

**PUESTA EN MARCHA DE
PVControl+ DESDE
IMAGEN microSD
MAY/22**





Contenido

1.	Instalación HW	4
1.1.	Introducción	4
1.2.	Diagramas de Instalación	6
	Instalación con Híbrido tipo Axpert	6
	Instalación con Híbrido tipo Axpert con shunt.....	7
	Instalación Genérica con Regulador e Inversor.....	8
1.3.	Material HW necesario.....	9
	Raspberry	9
	Placa Base (en caso de no capturar desde los equipos).....	9
	PCF8574P.....	10
	ADS1115	11
	Divisores de tensión (resistencias).....	12
	Sensor de Temperatura DS18B20	16
	Conector a la Raspberry	17
1.4.	Montaje en Protoboard	20
2.	Instalación SW	21
2.1.	Acceso a la Raspberry.....	21
	Monitor/Teclado/Raton	21
	Acceso por VNC.....	21
2.2.	Expandir la tarjeta SD	23
2.3.	WIFI	24
2.4.	IP.....	24
2.5.	Actualizar a la última versión de PVControl+	26
2.6.	Parámetros Instalación PVControl+	27
	Sección principal.....	30
	SECCION PARAMETRIZACION SEGUN EQUIPAMIENTO INSTALADO	35
	Pantallas OLED (OPCIONAL)	41
	SECCION PARAMETRIZACION SERVICIOS ADICIONALES.....	42
	TELEGRAM (OPCIONAL)	42
	CAMARA DE VIGILANCIA (OPCIONAL).....	44
	PVOUTPUT (OPCIONAL)	46



2.7.	ADAPTACION de la WEB	50
2.8.	Tabla parámetros en la Base de Datos	51
3.	Test y Resolución de Problemas	56
4.	Integración con Home Assistant	59
5.	Integración Tasmota.....	61

1.Instalación HW

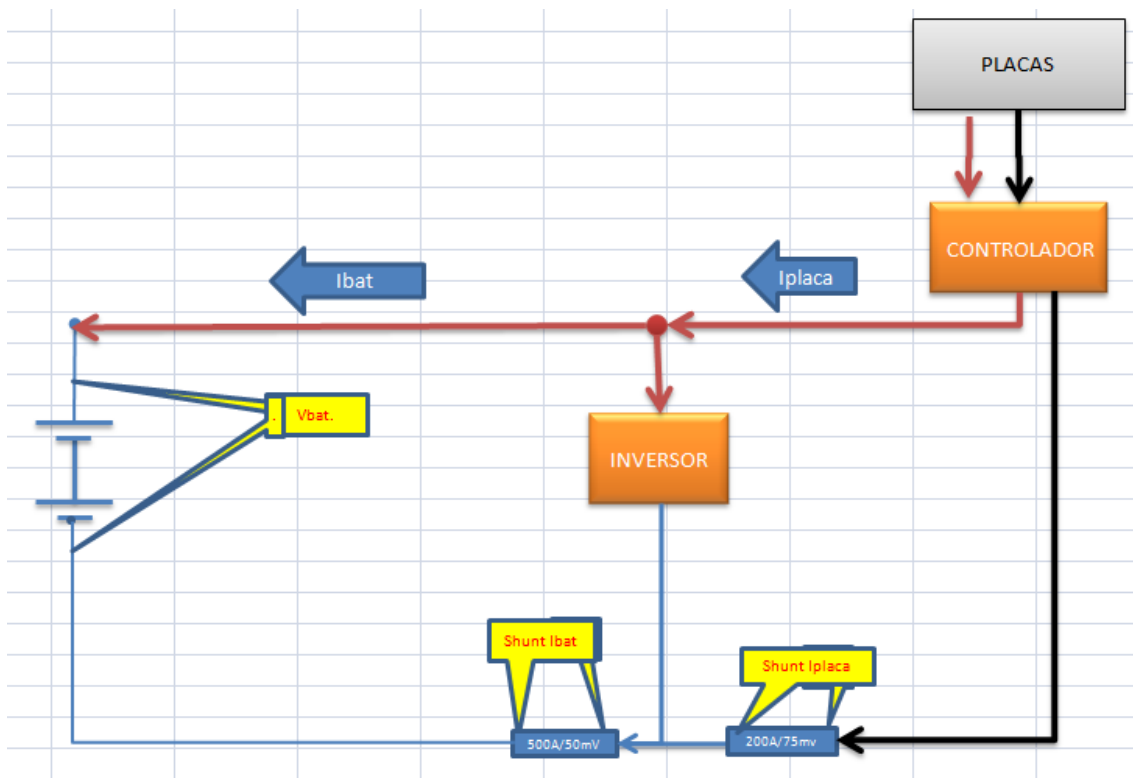
1.1.Introducción

PVControl+ se ha diseñado con el objetivo de servir a múltiples tipos de instalaciones FV, por lo que permite monitorizar y controlar diversos equipamientos FV con o sin batería (Victron, Goodwe, Fronius, Eastron, SMA, Huawei, Axpert...)

No obstante, lo más genérico es considerar una instalación simple estándar de FV en donde se tienen Placas, Regulador e Inversor

Por tanto hay configuraciones de PVControl+ que están ligadas a algún equipamiento específico (Híbridos Axpert. Monitores baterías Victron, BMS Daly, etc), que permiten minimizar el HW necesario para la monitorización mediante la conexión a dichos dispositivos y leer datos de los mismos (Voltaje batería, Watios consumidos, Voltaje celdas, etc), la idea inicial es poder leer los datos relevantes de una instalación FV independientemente del equipamiento que se tenga.

Un esquema básico de la instalación FV sería el siguiente:



Como se observa la idea básica es poder medir:



- Voltaje de batería: V_{bat}
- Intensidad que entra o sale de la batería; I_{bat}
- Intensidad que sale del regulador : I_{placa}

Adicionalmente se ha pensado en poder medir otros parámetros relevantes

- Temperatura de las Baterías
- Voltaje de las placas: V_{placa}
- Salida auxiliar del regulador: V_{aux} (disponible en muchos reguladores)
- Voltaje de cada celda de la Batería: V_{celda1} , V_{celda2} , ...
- Otras magnitudes que con un sensor se pueda pasar el valor a voltaje

Es importante señalar que NO hace falta instalar todas las posibilidades de PVControl+, solo lo que se necesite.

1.2. Diagramas de Instalación

Se dibujan a continuación unos diagramas “tipo” de distintas instalaciones posibles dependiendo de la configuración que cada uno tenga

No se especifican todas las posibilidades (instalaciones sin batería, etc) que permite PVControl+ y que en ese caso suele ser simplemente conectar el cable de comunicaciones entre en equipo y la Rpi

Tampoco se especifican todas las posibilidades de la PCB (medición de cada celda, control de relés, etc) pero da una idea básica de la instalación necesaria

Por simplificar el dibujo, NO se incluyen en los diagramas las protecciones (Fusibles, etc) que se deben poner en la instalación y que, normalmente ya estarán puestas cuando se instala PVControl+

Instalación con Híbrido tipo Axpert

Esta es la instalación más simple dado que los datos se capturan desde el propio Híbrido por lo que solo hace falta un cable USB entre la RPi y el Híbrido

No es estrictamente necesario alimentar la RPi desde las baterías con un convertidor DC-DC, se puede utilizar un cargador AC/DC estándar, aunque por seguridad es la mejor opción



Instalación con Híbrido tipo Axpert con shunt

Si se quiere tener una medida mas exacta de la intensidad que se carga o descarga de la batería (Ibat) y por tanto tener el SOC con mayor precisión se puede incorporar la PCB de PVControl+ junto con un shunt para medir esta intensidad

La PCB, dependiendo de la configuración que se ponga, permite además controlar hasta 32 Reles, sensores de temperatura y poder medir el voltaje de hasta 32 celdas

Igualmente recomiendo alimentar la RPi desde las baterías con un convertidor DC-DC ,



Instalación Genérica con Regulador e Inversor

Es la instalación mas genérica en donde los valores de la FV (V_{bat} , I_{bat} , I_{placa} , etc) se capturan desde la PCB por tanto es independiente del equipamiento instalado

Como hemos comentado la PCB, dependiendo de la configuración que se ponga, permite controlar hasta 32 Reles, sensores de temperatura y poder medir el voltaje de hasta 32 celdas

Igualmente recomiendo (yo diría que en este caso es casi obligatorio) alimentar la RPi desde las baterías con un convertidor DC-DC ,



1.3.Material HW necesario

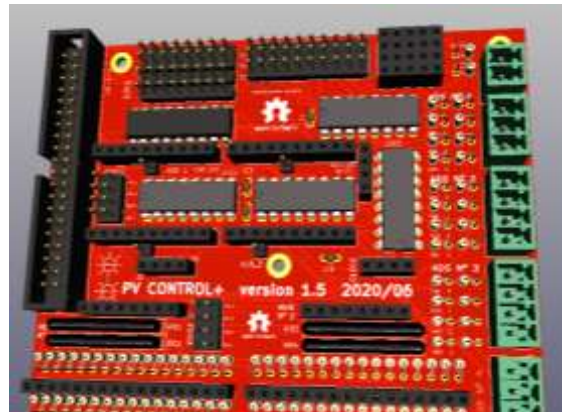
Raspberry

- Una Raspberry Pi(Rpi) al menos en versión 3 o 4 con una micro SD de 8GB o superior (dados los importes actuales yo recomiendo al menos 32GB y **microSD de tipo A2**
- Alimentador de la Rpi ...aquí podemos optar por un alimentador desde 220VAC (útil para ponerla inicialmente mientras se configura cerca de la TV etc.....o **un convertidor DC-DC para alimentarla directamente de las baterías (es la opción lógica en la instalación final)**
- Teclado/ratón...solo hace falta para la configuración inicial..luego si se tiene cualquiera por USB vale (en mi caso use uno inalámbrico con dongle USB)...pero insisto es solo para la fase inicial...después ya se puede quitar y controlar la Rpi desde cualquier PC que tengamos



Placa Base (en caso de no capturar desde los equipos)

Para el montaje HW se ha diseñado una PCB que facilita la instalación, no es obligatoria su utilización dado que se puede montar en una protoboard estándar, pero por el coste que tiene es recomendable su utilización, la tirada mínima de las PCB es de 10 uds si no se consiguen interesados para compartir PCB, la peor situación sería tener 1 PCB operativa y 9 PCB de repuesto ☺



De dicha PCB se han diseñado varias versiones ,por ejemplo la versión 1.6 como máximo puede tener:

- 24 salidas de relé en local utilizando 3 PCF8574P
- 8 salidas de relé con capacidad PWM utilizando los GPIO de la Rpi y un ULN2803
- 16 entradas Analógicas directas utilizando 4 ADS1115
- 32 entradas analógicas adicionales a través de dos multiplexores de 16 canales que usan dos de las entradas anteriores (en la versión de PCB 1.6 se permite capturar en modo diferencial para tener una medida mas precisa usando los dos MUX y un máximo de 16 entradas)

- 2 Sensores de temperatura DS18B20 (uno integrado en le PCB y otro para un sensor externo)

Como hemos comentado anteriormente, esta seria la capacidad máxima, pero lo normal es que no se necesite completa en una instalación FV normal, por lo que en este manual instalaremos en la PCB solo:

- 8 salidas de rele: instalaremos 1 PCF8574P
- 8 entradas analógicas : instalaremos 2 ADS1115
- 1 Sensor de temperatura DS18B20

PCF8574P

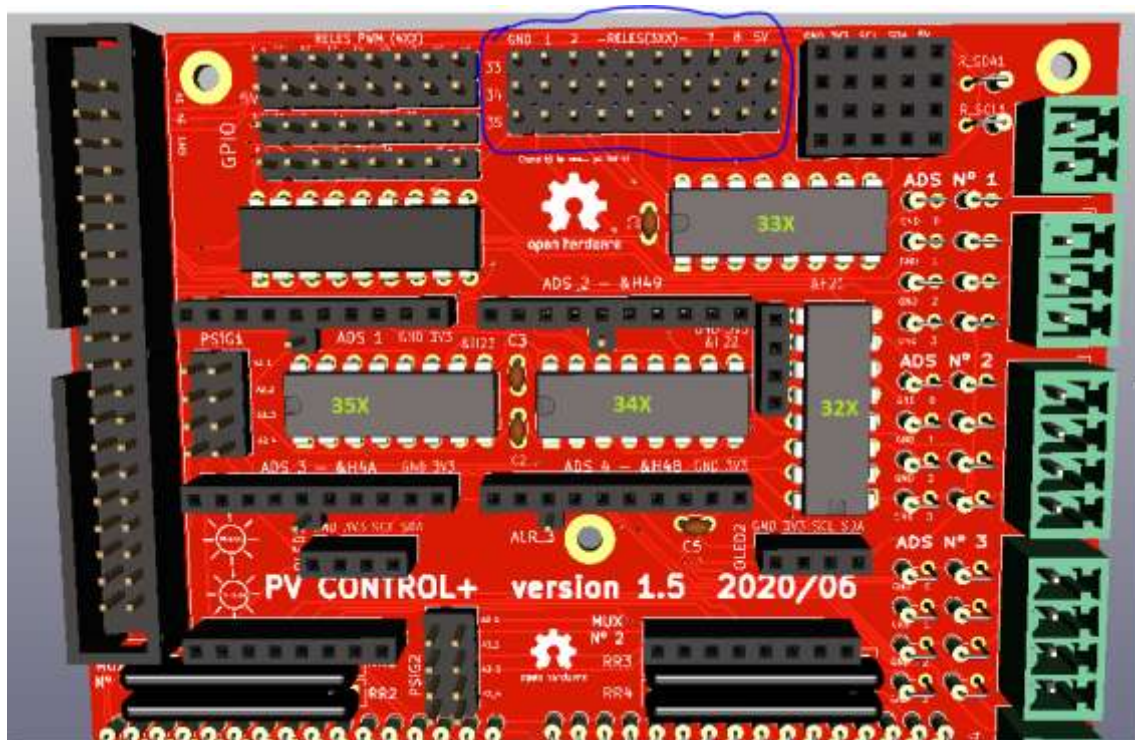
Este integrado tiene 8 GPIO que usamos para activar los relés que tengamos en local

Es buena práctica no soldar directamente los integrados sino poner un zócalo, por lo que mi recomendación es hacerlo así

Dado el escaso coste como siempre es mejor no comprar solo 1 sino tener alguno de repuesto

Por tanto, para nuestra instalación si necesitamos control de relés mecanicos en local instalaremos el PCF de dirección 33, 34 o 35 si queremos tener 8, 16 o 24 relés y sus correspondientes pines de salida

El PCF con dirección 32 se utiliza para el control del Multiplexor por lo que no se usa para controlar relés



En la PCB se puso unos condensadores de desacoplo de 100nf (C1, C2, ...) realmente, salvo que exista un ambiente de mucho ruido, no creo que haga falta su instalación

ADS1115

En PVControl+ se decidió usar el conversor analógico digital ADS1115 para medir la Vbat, Ibat, etc

La PCB admite hasta 4 , pero para la configuración estándar utilizaremos solo 2

En este caso, si recomiendo encarecidamente tener alguno de repuesto, dado que es fácil que se estropee alguno si “metemos algo la pata” (no conectar GND, etc)

La PCB se ha diseñado para instalar el ADS1115 según la imagen que se observa (placa roja)



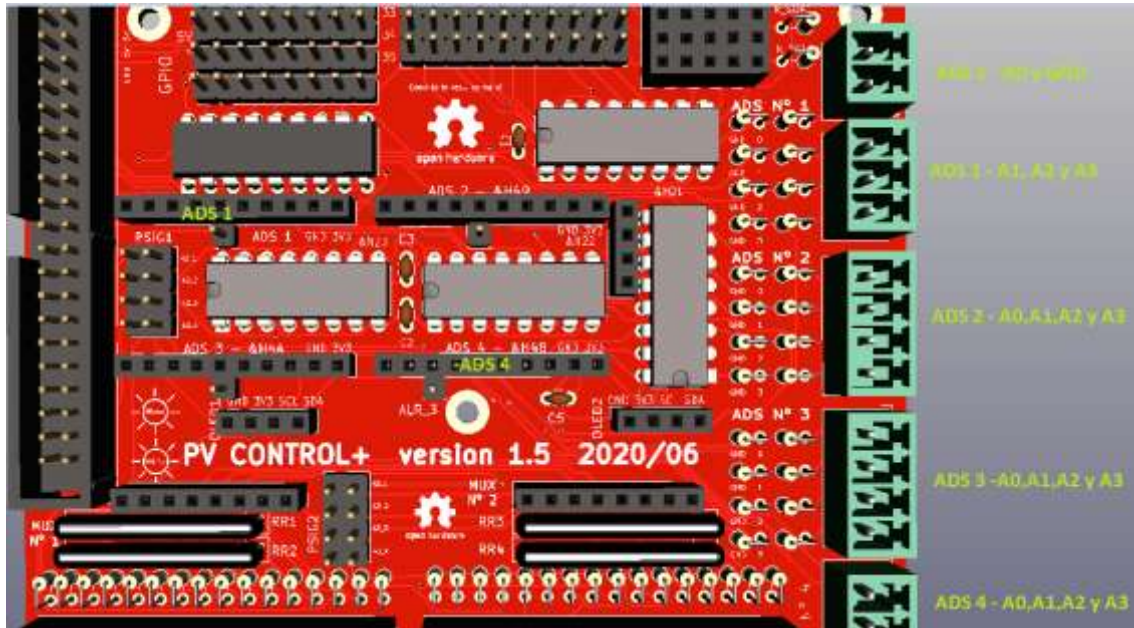
Hay otro tipo de placas en el mercado ,por ejm una placa azul mas grande, en ese caso, como los pines están a la izquierda en lugar de a la derecha de la placa, tened cuidado de soldar los pines macho en la cara contraria para que la placa enchufada en la PCB “vuele” sobre el PCF que está arriba

Lo importante en el ADS que se compre es que el orden de los pines VDD, GND, SCL y SDA etc sea el que se muestra en las imágenes



El proceso de instalación sería soldar los pines macho en dicha plaquita y soldar unos pines hembra en la PCB, si lo hacemos así, el cambio si se estropea es muy fácil dado que solo se enchufaría

Veamos qué es lo que se instalaría en el caso de querer capturar Vbat, Ibat,....



Utilizaremos el ADS número 1 para Vbat, Vplaca, etc y número 4 para Ibat e Iplaca

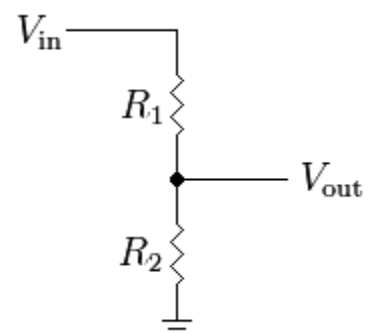
Por tanto pondremos la tira de pines hembra marcadas y los conectores respectivos

Aconsejo poner conectores enchufables para facilitar la instalación de los cables que vienen de la batería y de los shunts



Divisores de tensión (resistencias)

El ADS1115 solo admite señales de poco voltaje (5V) y nosotros queremos medir el valor de la batería por ejm que tiene 12V, 24V o 48V





Para hacer esto se utiliza lo que se llama un “divisor de tensión” que simplemente es poner dos resistencias en serie

Los valores de R1 y R2 dependerán del valor de la señal de entrada..luego serán distintos si el se quiere medir 24V o 48V u otro valor

$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in}$$

Es bastante fácil calcularlos, no obstante a modo de guía por ejm para medir Vbat y Vplaca seria:

Señal	Sistema	V Entrada maxima	R1 (Kohm)	R2(Kohm)	V salida	P_R1 (mW)	P_R2 (mW)	I (mA)
Vbat	12V	17	4,7	1	2,98	41,81	8,90	2,98
	24V	35	10	1	3,18	101,24	10,12	3,18
	48V	70	22	1	3,04	203,78	9,26	3,04
Vplaca		100	82	2,2	2,61	115,66	3,10	1,19
		150	100	2,2	3,23	215,42	4,74	1,47

Como se ve limitamos la salida máxima por debajo de 3,3V dado que es a lo que se alimentara los ADS1115 desde la Raspberry pero también miramos que la potencia que se disipa en las resistencias sea menos de 1/4w...si queremos poner resistencias de 1/8w tendremos que subir proporcionalmente los valores de R1 y R2

Una vez aclarado el concepto, veamos donde se ponen dichas resistencias en la PCB

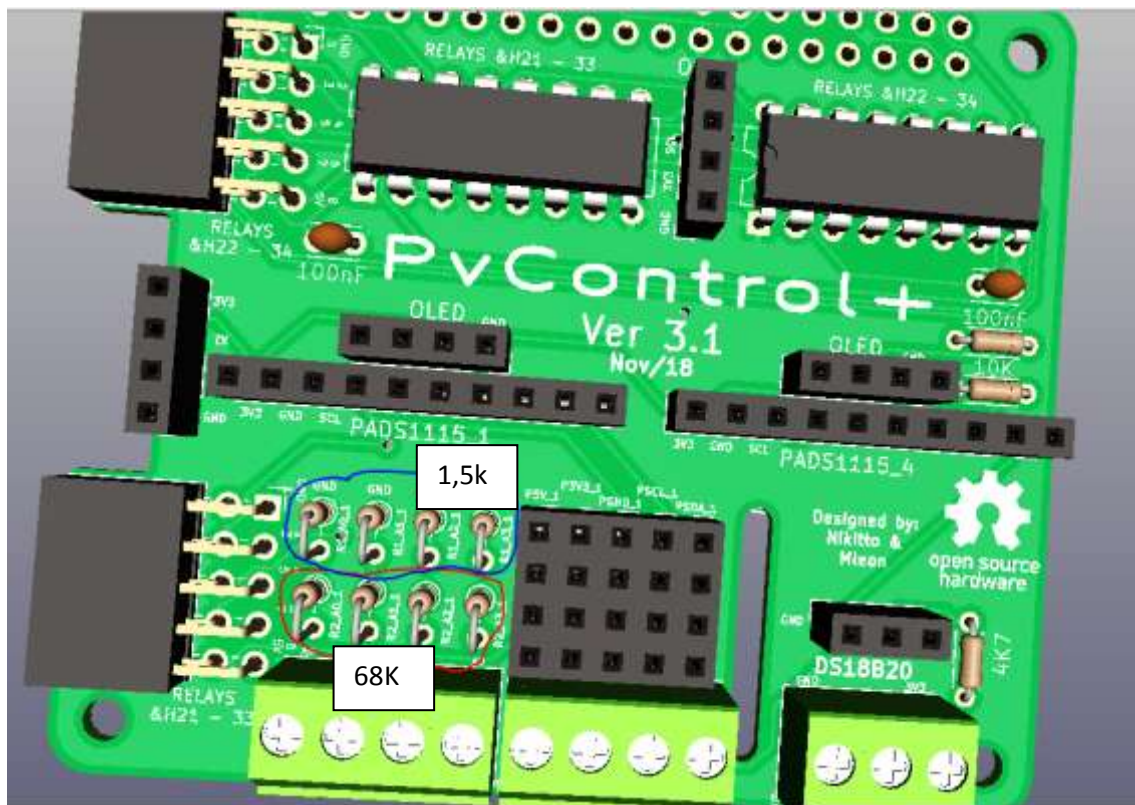
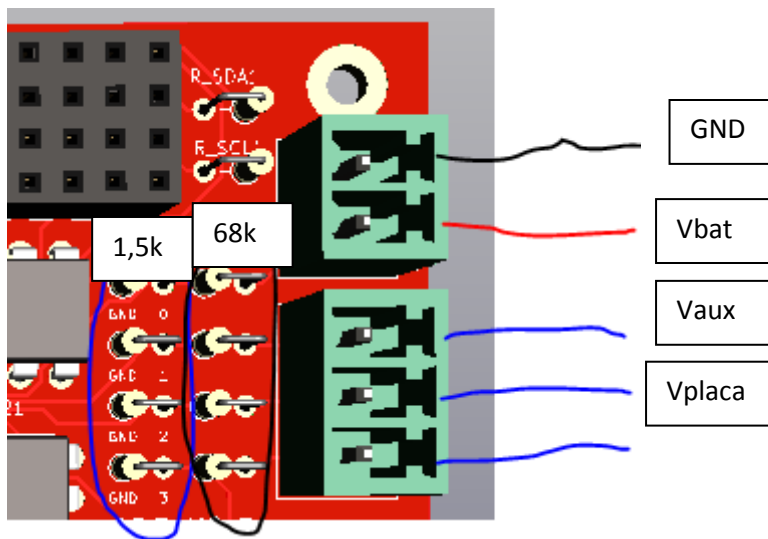
ES MUY IMPORTANTE COLOCAR LAS RESISTENCIAS BIEN PARA NO DAÑAR AL ADS1115, FIJARSE BIEN EN LAS IMÁGENES SIGUIENTES

Si por ejm, queremos leer Vbat, Vaux y Vplaca en las tres primeras entradas del ADS numero 1 podemos elegir dos opciones:

- Para cada entrada unos valores de resistencias para ser más exactos según la tabla anterior
- Optar por poner las mismas resistencias en todas las entradas, dado que el ADS tiene suficiente precisión como para que no sea crítico

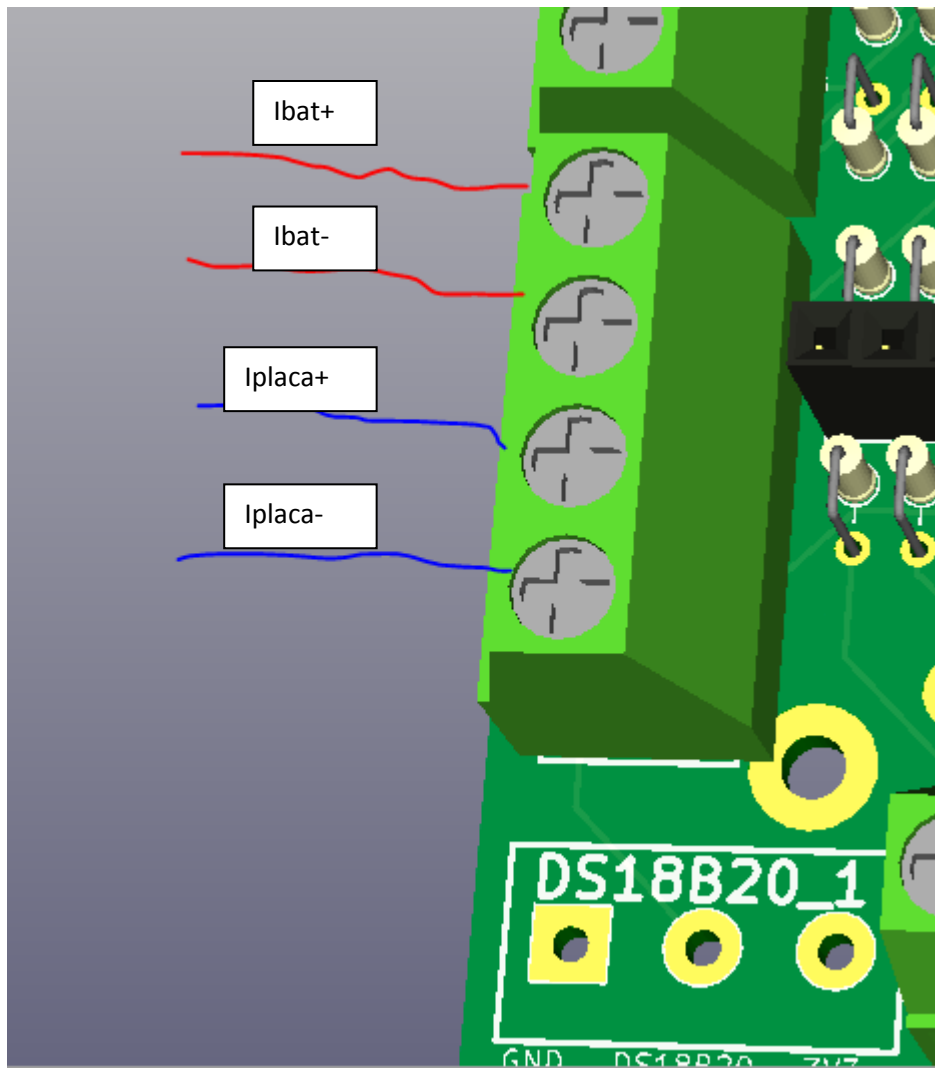
Por ejm con 68K y 1,5K nos permitiría un rango máximo de unos 150V en cada entrada

Esta sería la configuración de entrada:



Para medir las señales de Ibat e Iplaca usaremos dos shunts conectados según el primer esquema

En este caso la conexión al ADS número 4 de los cables que vienen desde los shunt es sencilla y NO se necesita poner resistencias para adaptar la señal de entrada:



No os preocupe mucho si ponéis el + o el – al revés.... Si lo ponéis al revés os saldrá la intensidad de batería negativa cuando carga y positiva cuando descarga luego simplemente intercambiar los cables y ya estará solucionado

Sensor de Temperatura DS18B20

En el Brico usamos el sensor de temperatura DS18B20

Hay muchos tipos de configuraciones en el mercado... en el Brico usamos los de este tipo si queremos instalarlo directamente en la PCB

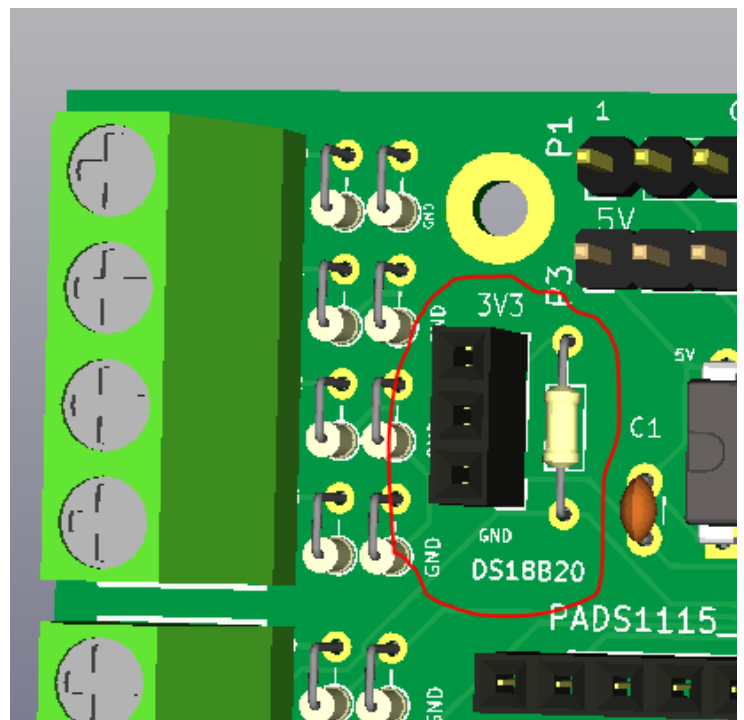


O los de este tipo si queremos que el sensor este cerca de las baterías



Si optamos por la primera opción en la PCB soldaremos una resistencia de 4,7Kohm y una tira de 3 pines hembra para poder enchufarlo

Yo recomiendo hacer esto aunque se opte por la segunda opción dado que la resistencia de 4,7K hace falta siempre y simplemente no enchufar en los 3 pines hembra el sensor



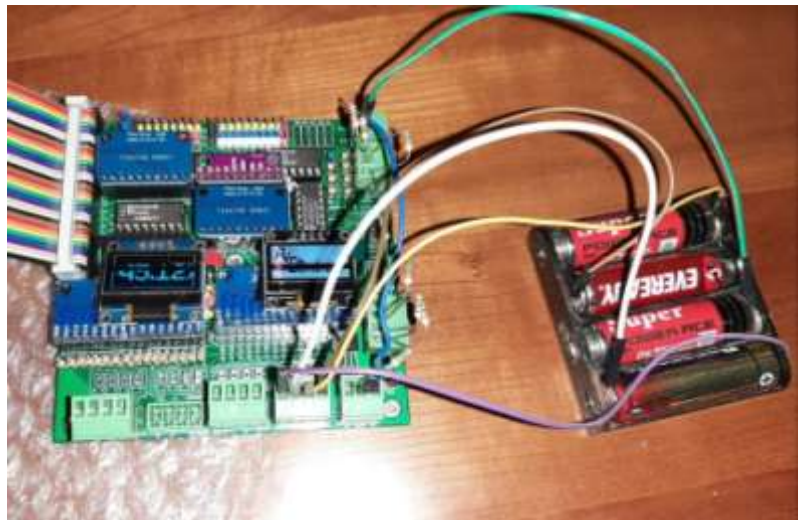
Un ejemplo de la placa versión 1.5 montada con :

- 8 salidas para reles con un PCF8574P
- 8 salidas de reles con capacidad PWM con el ULN2803
- Sensor Temperatura
- Dos ADS1115



Un ejemplo de la placa versión 1.6 montada casi "a tope" con :

- 24 salidas para reles con tres PCF8574P
- 8 salidas de reles con capacidad PWM con el ULN2803
- Sensor Temperatura
- Tres ADS1115
- Dos multiplexores para medir hasta 16 celdas de forma diferencial
- Dos OLED



Otro ejemplo de la placa versión 3.1 con capacidad para :

- 16 salidas para reles (con dos PCF8574P)
- Sensor Temperatura externa
- Dos ADS1115
- Dos pantallas OLED







2.Instalación SW

Marcaré en este color los puntos que **ES NECESARIO** revisar

Lo primero es cargar la imagen de PVControl+ en la microSD desde un PC

Hay muchos tutoriales en la web de cómo hacerlo, **por ejemplo usando la utilidad raspberry pi image o la utilidad de balenaetcher**

- <https://www.raspberrypi.org/blog/raspberry-pi-imager-imaging-utility/>
- <https://www.hwlibre.com/etcher/>

En la imagen de PVControl+ ya viene básicamente todo preinstalado, pero lógicamente hay que adaptar algunos parámetros a la instalación de cada uno

COMO SEGURIDAD MI CONSEJO ES GUARDAR UNA IMAGEN DE LA microSD EN EL PC ANTES DE TRASTEAR, PARA PODER VOLVER A LA SITUACION INICIAL SIEMPRE QUE SEA NECESARIO

UNA VEZ TENGAMOS LA INSTALACION FUNCIONANDO ES MUY RECOMENDABLE SACAR PERIODICAMENTE UNA IMAGEN DE LA microSD PARA TENERLA COMO SEGURIDAD

2.1. Acceso a la Raspberry

Lo primero es poder acceder a la RPi para poder configurarla con los datos de nuestra instalación (Wifi, etc)

<https://www.raspberrypi.org/documentation/remote-access/README.md>

Como veis hay varias formas, explico algunas:

Monitor/Teclado/Raton

Una forma muy cómoda y fácil de poder acceder a la RPi para configurarla es enchufarla a un Monitor/TV con entrada HDMI y usar un teclado/raton

Si esta opción es posible, es la mas recomendable dado que siempre funcionara sin problemas

Acceso por VNC

VNC nos permite ver el escritorio de la RPi en un PC, y por tanto manejar la RPi como si se tuviera enchufado un Monitor/Teclado/Raton



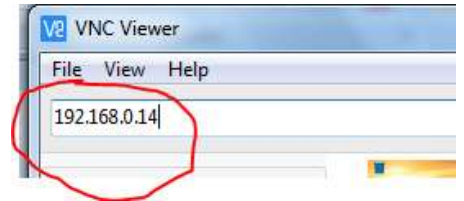
Para ello debemos instalar en nuestro PC un visor de VNC (VNCViewer por ejm)

<https://www.realvnc.com/es/connect/download/viewer/>

Una vez tengamos VNC Viewer instalado tendremos que decirle a que IP nos queremos conectar

En la imagen de PVControl se ha puesto el siguiente usuario/clave:

- **Usuario : pi**
- **Clave: PVControl+**



El valor de la IP a poner dependerá de cómo hemos conectado la RPi, veamos algunos ejemplos

- ***RPi conectada por cable Ethernet al Router***

En este caso por ejm, podemos ver que IP ha sido asignada en el Route

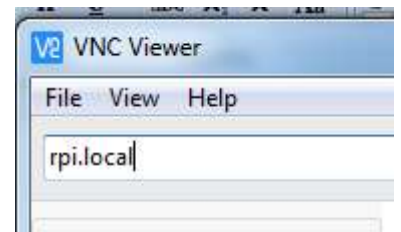
Una opción que muchas veces funciona es intentar

conectar poniendo **rpi.local**

Otra opción es usar una App en un movil para ver los equipos conectados a nuestra LAN

Yo utilizo Wifi Network Analyzer que permite esto además de ver la cobertura Wifi, pero seguro que hay muchas mas opciones

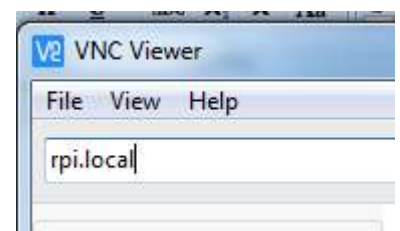
https://play.google.com/store/apps/details?id=com.pzolee.wifiinfo&hl=en_US



- ***RPi conectada por cable Ethernet directamente al PC***

Se puede probar una vez conectados si simplemente poniendo rpi.local se conecta

Si no funciona hay que analizar la configuración de la





red creada en el PC.. si quieres usar esta opción busca por la web los muchos tutoriales que hay

- *Incluir en la microSD desde el PC los archivos de configuración WIFI*

Otra opción es incluir en la microSD la información de nuestra WIFI (SSID y Clave) , en este caso hay que considerar que la IP que viene por defecto en la imagen es 192.168.0.X (X puede ser 12, 14, 15. etc .dependiendo de la version de la imagen que se tenga), luego nuestra LAN debe estar en el rango 192.168.0.X para poder conectarse

Hay que añadir desde el PC en la carpeta BOOT de la microSD el archivo “**wpa_supplicant.conf**” con esta estructura poniendo en el ssid y psk los valores de nuestra WIFI

```
wpa_supplicant.conf
1 ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
2 update_config=1
3 country=ES
4
5 network={
6   ssid="PVControl+"
7   psk="PVControl+"
8   key_mgmt=WPA-PSK
9 }
```

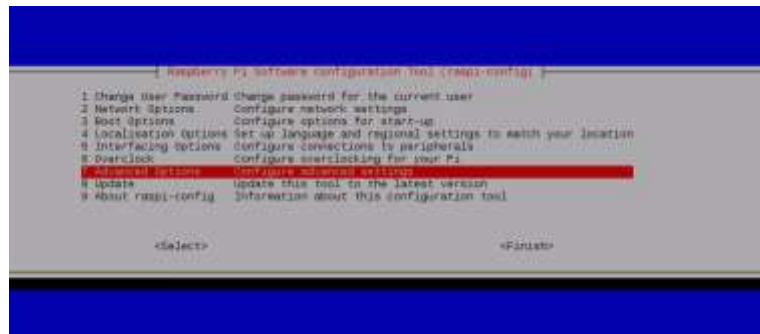
2.2. Expandir la tarjeta SD

Una vez que nos podamos conectar a la Rpi, podemos iniciar la configuración expandiendo la tarjeta microSD (en las últimas versiones de las Raspberry **NO es necesario expandir la tarjeta** dado que se autoexpande, en todo caso mirar la capacidad de la microSD con el comando `df -h` para ver si coincide con el tamaño de la tarjeta)

La imagen realizada se puede poner en una SD de al menos 8GB...se recomienda una de SD de 16GB o 32GB, por lo que una vez copiada la imagen en la SD hay que expandir la tarjeta para asegurar que se utiliza toda la capacidad disponible

Desde terminal se ejecuta:

- `sudo raspi-config`



opción Advanced



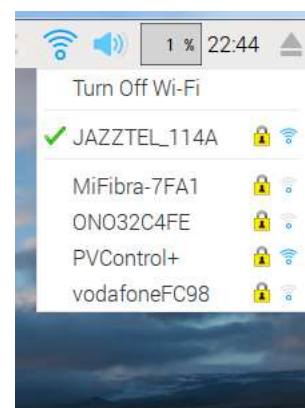
opción A1

Con esto la próxima vez que se reinicie ya se tendrá la capacidad completa de la tarjeta que podemos ver con el comando “df -h “

```
pi@rpi:~ $ df -h
S.ficheros      Tamaño Usados  Disp Uso% Montado en
/dev/root        30G    4,8G    24G   18% /
devtmpfs         460M     0    460M   0% /dev
tmpfs            464M     0    464M   0% /dev/shm
tmpfs            464M    13M    452M   3% /run
tmpfs            5,0M    4,0K    5,0M   1% /run/lock
tmpfs            464M     0    464M   0% /sys/fs/cgroup
/dev/mmcblk0p1   44M     23M     22M   52% /boot
tmpfs            93M     0     93M   0% /run/user/1000
```

2.3.WIFI

Si aun no lo hemos hecho en los pasos previos hay que dar de alta la red wifi que se tenga



2.4.IP

La IP que tendrá la RPi será la que le proporcione el router cuando se conecte por Wifi y/o cable

Si queremos una IP fija podemos establecerla en la propia Rpi o asignarla en el router





Una vez conectada la Rpi con la IP correcta, se puede trabajar perfectamente desde un PC externo con escritorio remoto como hemos hablado anteriormente usando el programa VNCViewer en el PC

2.5. Actualizar a la última versión de PVControl+

**Actualizamos
a la última
versión
de
PVControl+**

Reiniciamos la Rpi



Abrimos ventana de terminal, vamos a la carpeta PVControl+ y pulsamos git pull



```
Archivo  Editar  Pestanas  Ayuda
pi@rpi:~ $ cd PVControl+
pi@rpi:~/PVControl+ $ git pull
```



Se nos actualizarán los ficheros de PVControl+ con los últimos cambios que existan

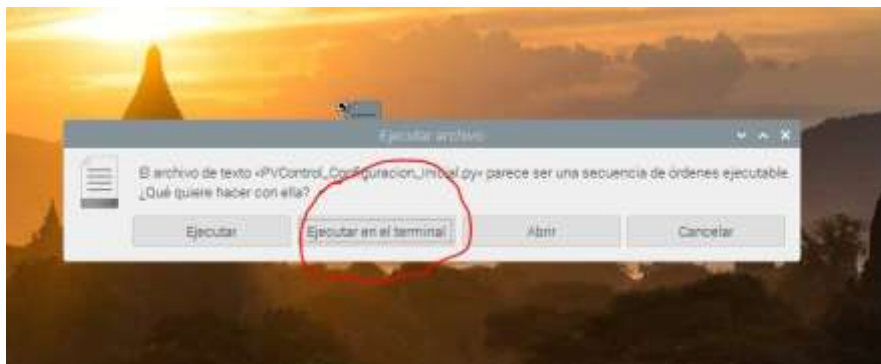
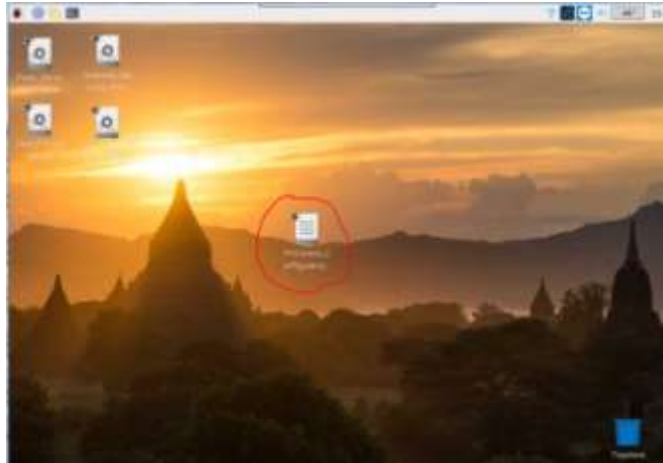
ESTE PASO DE ACTUALIZACION ES MUY IMPORTANTE

HACERLO SIEMPRE **ANTES DE EMPEZAR A CONFIGURAR
PVControl+**

2.6. Parámetros Instalación PVControl+

Para la configuración inicial que puede ser válida en muchos casos, se puede ejecutar la aplicación "PVControl_Configuracion_Inicial.py"

Esta aplicación se puede ejecutar directamente desde el escritorio haciendo doble click



```
PVControl_Configuracion_Inicial.py
Archivo Editar Pestañas Ayuda

#####
PROGRAMA DE CONFIGURACION INICIAL DE PVControl+

Este programa realiza la configuracion de PVControl+

#####

Si no esta seguro pulse 0 para salir o 1 para continuar [0]:
```

Pulsar 1 y después la tecla



Para que nos salga el siguiente paso

```

PVControl_Configuracion_Inicial.py
Archivo  Editar  Pestañas  Ayuda

#####
PROGRAMA DE CONFIGURACION INICIAL DE PVControl+

Este programa realiza la configuracion de PVControl+

#####

Si no esta seguro pulse 0 para salir o 1 para continuar  [0]: 1

#####
Se va a editar el archivo de configuracion Parametros_FV.py....
..... pulse 1 para usar el archivo actual de Parametros_FV.py
..... Pulse 2 si es la primera configuracion y usar el archivo patron Parametros_FV_DIST.py
..... Pulse 0 para seguir sin modificar el archivo Parametros_FV.py
[0]: █
  
```

- 0- Se salta este paso y no se modifica el archivo de Parametros_FV.py que ya existe
- 1- Usa el archivo Parametros_FV.py actual (se usa para cambiar el archivo previamente creado)
- 2- Si es la primera configuración de PVControl+

Introducimos la opción y pulsamos enter (las opciones que salen dependen de los archivos de ejemplo que estén dados de alta con la versión de PVControl+, elegir la que mas se adapte a la instalación FV que se tenga o, en caso de duda, la genérica DIST)

```

Archivo  Editar  Pestañas  Ayuda

#####

Si no esta seguro pulse 0 para salir o 1 para continuar  [0]: 1

#####
Se va a editar el archivo de configuracion Parametros_FV.py....

..... pulse 1 para usar el archivo actual de Parametros_FV.py
..... Pulse 2 si es la primera configuracion y usar el archivo patron Parametros_FV_DIST.py
..... Pulse 0 para seguir sin modificar el archivo Parametros_FV.py
[0]: 1

Se crea copia de seguridad de Parametros_FV.py anterior con el nombre...Parametros_FV_2021-12-21_21:25.back

#####
Como primer paso se abrirá en el editor de textos "geany" el archivo Parametros_FV.py.....

ES MUY IMPORTANTE RELLENAR BIEN ESTE ARCHIVO SIN ERRORES DE SINTAXIS
LEA EL MANUAL SI TIENE DUDAS DE COMO RELLENAR EL ARCHIVO
MODIFIQUE LO NECESARIO SEGUN SU INSTALACION, ...GUARDE el archivo..... y SALGA de geany para continuar
#####
pulsar una tecla para seguir..... [ ]: █
  
```

Nos avisa de que se abrirá el archivo Parametros_FV.py y de lo importante que es configurarlo bien

También realiza una copia de seguridad de archivo Parametros_FV.py que exista

Pulsamos una tecla y se abrirá el archivo Parametros_FV.py en el editor incluido con la Rpi llamado **Thonny**



```
Thonny - /home/pi/PVControl+/Parametros_FV.py @ 426:1
Fichero Editar Visualización Ejecutar Herramientas Ayuda
Argumentos del programa:

Parametros_FV.pyx
1 # -----
2 #####  PARAMETROS INSTALACION PVControl+  -- version: 2021-12-21
3 # -----
4
5 # =====
6 #
7 # 1.- SECCION PRINCIPAL PARAMETRIZACION
8 # =====
9 #
10
11
12 #####
13 ##### Simulacion #####
14 #####
15 simulador = 0 # Simulacion datos FV --- 1 para simular...0 para no simular
16 simulador_reles = 0 # Simular reles fisicos
17 # -----
18
19 #####
20 ##### Parametros sensores #####
21 #####
22
23 # 'Variable' : {'Equipo':"expresion captura",'Max':Valor maximo, 'Min':Valor minimo,.....},
24
25 ## Se pueden crear las variables que se quieran y con el nombre que se quiera
26 ## La "expresion de captura" puede ser cualquier expresion python3 valida
27 ## En los programas de captura que utilicen la tabla en BD RAM la sintaxis sera... "d_['CLAVE']['Variable']"... d_['ADS1
28 ## En los programas de captura que utilicen la tabla en BD RAM la sintaxis sera... "d_['CLAVE']['Variable']"... d_['ADS1
```

Dada su importancia veamos cada apartado de dicho archivo por separado

Parametros_FV.py se divide en 3 secciones

1. SECCION PRINCIPAL

Permite configurar lo necesario para el funcionamiento del programa principal (fv.py)

2. SECCION PARAMETRIZACION SEGUN EQUIPAMIENTO INSTALADO

Permite configurar el distinto equipamiento que tenemos instalado (baterías, PCB de PVControl+, Hibrido tipo Axpert, Fronius, Huawei etc)

3. SECCION PARAMETRIZACION SERVICIOS ADICIONALES

Permite configurar los distintos servicios adicionales disponibles en PVControl (Telegram, PVOOutput, motionEye...)

Sección principal

Simulación

Se dejara el valor por defecto a 0 salvo que queramos que el programa se ejecute con datos simulados

```
#####
##### Simulacion #####
#####
simular = 0 # Simulacion datos FV --- 1 para simular....0 para no simular
simular_reles = 0 # Simular reles fisicos
# .....
```

Sensores

```
#####
##### Parametros sensores #####
#####
# 'Variable' : ['Equipo':'expresion captura', 'Max':Valor maximo, 'Min':Valor minimo,.....],
## Se pueden crear las variables que se quieran y con el nombre que se quiera
## La "expresion de captura" puede ser cualquier expresion python3 valida
## En los programas de captura que utilicen la tabla en BD RAM la sintaxis sera... "d_['CLAVE']['Variable']"... d_['ADS1']['Vbat'], d_['HIBRIDO']['Ibat']
## En los programas de captura que usen utilicen los archivos pkl la sintaxis sera... "d_clave['Variable']"... d_sma['Vbat'], d_sche['Vbat'],...
## Si se definen las claves 'Max' y 'Min' se enviara un log si la captura esta fuera de dichos margenes

sensores = {
    'Vbat' : {'Equipo':'d_['ADS1']['Vbat']', 'Max':32, 'Min':0}, # Sensor de Voltaje bateria
    'Vplaca' : {'Equipo':'d_['ADS1']['Vplaca']', 'Max':100, 'Min':0}, # Sensor de Voltaje placa
    'Ibat' : {'Equipo':'d_['ADS4']['Ibat']', 'Max':200, 'Min':-200}, # Sensor de Intensidad bateria
    'Iplaca' : {'Equipo':'d_['ADS4']['Iplaca']', 'Max':200, 'Min':-1}, # Sensor de Intensidad Placa
    'Aux1' : {'Equipo':'d_['ADS1']['Aux1']', 'Max':100, 'Min':0}, # Sensor Aux1
    'Aux2' : {'Equipo':'d_['ADS1']['Aux2']', 'Max':100, 'Min':0}, # Sensor Aux2
    'Aux3' : {}, # Sensor Aux3
    'Aux4' : {}, # Sensor Aux4
    'Aux5' : {}, # Sensor Aux5
    'Aux6' : {}, # Sensor Aux6
}
```

En esta parte se definen las formulas que describen como se capturara cada variable (Vbat, Ibat etc)

En este sentido hay varias posibilidades:

- **Equipos que ya se han actualizado para usar la tabla Equipos de la Base de datos**

En este caso la sintaxis es **d_['CLAVE']['Variable']**

Asi por ejemplo si queremos que Vbat se capture desde un Hibrido tipo Axpert la línea quedaría así:

```
'Vbat' : {'Equipo':'d_['HIBRIDO']['Vbat']', 'Max':32, 'Min':0},
```

Si no queremos que se controle el máximo y mínimo valor (mandando un log si el valor capturado se sale de dichos márgenes podemos simplificar quedando

```
'Vbat' : {'Equipo':'d_['HIBRIDO']['Vbat']'},
```



ATENCIÓN a cada carácter { “ , etc debe se exactamente como se indica para que funcione... no olvidar la coma final

Entre los equipos que a fecha actual usan este sistema están las siguientes claves (programas que generan el registro en tabla equipos):

- **ADS** (fv_ads.py) : Para el uso de los ADS1115 de la PCB de PVControl+
- **HIBRIDO** (hibrido.py) : Captura de hasta 9 hibridos tipo Axpert
- **CELDAS**: Captura los valores de la distintas celdas , existen varuios programas para este caso según como capturemos los valores
 - fv_mux.py si se usa la PCB de PVControl+
 - fv_daly.py: si se usa el BMS marca DALY
 - fv_diybms.py: si se usa el BMS de diyBMS
- **TEMP** (fv_temp.py) captura los valores de temperatura de los sensores DS18B20
- **MQTT** (fv_mqtt.py) captura los topic MQTT que se definan
- **SDM120C** (fv_sdm120c.py) : Capturas desde un sdm120C
- **HUAWEI** (huawey.py): Capturas desde equipo Huawei
- **FRONIUS** (fronius.py): capturas dede equipos fronius
- **SI / SB** (si.py/sb.py): Capturas desde equipos SMA (Sunny Island y Boy)
- **BMV** (bmv.py): capturas desde puerto serie de monitores baterías Victron

- **Equipos que aun NO se han actualizado para usar la tabla Equipos de la Base de datos** (puede que se actualicen posteriormente a la edición de este manual, luego pruebe con la opción anterior si esta no funciona)

En este caso la sintaxis es **d_clave['Variable']**

Así por ejemplo si queremos que Vbat se capture desde un SMA la línea quedaría :

'Vbat' : {'Equipo':'d_srne['Vbat']', 'Max':32, 'Min':0},

Se iran cambiando progresivamente este tipo de programas para usar la tabla equipos

A fecha de escritura de este manual estos son los programas que usan esta forma:

- **victron** (victron.py) : capturas desde puerto serie de equipos Victron
- **sma_meter** (sma_meter.py): capturas desde el meter de SMA
- **goodwe** (goodwe.py): capturas desde equipos goodwe
- **must** (must.py): capturas desde equipos must por modbus
- **srne** (srne.py): capturas desde equipos srne

Por tanto, así quedaría esta parte en el caso de usar la PCB de PVControl+ con el sensor de temperatura incluido en una instalación FV aislada

```
sensores = {
  'Vbat' : {'Equipo': "d_['ADS1']['Vbat']", 'Max':32, 'Min':0}, # Sensor de Voltaje bateria
  'Vplaca' : {'Equipo': "d_['ADS1']['Vplaca']", 'Max':100, 'Min':0}, # Sensor de Voltaje placa

  'Ibat' : {'Equipo': "d_['ADS4']['Ibat']", 'Max':200, 'Min':-200}, # Sensor de Intensidad bateria
  'Iplaca' : {'Equipo': "d_['ADS4']['Iplaca']", 'Max':200, 'Min':-1}, # Sensor de Intensidad Placas

  'Aux1' : {'Equipo': "d_['ADS1']['Aux1']", 'Max':100, 'Min':0}, # Sensor Aux1
  'Aux2' : {'Equipo': "d_['ADS1']['Aux2']", 'Max':100, 'Min':0}, # Sensor Aux2
  'Aux3' : {}, # Sensor Aux3
  'Aux4' : {}, # Sensor Aux4
  'Aux5' : {}, # Sensor Aux5
  'Aux6' : {}, # Sensor Aux6
  'Aux7' : {}, # Sensor Aux7

  'Vred' : {}, # Sensor Voltaje de red
  'Ired' : {}, # Sensor Intensidad de red
  'EFF' : {}, # Eficiencia Conversion

  'Wbat' : {'Equipo': "Ibat * Vbat"}, # Potencia de/a baterias
  'Wplaca' : {'Equipo': "Iplaca * Vbat"}, # Potencia de placas
  'Wred' : {'Equipo': "Ired * Vred"}, # Potencia de/a red
  'Wconsumo' : {'Equipo': "Wplaca-Wred-Wbat"}, # Consumo

  'Temp_Bat' : {'Equipo': "d_['TEMP']['Ds18b20']['Temp0']", 'Max': 50, 'Min':-5} # Sensor Temperatura
}
```

En el caso de usar Hibrido tipo Axpert y además querer poner en variables auxiliares capturas del equipo sdm120C quedaría:

```
sensores = {
  'Vbat' : {'Equipo': "d_['HIBRIDO']['Vbat']", 'Max':66, 'Min':11}, #
  'Vplaca' : {'Equipo': "d_['HIBRIDO']['Vplaca']", 'Max':500, 'Min':-5}, #

  'Ibat' : {'Equipo': "d_['HIBRIDO']['Ibat']", 'Max':200, 'Min':-200}, #
  'Iplaca' : {'Equipo': "d_['HIBRIDO']['Iplaca']", 'Max':200, 'Min':-1}, #

  'Aux1' : {'Equipo': "d_['SDM120C']['Vac']", 'Max':300, 'Min':0}, # Sen
  'Aux2' : {'Equipo': "d_['SDM120C']['Wac']", 'Max':1, 'Min':-2500}, # S
  'Aux3' : {}, # Sensor Aux3
  'Aux4' : {}, # Sensor Aux4
  'Aux5' : {}, # Sensor Aux5
  'Aux6' : {}, # Sensor Aux6
  'Aux7' : {}, # Sensor Aux7

  'Vred' : {}, # Sensor Voltaje de red
  'Ired' : {}, # Sensor Intensidad de red
  'EFF' : {}, # Eficiencia Conversion

  'Temp_Bat' : {}, # Sensor Temperatura

  # Expresiones calculadas
  'Wbat' : {'Equipo': "Ibat * Vbat"}, # Potencia de/a baterias
  'Wplaca' : {'Equipo': "d_['HIBRIDO']['Wplaca']", # Potencia de placas
  'Wred' : {'Equipo': "Ired * Vred"}, # Potencia de/a red
  'Wconsumo' : {'Equipo': "d_['HIBRIDO']['PACW']"}, # Consumo
}
```




Se permiten expresiones algebraicas, por lo que si tenemos dos Híbridos y queremos sacar la Wplaca total se pondría:

....

```
'Wplaca' : {'Equipo':" d_['HIBRIDO']['Wplaca']+ d_['HIBRIDO1']['Wplaca']", 'Max':4000, 'Min':0},
```

...

Esta parte es vital y algo compleja luego se debe rellenar espacio y con cuidado de poner la sintaxis bien

Para asegurar que, al menos, no hay errores sintácticos ejecutar desde Thonny y ver que no da error

```
Fichero Editar Visualización Ejecutar Herramientas Ayuda
+ [Run] [Stop] [Clear] [Run and Debug] [Run and Test] [Run and Profile] [Run and Coverage] [Run and Coverage and Profile] [Run and Coverage and Test] [Run and Coverage and Profile and Test]
Argumentos del programa:

Parametros_FV.py
1 # -----
2 #####  PARAMETROS INSTALACION PVControl+  -- version: 2021
3 # -----
4 # -----
5 # -----
6 # -----
7 # 1.- SECCION PRINCIPAL PARAMETRIZACION
8 # -----
9 # -----
10
11
12 #####
13 ##### Simulacion #####
14 #####
15 simular = 0 # Simulacion datos FV -- 1 para simular.
16 simular_reles = 0 # Simular reles fisicos
17 # -----

Consola
Python 3.9.2 (/usr/bin/python3)
>>> %cd /home/pi/PVControl+
>>> %Run Parametros_FV.py
>>>
```

Base de datos y Broker MQTT

```
#####
##### Parametros Base de Datos #####
#####
servidor = "localhost"
usuario = "rpi"
clave = "fv"
basedatos = "control_solar"

grabar_datos_s = "False" # expresion para grabar cada muestra en la tabla datos_s
                        # Ejemplos: 'True'.. 'False'.. 'Vplaca > 10'.. 'PWM > 0'

t_muestra_max = 0 # valor para grabar en el log si tarda mas el bucle en ejecutarse
# -----

##### MQTT #####
#####
mqtt_broker = "localhost"
mqtt_puerto = 1883
mqtt_usuario = "rpi"
mqtt_clave = "fv"

##### Subscripciones #####
usar_mqtt_subcripciones = 0 # activa servicio fv_mqtt.py que se suscribe a los topics que se especifiquen en mqtt_subcripciones
                        # guarda lo capturado en la tabla ram 'equipos' ... diccionario=d['MQTT'] / servicio = fv_mqtt

mqtt_subcripciones=[] # lista de topics a los que se suscribe fv_mqtt.py para guardar en tabla equipos.. diccionario=d['MQTT']

##### Publicaciones #####
usar_mqtt_publicaciones = # 1 = Publica por MQTT los topic definidos en mqtt_publicaciones..... 0= No publica por MQTT
mqtt_topic_raiz = "PVControl/" # Raiz del topic a publicar

# Tuplas [Nombre Topic, Variable, frecuencia en sg (0 deshabilita, por defecto = Tmuestra * Nmuestra)]
mqtt_publicaciones = [{"Vplaca", "Vplaca"}, {"Vbat", "Vbat"}, {"Ibat", "Ibat", 10}, {"SOC", "SOC", 15}, {"DatosFV", "d_['FV']"}, {"d_['FV']", 0}]
```

Salvo que se sea usuario experimentado mejor no tocar la parte de Base de datos

En la parte de MQTT se pueden definir tanto los topic a los que se suscribe PVControl+ como los topic que publica

SECCION PARAMETRIZACION SEGUN EQUIPAMIENTO INSTALADO

Batería

```
#####
##### Parametros Bateria #####
#####
AH = 100.          # Capacidad en Ah de la Bateria a C20 (poner 0 para instalaciones sin Bateria)
CP = 1             # Indice Peukert
EC = 1             # Eficiencia Carga
vsys = 1           # Voltaje sistema - 1=12V 2=24V 4=48V
vflotacion = 13.7  # Valor por defecto de flotacion a 25°C a 12V (no se usa por ahora)
# -----
```

ADS1115 (PCB PVControl+)

Permite definir los distintos parámetros de captura de los conversores analógico-digitales ADS1115 que utiliza la PCB de PVControl+ (data_rate, ganancia,...)

Lo habitual es solo tener que modificar **res_ADS** para poner los ratios de los divisores de tensión que se han instalado en la PCB y los shunt de medición de Ibat/Iplaca

```
#####
##### Parametros ADS1115 - Permite hasta 4 ADS #####
#####
usar_ADS = [1,1] # activar o no el ADS
nombre_ADS = ['ADS1', 'ADS2']
direccion_ADS = [72, 75]

var_ADS = [['Vbat', 'Aux1', 'Vplaca', 'Aux2'], ['Ibat', 'Iplaca']] # Nombre de las variables a capturar

tmuestra_ADS = [0.200, 1] # tiempo en seg entre capturas
rate_ADS = [[250, 250, 250, 250], [250, 250, 250, 250]] # datarate de lectura
bucles_ADS = [[10, 0, 0, 0], [4, 2, 4, 2]] # Numero de bucles de lectura

gain_ADS = [[2, 2, 2, 2], [2, 2, 2, 2]] # Voltios Fondo escala 1=4.096V - 2=2.048V - 16= 256mV
modo_ADS = [[1, 1, 1, 1], [3, 0, 3, 0]] # 0=desactivado, 1=disparado, 2= continuo, 3=diferencial, 4=diferencial_continuo
res_ADS = [[47.48, 47.48, 47.48, 47.48], [100/75, 0, 100/75, 0]] # ratio lectura ADS - Lectura real
```

ATENCION... si alguien va a medir una señal por encima de 110V por ejm en Vplaca y no tiene claro cómo hacerlo que NO lo intente hacer

Por favor mucho cuidado con manejar tensiones altas... que todo esté perfectamente aislado y sin posibilidad de tocar por accidente

Si no lo veis claro..mucho mejor quedarse sin medir Vplaca que tener un accidente

Un cuadro de valores que llegarían a la entrada del ADS y potencia disipada en las resistencias dependiendo de la señal de entrada

R2= V1*R1/Vs - R1					mW		
GAIN	Ventr	R1	R2	Vsalida	P. R1	P. R2	
4	12	1500	68500	0.26	0.04	2.01	# -Tabla de Valores maximos segun ganancias ADS111 # - 2/3 = 6.144V # - 1 = 4.096V # - 2 = 2.048V # - 4 = 1.024V # - 8 = 0.512V # - 16 = 0.256V
	24	1500	68500	0.51	0.18	8.05	
	36	1500	68500	0.77	0.40	18.12	
2	48	1500	68500	1.03	0.71	32.21	
	60	1500	68500	1.29	1.10	50.33	
	72	1500	68500	1.54	1.59	72.47	
	84	1500	68500	1.80	2.16	98.64	
1	96	1500	68500	2.06	2.82	128.84	
	108	1500	68500	2.31	3.57	163.06	
	120	1500	68500	2.57	4.41	201.51	
	132	1500	68500	2.83	5.33	243.58	
	144	1500	68500	3.09	6.35	289.88	

MUX (Medida de Vceldas de la PCB de PVControl+)

Parametriza los MUX de la PCB de PVControl+ para la captura del valor de voltaje de las celdas

- Hasta 32 celdas en lectura simple
- Hasta 16 celdas en lectura diferencial

```
#####
##### Multiplexador #####
#####
user_mux = 0 # Poner el numero de celdas a monitorizar (0= desactivar)...diccionario = d_['MUX'] / servicio = fv_mux

t_muestra_mux = 5 # segundos entre capturas del mux
n_muestras_mux = 4 # grabar en BD en tabla permanente cada X capturas

pin_ADS_mux1 = "A2_1" #A2_1 = entrada A2 del ADS1, #A2_2 = entrada A2 del ADS2
#A2_3 = entrada A2 del ADS3, #A2_4 = entrada A2 del ADS4

pin_ADS_mux2 = "A3_1" #A3_1 = entrada A3 del ADS1, #A3_2 = entrada A3 del ADS2
#A3_3 = entrada A3 del ADS3, #A3_4 = entrada A3 del ADS4

captura_mux = "S" # 0 = lectura modo diferencial..., 5 = modo simple
# ATENCION si el modo de captura es diferencial se deben usar los 2 MUX y
# configurar en la PCB las salidas del MUX para usar las entradas A2 y A3 del mismo ADS

gain_mux = 1 # Voltios Fondo escala del ADS1119... 1=4.096 - 2=2.048

r_mux = [47] * 32 # Ratin Divisores de Voltaje de cada entrada de los Mux - Ejecutar el programa: python3 fv_mux_calibracion.py ... para calibrar
# Dicho programa creara en la BD la tabla "parametros1" y un registro donde se incluya la calibracion realizada

celdas_log_dif = 0.5 # diferencia entre la celda mas alta y la mas baja para suandar log
```

Usar esta funcionalidad nos permitirá ver distintas gráficas de la evolución de cada celda de la batería, por ejemplo en la pagina web "Inicio" se muestra el valor actual junto con el máximo y mínimo del día



Para la calibración de las capturas ejecutar desde terminal el programa ...

python3 fv_mux_calibracion.py



La lista de distintos equipos va creciendo con el tiempo

- `usr_XXX = 1` para activar el servicio que captura los datos
- `dev_XXX` = driver con el que reconoce la Rpi al equipo
- `t_muestra_XXX = 5` para capturar cada 5 sg los datos
- mas otros campos que dependen de cada equipo (IP, etc)

```
#####
##### HIBRIDO - Permite hasta 9 Equipos #####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Vbat, Vplaca,...) usa el Hibrido o se quiere guardar en BD en la tabla 'Hibrido'
## se debe poner usar hibrido = [1]

usar_hibrido = [1] #1 para leer datos Hibrido ..... 0 para no usar

dev_hibrido = ["/dev/hidraw0"] # puerto donde reconoce la RPi al Hibrido
usar_crc = [1] # 1 para comandos del hibrido con CRC... 0 para no añadir CRC

t_muestra_hibrido = [5] # Tiempo en segundos entre muestras del Hibrido
publicar_hibrido_mqtt = [1] # Publica o no por MQTT los datos capturados del Hibrido

grabar_datos_hibrido = [1] # 1 = Graba la tabla Hibrido... 0 = No graba
n_muestras_hibrido = [1] # grabar en BD en tabla 'hibrido' cada X capturas del Hibrido
```

Por ejemplo si queremos capturar valores de 2 Híbridos se pondría:


```
#####
##### HIBRIDO - Permite hasta 9 Equipos #####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Vbat, Vplaca,...) usa el Hibrdo o se quiere guardar en BD en la tabla 'Hibrdo'
## se debe poner usar hibrdo = [1]

usar_hibrdo = [1,1] #1 para leer datos Hibrdo ..... 0 para no usar

dev_hibrdo = ["/dev/hidraw0", "/dev/hidraw1"] # puerto donde reconoce la RPi al Hibrdo
usar_crc = [1,1] # 1 para comandos del hibrdo con CRC... 0 para no añadir CRC

t_muestra_hibrdo = [5,10] # Tiempo en segundos entre muestras del Hibrdo
publicar_hibrdo_mqtt = [1,0] # Publica o no por MQTT los datos capturados del Hibrdo

grabar_datos_hibrdo = [1,1] # 1 = Graba la tabla Hibrdo... 0 = No graba
n_muestras_hibrdo = [1,2] # grabar en BD en tabla 'hibrdo' cada X capturas del Hibrdo

#
```

En este caso las capturas del primer hibrdo serán cada 5 sg y del segundo cada 10sg etc

El valor de hidrawX dependerá de donde lo reconoce la Rpi, por lo que puede cambiar con cada instalacion

Ejemplos de distintos equipamientos

```
#####
#### DALY ####
#####
usar_daly = 0 # Poner el numero de celdas a monitorizar (0= desactivar)...diccionario = d_['DALY'] / servicio = fv_daly
t_muestra_daly = 1 # segundos entre capturas para tabla en RAM

grabar_datos_daly = 1 # 1 = Graba la tabla ... 0 = No graba
n_muestras_daly = 10 # grabar en SD en tabla permanente cada X capturas
dev_daly = "/dev/ttyUSB0" # puerto donde reconoce la RPI al Hibrido

# -----
#####
#### VICTRON ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa el Victron o se quiere guardar en SD en la tabla 'victron'
## se debe poner usar victron = 1

usar_victron = 0 # 1 para leer datos victron ..... 0 para no usar
dev_victron = "/dev/ttyUSB0" # puerto donde reconoce la RPI al Victron

grabar_datos_victron = 0 # 1 = Graba la tabla victron... 0 = No graba
t_muestra_victron = 5 # Tiempo en segundos entre muestras

iplaca_victron_max = 99
iplaca_victron_min = 0

# -----
#####
#### MUST ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa el MUST o se quiere guardar en SD en la tabla 'must'
## se debe poner usar must= 1

usar_must = 0 # 1 para leer datos victron ..... 0 para no usar
n_equipos_must = 0 # numero de inversores en paralelo. Si sólo hay uno, marcar 1.
dev_must = "/dev/ttyUSB0" # puerto donde reconoce la RPI al Must

grabar_datos_must= 0 # 1 = Graba la tabla Must... 0 = No graba
t_muestra_must = 1 # Tiempo en segundos entre muestras + numero de equipos

iplaca_must_max = 99
iplaca_must_min = 0

# -----
```

```
#####
#### BMV ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa el BMV o se quiere guardar en SD en la tabla 'bmv'
## se debe poner usar bmv = 1

usar_bmv = 0 # 1 para leer datos victron ..... 0 para no usar
dev_bmv = "/dev/serial0" # puerto donde reconoce la RPI al BMV

grabar_datos_bmv = 0 # 1 = Graba la tabla bmv... 0 = No graba
t_muestra_bmv = 5 # Tiempo en segundos entre muestras

# -----
#####
#### SMA ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa el SMA o se quiere guardar en SD en la tabla 'sma'
## se debe poner usar sma = 1

usar_sma = 0 # 1 para leer datos del sma ..... 0 para no usar
usar_si = 0 # 1 para leer datos del S1 ..... 0 para no usar
usar_sb1 = 0 # 1 para leer datos del SB1 ..... 0 para no usar
usar_sb2 = 0 # 1 para leer datos del SB2 ..... 0 para no usar
usar_smameter = 0 # 1 para leer datos del meter SMA ..... 0 para no usar
IP_S1 = "192.168.0.24" # IP del S1
IP_SB1 = "192.168.0.253" # IP del SB1
IP_SB2 = "192.168.0.252" # IP del SB2

grabar_datos_sma = 0 # 1 = Graba la tabla sma... 0 = No graba

# -----
#####
#### FRONIUS ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa fronius se debe poner usar fronius = 1

usar_fronius = 0 # 1 para leer datos del fronius..... 0 para no usar
usar_meter_fronius = 0 # 1 para activar lectura de contador de Fronius
IP_FRONIUS = "192.168.0.24" # IP del FRONIUS
t_muestra_fronius = 5

# -----
```

```
#####
### HUAMEI ###
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa fronius se debe poner usar huawei = 1

usar_huawei = 0          # 1 para leer datos del huawei..... 0 para no usar
IP_HUAMEI = "192.168.0.24" # IP del huawei
t_muestra_huawei = 5      # Tiempo entre capturas
# -----
#####
#### GOODWE ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa fronius se debe poner usar goodwe = 1

usar_goodwe = 0          # 1 para leer datos del goodwe..... 0 para no usar
IP_GOODWE = "192.168.0.100" # IP del goodwe
t_muestra_goodwe = 5
usar_batgoodwe = 0        # 1 para usar bateria y 0 para no usar
# -----
#####
#### SRNE ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor (Iplaca, Vplaca,...) usa el SRNE o se quiere guardar en BD en la tabla 'srne'
## se debe poner usar srne = 1

usar_srne = 0            # 1 para leer datos srne ..... 0 para no usar
dev_srne = "/dev/ttyUSB0" # /dev/ttyUSB0" # USB - "/dev/ttyS0" # TTL
grabar_datos_srne = 1     # 1 = Graba la tabla srne... 0 = No graba

iplaca_srne_max = 85
iplaca_srne_min = 0
# -----
#####
#### EASTRON ####
#####

## ATENCION ser congruente con lo que se ha puesto en el apartado de sensores
## Si algun sensor usa el easton se debe poner usar easton = 1

usar_eastron = 0          # 1 para leer datos ..... 0 para no usar
dev_eastron = ""          # /dev/ttyUSB0" # USB
# -----
```

```
#####
### BROADLINK ###
#####

## Indicar si se tiene instalado algún equipo de Broadlink para uso AA

array_IP = ['192.168.1.234', '192.168.1.235'] # Indicar IP's de equipos Broadlink
array_reles = [271, 281] # Indicar relés en el mismo orden que las IP's anteriores a asignar a los relés.

#####
#### Pantalla OLED ####
#####
OLED_salida1 = [0, 1, 2, 3] # secuencia de pantallazos modelo 1, 2, 3 o 4...0=logo
OLED_salida2 = [4] # secuencia de pantallazos modelo 1, 2, 3 o 4...0=logo
```

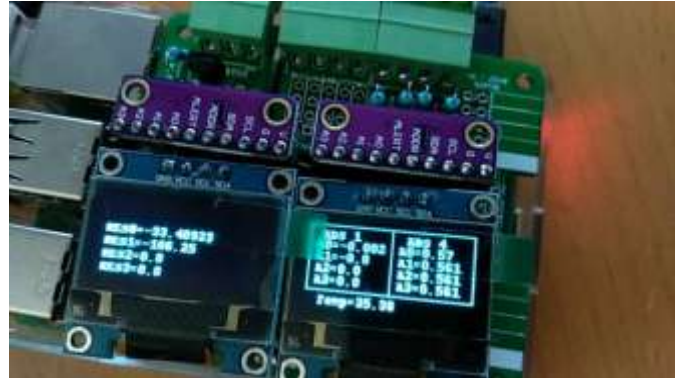
Pantallas OLED (OPCIONAL)

En la PCB versiones 1.2 o superiores y 3.1 se pueden poner hasta dos pantallas OLED

El programa reconoce automáticamente el número de pantallas instaladas

Se ha definido varios tipos de salida por pantalla:

- 0: el Logo de PVControl+
- 1: SOC, Vbat, Ibat, Vplac, Iplaca y los reles
- 2: Otra forma de mostrar los valores de Vbat, Ibat...
- 3: Situación de los reles en detalle (con nombre..)
- 4: Valor del SOC en grande...si llega a 100% se pone iluminado el fondo



Podemos definir la secuencia de pantallazos que queremos en cada OLED para que se vaya mostrando secuencialmente en esta parte

```
# -----
##### Pantalla OLED
OLED_salida1 =[0,1,2,3] # secuencia de pantallazos modelo 1, 2, 3 o 4...0=Logo
OLED_salida2 =[4] # secuencia de pantallazos modelo 1, 2, 3 o 4...0=Logo
# -----
```

En este ejm la OLED 1 mostraría secuencialmente los pantallazos 0,1,2,3 y en la OLED 2 siempre mostraría el SOC en grande

Se permite cualquier secuencia tipo por ejm suponiendo que capturamos datos cada 5 seg:

- [1,1,2] mostraría durante 10seg la pantalla 1 y 5seg la 2
- [1,2,1,3]...5 seg la 1, 5 seg la 2, 5 seg la 1, 5 seg la 3
- ...

SECCION PARAMETRIZACION SERVICIOS ADICIONALES

TELEGRAM (OPCIONAL)

Aquí estableceremos si queremos o no usar Telegram

```
##### Telegram
usar_telegram = 0 # 1 para usar ..... 0 para no usar
TOKEN = 'xxxxxxx' # ....cambiar por el que cada uno de de alta

# ID de Usuarios autorizados a mandar mensajes, los msg periodicos se mandan al primer declarado
Aut = [-11111, -22222, 33333]

cid_alarma = -22222 # Id Telegram a donde se enviara la foto/video de alarma

msg_periodico_telegram = 0 # 1 = Manda un mensaje resumen por Telegram cada Hora -- 0 = No manda mensaje
```

Yo recomiendo el uso de Telegram por lo que es lógico darse de alta y crease un Bot

Hay muchos tutoriales de cómo crearse un Bot, pongo aquí un extracto algo modificado, sacado de <https://elandroidelibre.lespanol.com/2018/02/como-crear-tu-propio-bot-de-telegram.html> (

1. Lo que primero que tienes que hacer es dar constancia a Telegram de que quieres crear un nuevo Bot. Para ello tenemos que mandar un mensaje al **BotFather** (@BotFather), en concreto el de «/newbot».
2. Posteriormente el propio bot te preguntará por el nombre que quieres para tu bot. Importante, tiene que acabar en Bot. Ejemplo: FV_Bot o Eal_Bot.
3. Si todo está correcto, te verificará la creación de tu Bot.
4. Ahora es el turno de configurar la privacidad de tu bot. Escribimos «/setprivacy» y posteriormente, el nombre de tu bot mencionándolo por su nombre «@Bot». El BotFather te responderá con las opciones y puedes hacer que sólo atienda a mensajes que lo mencionen o que empiecen por un «/» con el modo ENABLED. O recibir cualquier mensaje del grupo si marcamos la opción DISABLED... para **PVControl+ hay que elegir la opción DISABLED.**
5. Bot creado....con /mybots nos salen opciones y podemos ver el TOKEN que se ha generado para nuestro Bot y que debemos poner en el fichero Parametros_FV.py

También debemos saber que ID tenemos, cada chat de Telegram tiene un ID, por lo que nuestro usuario tendrá un ID, y si estamos en un grupo, dicho grupo también tendrá un ID

Por tanto, podemos definir desde que chat se le van a poder mandar ordenes a PVControl+, o a que ID se van a mandar los mensajes periódicos etc

El ID es un numero que será positivo si corresponde a un chat personal, o será un número negativo si es de un chat de grupo

Hay muchas formas de obtener el ID:

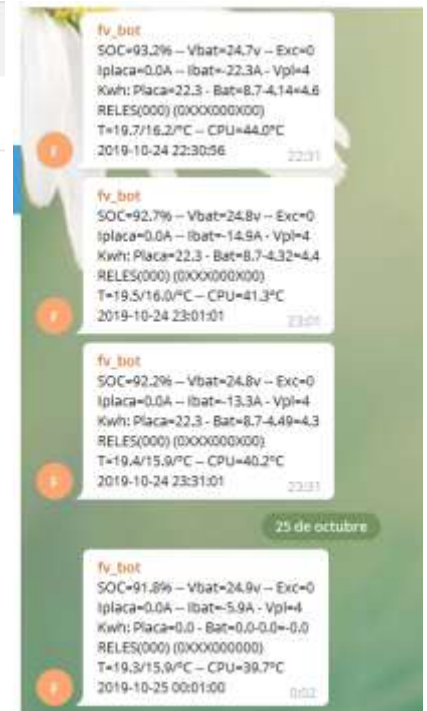
- Por ejm <http://iejo.pw/post/2018/03/17/Obtener-nuestro-ID-de-telegram>
- En mi caso uso mas el Bot llamado IDbot al que le podemos sacar fácilmente tanto el ID propio como de un grupo



Así, por ejemplo, en mi caso tengo un Grupo (formado por el Bot creado, mis hijos y yo) donde mando los mensajes periódicos para que nos llegue cada 30' el estado de la FV

Tengo otro Grupo donde me llegan las fotos/videos de alarma con la cámara instalada (según veremos en el apartado siguiente), etc

Lo mínimo necesario es poner en el archivo Parametros_FV.py el ID propio



CAMARA DE VIGILANCIA (OPCIONAL)

Se ha integrado en PVControl+ la posibilidad de utilizar una cámara para poder capturar fotos y videos (no es obligatorio usarlo)

Para ello estamos utilizando el programa "motionEye" instalado en la RPi

Hay muchos tutoriales en la web del uso de motion por si alguien quiere profundizar

En el caso de PVControl+, para evitar falsas alarmas también se ha integrado el análisis de la foto capturada usando la Inteligencia artificial que nos proporciona "clarifai", si queremos usar esta característica de análisis por inteligencia artificial debemos darnos de alta en Clarifai con una cuenta gratuita.

Esta característica es avanzada, no es muy difícil de usar, pero requiere crearse un workflow e ir "enseñando" a la inteligencia artificial con las fotos que se le manden, que consideramos alarma y que no, un pequeño detalle de cómo se usa y como utiliza la capacidad de reconocimiento de caras que tiene Clarifai lo veis aquí:

<https://adnsolar.eu/viewtopic.php?f=17&t=149&start=30#p6399>

En cualquier caso el uso de Clarifai no es obligatorio, aunque da un "toque" curioso a la funcionalidad

Si tenemos instalada la cámara, además de solicitar fotos o configurarlo via Telegram, tendremos un servidor web en el puerto 8765 de la IP de la Rpi (http://IP_rpi:8765) en donde podremos ver la imagen y configurar el programa motionEye



Los parámetros a configurar son los siguientes:

```
#----- Vigilancia por Camara con Motion y Clarifai
motion_telegram = 1 # 1 = Envia foto deteccion a Telegram
motion_clarifai = 1 # activa reconocimiento por Clarifai
api_key = 'XXXXXXXXXX' # Key Clarifai
workflow_id = 'PVControl-WF1' # Nombre del Workflow creado en Clarifai

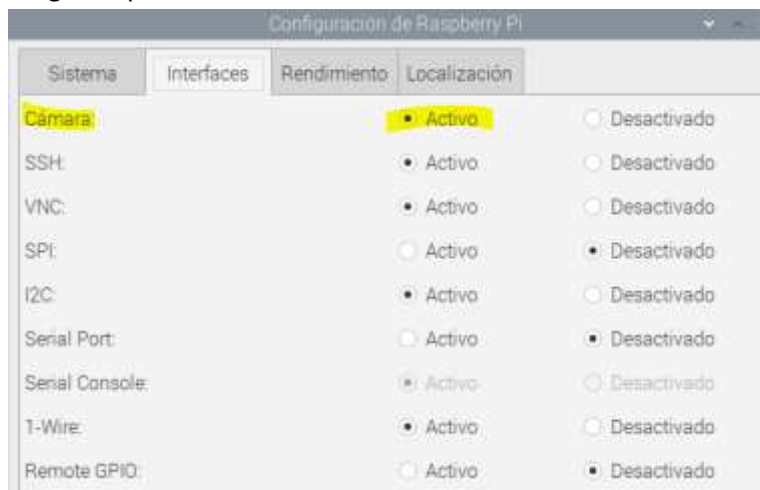
# fconfiguración horaria para motion
# dias de la semana 1-7. Horas 24 bits 0=no grabar 1=si
horario_alarma = {
    1: '111111110000000000000000',
    2: '111111110000000000000000',
    3: '111111110000000000000000',
    4: '111111110000000000000000',
    5: '111111110000000000000000',
    6: '111111110000000000000000',
    7: '111111110000000000000000'}

# -----
```

En horario_alarma se ha dividido cada día de la semana en 24 horas donde podemos poner si la detección esta activada o desactivada

Si tenemos la cámara propia de la Rpi, los pasos a seguir son:

- Asegurar que está activo el interface

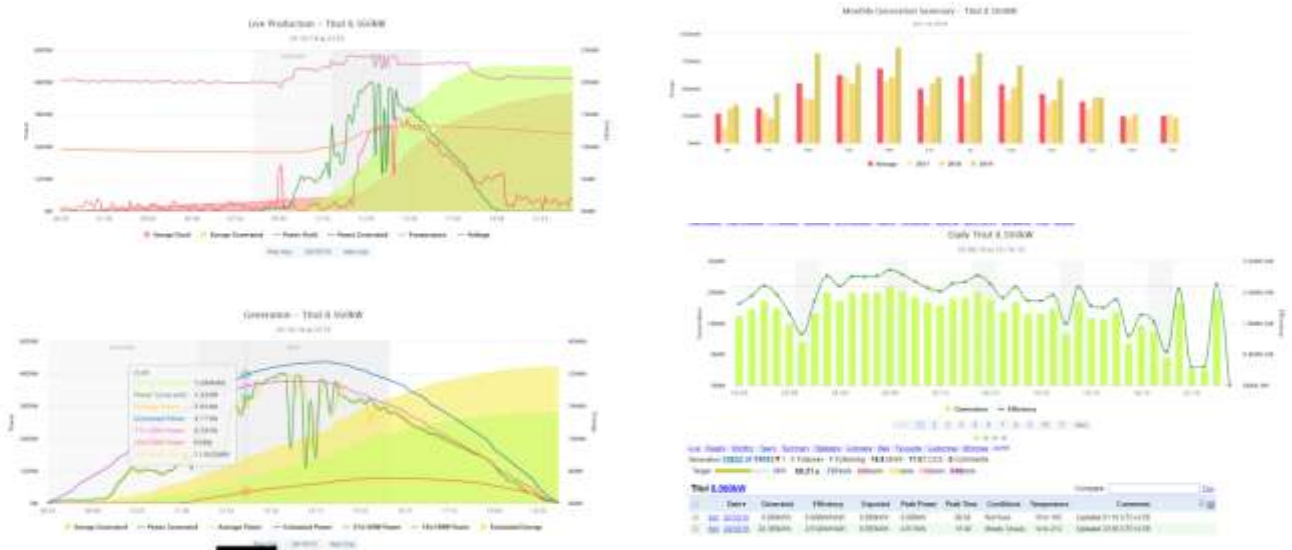


Si NO se tiene instalada la cámara hay que asegurarse de que dicha opción este desactivada... y además creo conveniente desactivar el programa motionEye con “`sudo systemctl disable motioneye`”

Si posteriormente instalamos la cámara, podremos sin problemas activar el interface y activar el servicio motioneye

PVOUTPUT (OPCIONAL)

Aunque PVControl+ tiene una web propia con gráficos etc, también se puede utilizar PVOutput que es una web australiana que nos permite mandarle los datos de nuestra instalación cada 5 minutos y tiene una gran cantidad de graficas, análisis, una app para android, etc



Para utilizar esta capacidad hay que darse de alta en dicha web y conseguiremos el TOKEN e id que debemos poner

```
##### PV_OUTPUT
usar_pvoutput = 0 # 1 para usar ..... 0 para no usar
pvoutput_key = "04b4b3fd09056888888abf3ce31cb18819d69a3a" #
pvoutput_id = "44444"
# -----
```

Asegurarse de que en el archivo crontab se tiene la entrada marcada en azul activa

Se puede modificar cambiando el archivo :

```
#
32 6 * * * root /home/pi/PVControl+/gestionbd.sh
28 * * * * root /home/pi/PVControl+/diario.sh
58 * * * * root /home/pi/PVControl+/diario.sh
10 0 * * * root python3 /home/pi/PVControl+/diario_ayer.py
23 5 * * * root python3 /home/pi/PVControl+/diario_ayer.py
00 * * * * root python3 /home/pi/PVControl+/fvbot_msg.py
01 3 * * * root python3 /home/pi/PVControl+/fv_soh.py
1-59/5 * * * * root python3 /home/pi/PVControl+/pvoutput_live.py
```

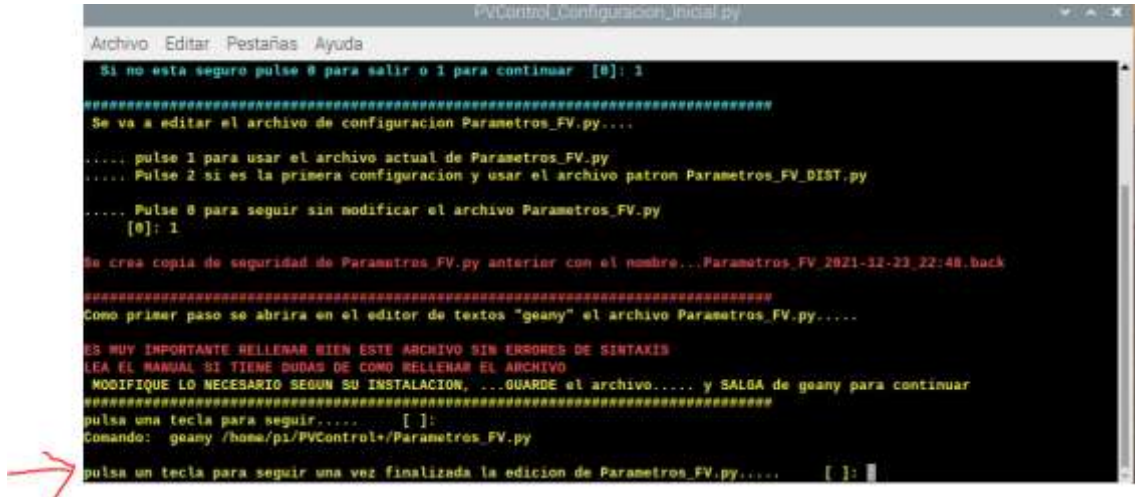
/home/pi/PVControl+/etc/cron.d/pvcontrol

Para editar este archivo que utiliza crontab se debe utilizar el usuario "root", esto se puede hacer simplemente tecleando desde terminal

sudo geany

para abrir el editor de textos geany con capacidad para modificar archivos de root

Una vez que terminemos de configurar el archivo Parametros_FV.py lo guardaremos y cerraremos Thonny para continuar con la instalación

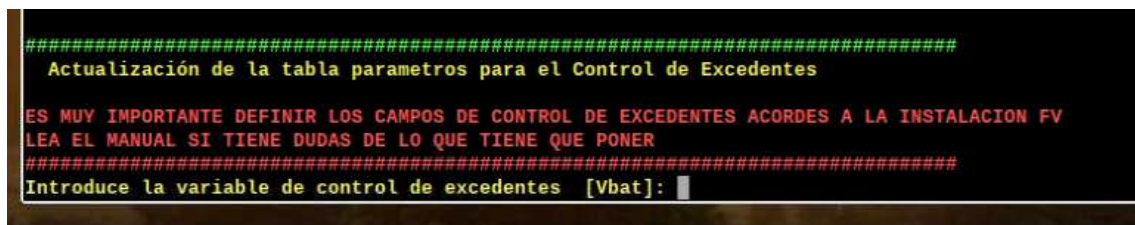


```

PVControl_Configuracion_inicial.py
Archivo Editar Pestañas Ayuda
Si no esta seguro pulse 0 para salir o 1 para continuar [0]: 1
=====
Se va a editar el archivo de configuracion Parametros_FV.py....
..... pulse 1 para usar el archivo actual de Parametros_FV.py
..... Pulse 2 si es la primera configuracion y usar el archivo patron Parametros_FV_DIST.py
..... Pulse 0 para seguir sin modificar el archivo Parametros_FV.py
[0]: 1
Se crea copia de seguridad de Parametros_FV.py anterior con el nombre...Parametros_FV_2021-12-23_22:48.back
=====
Como primer paso se abrira en el editor de textos "geany" el archivo Parametros_FV.py....
ES MUY IMPORTANTE RELLENAR BIEN ESTE ARCHIVO SIN ERRORES DE SINTAXIS
LEA EL MANUAL SI TIENE DUDAS DE COMO RELLENAR EL ARCHIVO
MODIFIQUE LO NECESARIO SEGUN SU INSTALACION, ...GUARDE el archivo.... y SALGA de geany para continuar
=====
pulsa una tecla para seguir..... [ ]:
Comando: geany /home/pi/PVControl+/Parametros_FV.py
pulsa una tecla para seguir una vez finalizada la edición de Parametros_FV.py.... [ ]:
  
```

En PVControl+ adicionalmente a la configuración del archivo Parametros_FV.py existe en la base de datos una tabla llamada "parametros" con diferentes campos para su configuración

La razón de estar en BD es que así se nos permite actualizar los valores "en caliente" sin tener que reiniciar el programa (control excedentes , etc)



```

#####
Actualización de la tabla parametros para el Control de Excedentes
=====
ES MUY IMPORTANTE DEFINIR LOS CAMPOS DE CONTROL DE EXCEDENTES ACORDES A LA INSTALACION FV
LEA EL MANUAL SI TIENE DUDAS DE LO QUE TIENE QUE PONER
=====
Introduce la variable de control de excedentes [Vbat]:
  
```

Iremos rellenando los distintos valores que nos pide de acuerdo a la instalación FV que tengamos



```

#####
Actualización de la tabla parametros para el Control de Excedentes
=====
ES MUY IMPORTANTE DEFINIR LOS CAMPOS DE CONTROL DE EXCEDENTES ACORDES A LA INSTALACION FV
LEA EL MANUAL SI TIENE DUDAS DE LO QUE TIENE QUE PONER
=====
Introduce la variable de control de excedentes [Vbat]:
Introduce valor de Vabs.. [14.4]:
Introduce valor de Vflat.. [13.2]:
Introduce valor de Veq.. [14.8]:
Introduce valor de Tabs.. [3600]:
UPDATE parametros SET sensor_PID = 'Vbat', objetivo_PID = '14.4', Vabs = '14.4', Vflat = '13.2', Veq = '14.8'

Introduce el valor del parametro Kp del control de excedentes [10]:
Introduce el valor del parametro Ki del control de excedentes [0]:
Introduce el valor del parametro Kd del control de excedentes [0]:
UPDATE parametros SET Kp = 10, Ki = 0, Kd = 0
tabla parametros configurada....
... Recuerda adaptar los parametros del control PID ....Kp, Ki, Kd para un control mas preciso
Introduce valor del SOC actual de la Bateria.. [100]:
  
```




Por último se necesita configura las paginas web para adaptarlas a la instalación que se tenga (con o sin baterías, con o sin control de celdas,...)

```
#####
PROGRAMA DE CONFIGURACION PAGINA PRINCIPAL WEB DE PVControl+

Este programa realiza una adaptacion de la WEB segun archivo Parametros_FV.py

#####

ATENCION.. SE CAMBIARAN LOS ARCHIVOS (version.inc y Parametros_Web.js

Si no esta seguro pulse 0 para salir o 1 para continuar [1]: █
```

```
#####
PROGRAMA DE CONFIGURACION PAGINA PRINCIPAL WEB DE PVControl+

Este programa realiza una adaptacion de la WEB segun archivo Parametros_FV.py

#####

ATENCION.. SE CAMBIARAN LOS ARCHIVOS (version.inc y Parametros_Web.js

Si no esta seguro pulse 0 para salir o 1 para continuar [1]: 1

Dada de alta BATERIA de 100.0 AH se configura la Web como ----- FV CON BATERIA a 12V -----

NO se ha dado de alta control de Celdas por lo que se configura la WEB como
----- FV CON BATERIA y SIN CONTROL CELDAS -----

1= Actualiza Web -- 0: No Actualiza [1]: █
```

```
WEB ACTUALIZADA
#####
--- SI HA MODIFICADO EL FICHERO Parametros_FV.py ES NECESARIO REINICIAR PVControl+..
#####

Pulsa 1 para reiniciar PVControl+ o 0 para reinicio manual [1]: █
```

Si pulsamos 1 se iniciaran los servicios de acuerdo a la parametrizacion seleccionada

```
#####

Pulsa 1 para reiniciar PVControl+ o 0 para reinicio manual [1]: 1
1
##### Activando Sevicios #####
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/sdm120c.service
b'systemd[1]: Started PVControl+ - Lectura SDM120C.\n'
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/fv_oled.service
b'49:57 rpi systemd[1]: Started PVControl+ -- OLED.\n'
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/huawei.service
b' systemd[1]: Started PVControl+ - Lectura HUAWEI.\n'
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/fv.service
█
```



```

Procesando archivo... /home/pi/PVControl+/etc/systemd/system/motion.service
b' :26 rpi systemd[1]: Started PVControl+ -- Motion.\n'
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/fv_ads.service
b' 02:50:28 rpi systemd[1]: Started FV lectura ADS.\n'
Procesando archivo... /home/pi/PVControl+/etc/systemd/system/victron.service
b'systemd[1]: Started PVControl+ - Lectura VICTRON.\n'

##### Activando WEB PVControl+ #####
---- OK ----

##### Activando Procesos CRONTAB #####
---- OK ----

##### Proceso Completado #####

--- SERVICIOS PVControl+ REINICIADOS ---
Pulsa INTRO para salir []:
  
```

El proceso se ha completado...pulsamos INTRO para salir

Con esto ya se debería poder ver la web funcionando correctamente y monitorizando los datos simplemente con poner en el navegador la IP de la Raspberry



Si queremos ajustar mas parámetros o nuestra instalación es algo mas compleja (varios equipos distintos, etc) hay que editar el archivo Parametros_FV.py ubicado en /home/pi/PVControl+ y cambiar lo que queramos según lo explicado

Asimismo si queremos personalizar la web (escala, valores y franjas de colores de cada reloj,..) editaremos el archivo Parametros_Web.js ubicado en /home/pi/PVControl+/html y cambiaremos lo que queramos

2.7.ADAPTACION de la WEB

Como hemos comentado las distintas páginas web creadas en PVControl+ se pueden adaptar a los valores de cada instalación

Esta adaptación se puede hacer en un primer paso de forma cómoda utilizando el programa descrito en el apartado 2.6 “Configuración básica”

También se puede realizar una adaptación más ajustada a nuestra FV editando el archivo **Parametros_Web.js** ubicado en **/home/pi/PVControl+/html**

En dicho archivo podremos ir cambiando los valores que definen cada grafico

PVControl+ usa las librerías gráficas de Highchart que hay documentación de sobra en internet, luego si se quiere realizar cualquier tipo de grafica es posible

```
Parametros_Web.js X
1
2 // Pagina inicio.php
3
4 // Reloj Vbat
5 Vbat_min = 22;
6 Vbat_bajo_amarillo = 24;
7 Vbat_verde = 25.4;
8 Vbat_alto_amarillo = 28;
9 Vbat_alto_rojo = 30;
10 Vbat_max = 32;
11
12 // SOC
13 SOC_min=60;
14 SOC_max=100;
15
16
17 // reloj Temp
18 Temp_bat_min = -10;
19 Temp_bat_baja = 10;
20 Temp_bat_normal = 20;
21 Temp_bat_alta = 30;
22 Temp_bat_max = 50;
23
24 Temp_rpi_min = 0;
25 Temp_rpi_normal = 40;
26 Temp_rpi_alta = 60;
27 Temp_rpi_max = 100;
28
29 // Reloj Consumo
30 Consumo_watios_min = 0;
31 Consumo_watios_amarillo = 3000;
32 Consumo_watios_rojo = 4000;
33 Consumo_watios_max = 5000;
34
35 Consumo_amperios_min= 0;
36 Consumo_amperios_amarillo = 125;
37 Consumo_amperios_rojo = 166;
38 Consumo_amperios_max= 210;
39
40 // Reloj Ibat/Iplaca
41 Intensidad_min = -130;
42 Intensidad_descarga_amarillo = -40;
43 Intensidad_carga_rojo = 80;
```

2.8.Tabla parámetros en la Base de Datos

Como se ha comentado, para mayor funcionalidad la parametrización de la instalación de PVControl+ se ha dividido en tres partes

- Archivo **Parametros_FV.py** : según lo que hemos visto
- Archivo de parámetros de la Web **Parametros_Web.js**
- Tabla **parametros** en la Base de datos

El concepto de cada uno de estos archivos es:

Si se modifica algo en el archivo **Parametros_FV.py** es conveniente reiniciar la Rpi para que se **active** (o dependiendo de la parte que se modifique reiniciar uno o todos los servicios fv, fvb0t y hibrido con “sudo systemctl restart fv” por ejm para reiniciar el servicio fv)

Si se modifica algo en el archivo **Parametros_Web.js** se debe refrescar la pagina web para ver los cambios

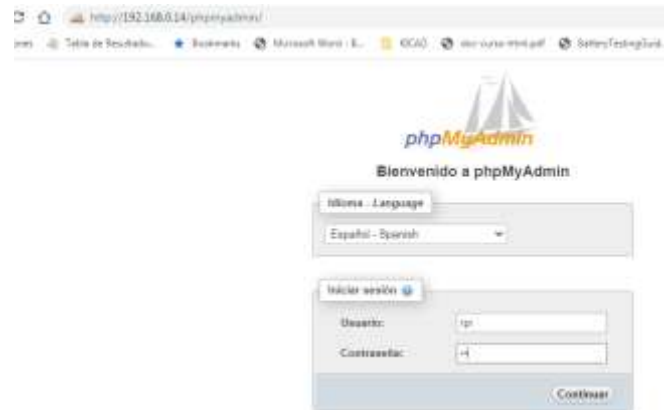
Sin embargo lo que se cambia en la tabla **parametros** de la Base de Datos afecta inmediatamente sin necesidad de reiniciar

Esto nos permite ir cambiando valores “en caliente” y ver cómo reacciona

Veamos cuales son los campos que se pueden cambiar en la tabla parámetros

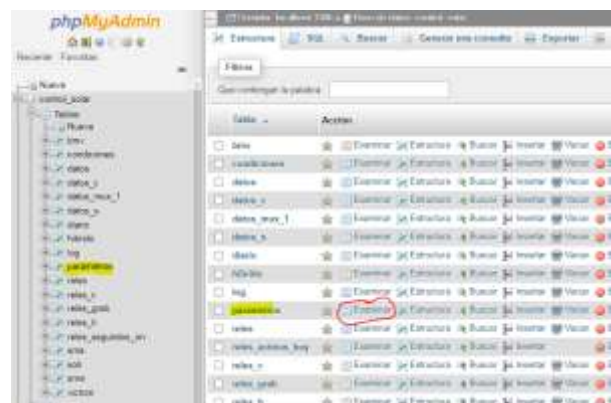
Abrimos la tabla parámetros con phpmyadmin poniendo en el navegador

- ip de la Rpi/phpmyadmin
- usuario/clave es rpi/fv



accedemos a la tabla parámetros pulsando sobre “examinar

y vemos lo siguiente:





grabar_datos	grabar_reles	t_muestra	n_muestras_grab	nuevo_soc valor distinto de 0 actualiza	objetivo_PID Valor objetivo a conseguir por el control PID	sensor_PID Variable de control PID	Kp Constante proporcional PID	Ki Constante integral PID	Kd Constante derivativa PID	Mod_bat	Vfot Valor de flotación	Vabs Valor de absorción	Tabs Tiempo de absorción en segundos	Vequ Valor de equalización	Tequ Tiempo de equalización en segundos	Icola Intensidad de cola para Compensación de Absorción	coef_temp Coeficiente Compensación Temperature para Vfot y
S	S	5	1	0	27.2	Vbat	25	0	25	BULK	27.2	28.8	3600	29.6	3600	4	0

- **grabar_datos**: Poniendo S o N grabaremos o no en la tabla “datos” las capturas de los valores
- **grabar_reles**: Poniendo S o N grabaremos o no los cambios de estado de los rele
- **t_muestra**: define el tiempo entra cada muestra, en el caso de usar un hibrido no se recomienda valores por debajo de 4 o 5 seg dado la lenta respuesta de su interfaz USB
- **n-muestras_grab**: se define cada cuantas muestras se grabara los valores en la tabla datos, asi por ejm :
 - t_muestra=2 y n_muestras_grab=3: se capturan los datos cada 2 seg y se almacenan cada 6 seg
 - t_muestra=1 y n_muestras_grab=5: se capturan los datos cada 1 seg y se almacenaran cada 5 seg (esta es la configuración que yo tengo capturando desde los ADS) para tener un control de excedentes que reaccione rápido
 -

Si queremos hacer un análisis de detalle por algún tiempo, podemos hacer que se guarden todas las capturas a una velocidad de t_muestra **definiendo en Parametros_FV.py la variable “grabar_datos_s” a una condición que se cumpla cuando queramos grabar los datos a alta velocidad**

En ese caso se guardaran las capturas en la tabla “datos_s” cada t_muestra y, como siempre, en la tabla “datos” cada “t_muestra*n_muestras”



La grafica Hist_8h coge los datos de la tabla datos_s

La grafica Hist_1 coge los datos de la tabla datos

La grafica Hist_3 y Hist_mes cogen los datos de la tabla datos-c (la tabla datos_C es una tabla resumen que se crea automáticamente desde la tabla datos en donde se guarda un registro cada 5 minutos

...

- **nuevo_soc:** si ponemos un valor distinto a cero, se actualizara el SOC de la instalación FV
- **objetivo_PID:** se indica el valor a conseguir por el control de excedentes
- **sensor_PID:** se indica la variable que se utiliza para el control de excedentes (Vbat, Vplaca, Ibat, etc)
- **Kp , Ki, Kd:** se indica las constantes del control PID que se utiliza para controlar los excedentes
- **Vabs , Vflot y Tabs:** se consigna el valor de Absorción, Flotación y el Tiempo de Absorción para la carga de la batería
- **Modo_Bat:** Nos indica el estado de carga de la Bateria (BULK, ABS, FLOT..) .. se puede cambiar manualmente pero normalmente será actualizado automáticamente por PVControl+

- Vequ, Tequ: Voltaje y tiempo de ecualización de baterías

Sin implementar actualmente

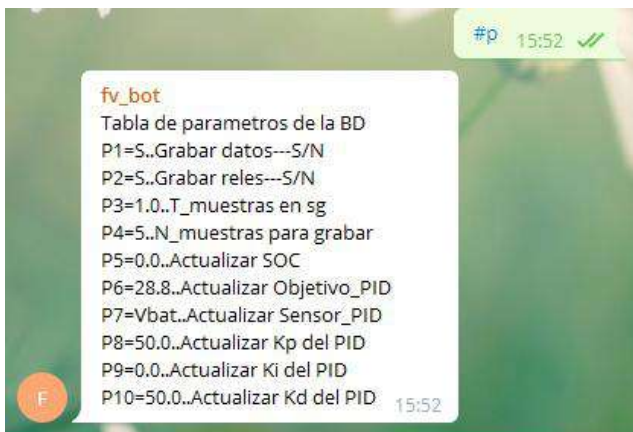
No obstante si se quiere ecualizar simplemente cambiar el valor de Vabs o Vflot a que se desee

- Icola: para futuros usos
- Coef_temp: para futuros usos

Así pues, si se consigna Vbat como “sensor_PID” y ponemos ir cambiando dinámicamente el valor objetivo a conseguir

Esta característica es muy potente y nos permite incluso poder controlar la carga de las baterías a nuestro antojo con excedentes independientemente de los reguladores que tengamos simplemente poniendo el valor de flotación y absorción en los reguladores a un valor algo superior al que consignemos en PVControl+

Si tenemos la posibilidad de usar Telegram, los datos de la tabla parámetros se pueden actualizar mandando un mensaje



Con todo esto el control de carga y de excedentes es más que aceptable como se puede ver en la curva diaria de carga en donde se aprecia perfectamente cómo se va adaptando la señal PWM a la potencia disponible en las placas



PVControl+ permite usar y controlar cualquier número de relés locales o via wifi

En el caso de usar excedentes, lo mas lógico es usar relees SSR para poder encenderlos en un % y que se adapten bien a la energía sobrante disponible.... En el caso de la grafica de arriba se esta usando 3 relees con SSR (Termo agua, calefactor1, calefactor2)

3. Test y Resolución de Problemas

Lo primero que hay que hacer una vez parametrizado PVControl+ es comprobar que esta funcionando correctamente en la web **pestaña Equipos/Equipos**



id_equipo	tiempo	estado
_PVControl+	2022-05-03 23:30:43	{"PVControl+": "OK"}
ADS1	2022-05-03 23:30:43	{"Vbat": 25.168, "Aux1": 25.586, "Vplaca": 4.358, "Aux2": 12.88}
ADS4	2022-05-03	{"Vbat": 10.484, "Inlaca": 0.138}

Además esta pestaña nos dará información del estado de captura de los distintos equipos configurados, por lo que es una buena forma de identificar la mayoría de los problemas

Si ya se necesita un análisis mas avanzado, se puede ver el estado de los distintos servicios activos usando el comando “sudo systemctl status XXXX” para ir viendo la situación de cada servicio, por ejemplo para el servicio principal “fv” sería:

- **sudo systemctl status fv**

```
pi@rpi:~/PVControl+ $ sudo systemctl status fv
● fv.service - Control Fotovoltaico
   Loaded: loaded (/home/pi/PVControl+/etc/systemd/system/fv.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-04-10 17:09:23 CEST; 3s ago
     Main PID: 31636 (python3)
       Tasks: 3 (limit: 2229)
      Memory: 12.7M
      CGroup: /system.slice/fv.service
              └─31636 /usr/bin/python3 /home/pi/PVControl+/fv.py

abr 10 17:09:23 rpi systemd[1]: Started Control Fotovoltaico.
```

Igual con el resto de los servicios principales:

- **sudo systemctl status hibrido** si se esta usando un Hibrido
- **sudo systemctl status fvbot** si se esta usando un bot de Telegram
- **sudo systemctl status fv_temp** si se esta usando un sensor de temperatura
- **sudo systemctl status fv_oled** si se esta usando unas pantallas OLED con la PCB
- **sudo systemctl status fv_mux** si se esta usando el multiplexor de la PCB
-

También se puede usar el pequeño programa que se ha puesto en el escritorio para listar los programas de PVControl+ activos

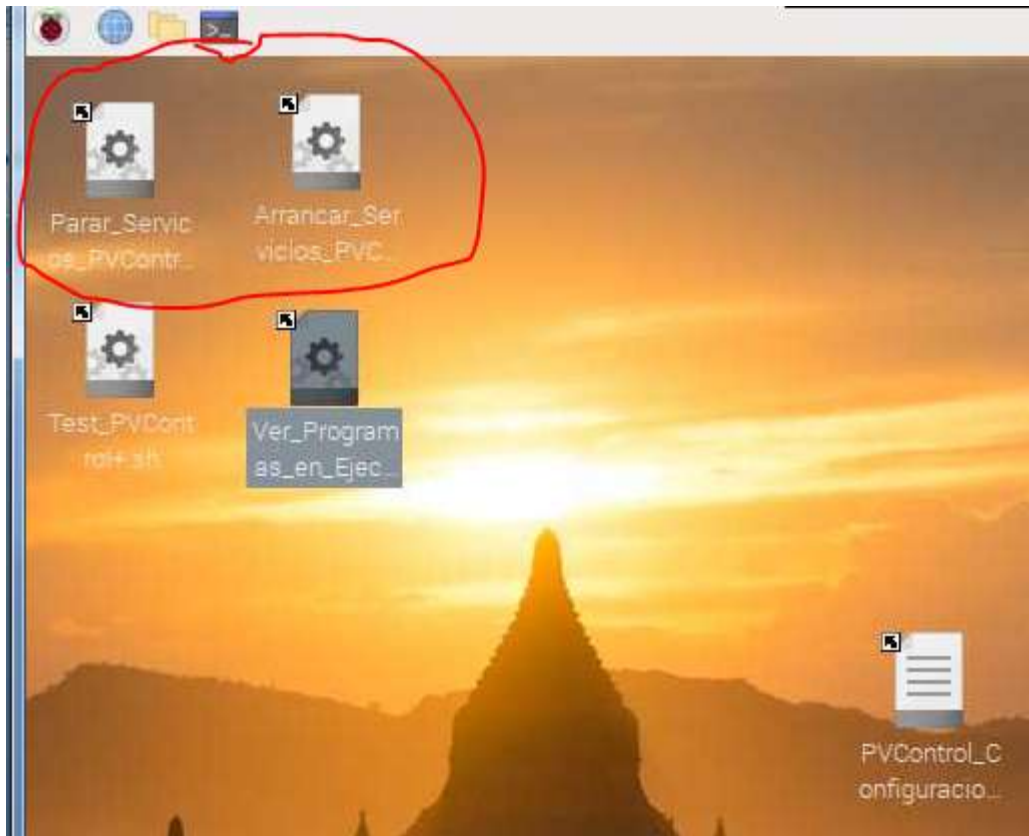


Si el programa fv.py no esta ejecutándose, PVControl+ no estará funcionando y por ejemplo la web no se actualizará

Por tanto lo primero es ver si hemos parametrizado correctamente el fichero Parametros_FV.py, cada vez que se cambia algo en dicho fichero y se guarda es mas seguro reiniciar los servicios con

- sudo systemctl restart fv
- sudo systemctl restart hibrido
-

También se puede utilizar una pequeña utilidad que se ha puesto en el escritorio para parar o reiniciar todos los programas de PVControl+



Si una vez se ha repasado (varias veces) que la instalación y parametrización se ha hecho correctamente y PVControl+ sigue sin funcionar lo mejor es parar el servicio y ejecutarlo de forma manual para ver donde está el fallo

Para ello los pasos son:

- 1.- Parar el servicio fv `sudo systemctl stop fv`
- 2.- Ir a la carpeta `/home/pi/PVControl+`
- 3.- Ejecutar `sudo python3 fv.py`
- 4.- Ver lo que sale y actuar en consecuencia (arreglándolo o llamando a quien sabe cómo arreglarlo ☺)

```
pi@rpi:~ $ sudo systemctl stop fv
pi@rpi:~ $ cd PVControl+
pi@rpi:~/PVControl+ $ sudo python3 fv.py
Arrancando_PVControl+
DEBUG= 0
```


4.Integración con Home Assistant

La imagen actual de PVControl+ no lleva preinstalado Home Assistant (HA) para ahorrar espacio y disminuir el tamaño del fichero de la imagen.

la instalación es muy simple de hacer si se quiere activar HA ya que corre en docker por lo que simplemente ejecutaremos:

- `python3 PVControl_Instalacion_HomeAssistant.py`

que ejecutará el comando docker para la descarga y puesta en marcha de HA

Una vez activado podremos acceder a HA desde el navegador con la IP de la Raspberry en el puerto :8123

HA tiene muchos tutoriales y manuales por la web, por lo que se recomienda su uso



Se ha realizado una pequeña integración HA-PVControl y, por ejemplo, se puede ver la web de PVControl+ en la pestaña PVCONTROL+ creada





También, a través de MQTT, se pueden pasar a HA los valores capturados por PVControl+ , por ejemplo se incluye unos ejemplos de integración usando el **archivo configuration.yaml** ubicado en la carpeta **/home/pi/PVControl+/HA**

Es necesario que los topic MQTT que publica PVControl+ y a los que se suscribe HA se llamen igual, luego comprobar que lo que se define en Parametros_Fv.py y configuration.yaml es congruente

```
configuration.yaml x
1  # Configuración PVControl+
2  sensor:
3    - platform: mqtt
4      name: "SOC"
5      device_class: "voltage"
6      state_topic: "PVControl/DatosFV"
7      unique_id: "PVControl/FV/SOC"
8      unit_of_measurement: "%"
9      icon: "hass:battery"
10     value_template: "{{ value_json.SOC }}"
11
12     - platform: mqtt
13       name: "SOC_min"
14       state_topic: "PVControl/DatosFV"
15       unique_id: "PVControl/FV/SOC_min"
16       unit_of_measurement: "%"
17       value_template: "{{ value_json.SOC_min }}"
18
19     - platform: mqtt
20       name: "SOC_max"
21       state_topic: "PVControl/DatosFV"
22       unique_id: "PVControl/FV/SOC_max"
23       unit_of_measurement: "%"
24       value_template: "{{ value_json.SOC_max }}"
25
26     - platform: mqtt
27       name: "Wplaca"
28       device_class: "power"
```

5. Integración Tasmota

Otra integración realizada en PVControl+ ha sido poder utilizar los ESP32/ESP01 con TASMOTA tanto para controlar relees como para poder leer los sensores que se les pongan

Tasmota a su vez se integra bastante bien con HA, por lo que hay muchas posibilidades de utilización

Igualmente para la programación y uso de Tasmota hay muchos tutoriales en la web y, además, **se ha editado un manual específico para su uso dentro de PVControl+**



id_equipo	tiempo	sensores
PV	2021-09-25 14:56:49	{ "Vbat": 25.2, "Batt": 1.0, "What": 25, "Whp_kat": 422, "Wbat_kat": 0, "Vbat_min": 25.2, "Vbat_max": 25.2, "DS": 500.0, "SOC": 100.0, "SOC_min": 100.0, "SOC_max": 100.0, "Mod_bat": "BULK", "Tbat": 0, "Tbat_bulk": 0, "Vplaca": 2.26, "Vplaca": 2.0, "Wplaca": 50, "Wb_placa": 752, "Vred": 0.0, "Wred": 0.0, "Whp_red": 0, "Wbat_red": 0, "Vred_min": 0.0, "Vred_max": 0.0, "EFF": 0.0, "EFF_min": 0.0, "EFF_max": 0.0, "Wconsumo": 25, "Wh_consumo": 330, "Temp": 27.12, "PWM": 72 }
RELES	2021-09-25 14:56:49	[{"Rela331(PRG-P0)", 0}, {"Tasmota(PRG-P1)", 72}]
TEMP	2021-09-25 14:56:49	[{"Temp0": 28.75, "Temp1": 28.88, "Temp_cpu": 53.536}]
MQTT	2021-09-25 14:56:14	{ "id": "PVControl+ TASMOTA", "Tasm": "2021-09-25T14:56:14", "DS18B20-1": {"id": "00000002A4B3", "Temperature": 28.7, "DS18B20-2": {"id": "0446CB35EFF", "Temperature": 28.5}, "ESP32": {"Temperature": 51.1, "TempUnit": "C"} }
MUX	2021-09-24 18:27:50	[{"Name": "C1", "C2", "C3", "C4", "C5", "C6", "C7", "C8", "C9", "C10", "C11", "C12", "C13", "C14", "C15", "C16", "Max": [1.3887, 1.3984, 1.3887, 1.5148, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0], "Min": [1.3697, 1.3984, 1.247, 1.5148, 0.0, 0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0, -0.0]}]
HIBRIDO	2021-09-24 18:10:51	{ "Vbat": 25.2, "Batt": 0.0, "Battp": 0.0, "Battm": 0.0, "Vplaca": 0.0, "Vplaca": 0.0, "Vplaca": 0.0, "Vbat": 407.0, "PACW": 0.0, "PACVA": 69.0, "Temp": 31.0, "Flot": 0, "OnOff": 1 }