

Autor: Elías Marín Domínguez

Tutor: Juan Carlos del Pino López.

**Monitorización-gestión consumo eléctrico de una vivienda.**

Proyecto Fin de Carrera

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Dpto. de Ingeniería Eléctrica

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera

Grado en Ingeniería Electrónica, Robótica y Mecatrónica.

**Monitorización-gestión consumo eléctrico de una vivienda.**

Autor:

Elías Marín Domínguez

Tutor:

Juan Carlos del Pino López

Profesor titular

Dpto. de Ingeniería Eléctrica

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera: Monitorización-gestión consumo eléctrico de una vivienda.

|  |  |
| --- | --- |
| Autor: | Elías Marín Domínguez |
| Tutor: | Juan Carlos del Pino López |

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

*A mi familia*

*A mis maestros*

Agradecimientos

Los estilos adoptados por nuestra Escuela y utilizada en este texto es una versión y adaptación a Word® del la versión LATEX que el Prof. Payán realizó para un libro que desde hace tiempo viene escribiendo para su asignatura. Por ello, la Escuela le está agradecida. Por otro lado, la adaptación se hizo sobre un formato que el prof. Aguilera arregló, basándose en su tesis doctoral. Su aportación ha sido muy relevante para que este formato vea la luz. Esta adaptación la llevamos a cabo el alumno Silvio Fernández, becario del Centro de Cálculo, y yo mismo, sobre un trabajo preliminar del alumno Julián José Pérez Arias.

A esta hoja de estilos se le incluyó unos nuevos diseños de portada. El diseño gráfico de las portadas para proyectos fin de grado, carrera y máster, está basado en el que el prof. Fernando García García, de la Facultad de Bellas Artes de nuestra Universidad, hiciera para los libros, o tesis, de la sección de publicación de nuestra Escuela. Nuestra Escuela le agradece que pusiera su arte y su trabajo a nuestra disposición.

*Juan José Murillo Fuentes*

*Subdirección de Comunicaciones y Recursos Comunes*

*Sevilla, 2013*

Resumen

Siguiendo la tendencia actual de conectar todo dispositivo-objeto cotidiano a internet (IOT), y atendiendo a una preocupación tan frecuente como es el ahorro en consumo eléctrico, ya sea por motivos económicos o por sostenibilidad se ha desarrollado un sistema para la gestión-ahorro del consumo eléctrico, basado en app. En la cual podrás consultar en remoto el consumo en euros en tiempo real que llevas el mes en cuestión, así como en kwh y los picos de potencia que se están registrando.

Este sistema, se encarga de llevar un recuento de consumo no solo con la comercializadora en la que estás sino con una serie de comercializadoras que se encuentran en una base de datos propia, con el objetivo de proponer cambios de compañía eléctrica por ahorro en dinero.

También, se encarga de llevar un estudio del uso de la potencia que tienes contratada, proponiendo también bajadas o subidas de potencia.

De momento el sistema está adaptado únicamente para el mercado libre. Se cuenta con una base de datos de elaboración propia de las principales compañías eléctricas (de momento solo Fenie Energia, Endesa, pero fácilmente ampliables), con los precios asociados a cada tipo de tarifa.

La app se podría adaptar muy fácilmente para poder lanzar notificaciones al usuario cada vez que se exceda un consumo límite impuesto, así como notificar cuando se entre en las horas baratas, de manera que el usuario pueda desviar todo el consumo a estas horas.

Se podría incluso tener no solo un sensor de tensión y de corriente para el consumo en global, sino tener medidas independientes de cada circuito de la instalación. Incluso, si se quiere de la posibilidad de cortar-cerrar estos circuitos en remoto se podría adaptar muy fácilmente el sistema.

Este aparato de bajo consumo eléctrico consta de una placa de desarrollo junto con dos sensores, que iría al cuadro de luz del cliente.

Además, se ha montado un servidor basado en una raspberry, que se encarga de la gestión de la app para Android/IOS, servir datos de la base de datos y de la comunicación con el microcontrolador para lanzarle órdenes y recoger datos de los sensores del mismo.

Nota: Finalmente, como la conexión a internet donde se encontraba el servidor no era muy estable (red doméstica), se ha trasladado el mismo a una “virtual machine” perteneciente a Google.

Esta plataforma de Google Cloud ofrece numerosos servicios basados en la nube, en este caso se está alquilando un “servidor”-máquina virtual, configurada con Debian9 como SO, 25GB HDD, 1Vcpu, 2GB RAM…

El alquiler de la máquina supone un coste de 3 cent/hora.

A modo de resumen, se tiene un servidor, con las características deseadas dentro de la infraestructura de Google.

Una vez se crea la instancia, se suministra la dirección IP externa para poder controlar en remoto a la máquina.

En ella, sobre Debian 9, he montado el stack LAMP, servidor Blynk (como está basado en Java, también Java 8).

LAMP 🡪 Linux: Debian9

Apache

Mysql 5.6, servidor base de datos

PHP 7

Abstract

In our school there are a considerable number of documents, many teachers and researchers. Our students also contribute to this production through its work in order of degree, master's theses. The aim of this material is easier to edit these documents at the same time promote our corporate image, providing visibility and recognition of our Center.

... -translation by google-

Índice

Agradecimientos ix

Resumen xi

Abstract xiii

Índice xiv

Índice de Tablas xv

Índice de Figuras xvi

Notación xviii

1 Introduccion 1

1.1. Motivación y objetivo 2

1.2. Estado del arte 3

1.3. Resumen del hardware usado 3

2 Hardware 4

2.1. Linkit Smart 7688 Duo 11

2.2. Raspberry Pi3 ModelB 12

2.3. Sensor SCT013 13

2.4. Transformador AC/AC 13

2.5. Circuito de acondicionamiento para sensores 14

2.6. Lista de precios para el prototipo 16

2.7. Ideas para continuar el Proyecto 16

3 Software 18

3.1. Sistemas Operativos 19

3.2. Conexión a Internet 20

3.3. Programa MCU 20

3.4. Programa en C++ para la RPI 22

3.5. Plataforma IoT Blynk 25

3.6. Plataforma Google Cloud 29

3.7. Puesta a punto de la máquina virtual dentro de GCP 29

3.8. Base de datos Mysql 33

3.9. Servidor LAMP 33

4 Resultados 34

4.1. Muestreo de tension y corriente 35

4.2. Consideraciones para el software 35

Referencias 35

Índice de Conceptos 36

Glosario 38

# **Índice de Tablas e ilustraciones**

[Ilustración 1. Esquema general del sistema 1](#_Toc530679412)

[Table 1. BOM- Coste componentes que componen el sistema 16](#_Toc530481434)

# **Índice de Figuras**

[Figure 1.Placa de desarrollo Linkit Smart 7688 Duo 11](#_Toc530480728)

[Figure 2.Raspberry Pi3 Model b 12](#_Toc530480729)

[Figure 3.Sensor de Corriente SCT013 13](file:///C:\Users\Elias\Documents\Desktop\universidad\tfg.docx#_Toc530480730)

[Figure 4.Transformador AC/AC 13](#_Toc530480731)

[Figure 5.Mini Transformador AC/AC PCB 14](#_Toc530480732)

[Figure 6.Circuito OFFSET DC 15](#_Toc530480733)

[Figure 7.Circuito divisor resistivo 15](#_Toc530480734)

[Figure 8.Funcion código .ino inicializa el ADC 21](#_Toc530480735)

[Figure 9.Función código .ino inicializa el TIMER 22](#_Toc530480736)

[Figure 10.Función loop() del código gestiona la app dentro de la raspberry pi c++ (I) 23](#_Toc530480737)

[Figure 11.Función loop() del código gestiona la app dentro de la raspberry pi c++ (II) 24](#_Toc530480738)

[Figure 12.App Blynk Android (I) 25](#_Toc530480739)

[Figure 13.App Blynk Android (II) 26](#_Toc530480740)

[Figure 14.App Blynk Android (III) 26](#_Toc530480741)

[Figure 15.App Blynk Android (IV) 27](#_Toc530480742)

[Figure 16.Port Forwarding 28](#_Toc530480743)

[Figure 17.Tabla de datos Mysql 33](#_Toc530480744)

[Figure 18.Rango de Potencias a contratar dentro de Fenie Energia 33](#_Toc530480745)

[Figure 19.Test peticion GET al servidor 34](#_Toc530480746)

Notación

|  |  |
| --- | --- |
| A\* | Conjugado |
| c.t.p. | En casi todos los puntos |
| c.q.d. | Como queríamos demostrar |
| ∎ | Como queríamos demostrar |
| e.o.c. | En cualquier otro caso |
| e | número e |
| IRe | Parte real |
| IIm | Parte imaginaria |
| sen | Función seno |
| tg | Función tangente |
| arctg | Función arco tangente |
| sen | Función seno |
| sin*xy* | Función seno de *x* elevado a *y* |
| cos*xy* | Función coseno de *x* elevado a *y* |
| Sa | Función sampling |
| sgn | Función signo |
| rect | Función rectángulo |
| Sinc | Función sinc |
| ∂y ∂x  *x*◦ | Derivada parcial de *y* respecto  Notación de grado, *x* grados. |
| Pr(*A*) | Probabilidad del suceso *A* |
| SNR | Signal-to-noise ratio |
| MSE | Minimum square error |
| : | Tal que |
| < | Menor o igual |
| > | Mayor o igual |
| \ | Backslash |
| ⇔ | Si y sólo si |

# 1 Introduccion

Esto es una cita al principio de un capítulo.

- El autor de la cita -

E

n la actualidad, se aprecia cada vez más fácilmente la tendencia de conectar todo tipo de dispositivos cotidianos a internet, ya sea buscando comodidad (todo controlado desde el Smartphone) o productividad, con tal de aprovechar todos los datos que nos brinda internet, que no son pocos.

Desde bombillas gestionadas por una app móvil a asistentes virtuales capaces de atender nuestras preguntas a través de comandos de voz.

¿Por qué no darle más posibilidades a un cuadro de luz de vivienda, con tal de volverlo más “inteligente”?

Y si lo enfocamos para que no solo sea más cómodo y rápido llevar un control de tu consumo en euros en luz en tiempo real, sino que se preocupe también, de lanzarte sugerencias de ahorro.

El esquema de trabajo general del sistema sería el siguiente:

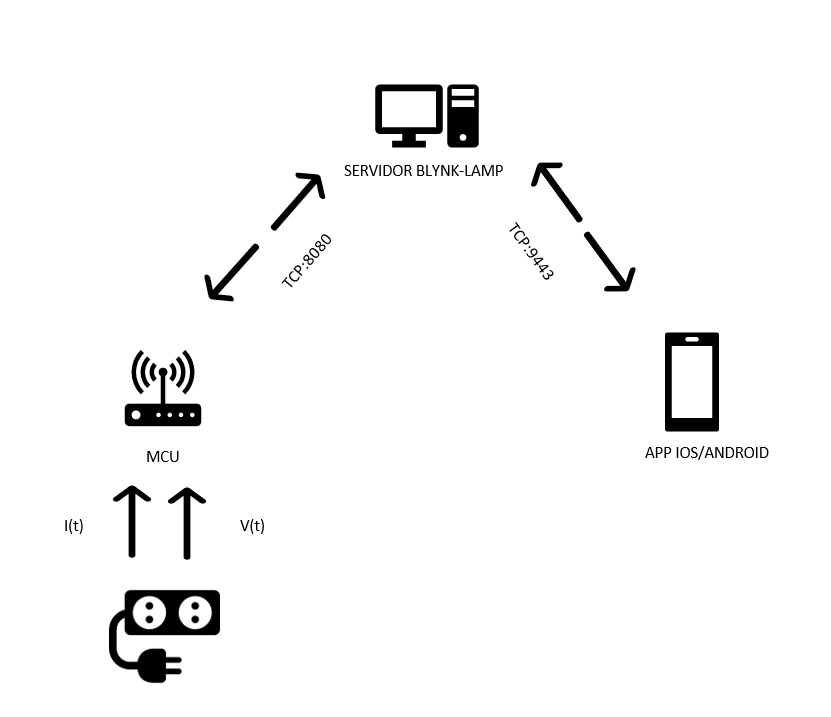


Ilustración 1

## Motivación y objetivo

Este trabajo de fin de grado simplemente sigue la tendencia actual del IOT, más concretamente siguiendo el término de las Smart homes, a la par que se presenta una solución a una preocupación tan común como es el ahorro energético, realmente, ahorro de dinero.

Si además a este sistema se le diera la posibilidad de tomar decisiones, del tipo abro circuito A, cierro circuito B, en función del precio de la luz, lo cual sería fácilmente implementable, se podría extender su uso a industria, donde un pequeño ahorro en luz, supondría una gran diferencia en la factura de luz.

Además, el disponer de una app, muy sencilla de usar y con posibilidad de realimentación con notificaciones en función del estado del consumo en la casa es muy atractivo. Piénsese, que el usuario desde fuera de casa y en cualquier momento tiene a su disposición el consumo en euros en tiempo real de la vivienda, el consumo en kwh, los picos de potencia que se han registrado y a la hora que se han producido…

## Estado del arte

La automatización o monitorización de actividades en remoto, está experimentando un crecimiento notable, con la aparición de nuevas e incontables placas de desarrollo y plataformas IOT. Actualmente la mayor parte de dispositivos conectados, que quieren poder disfrutar de todos los datos que brinda internet, usan Wifi, el principal problema que se presenta es el elevado consumo que tienen. Se están investigando distintas alternativas que necesiten menos energía.

La idea tuvo su aparición gracias a Kevin Ashton en el AutoId Center del MIT en 1999, donde se andaba trabajando con técnicas de sensores y RFID.

La flexibilidad que da una placa de desarrollo, ya sea basada en microprocesador o microcontrolador, incluso actualmente muy común el uso de placas con mcu y mpu mixtas, hacen que el ámbito de aplicación sea enorme.

Ya sea una placa extrayendo datos de internet o una placa que recoge datos de sensores y los procesa y los hace accesibles desde cualquier parte de la red, lo hace muy llamativo.

Si a esto se le unen tendencias actuales del tipo machine learning, para el procesado de datos de una base de datos que proviene de un dispositivo conectado con el fin de predecir comportamientos y tendencias, su ámbito de aplicación crece aún más.

## Resumen del hardware usado

El sistema global se compone:

- Parte destinada a las lecturas con procesado intermedio:

Placa de desarrollo **linkit Smart 7688 duo**, cuenta con microcontrolador ATMEGA32U4 (se puede programar desde el ide de arduino), cuenta también con una mpu Mediatek mt7688, corriendo OpenWrt.

Esta placa se encarga de la recogida de los valores de corriente y de voltaje de la red, entre muestra y muestra hace cálculos intermedios para que el resultado posteriormente sea más rápido de obtener. Una vez realiza la lectura completa de un periodo de red, calcula valores eficaces, así como potencia activa, reactiva y cos de phi y se lo transmite a la rpi, para que la misma gestione la app con esos nuevos datos.

Sensor de corriente no invasivo de la serie **SCT013**, en concreto he trabajado con el SCT013-30, el cual trabaja hasta con 30 A en ac.

**Transformador ac/ac**

**Circuito acondicionamiento**, para adaptar el rango de medidas tanto del sct como del trafo a un rango admisible por la placa. Para el transformador además de ajustar el rango completo a los 0-3.3V de la placa, se añade un offset en dc para que la señal alterna, se mueva siempre entre valores positivos.

Para el sensor de corriente simplemente añadir el punto medio, para que los valores de tensión sean siempre positivos.

- Servidor:

**Raspberry pi 3 Model B**, corriendo raspbian(distribución Linux). La misma se encarga del procesado de datos provenientes de la placa y de la gestión de la app, además de la gestión de la base de datos propia.

Este servidor puede ser común para varias de estas placas, si por ejemplo se tuvieran varios proyectos, todos ellos colgarían del mismo servidor.

Como ya se ha mencionado, finalmente a modo de servidor se ha migrado desde la Raspberry a una máquina virtual dentro de un servidor perteneciente a Google con Debian 9 como sistema operativo.

# Hardware

Texto elaborado por Marísa Balsa, de la Biblioteca de la Escuela Técnica Superior de Ingeniería de la Universidad de Sevilla

L

A idea que se tenía desde el primer momento para abordar el trabajo era dividir el sistema en 3 módulos. Por un lado, una placa basada en microcontrolador que se encargara de las lecturas, una placa basada en microprocesador que se encargara de realizar procesado y envio de datos al servidor, y el mismo servidor para el control de la base de datos y la gestión de la app.

Primeramente, se comenzaron las pruebas con la “famosa” placa esp8266, placa ampliamente usada para el IoT, con un modesto precio y unas características bien sobradas. También para las primeras pruebas se hicieron tests con un Arduino Uno.

Del esp, existía la desventaja de la poca documentación relativa al uso de su ADC interno. Tampoco fue fácil encontrar información relativa a sus TIMERs internos. Como lo que se buscaba era poder asociar un TIMER con el ADC, para tener control sobre cuando se realizaba la lectura de voltaje e intensidad, se requería programar estos periféricos a nivel de registros.

Con el arduino Uno, y su atmega328p, se consiguió programar estos dos periféricos y hacer que el tiempo de muestreo de las lecturas estuviera acordado.

Posteriormente, se optó por usar la placa de desarrollo linkit Smart 7688 duo, la cual integra, como se comentó anteriormente un mcu y un mpu, comunicados por puerto serie entre sí. Sería equivalente al Arduino Yun.

Por unos 15€, se puede obtener esta placa. Las ventajas que aporta son las siguientes:

* El mcu se puede programar usando el IDE de Arduino.
* El mpu corre un SO, OpenWrt, que brinda muchas oportunidades complementando al mcu anterior.
* Cuenta con slot SD, para ampliar la memoria interna.
* Cuenta con comunicación Wifi-ethernet.

Entonces, la mcu programada desde el entorno de arduino, se encarga de la lectura de los sensores de manera periódica, así como de enviar los datos finales a la raspberry, para que los dirija a la app.

La mpu de la misma placa, apoya a la mcu, con su chip Wifi.

Anteriormente, se quiso cargar toda la parte de gestión de la app dentro del mpu de la placa Linkit, pero tras varios intentos de compilación cruzada fallidos, todo debido a falta de librerías de ciertas plataformas IOT, para este SO(OpenWrt), se optó por trasladar este programa escrito en C a la rpi.

Se darán más detalles, en el siguiente punto dedicado a software.

Para el prototipo se monta una caja estanca con carril DIN, para montar la placa de prototipado, junto al transformador y la salida del sensor de corriente tipo pinza y el conector para la lectura de la tensión.

## Linkit Smart 7688 Duo

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figure 1

Como ya se adelantó anteriormente esta placa de desarrollo cuenta con una mpu apoyada por una mcu. Es una placa diseñada para el IoT. En concreto cuenta con un ATmega32U4, como mcu, el cual puede ser programado usando el ide de Arduino. La mpu, es una Mediatek MT7688AN corriendo Linux, en concreto OpenWrt. Posee conectividad WiFi, numerosos periféricos internos (asociados a la mcu).

Problemas detectados durante su uso:

* De fábrica sin SO, lo cual no suele ser habitual. Se procede a descargar la imagen desde la página del fabricante y cargarla sobre la placa haciendo uso de memoria usb formateada en un formato en específico(FAT32).
* Pasar de modo estación a punto de acceso para reconfigurar la red wifi a la que se conecta puede llegar a resultar un poco tedioso.
* Memoria de programa de ATMEGA32U4 se queda corta muy fácilmente.

**Especificaciones técnicas y otras características:**

\*580MHz (MPU) 8MHZ(MCU)

\*Voltaje de operación: 3.3V

\*ADC 🡪10bits(MCU), 12 canales

\*Wi-Fi 802.11 b/g/n(2.4G)

\*GPIO,I2C,I2S,SPI,UART,PWM, ETHERNET PORT

\*32MB flash, 128MB DDR2 RAM (MPU)

\*32KB flash,1KB EEPROM

\*Soporta USB host y tarjetas SD, para expandir la memoria interna.

\*Tamaño de la placa 60.8\*26.0mm

En resumen, placa muy completa por un precio bastante modesto.

## Raspberry Pi3 ModelB

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figure 2

Este tipo de placas, bien conocidas, se engloban dentro de lo que se conoce como ordenadores de placa reducida.

Se trata de un mini pc, con unas especificaciones limitadas, si se compara con un pc doméstico, pero sobrado en potencia para las tareas a las que suele estar destinado (como servidor web, manejo de base de datos, domótica …).

En concreto para el desarrollo de este sistema, se ha usado la Pi3 Model B. Comentar que aun más reciente, la Pi3 Model B+.

La Pi3 cuenta con una CPU quad core, 1GB de RAM, chip bluetooth-wifi, puerto ethernet, 40 pines GPIO, salida HDMI… Todo esto por un precio de unos 35€.

Estas placas se comercializan sin SO, el primer paso que hay que hacer con ellas es bajar un sistema operativo compatible, como el propietario de la fundación 🡪 Raspbian, basado en debian. Este SO debe ser montado sobre una SD apta.

Problemas detectados durante su uso:

* Se optó por una SD de 8gb que rápidamente se quedó sin memoria y empezó a funcionar de manera anómala, por ejemplo el servidor Blynk no llegaba siquiera a dispararse al arrancar la raspberry por falta (necesita almacenar en la placa cierta información acerca de la nueva conexión entrante).
* La SD que se estaba usando era de clase 4, como el SO se carga sobre la sd lo ideal es invertir en una buena SD desde primera hora, de manera que el conjunto se comporte de mejor forma.
* Se presentaron también problemas relacionados no con la placa en si sino con la red local que se saturaba por el tráfico entrante y saliente hacia el exterior. El problema venía del plan de internet contratado que tenía una pésima velocidad de subida.

Pasando a las especificaciones técnicas más en detalle:

\*Chip BCM2837: quad core cluster ARM Cortex A53

\*1GB de RAM

\*Chip BCM43438 LAN & BLE

\*40GPIO

\*4 USB

\*HDMI

\*MicroSD slot

## Sensor SCT013

Figure 3

Sensor de corriente no invasivo de tipo pinza amperimétrica, para la medida de la corriente. Su principio de funcionamiento se basa en un transformador de corriente, obteniéndose una medida de corriente proporcional a la intensidad que atraviesa el circuito. Dependiendo del tipo de modelo que escojamos, podemos obtener a la salida del mismo una medida de tensión o intensidad, lo preferible a no ser que se quiera adaptar el rango de medida para ajustar-afinar la precisión del mismo es que la salida sea en tensión.

La relación de transformación que rige a este sensor de corriente:

{\frac {I_{s}}{I_{p}}}={\frac {Vp}{Vs}}={\frac {Np}{Ns}}

Donde el número de espiras del devanado primario suele ser simplemente el conductor, jugando con el número de espiras del secundario podemos conseguir que la intensidad en el secundario decrezca más o menos.

La precisión del sensor suele ser de un 1-2%. La única desventaja que se le puede achacar a este tipo de sensores, es que al ser una carga inductiva introduce un desfase que puede llegar a los 3º.

Todos los modelos tienen un precio similar, en vendedores internacionales suele rondar los 5€.

En este proyecto se ha usado el sct013-030, el cual produce una salida de hasta 1vrms para un máximo de corriente de 30Arms. Por lo tanto, con el modelo actual de sensor de corriente, se puede llegar a monitorizar una vivienda de hasta aproximadamente 7kw de potencia contratada.

## Transformador AC/AC

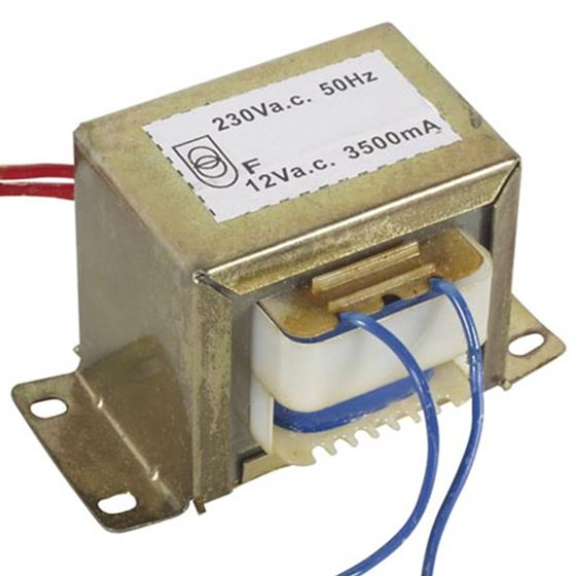


Figure 4

Para medir la tensión de red, se han hecho pruebas con dos tipos de transformadores de tensión. En el caso de España, la tensión de red 230Vac a 50 Hz. De nuevo aparece el problema del desfase que introducen los transformadores.

Se han usado para las pruebas un transformador 230VAC/12VAC para montar en carril DIN y un mini transformador de 230VAC/3VAC.

Como se sabe, un transformador consta de dos devanados, arrollados en torno a un núcleo magnético. Si hacemos circular por el devanado primario corriente alterna, se genera e induce un flujo magnético en el núcleo, que, al atravesar el devanado secundario, genera una tensión inducida.

Como se indica en el punto anterior, sensor de corriente sct, se juega con el número de devanados en el primario y secundario, para conseguir una relación de transformación determinada.

En general, aunque un transformador tiene pérdidas, son máquinas de elevado rendimiento, en torno al 95%.

Para un dispositivo final lo ideal sería sustituir el transformador tipo carril DIN, por un mini transformador del tipo:

Imagen que contiene electrónica, batería

Descripción generada automáticamente

Figure 5

## Circuito de acondicionamiento para sensores

Para el sensor de corriente, el cual aporta una salida de tensión de 1Vac, lo único que se ha incluido es un offset DC, de 1.5V para que la tensión alterna siempre este entre valores positivos.

En concreto tras añadir este offset, la medida de tensión del sensor se mueve en este rango [0.09V,2.91V].

Incluir circuito con resistencias y condensador para los 1.6V de offset DC.

Esquema del circuito usado para conseguir este offset en DC:

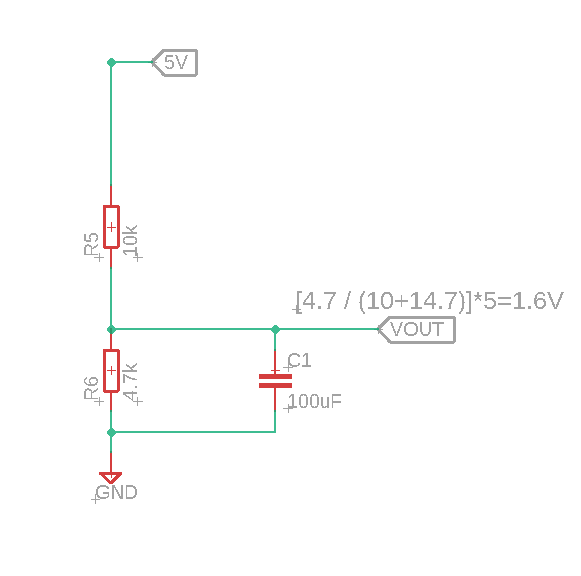


Figure 6

Para el transformador, aparte de incluir el mismo offset dc para que las medidas siempre sean positivas, se ha incluido un divisor resistivo para bajar de los 12VAC (10.2VAC realmente, según voltímetro) a 0.65VAC.

A continuación, se presenta el circuito usado a modo de divisor resistivo, para hacer la conversión de voltaje desde la salida del transformador y obtener unos valores admisibles para la placa que realiza el sensado.

Donde:

* AC1 representa uno de los dos polos de la señal alterna de salida del transformador
* AC2 el segundo polo de la señal alterna, este nodo a su vez se conecta al punto que añade el offset en DC al circuito
* Vout representa el nodo de salida para el voltaje reducido desde el transformador
* Relación de transformación usando este circuito 🡪 (1/15.7)\*Vin

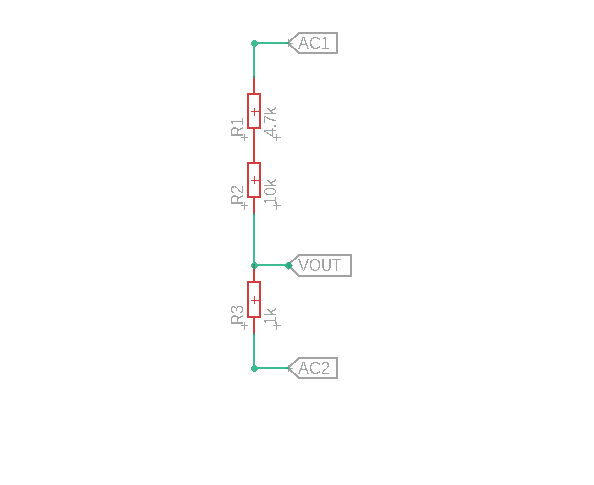


Figure 7

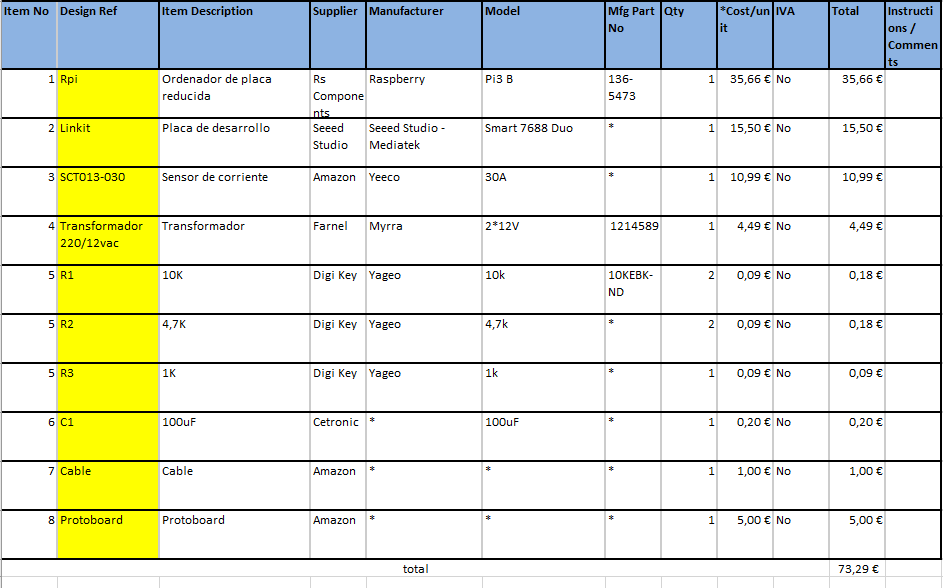
## Lista de precios para el prototipo

A continuación se presenta una table excel a modo de “BOM”, término usado para hacer referencia a la lista de costes de un prototipo-producto:

Como se puede apreciar en la tabla el componente que más encarece el total de la lista, es la placa Rpi 3b, con un precio de entorno a los 35 euros. Aunque ha de tenerse en cuenta que cada Raspberry gobernaría no solo a una placa de sensado sino a varias de ellas. Si el número de prototipos instalados en casas de usuarios que cuelgan de la Rpi comenzara a crecer en exceso lo ideal sería trasladar el servidor blynk a una computadora de mayores recursos.

Entonces si descontamos el precio de la rpi del total(el cual habría que repartir en function del numero de proyectos que cuelguen de ella) se tiene un total de 37.63 euros de coste hardware.

Table 1



## Ideas para continuar el Proyecto

Como extras para complementar la funcionalidad del producto (algunos de los siguientes puntos ya se han mencionado anteriormente en su punto respectivo):

* Incluir varios sensores de corriente para hacer medidas de consume por circuito de la instalación y no solo del circuito general. El software está preparado para que sea modular por lo que únicamente estaríamos limitados por el numero de entradas analógicas del controlador y por la tasa de muestreo que se quiera una medida de consume de un mismo circuito. El adc del mcu en cuestión cuenta con 12 canales para entrada del valor analógico por lo que se podría montar un Sistema de hasta 11 circuitos, ya que necesitamos una entrada para la lectura del votltaje.
* Ahora mismo el Sistema está preparado para funcionar con un tarifa perteneciente al Mercado libre con una potencia contratada menor de 10Kw. Fácilmente se podría preparer el sisema para que también se pudiera tener control del consume en viviendas de más de esta potencia. Simplemente incluyendo otra table de precios en la base de datos similar en estructura a la que hay de potencia <= 10kw.

También se podría preparer al Sistema como se comentó en puntos anteriores para que funcione dentro del Mercado regulado. Para ello el servidor debería ser capaz de extraer de una página web el precio del kwh. Esta funcionalidad daría mucho juego ya que si este precio temporal comenzara a guardarse en una base de datos y se le aplicara tratamiento de datos (BigData) podríamos empezar a predecir comportamientos y adelantarnos a ciertos valores futuros.

* Lo ideal sería que el programa que ahora mismo cuelga de la rpi para comunicarse con el servidor blynk y con el hardware, para llevar el control del consumo estuviese dentro de la placa encargada del sensado para evitar tanta dependencia de la red.
* Todos los datos que se mandan al servidor blynk podrían ser almacenados en una base de datos para aplicar un tratamiento de datos(BigData).
* Migrar de la placa de prototipado a una PCB, donde se situen el alojamiento de la placa Linkit, el circuito de acondicionamiento, conectores tipo “click” para la facil integracion de los sensores. Lo ideal sería o bien preparer distintos modelos de pcb con más o menos salidas de sensores o dejarlas todas accesibles para que se usen las que se requieran.
* Migrar a trifásica para un entorno industrial. La idea para el sensado sería similar a la de hacer lectura de circuitos por separado de una instalación eléctrica monofásica.
* Como responder ante la falla de la conexión a internet o pérdida de alimentación. Lo ideal que ante la pérdida de alimentación se pudiera notificar al usuario vía sms. Para esto habría que incluir un modulo GSM con una tarjeta sim para que salte la notificación cada vez que se corte la luz via sms, este modulo se encuentra por un precio en torno a los 15 euros.

# Software

The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

Claude Shannon, 1948

E

n cuanto a software se tienen de nuevo 3 partes bien diferenciadas, asociadas a cada módulo hardware, los cuales se han descrito en el punto anterior. Por una lado se encuentra el programa desarrollado usando el IDE Ardunino, que se encarga de las lecturas, de cierto procesado y del envío de datos hacia el servidor usando la plataforma Blynk, la cual se detellará posteriormente. Por otro lado, el programa escrito en C que se encarga de recibir datos del servidor blynk, el cual se encuentra gestionando la app, y también los que proceden del mcu. Este programa se encarga de llevar un recuento de consumo y mostrar datos en los distintos widgets de la app. También se tiene el servidor Blynk, montado en la Rpi para el control de la app, es Open-Source, y está basado en java. Lo único que se ha tendido que hacer con el servidor es configurarlo para adaptarlo a los requerimientos de la app. Por último se ha montado un servidor LAMP en la Rpi, para poder gestionar el acceso a la base de datos propia creada para almacenar información respesctiva a las distintas comercializadoras.

Como software usado, ya se ha mencionado que se ha usado el IDE de Arduino para programar la placa de desarrollo Linkit, también Putty, usándolo como cliente SSH, para el control en remoto de la Rpi y el mpu de la placa Linkit.

También citar en este punto, el uso del protocolo Samba, el cual ha sido de gran ayuda para facilitar el intercambio de archivos entre mi computadora personal y las placas con SO propio.

En cada punto, se incluirán los enlaces necesarios y que han sido de más ayuda para la descarga de software.

## Sistemas Operativos

A la hora de la hacerte con la Raspberry, debes de tener en cuenta, que se comercializa sin el SO instalado. De hecho, has de hacerte con una tarjeta microSD, para poder instalar sobre ella el SO. Esto, aun pareciendo una desventaja, aporta mucha flexibilidad. Se podrían tener distintos SO en distintas SD, y alternar entre ellos tan solo cambiando la tarjeta de memoria.

Aunque el SO oficial para la RPI es Raspbian, existen numerosas alternativas de SO de terceros, en la misma página oficial de Raspberry se incluyen enlaces de descarga de las imágenes tanto del SO oficial como de SO´s de terceros.

Raspbian, es una distribución de GNU/LINUX, libre y basada en Debian, se desarrolló en específico para esta placa (Raspberry) y con la idea inicial de la enseñanza informática. Esa es la esencia de esta placa y de su SO asociado.

En concreto, la versión de Raspbian que se ha usado es la 8, conocida como Jessie.

Para instalar el SO, simplemente acudiendo a la pagina web oficial de la fundación Rpi, descargando la imagen, y apoyándonos en software del tipo Win32 Disk Imager, se puede tener lista la placa para jugar, en menos de 10 minutos.

Por otro lado, para la placa Linkit Smart 7688 duo, hay que seguir un proceso similar, pero solo en el caso de que uno se encuentre con una imagen dañada o corrupta de fábrica. Por norma general, el fabricante proporciona la placa con el SO ya instalado. En este se caso se trata de OpenWrt.

Si se acude a la página de descargas de la placa en cuestión:

<https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/downloads>

Se pueden acceder a distintas versiones firmware de la placa. Así como a SDKs para hacer la compilación cruzada desde tu computadora personal, por ejemplo, se usó el SDK para ejecutar código en C en la propia mpu.

Aunque los resultados no fueron los esperados, por problemas de librerías, se tuvo que traspasar este código a la Rpi, donde si se pudo ejecutar con éxito.

En caso de que se encuentre con una placa con imagen corrupta seguir los pasos descritos en este tutorial:

<https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/tutorials/firmware-and-bootloader/update-the-firmware-with-a-usb-drive>

Para la máquina virtual estamos usando Debian 9 Stretch como imagen del sistema. Para configurar la misma con esta imagen es tan fácil como seleccionar la imagen en el desplegable que aparece al crear la instancia en la página de Google Cloud Platform.

## Conexión a Internet

Simplemente para la Rpi conectar cable Ethernet, podría estar fuera de la red local, donde se coloque la placa para el sensado, pero dentro de la red local que se encuentre se deberían abrir los puertos respectivos para el funcionamiento de la app Blynk. La misma se puede conectar también via WiFi si se quiere.

Para la placa Linkit, igualmente la conexión puede ser física o sin cable. Para la conexión sin cable, simplemente tras el primer encendido la placa arranca en modo punto de acceso, y consultando una página web que sirve en una dirección en concreto puedes configurar la red WiFi a la que quieres que se conecte en modo estación.

## Programa MCU

Para poder programar la placa Linkit (mcu ATMEGA32U4), desde el entorno Arduino simplemente limitarse a seguir estos sencillos pasos:

<https://docs.labs.mediatek.com/resource/linkit-smart-7688/en/get-started/get-started-with-the-linkit-smart-7688-duo-development-board/install-arduino-ide-with-board-support-package>

Entonces, tras esto la placa se encuentra lista para ser programada.

El programa .ino desarrollado para el mcu de esta placa se encarga de:

* Sensado periódico de tensión y corriente haciendo uso del ADC interno y del TIMER, de manera que en resumen, el ADC salta, cada periodo del TIMER, recogiendo una medida de tensión o corriente, según toque. Una vez el ADC finaliza la conversión(tiene el resultado de la conversión) y hasta que el timer vuelve a saltar, dedicamos ese intervalo de tiempo sobrante en hacer cálculos intermedios. De manera que llevamos la suma del cuadrado de los valores instantáneos de corriente y tensión, además de un contador de medidas para finalmente poder hallar el valor RMS.
* Cálculo del cosfi. Mediante detección de cambio de signo de valor de tensión o corriente, se recogen los cruces por cero con su respectivo momento de tiempo. Esto se realiza como parte de procesado intermedio entre que el ADC devuelve dato convertido y el TIMER vuelve a indicar comienzo de siguiente conversión. Se recogen los cambios de signo tanto pot flanco de subida como bajada, y posteriormente se contemplan todas las posibles combinaciones para el cálculo del cosfi.
* Envío de datos al programa de la Rpi escrito en C, mediante MQTT usando Blynk. Una vez completado un ciclo de lectura de un periodo de red, se realizan los cálculos finales para obtener, P, cosfi ,Q …
* Si se quisiera poder actuar en remoto, o de manera autónoma por toma de decisiones, este seria el momento de activar alguna salida.

Comentar que para tener control sobre la periodicidad del muestreo se ha tenido que ajustar el ADC mediante el uso de sus registros internos, lo mismo se ha tenido que hacer para ajustar el TIMER a las necesidades de esta aplicación.

Detallando acerca de estas dos piezas hardware:

* **TIMER**: están destinados a temporizar eventos sin tener que recurrir a funciones del tipo delay, bloqueando tontamente el programa.

Nos permiten lanzar interrupciones una vez se haya alcanzado el tiempo configurado sin necesidad de bloquear el programa durante ese tiempo acordado.

Se utiliza el TIMER en modo CTC, saltando la interrupción cuando se alcance el tiempo asignado. El mcu que usa la placa Linkit posee un oscilador interno que corre a 8Mhz, pudiendo llegar a disparar una interrupción cada 125ns. Existe un registro interno que es el que indica cada cuantos ticks de 125ns debe dispararse la interrupción. Lo siguiente a tener en cuenta es el valor máximo de ticks que se le puede indicar al timer asociado al número de bits de ese timer. En este caso se recurre al TIMER1 del mcu de 16 bits. Pudiendo llegar a un máximo de ticks acordados de 2^16 -1 🡪 65535=Vmaxespecificar. Por último tenemos el prescaler, que se encarga de dividir la frecuencia del oscilador interno por varios factores posibles, modificando así la cadencia de los ticks del timer.

En este caso se ha recurrido a un prescaler de 8, teniendo así un “oscilador” final de 1Mhz. Cada tick equivaldría a 1us. Y usando un compare match de 190, se tiene un timer que dispara interrupciones cada aproximadamente 190us.

Si el periodo del timer es, redondeando, 200us, y el periodo de la onda de red es de 20ms, se tienen 100 muestras por cada periodo de red. En total, 50 muestras de corriente y 50 mustras de voltaje.

* ADC: para la lectura de señales analógicas como las señales de tensión y corriente, se requieren conversores analógicos a digitales. Hay que tener en cuenta que el mcu es un chip digital y por lo tanto trabaja solo con valores binarios. Las tareas fundamentales y básicas que cumplen estos conversores son:
* Muestrear esa señal en un tiempo acordado.
* Cuantificar ese valor muestreado de acuerdo a la resolución del ADC.
* Codificar ese dato en binario.

El adc del atmega32u4 es un adc de 8/10 bits de resolución 🡨🡪 dependiendo de la velocidad de conversión que se le solicite. Además, jugando con el voltaje de referencia, se puede conseguir un mejor ajuste para el resultado final. De nuevo como con el TIMER, se tiene el prescaler, para determinar la cadencia de los ticks y con ello el tiempo de conversión. De nuevo se usa el prescaler de 8, resultando en un tick de 1us.

En el datasheet del mcu se especifica que el ADC requiere de 25 ciclos de reloj para la primera conversión y 13 para el resto. Por lo tanto, aproximadamente el tiempo de conversión será de 25us y 13us respectivamente.

El detalle, que de esos 200 us de periodo de timer, 13 o 25 us se emplean en la conversión del dato y el resto para procesado.

El problema o inconveniente que se tiene aquí es el siguiente: estamos recogiendo muestras alternativamente de tensión y coriente, por lo que constantemente se está reprogramando el ADC, y como consecuencia el tiempo de conversión es de 25 us.

Programa mcu 🡪 ADCTIMER\_blynk

**Detalle función que inicializa el ADC del mcu ATMEGA32u4**

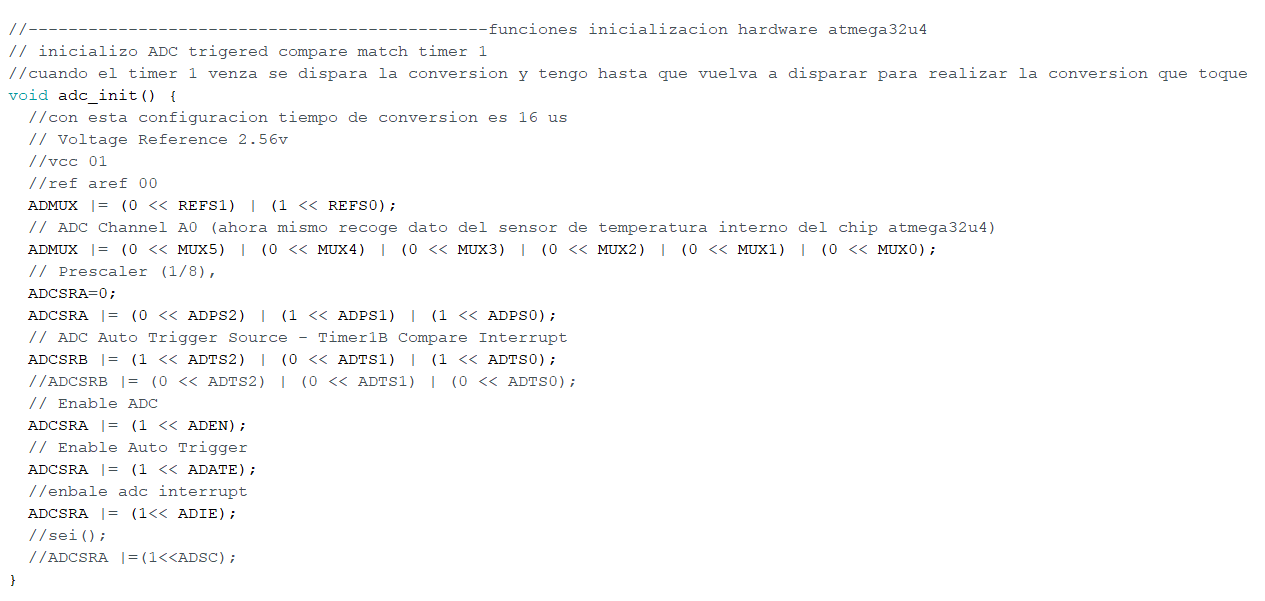


Figure 8

Lo primero que se hace es fijar la referencia del ADC, se tomo voltaje de referencia los 3.3V de voltaje de operación de la placa.

Posteriormente se selecciona el canal de interés para realizar la primera lectura, en este caso el A0.

Ahora se configura la velocidad de la conversión. 🡪 16us es la velocidad de conversión programada, salvo para la primera conversión que el ADC requiere mas tiempo.

Se asocia un timer en concreto el TIMER 1 para que vaya marcando el inicio de la conversión de una nueva muestra.

Por ultimo habilito el ADC, el auto TRIGGER, y las interrupción asociada al fin de la conversión.

**Detalle función que inicializa el TIMER 1**

* Contador se resetea
* Configuro el tiempo de disparo de la interrupción jugando con los prescaladores y con el valor a llegar con el contador
* Inicializo el contador
* Habilito las interrupciones del timer, en concreto del subtimer B

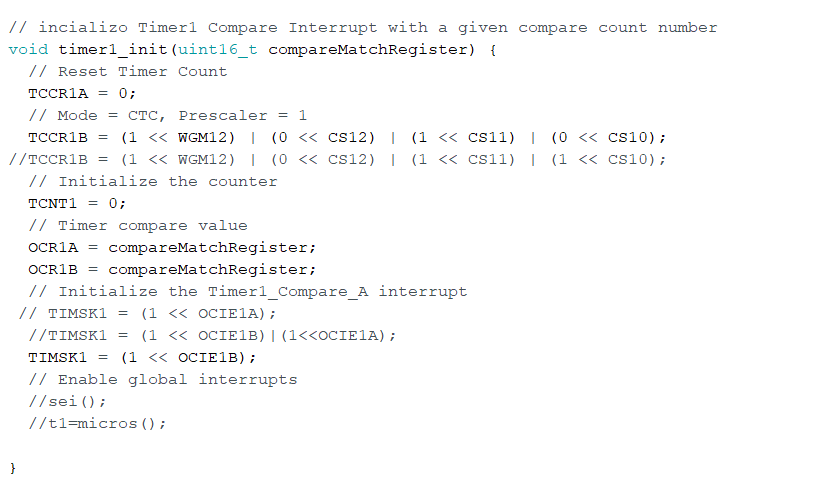


Figure 9

Por útimo, y como parte fundamental del programa, se tiene la **función asociada al fin de la conversión** de un dato dentro del ADC.

Para esta parte aun siendo analizada en el documento presente, se adjunta enlace para la descarga del archivo de programa completo comentado. Entonces, por extensión de la función mejor dirigirse dentro del código a 🡪

ISR(ADC\_vect){

………………..

}

## Programa en C++ para la RPI

Para el programa en c++ usando la plataforma Blynk, se ha seguido la siguiente guía para la puesta en marcha:

<https://community.blynk.cc/t/using-c-on-a-raspberry-pi-with-blynk/11864>

Simplemente ajustar una serie de parámetros para advertirle al compilador gcc que estamos trabjando con Linux pero desde una raspberry.

Este programa se ocupa de lo siguiente:

* Se conectar al servidor blynk, el cual esta montado en la misma Rpi.
* Extrae datos de las comercializadoras de la base de datos propia. De manera que recoge los nombres y precios de las mismas, para los cálculos de consumo y sugerencias de ahorro.
* Lleva un control del consumo en kwh en función de los datos que le pasa la placa Linkit, además de un control del consumo en euros en función de los datos extraidos de la base de datos.
* Lleva un control del periodo y franja horaria en la que se está realizando el consumo, para escoger el precio asociado del kwh.
* Lleva un contol del consumo uso de la potencia contratada, lanzando sugerencias.
* Lanza notificaciones de consumo-ahorro a la app.
* Debería almacenar variables críticas en cada ciclo por posibles pérdidas de alimentación.
* Realmente, si se pudiera implementar en la mcu, o en la propia mpu que acompaña al mcu dentro de la placa linkit, estaríamos quitando tanta dependencia de red. Ahora mismo, el mcu envía datos a la Rpi por internet al programa c++, y ahora el programa en c++ tiene que volcar estos datos a la app. De la otra forma solo tendríamos un envío de datos por red.

Se hicieron pruebas para esto mismo, para tener programa de sensado y gestión de app blynk en el mismo mcu, pero la memoria de programa no era suficiente.

A la hora de detallar el programa main.cpp, simplemente mostrar fragmentos de código de los puntos más clave del programa, a parte de adjuntar un enlace de descarga del mismo:

Interior del bucle loop:

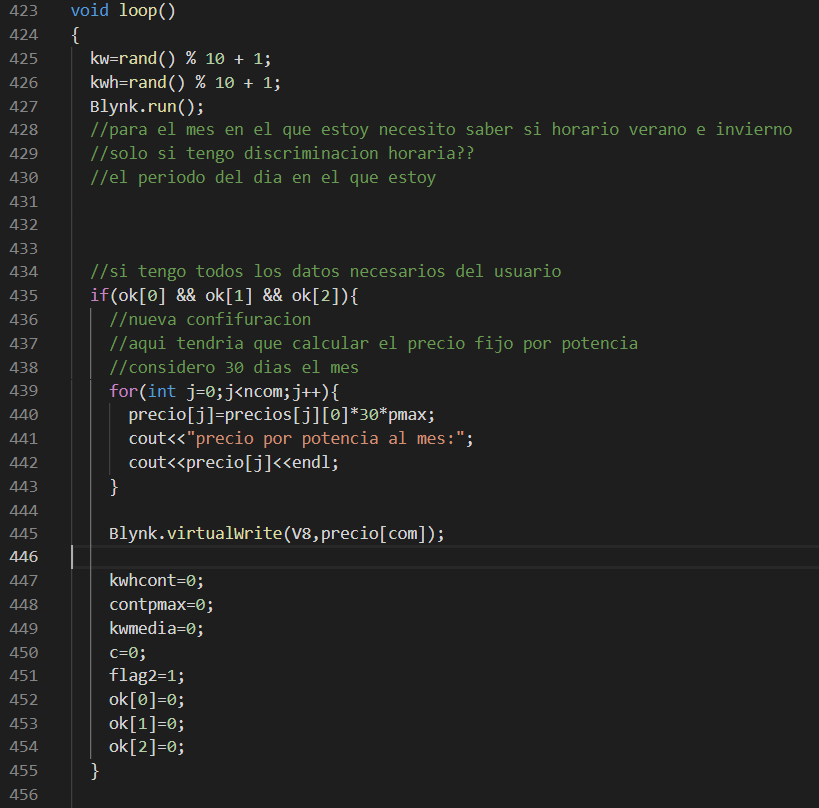


Figure 10

En esta primera parte del código se comprueba si se tienen los datos del usuario: comercializadora, tarifa y potencia contratada (de ahí lo del vector ok de longitud 3). Si se tienen los datos del usuario se calcula el precio por potencia para ese mes para cada una de las distintas comercializadoras. También se inicializan una serie de variales para el periodo en cuestión, como:

* Kwhcont: para el recuento total de consumo en ese periodo
* Contpmax: para el recuento de picos de potencia máxima
* C:
* Flag2: variable bandera para indicar al programa que ya se ha configurado la app con los datos de los usuarios.

Tras hacer esta comprobación, se comprueba si se tiene la app configurada y si ha llegado un dato nuevo desde la placa de sensado.

Tras esto se llama a la función horper(), la cual chequea si es el inicio de un nuevo periodo de facturación y contabiliza gastos del usuario en función de la hora que sea (lleva el recuento con su comercializadora y con las demás de la base de datos para sugerir cambios de tarifa para ahorrar dinero).

Se actualiza el recuento de consumo en kwh, se chequea si en este periodo de sensado y procesado se ha alcanzado un pico de potencia, se va llevando un recuento de la media de potencia y se actualizan los widgets respectivos con los datos obtenidos hacia la app.

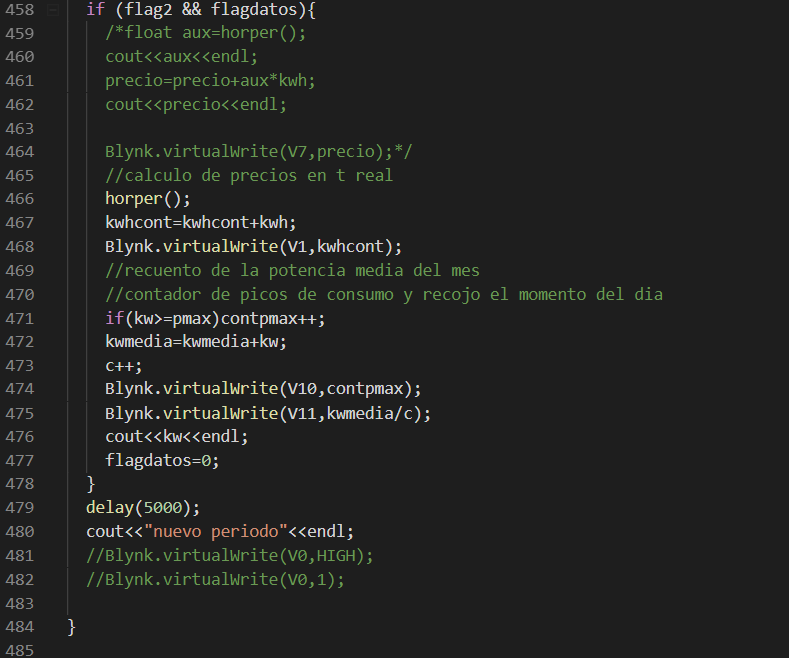
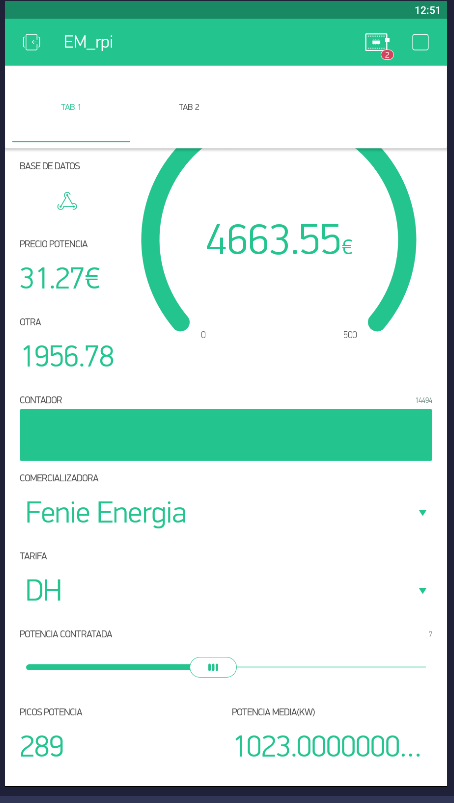


Figure 11

## Plataforma IoT Blynk

Figure 12



Las fotos de arriba se corresponden con la página principal de la app Blynk, widgets que la conforman:

* Consumo en tiempo real en euros
* Precio por potencia contratada ese mes
* Precio por consume en euros de un competidor seleccionado (otra comercializadora)
* Contador de consume en kwh del mes
* Seleccionable tipo desplegable para el tipo de tarifa 🡪
* SDH : sin discriminación horaria
* DH: con discriminacion horaria
* DH-Supervalle: con discriminacion horaria supervalle

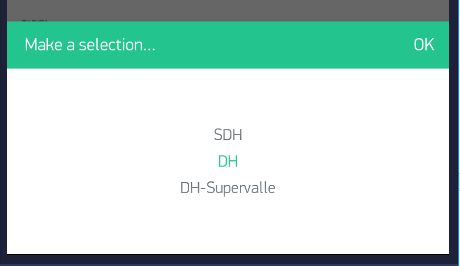


Figure 13

* Slider para seleccionar la potencia contratada
* Seleccionable tipo desplegable para sellecionar una de las comercializadoras disponibles.

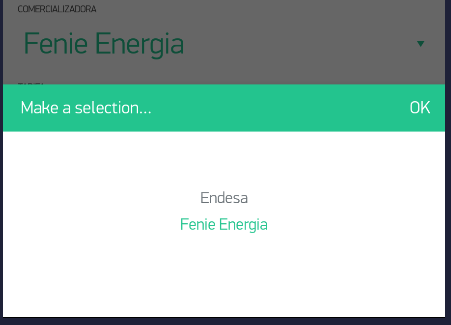


Figure 14

* Indicador de picos de potencia
* Indicador de media de potencia en ese period de factuación

Por otro lado en el menu secundario nos encontramos con lo siguiente

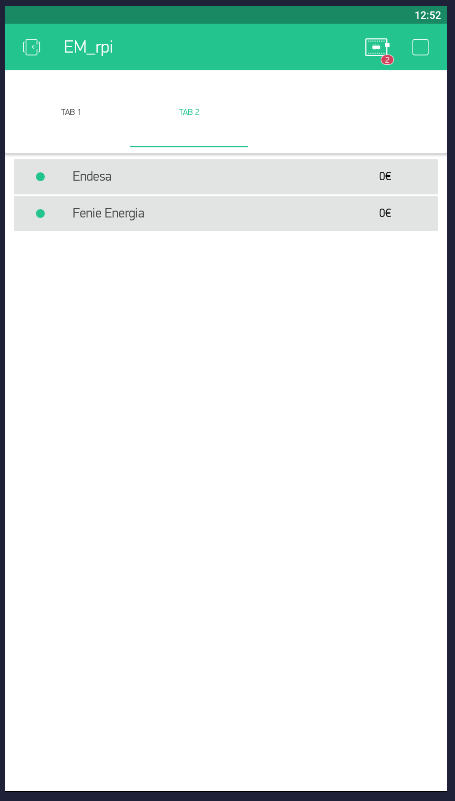


Figure 15

Aquí se va llevando el recuento en consume en euros no solo con tu comercializadora sino también con todas las que component la base de datos.

Hoy en día existen numerosas plataformas orientadas al IoT. En general estas plataformas mínimamente se encargan de comunicarse con el hardware (mcu), recogiendo datos desde el microcontrolador o lanzándole órdenes, además de guardar estos datos en la nube. Además de éstos, se pueden añadir más servicios del tipo, tratamiento Big Data.

Algunas plataformas conocidas serían: Thingspeak, Adafruit IO, Thinger…

Blynk, al igual que la mayoría de estas plataformas se encarga del control en remoto del hardware, visualización de datos asociados a los sensores, almacenamiento de datos en la nube, incluso en un servidor propio en local.

Esta es una de las grandes ventajas que te aporta Blynk, y es que te liberan el servidor, para que puedas montar tu propio servidor local y no tener que depender de servidores propios de la empresa. Además de esta manera, muchas características propias del servidor puedes modificarlas y adaptarlas de manera más específica a tu aplicación, como por ejemplo:

* Puertos asociados a cada canal de comunicación.
* Montar la base de datos en local, para posteriormente aplicar Big Data.
* Controlar parámetros de comunicación, de tiempo de espera, trafico de mensajes máximo…

En este caso se ha montado el servidor Blynk, como ya se ha mencionado dentro de una Rpi pi 3.

Un proyecto elaborado con blynk se compone de 3 partes fundamentales:

* La app, para IOS y Android. En ella se tiene que configurar el servidor asociado para que la app funcione, se tiene que componer la app usando distintos tipos de widgets, configurarlos y asociarlos a distintos pines virtuales para que luego desde el programa mcu se puedan controlar.
* El servidor Blynk, que puede ser local o usando servidores propios de la compañía.
* Hardware final, el cual va a ser finalmente goberando por la app. Este hardware tiene que contar con un programa basado en librerias que se nos proporcionan.

Entonces cuando desde la app se actue un botón, esta acción llega al servidor, se procesa y reenvía la orden al hardware para que realice la acción final. O vicecersa, si se manda un dato desde el hardware hacia la app, almacenándose ese dato en la base de datos a su paso por el servidor.

Pasos a seguir para poder usar la plataforma:

* Descarga de la librería para el harware en cuestión
* Descarga de la app móvil para IOS o Android, y plantear los widgets que van a ser usados, asi como asociarlos a los pines virtuales que toque para hacer referencia a ellos posteriormente desde el programa hardware
* Descargar el servidor blynk si se quiere explotar al máximo las ventajas que esta plataforma ofrece e instalarlo dentro del hardware respectivo.

Para hacerlo mas robusto lo ideal es hacer que el servidor se dispare automáticamente tras el inicio del ordenador o placa donde se haya alojado. Por ejemplo tras la perdida de alimentación y recuperarla, el servidor arrancaría automáticamente.

En este caso se ha usado el administrador de procesos Cron.

También es necesario que el servidor tenga asociada una ip estática, así como hacer “port forwarding” para redirigir todas las peticiones en relación al servidor blynk hacia la maquina que lo aloja.

Por ejmplo para la comunicación del servidor con el hardware se ha usado el puerto 8082, para la comunicación con la app Blynk se ha usado el 9443, y para el control en remoto de la rpi via ssh, incluso desde fuera de la red local se ha usado el puerto 22.

Captura asociada a los puertos del router para abrir la comunicación hacia el exterior de la red:

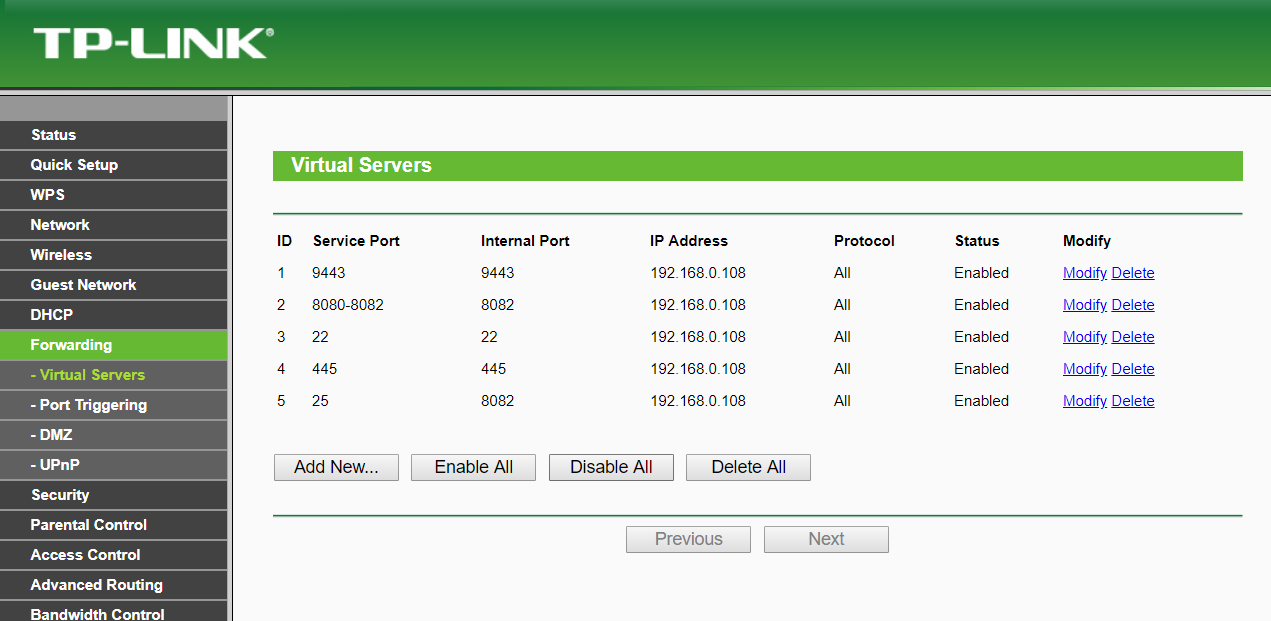


Figure 16

Cabe destacar el uso de un rango de puertos externos asociados a un único puerto interno. Como por ejemplo el [8080,8082] 🡪 8082. En ciertas redes locales se detectó que el firewall del router bloqueaba la comunicación asociada al puerto 8082, sin embargo usando puertos más usuales como el 8080, si dejaba realizarla. Es como si se le hicera pensar al router primero donde esta instalado el harware de sensado que realiza comunicaciones usando el puerto 8080, cuando en realidad se esta usando el 8082 dentro del router donde se encuentra el servidor. De esta manera el hardware instalado en las casas de usuarios podría usar cualquier puerto dentro del rango [8080-8082].

Como documento de guía se ha seguido el siguiente enlace:

<https://github.com/blynkkk/blynk-server>

## Plataforma Google Cloud

Plataforma similar a las que ofrecen Microsoft (Azure) y Amazon (Amazon Web Services), que basadas en la nube reunen una serie de servicios de distinto ámbito.

Destacan:

* Compute engine, basado en máquinas virtuales que usan la infraestructura de Google, escalables y altamente editables-configurables tanto en “hardware”, como en imagen que corren, se alquilan por tiempo de uso y en función también de la configuración usada.

En cuanto al hardware puedes seleccionar el tamaño y tecnología de disco duro, RAM, CPU’s, GPU…

* Cloud Storage, almacenamiento basado en la nube de todo tipo de datos.
* Cloud SQL.
* Persisten Disk, usado en máquinas virtuales.
* Análisis de datos, plataforma de big data para análisis de datos
* Servicios de inteligencia artificial
* Aprendizaje automático
* Análisis de video
* Análisis de imagen
* Análisis de texto
* Reconocimiento de voz

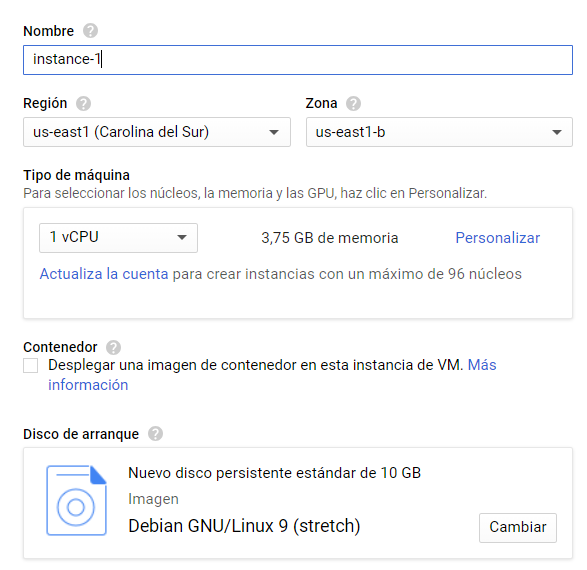
## Puesta a punto de la máquina virtual dentro de GCP

Primero se necesita una cuenta de google, posteriormente hay que enlazar esta cuenta con la plataforma de Google cloud. Se adjunta enlace:

<https://cloud.google.com/?hl=es>

En el momento de escribir el presente documento Google ofrece una prueba gratuita al inscribirte, con un crédito de 300$, con caducidad un año, para que pruebes la plataforma.

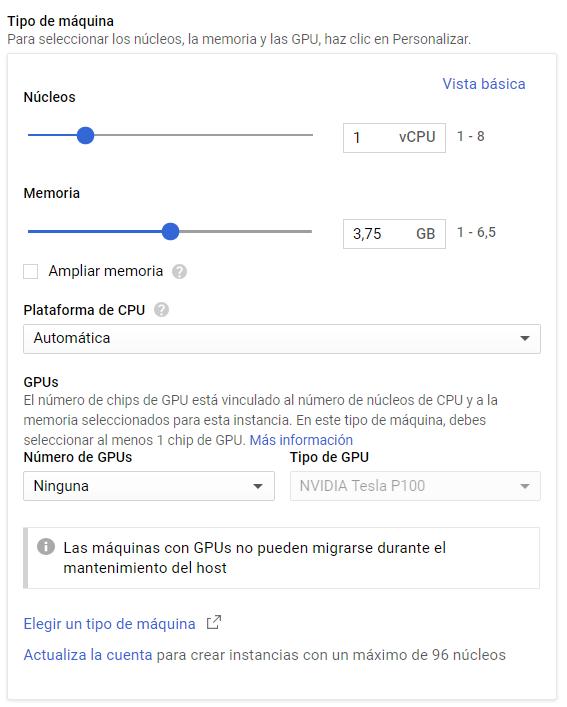
El siguiente paso acceder a la pesaña de Compute Engine, crear instancia, apareciendo el siguiente menu de configuración…



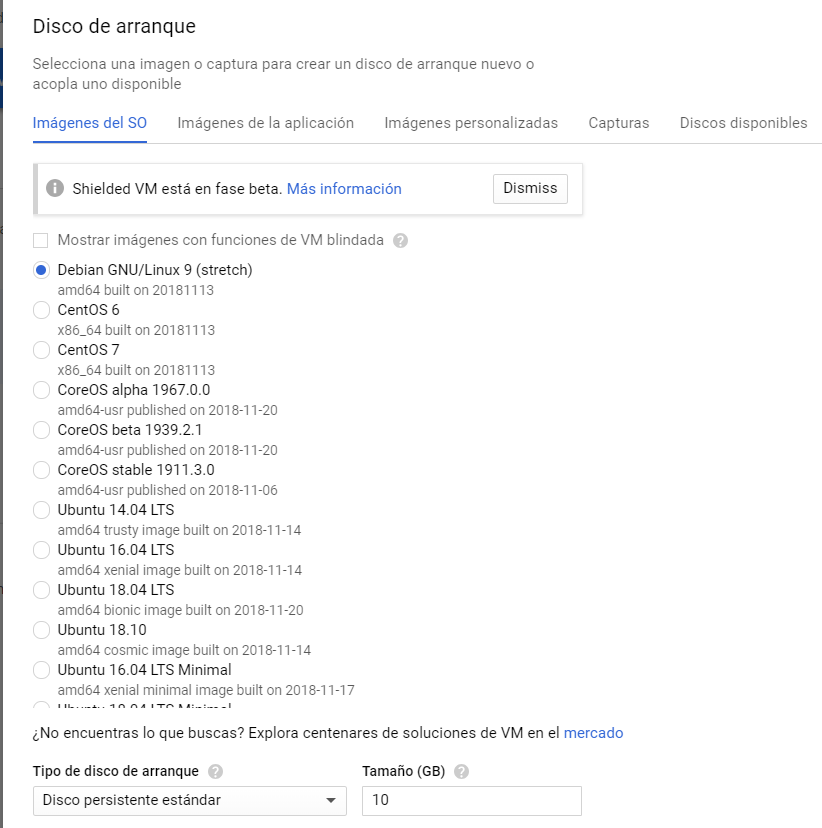
Donde hay que seleccionar la región en la que queremos ejecutar los recursos.

También la configuración de la máquina:

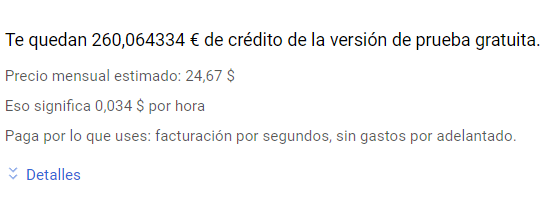
* Número de nucleos virtuales
* RAM
* Generacion del procesador de la máquina que ejecuta la VM
* Habilitar GPU y tipo de de la misma en caso de que aplique



Por último antes de crear la instancia, seleccionar la imagen que queremos que ejecute la VM, así como la tecnología y capacidad del disco duro.



Una vez se haya configurado la máquina la plataforma muestra una estimación del coste por alquiler de la máquina, que en mi caso queda a unos 3 centimos por hora de alquiler.



Tras crear la instancia se proporciona la IP externa de la máquina para acceder a ella en remote usando SSH, la plataforma ofrece un shell online para controlar en remote la misma. Si se quiere controlar la máquina usando la shell propia del pc personal o por ejemplo putty, hay que generar una clave y referenciarla dentro de la plataforma.

## Base de datos Mysql

Se ha montado una base de datos propia como se ha mencionado anteriormente, que almacena los precios asociados a las distintas comercializadoras.

Entonces para el mercado libre, para una potencia contratada de hasta 10 kw, tenemos por ejemplo:

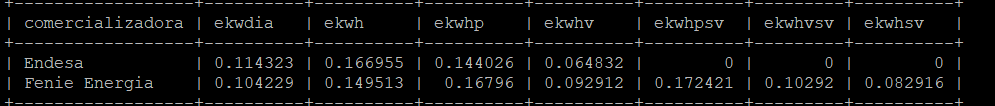


Figure 17

De izquierda a derecha:

* Nombre de la comercializadora
* Precio por potencia contratada por dia
* Precio por consumo con tarifa sin discriminación horaria
* Precios por consumo con tarifa con discriminación horaria
* Precios por consumo con tarifa con discirminacion horaria supervalle.

La idea es tener una tabla asociada a cada rango de potencia, para cada comercializadora. Además de ampliar el numero de comercializadoras.

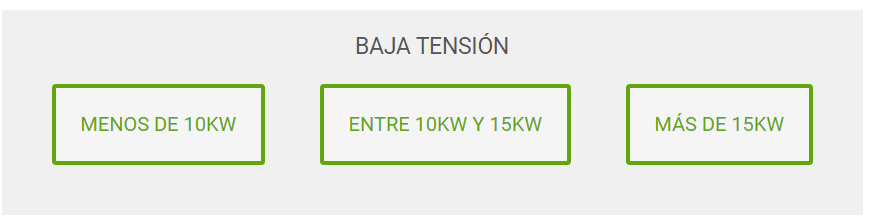


Figure 18

## Servidor LAMP

Tutorial seguido para instalar todas las dependencias:

<https://geekytheory.com/tutorial-raspberry-pi-15-instalacion-de-apache-mysql-php>

Para poder servir los datos de la base de datos hacia el mcu o mpu para que realice los cálculos de consumo en tiempo real, se monta un servidor LAMP.

LAMP es un término usado para hacer referencia al sistema de infraestructura de internet que usa las herramientas

* Linux, el sistema operativo de la máquina que lo alberga. En este caso usamos Raspbian, que ya mencionamos en el primer punto de este apartado.
* Apache, el servidor web como tal.
* Mysql o MariaDB como gestor de base de datos. En cuanto a versiones usadas, estamos usando Mysql 5.5 en Raspberry y 5.6 dentro de la máquina virtual alquilada a Google.
* Php, Python… como lenguajes de programación.En raspberry usamos PHP5, y en la máquina de Google estamos usando la evolución PHP7

Este servidor montado dentro de la Rpi a modo de complemento al servidor local blynk, se encarga de devolver los datos a todo cliente que realice la petición.

Además se tiene un código PHP que se encarga de recoger los datos de una petición GET y almacenarlos dentro de la base de datos y tabla respectivos. Este código está pensado para almacenar los datos de los sensores, y otros datos que se consideren de interés para ser explotados posteriormente aplicando BigData.

Ejemplo de consulta con respuesta del servidor:

Consulta: <http://89.39.22.36/mysql2.php>

Respuesta:

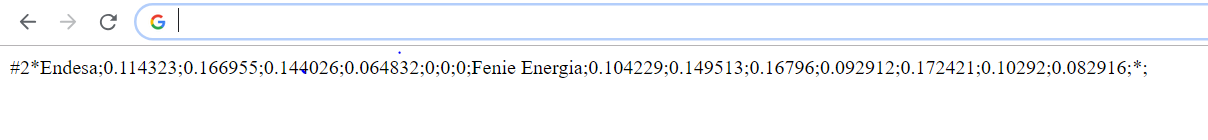


Figure 19

Como se aprecia en la imagen anterior, los datos los separo usando como delimitador una coma. También previo al envio de datos masivo, indico el numero de comercializadoras que componen los datos que voy a enviar.

# Resultados

E

n cuanto a los resultados finales obtenidos y las características del sistema obtenido hay que hablar por un lado de los datos referentes a los sensores (velocidad de lectura, veracidad de los mismos…), las suposiciones y consideraciones tenidas en software (como por ejemplo el tiempo durante el cual se considera que el consumo de potencia es el mismo)

## Muestreo de tension y corriente

Como se comentó en el apartado 3.3, el sensado consiste en ir recogiendo muestras alternativas de tension y corriente en un intervalo de aproximadamente 200 us. Cada ciclo básico de sensado tiene un periodo de 200 us, 25 us supone la conversion del dato por parte del ADC y el resto se emplea para cálculos intermedios:

* Suma cuadrática de valores instantáneos de tension o corriente según toque.
* Detectar cruces por cero de tension y corriente para el cálculo del desfase
* Recoger los valores pico para obtener un segundo valor de RMS.

Este periodo de sensado equivale a una frecuencia de muestreo de 5khz, si tenemos en cuenta que son muestras intercaladas de tensión y corriente tenemos una frecuencia real de 2.5khz.

Finalmente se contrastan los datos obtenidos de tensión y corriente con valores obtenidos con un polímetro certificado:

* Para voltaje
* Para corriente

Si ahora comparamos el valor obtenido para el cosfi, potencia activa y reactiva con los datos proporcionados por un analizador de red tenemos:

## Consideraciones para el software

Características y consideraciones referentes a la programación :

* Tiempo entre periodo de lectura y siguiente donde considero que la potencia consumida se ha mantenido constante: 5s
* Dias en media de un mes para el gasto por potencia contratada: 30 días.
* Considero que las cargas conectadas a mi Sistema son lineales y por tanto ondas de tensión y corriente son sinoidales, no considero armónicos.
* Pequeño ajuste por software para corregir el error del cosfi, relacionado con los 200us de desfase entre las muestras de tensión y corriente.
* Programa que se encarga de gestionar la app móvil y los datos desde el mcu están en el mismo servidor. Lo ideal que el mcu pudiera albergar el sensado y gestión de la app.

## Consideraciones para el hardware

Consideraciones y características referentes al hardware:

* Uso sensores de corriente de hasta 30Arms, luego el máximo de potencia contratada admisible sería de 7KW.
* Solo un sensor de corriente 🡪 circuito general
* El sensor de corriente desvirtúa la medida en torno a los 3º por ser una carga de tipo inductiva.
* Igualmente ocurre con el transformador que desvirtúa la medida.

## Test del sistema completo

Se procede a realizer una prueba real del sistema durante un periodo de tiempo considerable (x días).

Problemas detectados:

* 1 esto es nuevo

Avances durante ese periodo de tiempo conectado:

* 2 esto también

# Referencias

|  |  |
| --- | --- |
| [1] | Autor, «Este es el ejemplo de una cita,» *Tesis Doctoral,* vol. 2, nº 13, 2012. |
| [2] | O. Autor, «Otra cita distinta,» *revista,* p. 12, 2001. |

# **Índice de Conceptos**

conceptos 9

# **Glosario**

ISO: International Organization for Standardization 4

UNE: Una Norma Española 4