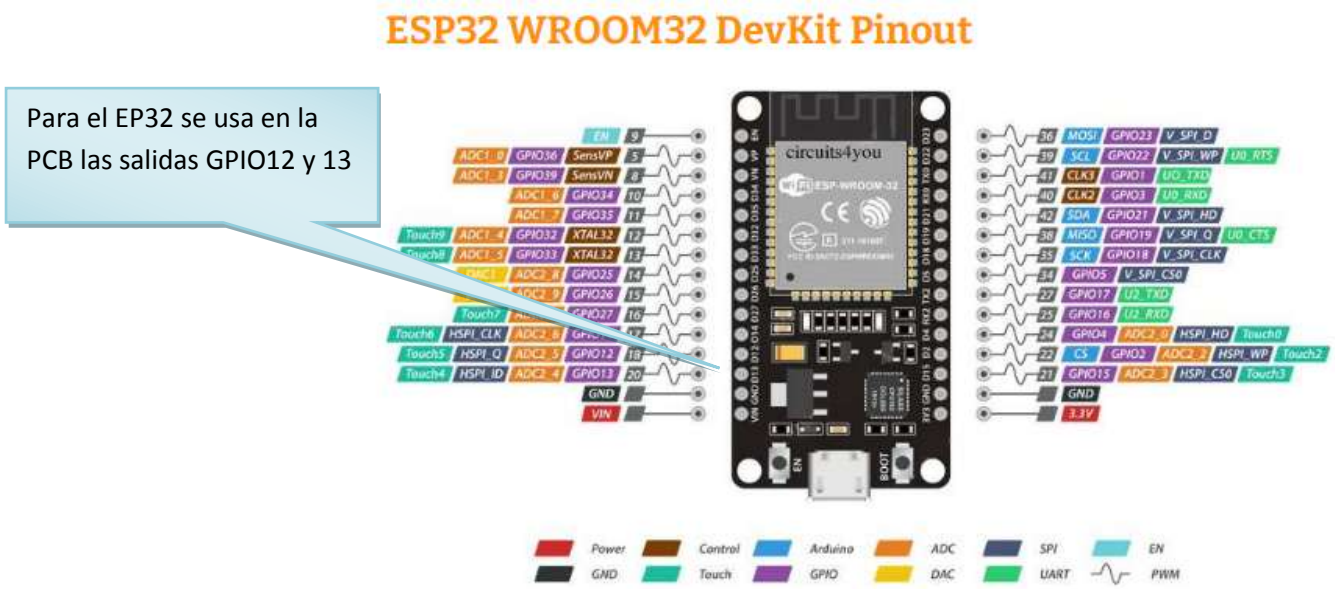
Para usar la PCB se tiene que usar las siguientes versiones **para que el pinout sea correcto**:

- **ESP01 en su versión NodeMCU V3**

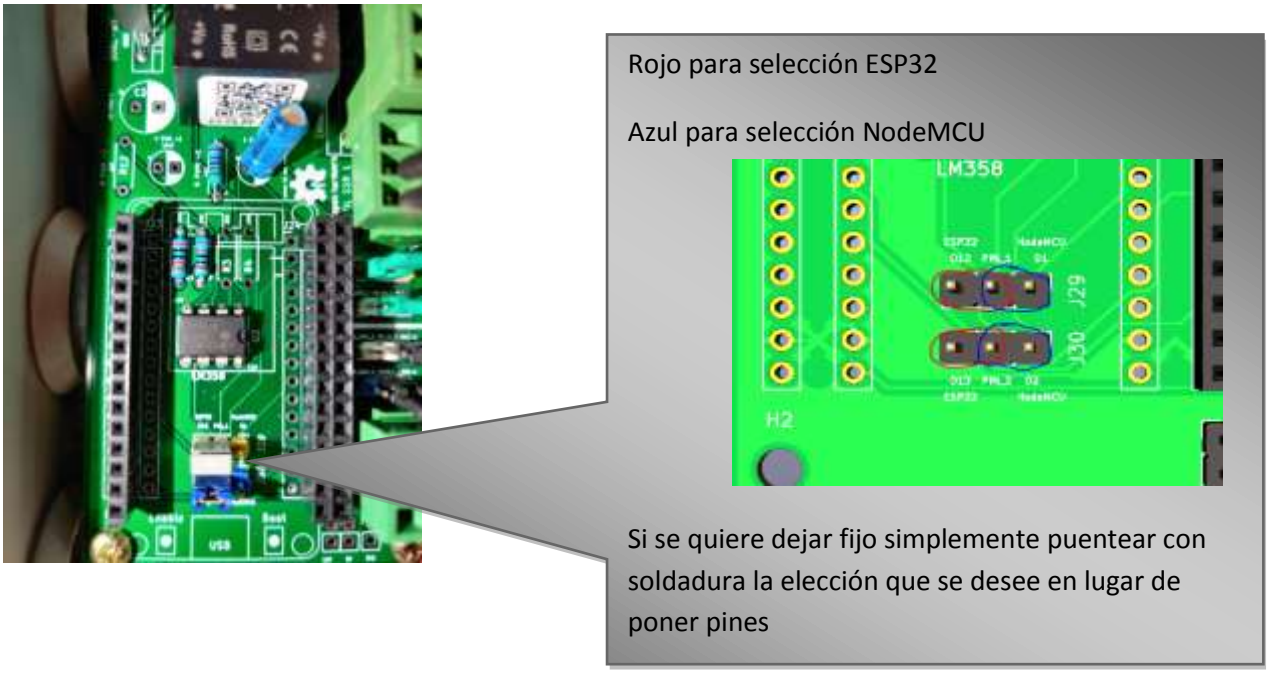
Para el NodeMCU se usa en la PCB las salidas GPIO 4 y 5 (D2 y D1)



- **ESP32 en su versión WROOM32 DevKit**



La selección del tipo de micro se realiza utilizando estos puentes



2.2 Modo de funcionamiento del SSR

En PVControl+ se pueden usar 2 tipos de funcionamiento para regular la potencia

El modo de fucionamiento se define el el programa de micropython que se carga en el mixro

:

- **Regulación con semiciclos AC completos**
  Se usa un SSR que tiene un control de disparo por detección de paso por cero como el que se muestra

  Consiste en regular el numero de semiciclos AC en ON y en OFF, asi por ejemplo:

  - Para un 50% se tendría 1 semiciclo ON y 1 semiciclo OFF
  - Para un 25% … 1 semiciclo ON y 3 semiciclos OFF
  - Para un 5% … 1 semiciclo ON y 19 semiciclos OFF

  Dado que cada semiciclo AC dura 10ms (para AC de 50hz) esta regulación produce un parpadeo visible

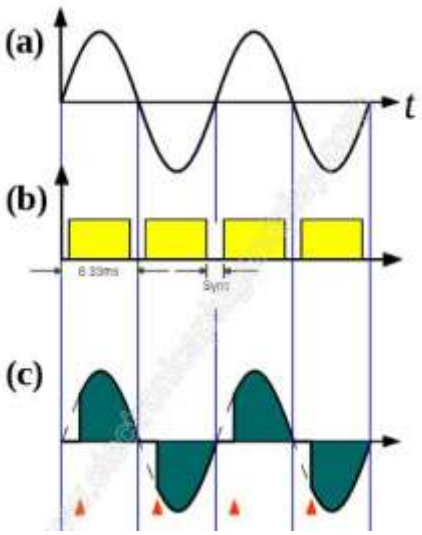
  Estos SSR son baratos y su uso es sencillo dado que se puede utilizar directamente la salida GPIO del microcontrolador

- **Regulación por ángulo de fase**

  Se usa un SSR con disparo por ángulo de fase lo que permite una regulación con menor parpadeo visible complicando un poco el circuito a poner

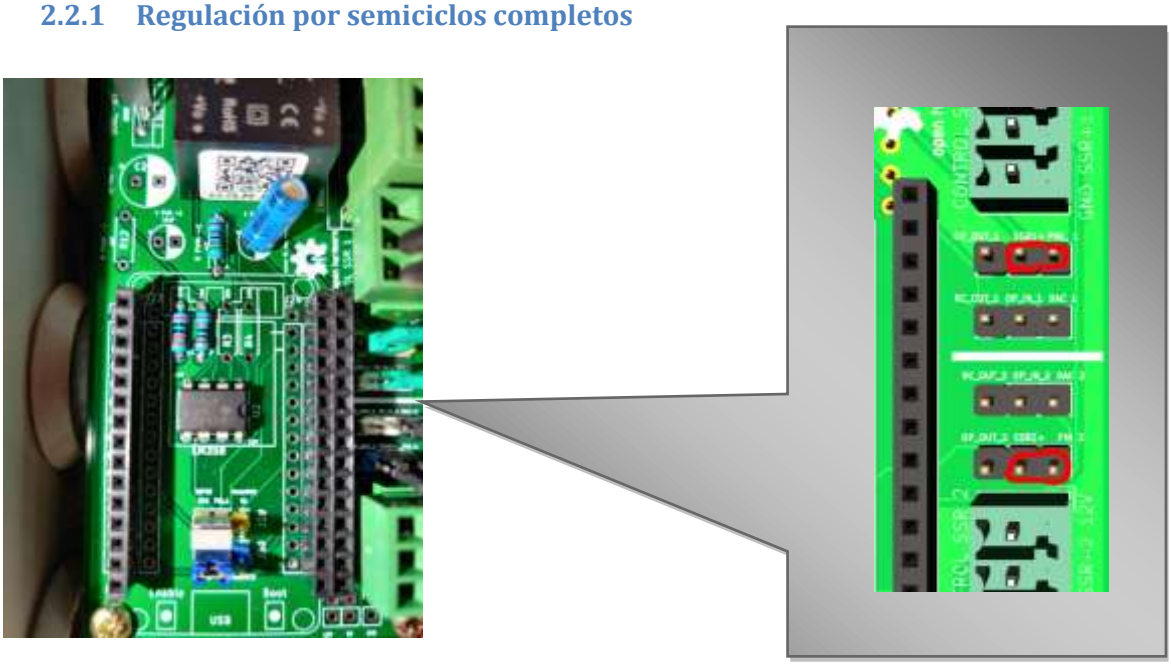
  En este caso el SSR conduce durante un “trozo” de casa semiciclo AC

  En este caso se usa un SSR que suele tener una entrada entre 0-10V por lo que para usarlo con un ESP32/NodeMCU hay que añadir un amplificador operacional alimentado a 12V que nos permita amplificar la señal



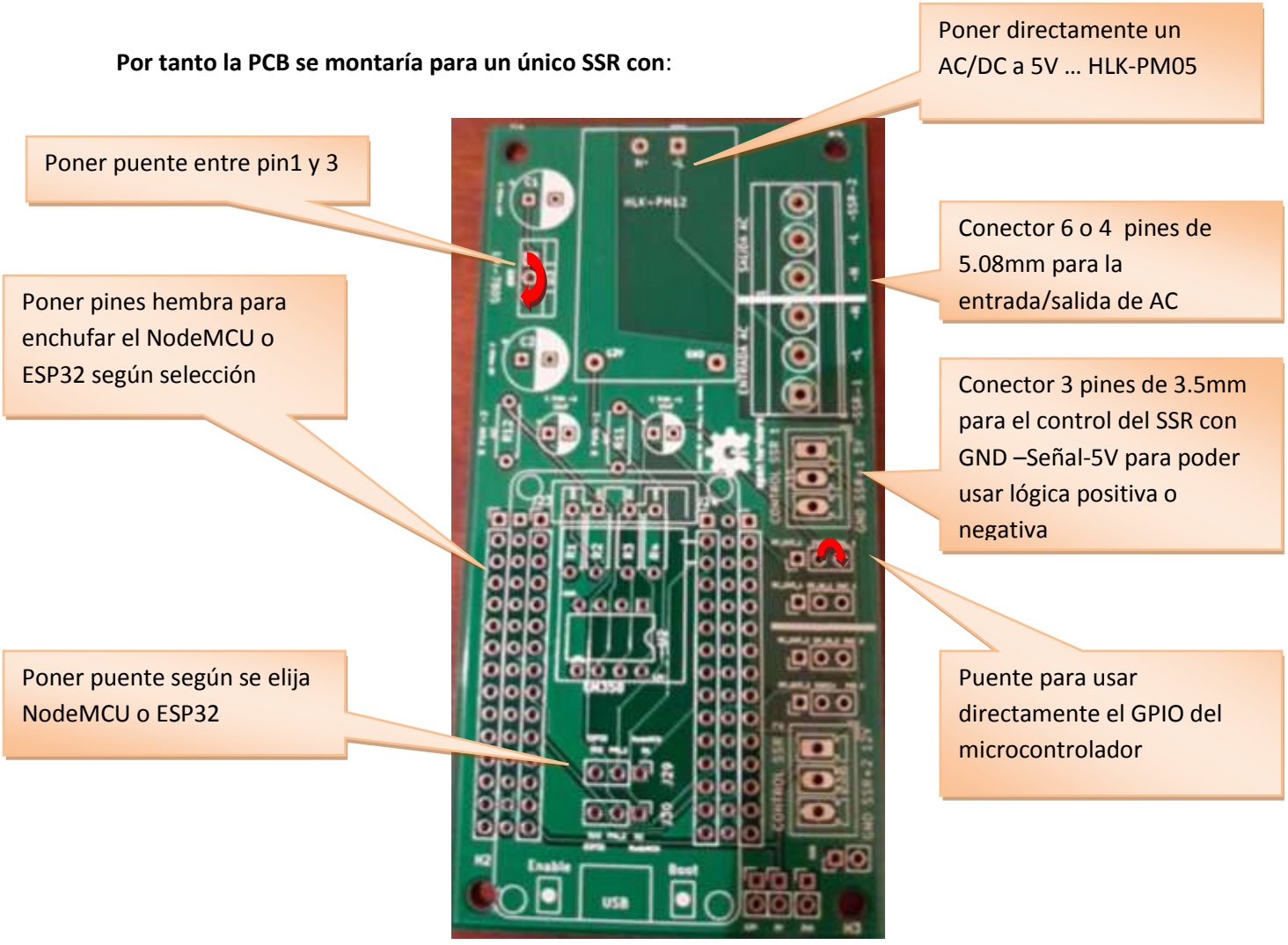
Podemos configurar la PCB con estos modos de funcionamiento

2.2.1 Regulación por semiciclos completos



Este es el caso de configuración mas simple dado que no necesita el uso de Amplificador Operacional, filtro RC , fuente doble 12V/5V…

Por tanto la PCB se montaría para un único SSR con:



2.2.2 Regulación por ángulo de fase

En este caso tenemos que usar como minimo:

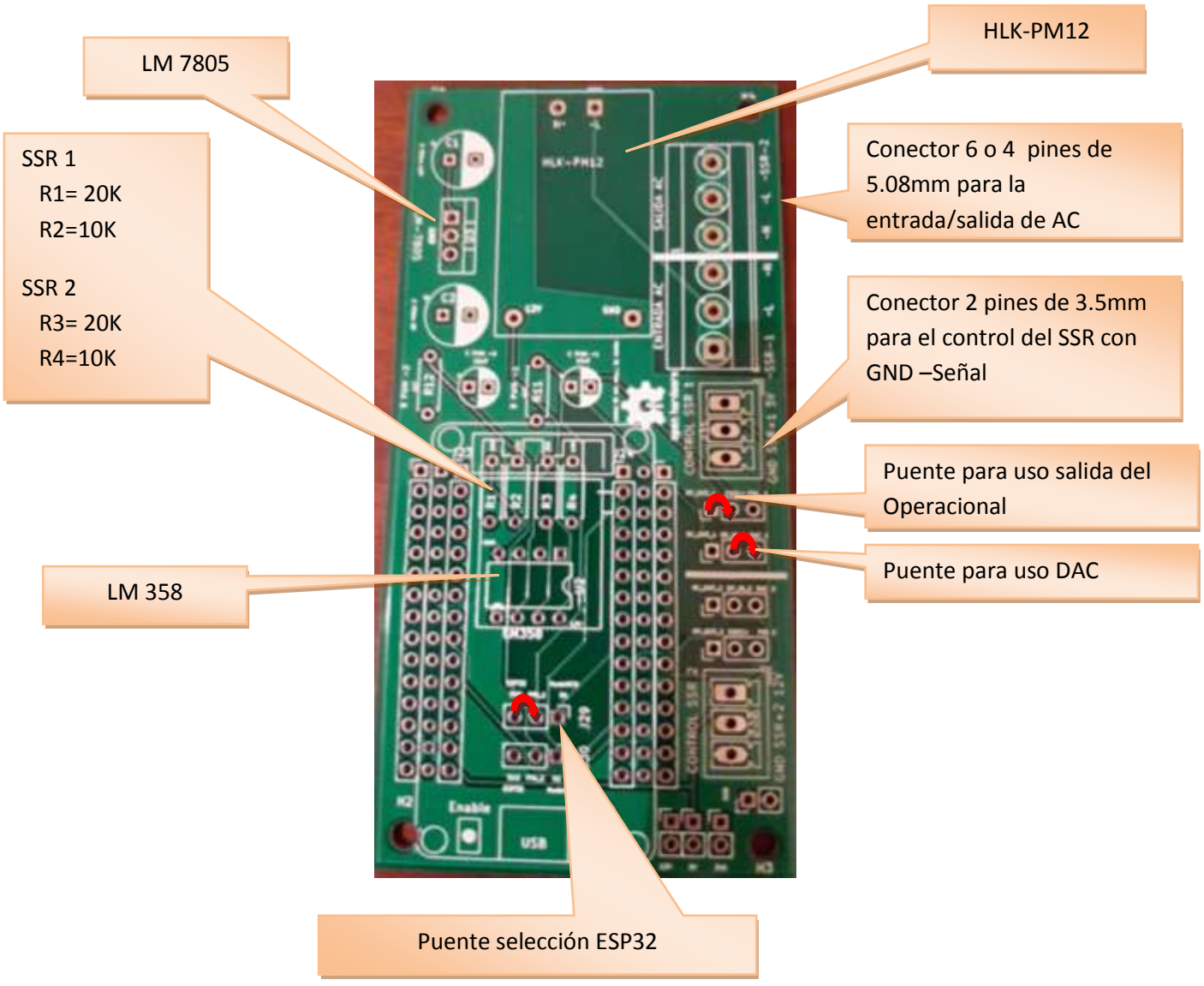
- Amplificador Operacional LM358
- Una fuente de 12V…HLK-PM12
- Un regulador 7805
- 1 resistencia de 10K por cada SSR
- 1 resistencia de 20K por cada SSR

En esta forma de regulación podemos elegir entre dos posible formas de funcionamiento

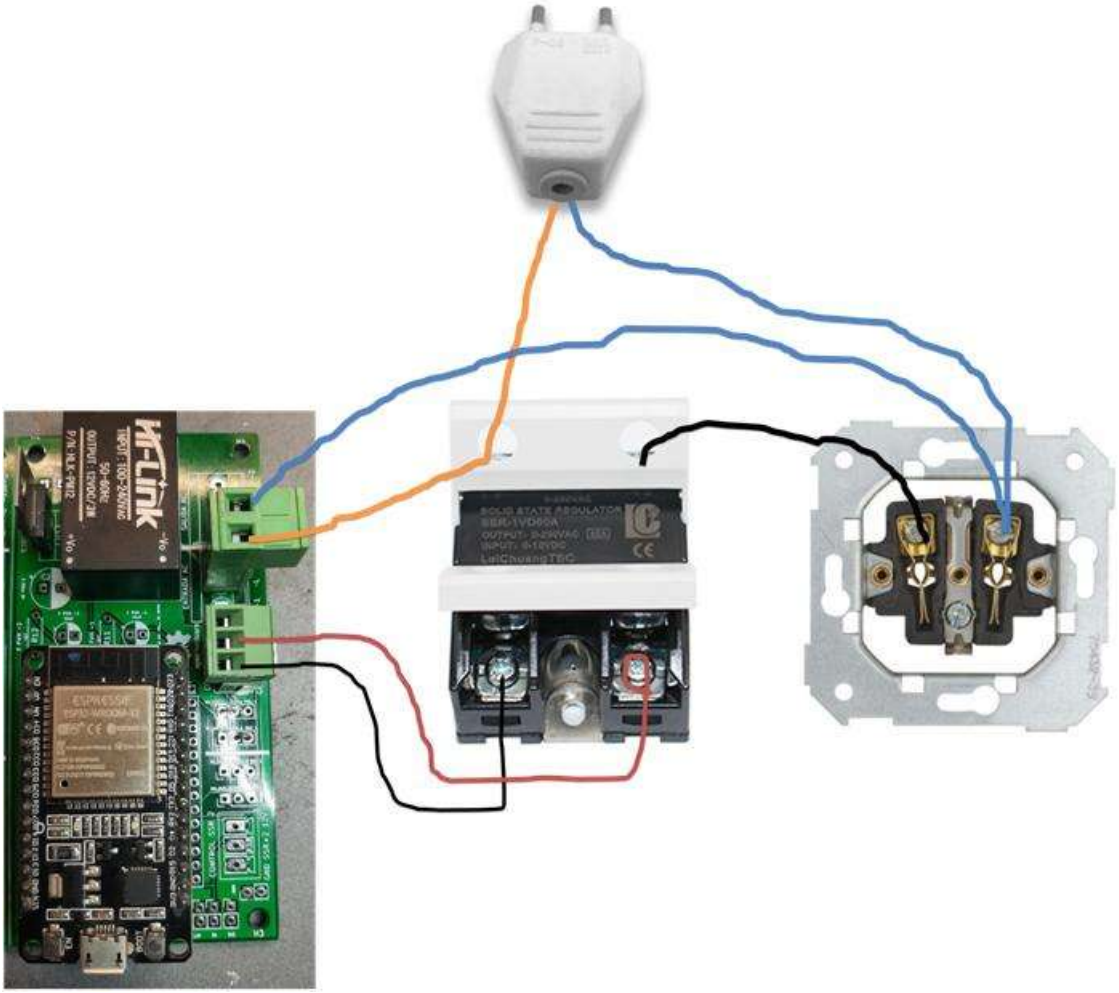
- Usar salidas GPIO mas una resistencia de 4,7K y Condensador de 10uF por cada SSR (para NodeMCU o ESP32)

- Usar salidas DAC (solo posible en ESP32)

2.2.2.1 Salidas DAC del ESP32- Ejemplo 1 SSR AF



ESQUEMA DE MONTAJE - 1 SSR ANGULO DE FASE



2.2.2.2 Salidas PWM del NodeMCU o ESP32 – 2 salidas SSR – Angulo de fase