# DISEÑO Y MONTAJE DEL PROTOTIPO

Resumiendo todo lo estudiado en los capítulos anteriores del proyecto:

El dispositivo electrónico va a estar formado por una Raspberry Pi Zero que va a ser alimentado por una sola celda de batería LiFePo4 mediante un HAT llamado LiFePo4wered/Pi, que está diseñado para dicha tarea.

Dicha celda será cargada sin contacto directo, mediante un cargador de inducción que nos proporciona la tensión necesaria para cargar la batería.

Por un lado vamos a tener el montaje del sistema de carga, y por otro lado, el montaje del sistema de alimentación.

## SISTEMA DE CARGA

Para el funcionamiento de la carga, es necesario un transformador, para transformar la corriente alterna (AC) que proporciona cualquier enchufe convencional de una vivienda (100-240V) en una salida de 9 a 12V de corriente continua (CC).

Éste irá conectado a la entrada del circuito inductor. (En este proyecto se utiliza un transformador que ofrece 12V a la entrada del circuito inductor).

Una vez ofrecida la tensión necesaria al circuito inductor, se conecta la parte del circuito receptor al módulo de carga de la batería, y cuando las bobinas se unen comienza la carga.

FOTO

El módulo de carga, ya integrado en el HAT de alimentación, se basa en un proyecto llamado LiFePO4wered/USB también diseñado por silicognition LLC y de venta en tindie.

Data sheet disponible en ANEXO II.

Es un módulo de carga diseñado para una correcta carga de baterías LiFePO4 , que permite a LiFePo4wered/Pi estar en estado de carga mientras se encuentra en funcionamiento y alimentando la Raspberry Pi, evitando sobrecargas y proporcionando seguridad a la batería.



1. Módulo de carga. Fuente: Tindie.

Características.

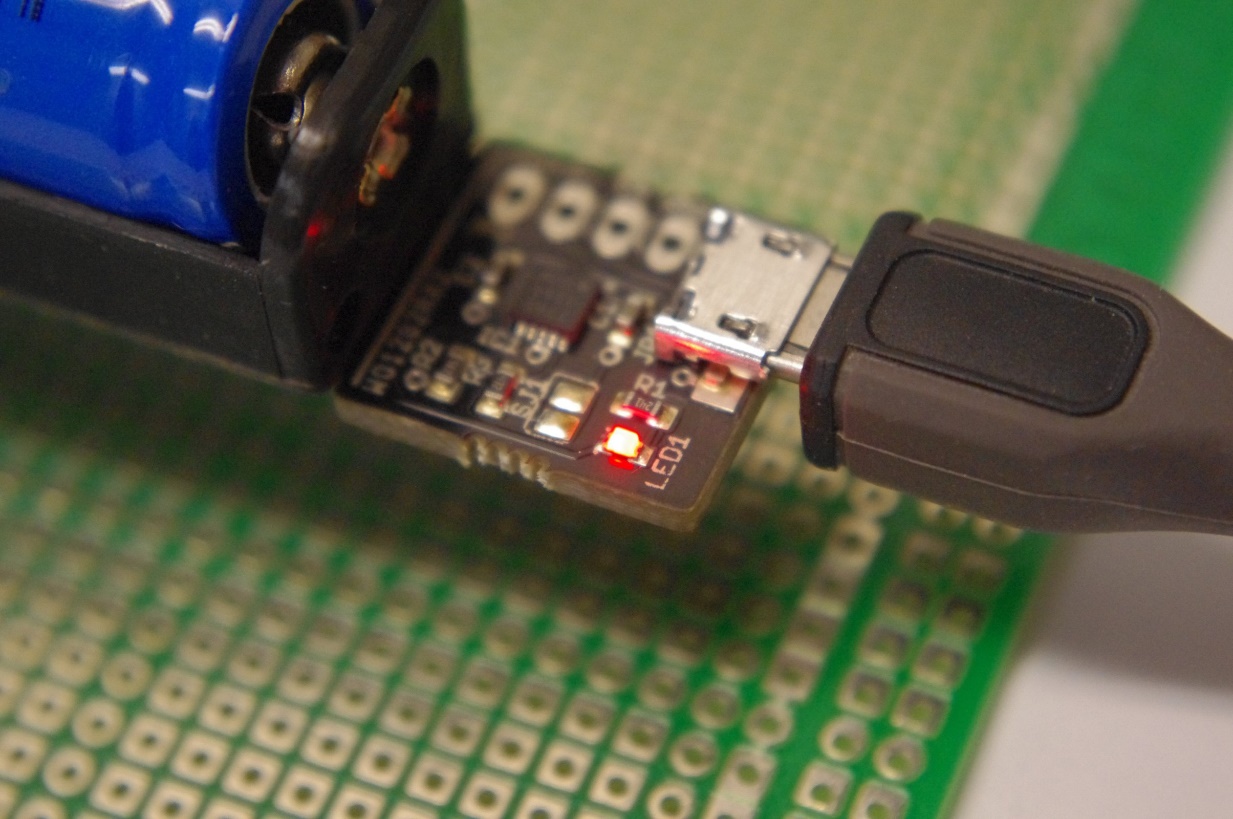
Cuando el cargador está desconectado ofrece una fuga ultra baja de 0,5µA.

Ofrece una carga mediante un conector de carga micro USB, por lo que tiene compatibilidad con cargadores habituales de teléfonos móviles.

La tensión de entrada puede oscilar entre 4.5V y 6.5V, siendo el valor más corriente a usar una tensión de 5V (que es la tensión que ofrece a la salida el cargador de inducción utilizado).

La corriente de carga es 245mA tardando en cargar una batería totalmente descargada 2.5h, pero ofrece la posibilidad de puentear SJ1 ofreciendo una corriente de carga de 480mA, tardando la mitad en cargar (carga rápida).

Tiene una tensión mínima de protección de 1,45V, así como una corriente mínima de corte de 25mA.



1. Conexión carga micro USB. Fuente: Tindie.

## SISTEMA DE ALIMENTACIÓN

Para la conexión del HAT a la Raspberry Pi, es necesario conocer cada una de las entradas y salidas que nos ofrecen y como van conectadas para un correcto funcionamiento.

Los pines del modelo de Raspberry Pi Zero son los mismos del modelo B+. Son conocidos como GPIO.

Consta de 40 pines de entrada y salida y cada uno de ellos está configurado para una función específica.

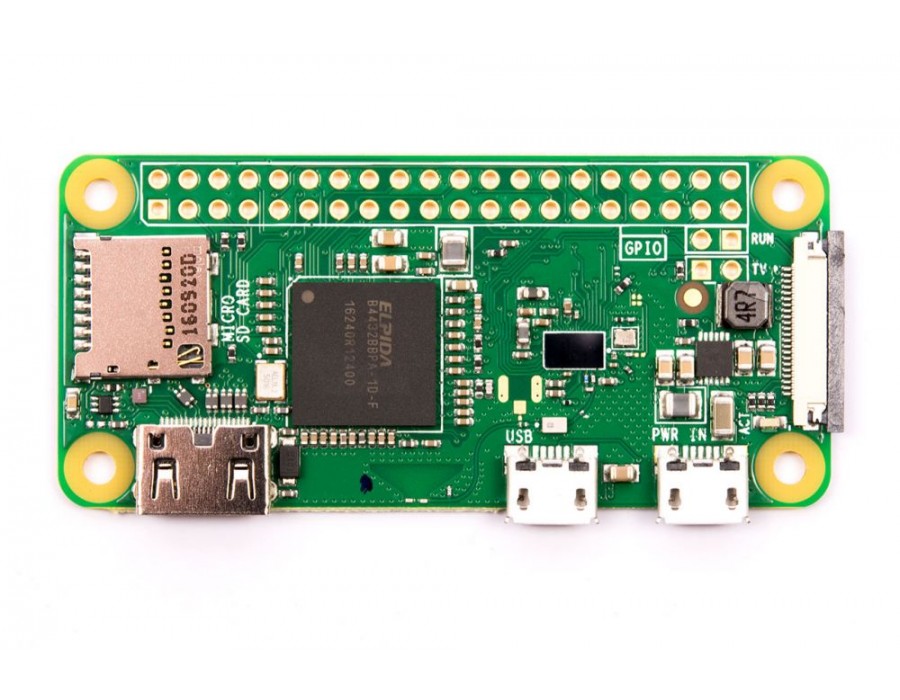
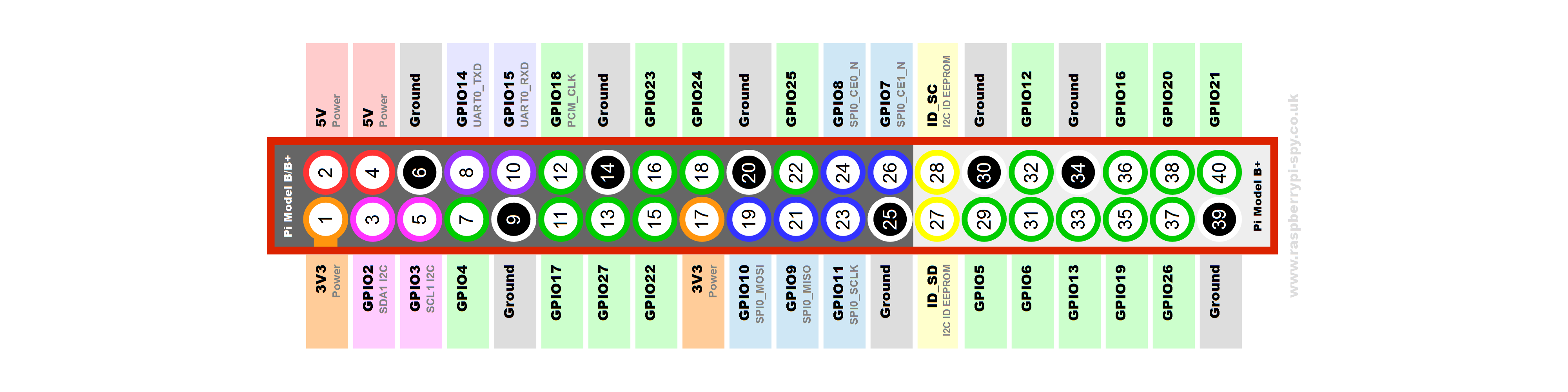
### INSTALACIÓN HARDWARE

En el ANEXO I se puede ver que los pines utilizados por este HAT van a ser los 8 primeros pines de la Raspberry Pi Zero.

Para conectar el módulo es necesario apagar Raspberry Pi, desconectar cualquier tipo de alimentación que pueda tener, y conectar el LiFePo4wered/Pi a los 8 primeros pines de Raspberry Pi Zero. Una vez instalado el modulo, basta con tener presionado el botón táctil unos segundos (2s) y se enciende.

Este módulo espera unos segundos antes de encender para evitar que se encienda de forma accidental.

¿Qué función tienen los pines?



1. GPIO Raspberry Pi Zero

* PIN1 – 3.3V

Es un pin de alimentación a 3.3V que tiene una corriente máxima de 50mA.

El HAT lleva este pin desconectado.

* PIN2/PIN4 – 5V

Es un pin de alimentación a 5V que va conectado directamente con la alimentación de entrada de la Raspberry Pi.

En el HAT es el pin mediante el cual el LiFePO4wered/pi proporciona energía a la Raspberry Pi.

* PIN3 – SDA

Es uno de los pines de comunicación I2C.

I2C (Inter Integrated Circuit) son unos pines de Raspberry que sirven para comunicar controladoras con diferentes periféricos. Es un bus maestro- esclavo, y lo forman dos líneas de señal (la señal de reloj CLK y la línea de datos SDA).

El Pin 3 es el pin de datos de I2C utilizado por Raspberry para comunicarse con LiFePO4wered/Pi.

* PIN5 – SCL

Es la señal de reloj del bus I2C que utiliza Raspberry para comunicarse con LiFePO4wered/Pi.

* PIN6 – GND

Es el pin de referencia a tierra. Hay 8 pines de masa conectados entre sí, pudiendo usar cualquiera de ellos.

* PIN7 – GPIO4

En el HAT va desconectado.

* PIN8 – TX0

Este pin se puede usar como pin de transmisión UART (receptor/transmisor universal asíncrono) o como serial para mostrar una consola.

Pero LiFePO4wered/Pi no lo usa como tal, sino que solo monitorea el nivel lógico a través de él.

Durante el estado de apagado se controla esta señal, y cuando esta deja de conducir, la Raspberry Pi habrá terminado su apagado.

FOTO

### INSTALACIÓN SOFTWARE

Para la recogida de datos sobre el estado de la betería que ofrece este HAT, es necesaria la instalación de bibliotecas disponibles en GITHUB.

Pero antes de ver eso, es necesario saber el sistema operativo con el que se va a trabajar.

#### Linux

El sistema operativo libre Linux o GNU/Linux, es el software necesario para que el ordenador te permita realizar diversas acciones (editar textos, internet…).

Es posible utilizar Linux con una interfaz gráfica como Windows o MacOS, o mediante una ventana de comandos como DOS.

Raspbian o Debian son distribuciones de este sistema operativo.

Para la utilización de Raspberry Pi, Raspbian es el sistema operativo oficial, y en la página oficial se ofrece NOOBS que es un instalador de Raspbian para poner en funcionamiento la Raspberry Pi.

Una vez instalado y puesto en marcha, se va utilizar este sistema desde la ventana de comandos, que te permite una fácil navegación entre directorios, instalar, crear archivos, programas, leer y escribir texto, etc. A través de unos comandos básicos para realizar las diferentes acciones necesarias.

#### Instalación de Bibliotecas

Para obtener los datos facilitados por LiFePO4wered/Pi sobre el estado de energía de la batería es necesario ejecutar el software proporcionado.

Este software proporciona acceso a unas bibliotecas compartidas que da un acceso fácil al LiFePO4wered/Pi con comandos de usuario, así como nos proporciona un programa CLI que permite al usuario configurar desde la línea de comandos.

El paquete del software se puede encontrar el Github:

<https://github.com/xorbit/LiFePO4weredPi>

Se ejecuta primero el siguiente comando para asegurar que las herramientas de compilación y Git están instaladas:

Sudo apt-get –y install build-essential git

Luego, se clona el paquete de software de Github:

git clone <https://github.com/xorbit/LiFePO4weredPi.git>

Y se entra en el directorio recién creado llamado LiFePO4wered-Pi con el comando “cd”.

Dentro de este directorio se construye el software ejecutando:

python build.py

Y se instala ejecutando:

sudo ./INSTALL.sh

Esto no solo instala el software en el sistema sino que realizará la configuración necesaria para su funcionamiento como habilitar el bus I2C y el GPIO UART.

# ENSAYOS

## RECOGIDA DE DATOS DE LA ALIMENTACION DE RASPBERRY PI

En este capítulo se van a describir los programas usados, tanto para la recogida de datos de la alimentación de la Raspberry PI Zero, como para los datos de la batería cuando esta está alimentando a la Raspberry Pi Zero.

El LiFePO4wered/Pi permite acceder a un conjunto de registros desde la Raspberry Pi a través de bus I2C. Por defecto la dirección del dispositivo de 7 bits es 0x43 pero puede cambiarse en caso de conflicto.

Como se ha comentado en el capítulo anterior, se ha instalado con el paquete del software una herramienta llamado lifepo4wered-cli que permite obtener y establecer valores de los registros que se quieren conocer o establecer, sin tener que conocer detalles de implementación.

Con esta herramienta, basta con escribir el comando lifepo4wered\_cli get y el registro que se quiera obtener, para que devuelva ese valor deseado. O bien, lifepo4wered-cli set, para establecer un valor (mostrando los datos en mV o segundos).

Enumeración de los registros a los que se puede acceder obtenidos del ANEXO I:

* I2C\_ADDRESS

Dirección del bus de 7 bits del dispositivo.

Valor por defecto 0x43.

* LED\_STATE

Se usa para configurar el estado del LED cuando Raspberry está encendida.

Esto se hace para ahorrar energía o para indicar el estado de un programa.

Los estados posibles son: LED apagado, LED encendido, LED pulsado o flash rápido.

Valor por defecto encendido.

* TOUCH\_STATE

Indica estado del botón.

Si el botón no se toca el valor va a ser cero, y distinto de cero si es presionado.

En caso de presionar durante unos segundos el sistema se apagará.

* TOUCH\_CAP\_CYCLES

Número total de ciclos de carga y descarga generados y medidos.

* TOUCH\_THRESHOLD

Indica el umbral del toque táctil y es uno de los parámetros que se pueden personalizar para ajustar la sensibilidad del toque táctil.

Valor por defecto 12.

* TOUCH\_HYSTERESIS

El sistema de detección táctil tiene una histéresis para garantizar la detección táctil y se puede personalizar en caso de querer equilibrar la sensibilidad.

Valor por defecto 2.

* DCO\_RSEL y DCO\_DCOMOD

Estos valores vienen establecidos de fábrica para que el reloj del microcontrolador funcione a 12MHz.

* VBAT

Este valor representa el voltaje de la batería en mV.

* VOUT

Este valor representa la tensión de salida. Es decir, representa el suministro a Raspberry Pi.

* VBAT\_MIN

Este valor determina el voltaje mínimo de la batería. Si el voltaje de la batería cae por debajo de este valor, la Raspberry será apagada. Este valor es solo un protocolo de seguridad que nunca se debe dar, ya que hay otro valor más alto en el que la Raspberry debe de ser apagada.

Valor por defecto 2,85V.

(Se recuerda que el tipo de batería usada no debe de caer por debajo de 2V).

* VBAT\_SHDN

Este valor determina el valor de tensión de la batería a la que la Raspberry debe de ser apagada.

Valor por defecto 2,95V.

* VBAT\_BOOT

Este valor determina el valor del voltaje de la batería a la que puede ser arrancada la Raspberry Pi.

Valor por defecto 3,15V.

* VOUT\_MAX

Este valor determina el voltaje de salida mínimo para el cual LiFePO4wered/Pi se niega a arrancar la Raspberry cuando está apagada. Este valor es un valor de seguridad para evitar daños cuando las Raspberry se alimentan desde una fuente diferente.

Valor por defecto 3,88V.

* VOFFSET\_ADC

Proporciona un valor de calibración para las mediciones de voltaje ADC.

Valor por defecto 0V.

* AUTO\_BOOT

Este registro por defecto tiene un valor 0 que hace que la Raspberry permanezca apagada mientras no se pulsa el botón táctil de encendido.

Pero es un valor que se puede modificar a 1 para que la Raspberry se encienda sola cuando tenga un valor de tensión suficiente (VBAT\_BOOT).

* WAKE\_TIME

Este valor permite establecer un tiempo en minutos para que la Raspberry permanezca apagada antes de que se encienda. Es una función despertador.

El valor por defecto es cero por lo que la Raspberry permanecerá apagada hasta que se presione el botón.

* SHDN\_DELAY

Establece el número de tics que transcurren desde que el sistema es apagado hasta que se apaga, medido por la línea UART TX).

Valor por defecto 65.

* PI\_BOOT\_TO

Si LiFePO4wered/Pi es arrancado y no detecta respuesta de Raspberry en un tiempo, se apaga.

Este valor determina este tiempo de espera para recibir una respuesta.

Valor por defecto 5min.

* PI\_SHDN\_TO

LiFePO4wered/Pi se apagará cuando este sea puesto en estado de apagado y la señal TX del UART pase un nivel bajo. Si esa señal se queda en nivel alto, el sistema permanecerá en estado de apagado si apagarse hasta que se finalice la batería.

Por eso se ha establecido este valor para que aunque esa señal siga en nivel alto, se apague cuando pase un tiempo determinado.

Valor por defecto 2min.

* WATCHDOG\_CFG

Es una función de vigilancia para garantizar al usuario que la aplicación se está ejecutando correctamente, mediante un temporizador que cuenta regresivamente cada 10 segundos y cuando llega a cero se toma una acción. Esa acción es determinada por este registro.

Cuando el valor es cero no se realiza ninguna acción y la función de vigilancia es desactivada.

Valor por defecto 0.

* WATCHDOG\_GRACE

Establece un periodo de gracia desde que el sistema comienza a arrancar hasta que la aplicación arranque el temporizador de vigilancia.

Valor por defecto 20s.

* PI\_RUNNING

Este valor determina el estado de potencia de la Raspberry, cuando esta desactivada o en arranque este valor es 0 y cambia a 1 cuando esta es arrancada. Esto cambiará el estado del LED de pulsante a encendido y permitirá entradas por el usuario.

* CFG\_WRITE

Este registro hace posible guardar de forma permanente el flash los cambios de configuración por el usuario.

Valor por defecto 0.

Una vez estudiados los registros posibles se puede proceder a la creación del programa que devuelve los valores de la batería deseados.

### Shell Scripts [30]

Para realizar la recogida de datos de una forma automática se van a usar Shell scripts.

Esto es una herramienta para la automatización de tareas y ejecutarlas desde la consola.

Está formado por un archivo de texto, el cual contiene unos comandos que se ejecutan de principio a fin cada vez que se ejecute el archivo.

La forma de crearlo es creando un archivo con la extensión .sh, y luego abrirlo con cualquier editor de texto para editarlo.

#### Shell Scripts Descarga

Para hacer el programa de recogida de datos de los valores de tensión de la batería cuando esta está alimentando a la Raspberry Pi Zero, lo primero que se hace es crear un Shell scripts que se ha llamado curva-carga.sh y un archivo .txt llamado log.txt, que servirá como argumento para el scripts y donde se guardaran de forma automática y secuencial los datos obtenidos por el programa.

Estor archivos se crean con el comando touch. Y antes de ejecutar por primera vez el archivo se debe dar permiso para ello al usuario con el comando chmod.

Una vez creados estos programas se procede a escribir los comandos del scripts con un editor de texto como puede ser vi, que es el utilizado en este caso.

**Curva-carga.sh**

(1) #!/bin/sh

If [ $# -lt 1 ]; then

echo “indica el archivo de log como argumento”

exit

fi

LOG=”$1”

echo “logging data to $LOG”

:>”$LOG”

while true: do

(2) VBAT=$(lifepo4wered-cli get vbat)

echo $VBAT >> “$LOG”

sync

echo “Read $VBAT”

sleep 2m

done

Para comenzar el scripts hay que indicar al sistema que se va a presentar un conjunto de instrucciones a procesar seguido del programa a usar para interpretar el scripts (1).

A continuación se añade un condicional que en caso de que al ejecutar el archivo no se indique el argumento, muestre por pantalla un mensaje que pida el argumento y se cierre.

Si $# que es el número de parámetros o argumentos, es menor que uno, muestra el mensaje y cierra el programa. Si el parámetro si ha sido indicado continuara con las siguientes funciones.

El resto del programa lo que realiza es lo siguiente:

Primero declara una variable, cuo valor es el primer parámetro indicado al ejecutar el archivo. Que en el caso del proyecto va a ser el archivo .txt donde se quieren guardar los datos.

Esa variable luego va a ser llamada con el valor “$1”.

Se define la variable VBAT que en este caso es el valor de la tensión de la batería que nos es proporcionada por el programa lifepo4wered-cli. (2)

Se le dice al programa que ese dato se guarde en el archivo txt (>>) y lo muestre por pantalla.

Y con el comando sync se obliga a que esos datos obtenidos se guarden en el disco y no en la memoria, para que en caso de fallo del sistema no se pierdan dichos datos.

Con el comando sleep le indicamos que todo esto se repita cada cierto tiempo hasta que se interrumpa la ejecución ya sea de forma intencionada o por agotamiento de la batería.

#### Shell Scripts Alimentación

En el caso de querer obtener los valores de la tensión con la que se alimenta la Raspberry Pi Zero el programa a realizar es el mismo (curva-descarga.sh) cambiando la variable que pide el dato a obtener por lifepo4wered-cli, que en este caso no será VBAT sino VOUT.

El argumento que se le indique en este caso va a ser log2.txt para que los datos obtenidos sean guardados en un archivo distinto.

**Curva-descarga.sh**

#!/bin/sh

If [ $# -lt 1 ]; then

echo “indica el archivo de log2 como argumento”

exit

fi

LOG=”$1”

echo “logging data to $LOG”

:>”$LOG”

while true: do

VOUT=$(lifepo4wered-cli get vout)

echo $VOUT >> “$LOG”

sync

echo “Read $VOUT”

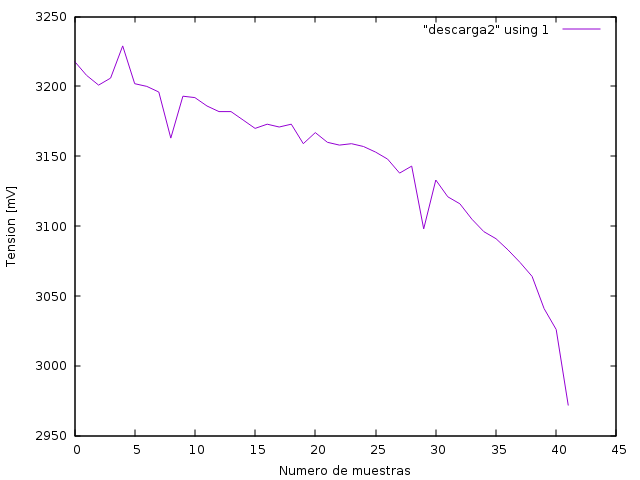
sleep 2m

done

### Resultados

Los resultados obtenidos en los archivos .txt son representados en graficas mediante GNUPLOT, que es un programa para generar graficas a partir de funciones o datos, compatible con Linux.

#### Descarga

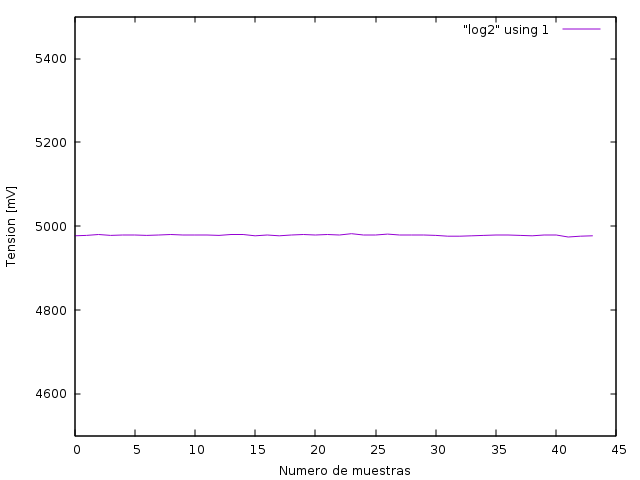


1. Datos descarga.

Los datos obtenidos son muestras recogidas cada dos minutos mientras la Raspberry es alimentada por el LiFePO4wered/Pi después de haberse cargado por completo.

Como se puede observar la batería se va descargando durante aproximadamente una hora y media, hasta que la tensión alcanza el valor definido por VBAT\_SHDN para que la Raspberry proceda a apagarse, que como antes se ha indicado es un valor de 2,95V.

#### Alimentación



1. Datos alimentación.

Como se puede observar en la gráfica esta celda de LiFePO4 suministra a la Raspberry Pi una tensión estable de alimentación de casi 5V desde que esta es arrancada, hasta que se apaga.

## RECOGIDA DE DATOS DE CARGA DE LA BATERÍA

# CONEXIONES