uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías de Telecomunicación
Curso 2016/2017

Trabajo Fin de Grado

Monitorización de la energía consumida mediante Raspberry Pi para sistema domótico

---

Alejandro Barón Cuevas

Tutor

Victor P.Gil Jiménez

Leganés, a 26 de septiembre de 2017

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ACRÓNIMOS

- **AC:** Alternating Current (Corriente Alterna).
- **ACS:** Alternating Current Sensor (Sensor de Corriente Alterna).
- **AGND:** Analogic Ground (Tierra Analógica).
- **ARM:** Advanced RISC Machine (Máquina RISC Avanzada).
- **A/D:** Analogic/Digital (Analógico/Digital).
- **CE:** Conformidad Europea.
- **CEE:** Comunidad Económica Europea.
- **CEM:** Compatibilidad Electromagnética.
- **CH:** Channel (Canal).
- **CLK:** Clock (Reloj).
- **CPU:** Central Processing Unit (Unidad Central de Procesamiento).
- **CS:** Chip Select (Selector de Chip).
- **CSI:** Camera Serial Interface (Interfaz Serie de la Cámara).
- **CTRL:** Control.
- **DGND:** Digital Ground (Tierra Digital).
- **DSI:** Display Serial Interface (Interfaz Serie del Monitor).
- **GB:** Gigabyte.
- **GLCD:** Graphic Liquid Crystal Display (Pantalla Gráfica de Cristal Líquido).
- **GND:** Ground (Tierra).
- **GNU:** GNU's Not Unix (GNU no es Unix).
- **GPIO:** General Purpose Input/Output (Entrada/Salida de Uso General).
- **GPU**: Graphics Processor Unit (Unidad de Procesador de gráficos).
- **HBES**: Home and Building Electronic Systems (Sistemas Electrónicos para el Hogar y la Construcción).
- **HDMI**: High-Definition Multimedia Interface (Interfaz Multimedia de Alta Definición).
- **ICT**: Infraestructura Común de Telecomunicaciones.
- **IoT**: Internet of Things (Internet de las cosas).
- **IP**: Internet Protocol (Protocolo de Internet).
- **I2C**: Inter-Integrated Circuit (Circuito Inter-Integrado).
- **LAN**: Local Area Network (Red de Área Local).

- **LCD**: Liquid Cristal Display (Pantalla de Cristal Líquido).
- **LED**: Light-Emitting Diode (Diodo Emisor de Luz).
- **MISO**: Master Input Slave Output (Salida Esclava de Entrada Maestra).
- **MOSI**: Master Output Slave Input (Entrada Esclava de Salida Maestra).
- **OTG**: On-The-Go (Sobre la marcha).
- **PC:** Personal Computer (Ordenador Personal).
- **PCB**: Printed Circuit Board (Placa de Circuito Impreso).
- **RAM**: Random Access Memory (Memoria de Acceso Aleatorio).
- **RCA**: Radio Corporation of America (Radio Corporación de América).
- **RD:** Real Decreto.
- **RJ:** Registered Jack (Jack Registrado).
- **RMS:** Root Mean Square (Media cuadrática).
- **SBC:** Single Board Computer (Ordenador de placa reducida).
- **SCL:** Signal Clock (Señal de Reloj).
- **SCLK:** Serial Clock (Reloj de Serie).
- **SCT:** Sensor Current Transformer (Transformador Sensor de Corriente).
- **SD:** Secure Digital (Seguro Digital).
- **SDA:** Signal Data (Datos de Señal).
- **SPI:** Serial Peripheral Interface (Interfaz Periférica de Serie).
- **SS/Select:** Selector-Slave (Selector de Esclavo).
- **SSH:** Secure SHell (Intérprete de Órdenes seguras).
- **TFG:** Trabajo Fin de Grado.
- **TIC:** Tecnologías de la Información y la Comunicación.
- **UART:** Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal).
- **UNE-EN**: Una Norma Española-European Norm.
- **USB:** Universal Serial Bus (Bus Serie Universal).
- **VDD:** Voltaje Drenador Drenador.
- **VNC**: Virtual Network Computing (Computación Virtual en Red).
- **VREF:** Voltaje de Referencia.
- **XBMC:** Xbox Media Center (Centro Multimedia de Entretenimiento).
- **ZIP:** Zone Improvement Plan (Plan de Mejora de Zona).

Resumen

La actual situación económica mundial y el gran aumento de la demanda energética han favorecido, junto con el fuerte empuje del sector TIC, el desarrollo de técnicas que buscan transformar completamente la vida de las personas.

Dentro de estas ellas se encuentra el IoT, cuyo objetivo es conectar el máximo de objetos que nos rodean, entre ellos y con las personas, con el fin de obtener datos en tiempo real que permitan optimizar los aparatos electrónicos y mejorar la calidad de vida de sus usuarios.

Una de sus aplicaciones más importantes se basa en el ahorro energético, ya que permite la creación de sistemas inteligentes capaces de optimizar de forma autónoma cada uno de los aparatos eléctricos que contengan. Gracias al uso de sensores, estos dispositivos recogen una gran cantidad de datos acerca de su comportamiento, los cuales son procesados con el fin de extraer información relevante para conseguir una reducción del gasto energético.

El presente proyecto se encuentra dentro de este último ámbito. El objetivo es realizar un sistema de monitorización de energía consumida para una instalación domótica mediante el uso de una Raspberry Pi, en el que se debe realizar el diseño del circuito y la programación correspondientes para poder extraer datos de diez sensores de corriente de forma simultánea.

Para desarrollar el sistema de forma completa faltaría el hecho de crear un servidor con toda la información recogida, al que accedería un cliente para el visionado de los datos de su instalación, y en cuyo desempeño se basa el TFG de otra compañera.

Abstract

The current global economic situation and the increase in energy demand have favored, along with the strong push of the TIC sector, the development of techniques which looked about for transform completely people's lives.

Among these techniques, we can find the IoT, whose objective is to connect the maximum of the objects which surround us, between them and with people, in order to obtain data in real time which allows optimizing electronic devices and improving the quality users' life.

One of its most important applications is based on saving energy, considering that it permits to create intelligent systems which are allowed to optimize in an autonomous way each electric device which forms it.

Thanks to sensors, these devices pick up a big quantity of data about its behavior, each one of which are processed with the goal to get relevant information to obtain a great reduction about the electric waste.

The current project is within the latest scope. The goal of this project is to create a system which monitoring the consumed energy in a domotic installation thanks to use a Raspberry Pi where to make the circuit design and the correct program to extract data in ten electric sensors in a simultaneous way.

To develop the system in a completely way, it would be necessary to create a server with all the collected information where the client would access to view the facility's information. This performance is the goal of another classmate's TFG.

1. Introducción

1.1. Motivación

Este trabajo fin de grado ha sido escogido debido principalmente a la cantidad de conocimientos que son necesarios aplicar para su consecución, desde llevar a cabo algoritmos de programación hasta construir un circuito capaz de gestionar la energía que circula a través del mismo.

Todos ellos, adquiridos a lo largo de la carrera, se unen a la necesidad de trabajar a través de un ordenador de tamaño reducido, en este caso una Raspberry Pi[1], que es quién se encargará de la inteligencia del sistema, y que supone una novedad con respecto a lo trabajado anteriormente.

Además, el hecho de que el trabajo tuviera la oportunidad de adentrar en el mundo de los sistemas de comunicaciones domóticos y el IoT[2] terminó por convencerme plenamente, puesto que desde el primer momento mi intención era realizar un proyecto que permita mejorar la calidad de vida del usuario que lo utilice.

Todo ello, unido al hecho de trabajar de forma autónoma ha supuesto un gran desafío, pero que me ha servido para culminar el proyecto de forma satisfactoria y mejorar tanto a nivel personal como profesional.

1.2. Objetivos

El objetivo de este proyecto es diseñar e implementar un sistema de monitorización de energía consumida mediante una Raspberry Pi para una instalación domótica, no invasivo y de fácil instalación para el cliente final. Está basado en una aplicación del IoT para reducir el consumo energético de una vivienda, y por tanto, favorecer su ahorro energético,

Además, este tiene que ser de bajo precio y debe obtener datos de diez sensores de corriente de manera simultánea, donde cada uno de ellos recoge información correspondiente a una línea del cuadro de electricidad donde irá instalado. Dentro del desarrollo del sistema se pueden diferenciar dos partes:

- Por un lado, se debe realizar el diseño y montaje del circuito de sensado de la señal, que es aquel que lleva los datos obtenidos en los sensores de corriente hacia la Raspberry Pi para que puedan ser gestionados correctamente. Para ello se ha utilizado una placa protoboard sobre la que se ha implementado el circuito.

- Por otra parte, se debe aprender a configurar la Raspberry Pi y su funcionamiento, así como realizar la programación necesaria para la lectura y procesado de los datos obtenidos a través de cada uno de los sensores, que serán guardados en diferentes ficheros dependiendo del canal de los que provengan, denominados datos_canalX.txt, donde X es el número de canal/sensor.

  Dentro de ello, además del programa principal se debe llevar a cabo la configuración de un demonio de arranque automático, de forma que cada vez que se encienda la Raspberry Pi el sistema recoja los datos a través de los sensores en segundo plano, sin necesidad de ejecutar ninguna instrucción desde la consola.

Por último, el sistema debe ser capaz, introduciendo un comando determinado para cada opción, de:

1) Modificar el tiempo existente entre la recogida de cada muestra para un canal específico, que será guardado en un fichero de tiempos para su continua lectura.

2) Recoger datos a partir de un canal, fecha y hora, de forma que se leerá el fichero datos_canalX correspondiente al canal introducido, y se guardarán en otro fichero todos aquellos datos posteriores a la fecha y hora introducidos.

Por tanto, se puede decir que para diseñar el sistema en su plenitud es necesario tener conocimientos, básicos al menos, tanto de programación como de diseño de circuitos.

1.3. Recursos utilizados

Los medios utilizados para la realización de este proyecto han sido los siguientes:

- **Raspberry Pi 3 Model B:** computadora de bajo coste empleada como nodo central del proyecto, cuyas características y funciones se explicarán con detalle más adelante.
- **Cargador micro-USB**: alimentación de la Raspberry Pi.
- **Tarjeta micro-SD de 16Gb:** proporciona almacenamiento a la Raspberry Pi.
- **Cable Ethernet:** para conectar la Raspberry Pi al router.
- **Bus GPIO conexión Raspberry Pi B+:** para conectar los pines GPIO de la Raspberry Pi con la protoboard.
- **Dos protoboard de 830 contactos:** placa de pruebas para la realización del circuito de sensado.
- **Tres amplificadores TL084:** para rectificar la señal del sensor.
- **Cable de conexión:** para unir elementos en la protoboard.
- **10 sensores de corriente SCT-013-030:** para recoger la corriente del aparato a medir.

- **10 conectores Jack hembra 3.5mm:** para conectar el sensor con la protoboard.
- **2 conversores analógico/digital MCP3008:** para convertir la señal analógica que proviene del sensor a digital para introducirla en la Raspberry Pi.
- **Ordenador portátil:** necesario para la realización general del proyecto.

1.4. Estructura del documento

El documento está estructurado en diferentes apartados con el objetivo de facilitar su comprensión y una rápida búsqueda de información. Dentro de ellos, se explican los siguientes contenidos:

1) Introducción: las motivaciones que hicieron escoger este proyecto, en qué consiste, cuáles son los objetivos a conseguir, qué medios han sido utilizados para ello, la planificación temporal del mismo, su entorno socio-económico y el marco regulador en el que se encuentra.

2) Estado del arte: el contexto en el que se encuentra el proyecto, así como dar a conocer todos los conceptos necesarios para el correcto entendimiento del mismo, con el objetivo de introducir al lector en los elementos y tecnologías que lo componen. Además, contiene un estudio de las herramientas y plataforma empleadas, justificando por qué se han usado unos medios y materiales en vez de otros, así como se presentarán soluciones similares ya existentes, cuyos conceptos fueron de gran importancia para aportar una base de conocimiento sobre la que partir.

3) Raspberry Pi: las características y componentes de la Raspberry Pi, se explica paso a paso la configuración en su primer encendido, el funcionamiento de los terminales GPIO (los cuales se usarán para conectar la Raspberry Pi y el circuito de sensado), y cómo conectarla a un PC para trabajar desde el mismo de forma remota.

4) Programación básica del sistema: cómo instalar el paquete Python en la Raspberry Pi y la configuración del demonio de arranque automático.

5) Desarrollo de la solución: parte hardware: toda la parte hardware que compone el proyecto, así como el circuito completo y la conexión entre el mismo y la Raspberry Pi.

6) Desarrollo de la solución: parte software: el código implementado en la Raspberry Pi para poder mostrar la energía obtenida en cada sensor de corriente, así como las opciones adicionales a la recogida de datos.

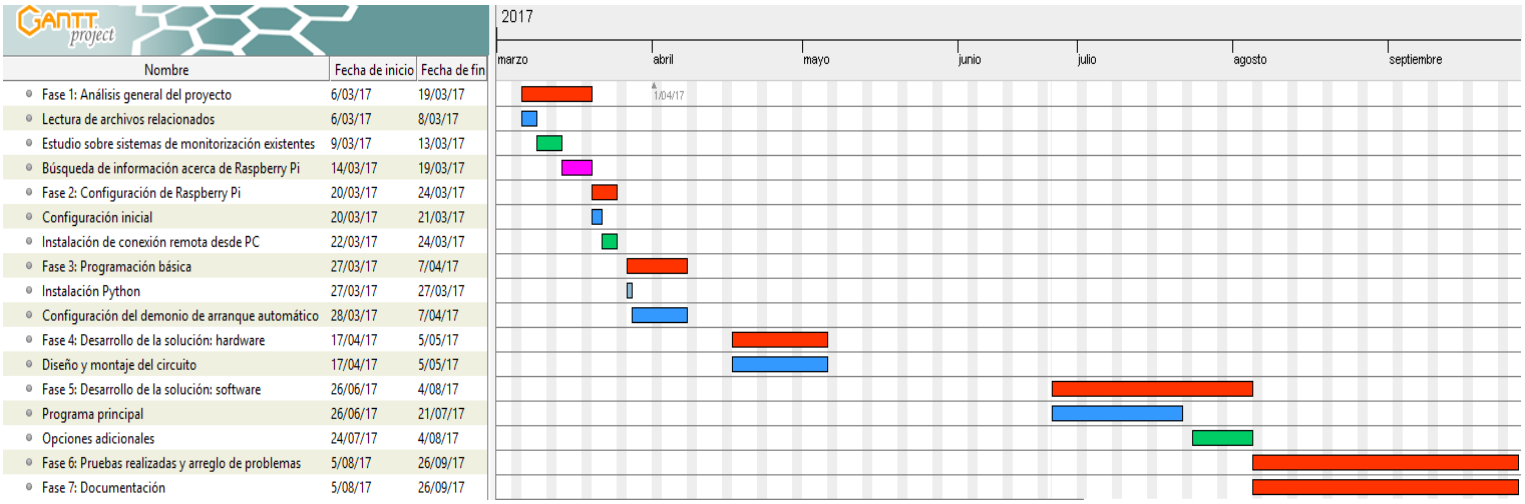
7) Valoración final: las pruebas realizadas al finalizar el sistema, los problemas encontrados a lo largo del mismo, una conclusión en la que se detalla si los objetivos planteados en la introducción se han visto cumplidos y en qué medida, qué se ha aprendido durante la realización de este proyecto, y las posibles mejoras o líneas futuras sobre las que seguir desarrollando el sistema.

8) Fuentes consultadas: la bibliografía y referencias consultadas a la hora de realizar este proyecto. Cada una de ellas tendrá un número asociado, de forma que cuando se mencione algo en el texto se pueda encontrar su información en este apartado con facilidad.

A) Anexos: todo el contenido relevante para el proyecto pero que no ha sido incluido dentro de la propia estructura del documento.

1.5. Fases del proyecto

Las fases desarrolladas a lo largo del proyecto son las que se muestran en el siguiente diagrama de Gantt[3].



Nombre	Fecha de inicio	Fecha de fin
Fase 1: Análisis general del proyecto	6/03/17	19/03/17
Lectura de archivos relacionados	6/03/17	8/03/17
Estudio sobre sistemas de monitorización existentes	9/03/17	13/03/17
Búsqueda de información acerca de Raspberry Pi	14/03/17	19/03/17
Fase 2: Configuración de Raspberry Pi	20/03/17	24/03/17
Configuración inicial	20/03/17	21/03/17
Instalación de conexión remota desde PC	22/03/17	24/03/17
Fase 3: Programación básica	27/03/17	7/04/17
Instalación Python	27/03/17	27/03/17
Configuración del demonio de arranque automático	28/03/17	7/04/17
Fase 4: Desarrollo de la solución: hardware	17/04/17	5/05/17
Diseño y montaje del circuito	17/04/17	5/05/17
Fase 5: Desarrollo de la solución: software	26/06/17	4/08/17
Programa principal	26/06/17	21/07/17
Opciones adicionales	24/07/17	4/08/17
Fase 6: Pruebas realizadas y arreglo de problemas	5/08/17	26/09/17
Fase 7: Documentación	5/08/17	26/09/17

Figura 1: Diagrama de Gantt.

Teniendo en cuenta que entre el 5 de mayo y el 22 de junio no se trabajó el proyecto, finalmente la duración de realización del proyecto ha sido de algo más de 4 meses (128 días), lo que a una media de trabajo de 2’5h/día ha provocado que el número de horas empleadas en su totalidad haya sido de 330 h.

1.6. Marco socio-económico

1.6.1. Presupuesto

COSTES DE PERSONAL

	Descripción	Unidades	Horas	Precio/hora	Precio Total
1	**Ingeniero de Telecomunicaciones:** Conocimientos básicos en programación, microcontroladores y sistemas de comunicaciones.	1	325	30€	9750€
2	**Ingeniero Técnico de Telecomunicación.** Tutor, con conocimientos amplios de sistemas de comunicación y telemática. Consultas de 1h.	1	4	60€	240€
				COSTE TOTAL:	**9990€**

Tabla 1: Costes de personal.

COSTES DE HARDWARE

	Descripción	Precio	Cantidad	Precio total
1	**Raspberry Pi 3 Model B**	36,72€	1	36,72€
2	**Cargador micro-USB**	6,99€	1	6,99€
3	**Tarjeta micro-SD de 16Gb**	5,63€	1	5,63€
4	**Cable Ethernet**	5,95€	1	5,95€
5	**Bus GPIO conexión Raspberry Pi B+**	8,10€	1	8,10€
6	**Protoboard de 830 contactos**	3,25€	2	6,50€
7	**Amplificador TL084**	0,50€	3	1’50€
8	**Cable de conexión**	1€	2	2€
9	**Sensor de corriente SCT-013-030**	5,05€	10	50,50€
10	**Conector Jack hembra 3.5mm**	1€	10	10€
11	**Conversor A/D MCP3008**	7,80€	2	15,60€
			PRECIO TOTAL:	**149,49€**

Tabla 2: Costes de hardware.

COSTES DE SOFTWARE

	Descripción	Precio
1	**Microsoft Office 2017 Enterprise**	78,40€/año
2	**Descarga de Python**	0€ (Software libre)
3	**Descarga de PuTTY**	0€ (Software libre)
4	**Descarga de VNC Viewer**	0€ (Software libre)
5	**Descarga de Win32 Disk Imager**	0€ (Software libre)
	PRECIO TOTAL:	**78,40€**

Tabla 3: Costes de software.

COSTES INDIRECTOS

	Descripción	Precio	Cantidad	Precio Total
1	**Ordenador portátil ASUS**	400€	1	400€
2	**Gastos de luz**	10€/mes	4	40€
3	**Gastos de conexión a Internet**	15€/mes	4	60€
			PRECIO TOTAL:	**500€**

Tabla 4: Costes indirectos.

COSTE TOTAL

	Descripción	Precio
1	**Costes de personal**	9990€
2	**Costes de software**	78,40€
3	**Costes de hardware**	149,49€
4	**Costes indirectos**	500€
	PRECIO TOTAL:	**10717,89€**

Tabla 5: Coste total.

1.6.2. Impacto socio-económico

Gracias al desarrollo de la domótica es posible transformar cualquier hogar en una vivienda inteligente, donde se automaticen procesos que harán la vida de sus propietarios más cómoda, tranquila y centrándose principalmente en aspectos como el ahorro energético, el confort y las comunicaciones.

En este sentido, una de las ventajas que ofrece la domótica es la instalación de sistemas de monitorización de energía, que es el producto en que se basa el presente proyecto. Éstos ofrecen una serie de beneficios para sus usuarios como:

- **Sociales:** posibilidad de una mejor gestión de sus equipos y dispositivos eléctricos, permitiendo configurar el uso de su instalación dependiendo del gasto de cada uno de ellos. Además, las futuras líneas del producto permiten manejar el control de gasto energético desde un ordenador, tablet o smartphone, lo que supone una mayor comodidad para el usuario

- **Económicos:** la instalación de un sistema de monitorización como el presente permite a los usuarios de una vivienda un consumo de sus recursos energéticos mucho más eficiente, lo que influirá, por tanto, en una disminución del gasto económico en electricidad.

- **Medioambientales**[4]**:** la monitorización de los equipos y electrodomésticos permite también fomentar acciones responsables con el medio ambiente al controlar el gasto de recursos eléctricos, contribuyendo de esta forma a un uso más eficiente de los mismos.

A la hora de entrar en el mercado, existen más y mejores sistemas similares (como se comentará en el apartado 2.3 de la memoria) aunque el hecho de que su fabricación suponga un bajo coste (únicamente teniendo en cuenta los gastos de hardware), así como su facilidad de instalación, hacen que el producto sea atractivo para aquellos usuarios que deseen un sistema de monitorización en sus viviendas sin desembolsar una gran cantidad de dinero.

1.7. Marco regulador

En toda instalación domótica se debe cumplir una serie de disposiciones legales y directivas específicas [5], contempladas en el Real Decreto 346/2011 del 11 de Marzo del 2011 dentro del reglamento ICT, entre las que se encuentran:

- **Directiva CE 2006/95/CE de Baja Tensión**: su objetivo es garantizar la seguridad al emplear materiales eléctricos.

- **Directiva CE 89/336/CEE de Compatibilidad Electromagnética**: tiene la finalidad de garantizar la protección de las personas contra los problemas causados por las perturbaciones electromagnéticas provocadas por los dispositivos electrónicos.

- **Reglamento Electrotécnico de Baja Tensión (RD 842/2002):** establece el marco de las condiciones técnicas y garantías que debe reunir una instalación eléctrica de baja tensión para:

    1) Afirmar la seguridad de las personas y bienes.
    2) Asegurar su normal funcionamiento y prevenir las perturbaciones de otras instalaciones y servicios.
    3) Contribuir a su fiabilidad técnica y eficiencia económica.

Por otra parte, entre las normativas y especificaciones aplicables que afectan a las instalaciones domóticas[6] se encuentran: las normas UNE-EN 50090 y UNE-EN 50491, la especificación nacional EA0026 y la especificación de ámbito europeo CLC/TR 50491-6-3:

- **Normas UNE-EN 50090 para sistemas electrónicos de viviendas y edificios (HBES)**: normalizan las aplicaciones de control del sistema de comunicación destinado a edificios y viviendas, cubriendo cualquier combinación de dispositivos conectados a través de una red de transmisión digital.

- **Normas UNE-EN 50491 para sistemas electrónicos de viviendas y edificios (HBES) y sistemas de automatización y control de edificios:** recogen los requisitos ambientales, de compatibilidad electromagnética (CEM) y seguridad de los sistemas electrónicos para viviendas, edificios y sistemas de automatización..

- **Especificación EA0026 para instalaciones de sistemas domóticos de viviendas**: establece los requisitos mínimos que debe cumplir el sistema domótico, fijando los requirimientos de instalación y evaluación. Surge con el objetivo de impulsar el desarrollo del mercado domótico.

- **Especificación CLC/TR 50491-6-3 para instalaciones de sistemas domóticos de viviendas:** incluye una clasificación que indica el nivel de ahorro energético ofrecido por los sistemas domóticos.

2. Estado del arte

2.1. Internet de las cosas

Para contextualizar este proyecto es necesario hablar sobre el IoT. Se trata de un nuevo escenario en el que cualquier aparato u objeto tiene conectividad a Internet, y con ello la capacidad para aportar información acerca de su conducta.

La mayor importancia de estos reside en que contienen sensores que permiten extraer información sobre su actividad, que será tratada para extraer una relación entre la misma y los comportamientos de sus usuarios. De este modo, es posible obtener una gran cantidad de ventajas como la monitorización de dispositivos, o aumentar el ahorro energético de una instalación, entre otros.

De este modo, por ejemplo, un frigorífico podría avisar sobre las existencias disponibles en su interior, y encargar autónomamente al supermercado la compra de aquellos elementos que se hayan agotado, o informar a su usuario de las posibles comidas que podría elaborar con ellos.

Todo ello también permite beneficiar tanto a la economía global como al medio ambiente, ya que se realiza un uso más eficiente de los recursos energéticos.

Gracias a toda esta información surgió el diseño del presente proyecto, un sistema de monitorización y control de consumo eléctrico que se basa en las aplicaciones del IoT en cuanto al ahorro energético, pero centrándose en el ámbito de la domótica, y que permita a los usuarios conocer el comportamiento de sus aparatos.

2.2. Herramientas utilizadas

2.2.1. Plataforma

La plataforma proporcionada por la universidad para implementar el sistema de monitorización de energía ha sido una Raspberry Pi 3 Model B. Se trata de un computador de placa reducida que presenta las funciones básicas de un ordenador corriente. Dentro de Raspberry Pi se pueden encontrar diferentes modelos:

Modelo	Raspberry Pi 1 Model B+	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B
CPU	ARM1176JZF-S (700 MHz)	ARM Cortex A7 Quad Core de 900 MHz	ARMv8 64-bit Quad-Core de 1.2 GHz
GPU	Broadcom VideoCore IV	Broadcom VideoCore IV	Broadcom VideoCore IV
Memoria RAM	512 MB	1 GB	1 GB
USB	4 USB 2.0	4 USB 2.0	4 USB 2.0
Ethernet	Sí	Sí	Sí
Salida de vídeo	HDMI, RCA	HDMI, RCA	HDMI, RCA
Salida de audio	3.5 mm Jack, HDMI	3.5 mm Jack, HDMI	3.5 mm Jack, HDMI
Almacenamiento	Micro SD	Micro SD	Micro SD
Tamaño	85.6 x 56 mm	85.6 x 56 mm	85.6 x 56 mm
Consumo energético	600 mA (3 W)	800 mA (4 W)	800 mA (4 W)

Tabla 6: Comparativa entre los 3 modelos más novedosos de Raspberry Pi.

Como se puede comprobar, la Raspberry Pi 3 Model B es la que mejores prestaciones ofrece de entre los modelos existentes. Para hacerla funcionar, basta con grabar el sistema operativo en una tarjeta SD, introducirla posteriormente en la Raspberry Pi y enchufar el aparato a la corriente.

Figura 2: Mini PC Raspberry Pi. Recuperada de: https://www.raspberrypi.org/forums/viewtopic.php?f=40&t=148891

Aunque el Rapsberry Pi es el ordenador de bajo coste más vendido y con mayor soporte, no es el único. Existen muchas otras alternativas[7], tanto en precio como en prestaciones y funcionalidades, que probablemente se adapten mejor a otros usuarios en situaciones concretas:

- **Banana Pi Pro**: es un mini ordenador con características similares a Raspberry Pi aunque con mejores características, como un procesador ARM de doble núcleo de 1 GHz o memoria RAM de 1 GB compartida con GPU.

Figura 3: Mini PC Banana Pi Pro. Recuperada de: http://raspi.tv/2014/banana-pi-review-first-impressions

- **Odroid-C2**: SBC que cuenta con 4 puertos USB 2.0, procesador QuadCore de 1.5 GHz y 2 GB de memoria RAM. Además, contiene una tarjeta de red Gigabit Ethernet, un puerto HDMI 2.0 y un puerto USB OTG. Es una opción similar a la anterior aunque con mejor rendimiento.

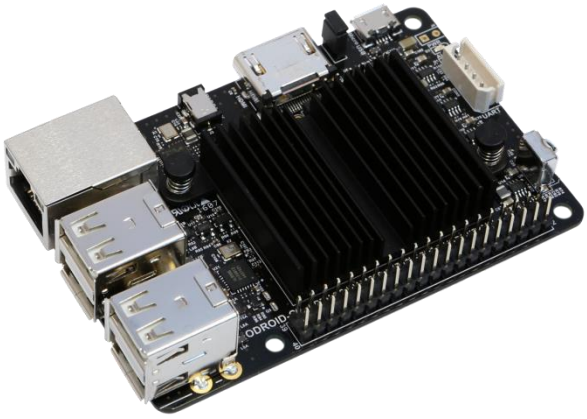


Figura 4: Mini PC Odroid-C2. Recuperada de: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145457216438

- **Jaguarboard**: contiene un procesador X86 en lugar de un procesador ARM, lo que le permite funcionar como un ordenador convencional y ejecutar los mismos programas. Sus principales características son: procesador Inter Atom Z3735G @1.83 GHz, memoria RAM de 1GB, memoria interna de 16GB, 3 puertos USB 2.0 y un puerto HDMI 1.4.

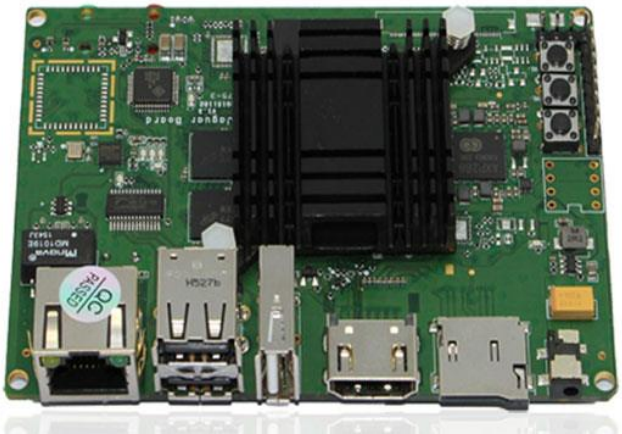


Figura 5: Mini PC Jaguarboard. Recuperada de: https://www.digikey.com

- **BeagleBone Black:** se diferencia con las demás en que posee almacenamiento interno de 4 GB. También cuenta con un procesador ARM de 1 GHz y una memoria RAM de 512 MB.

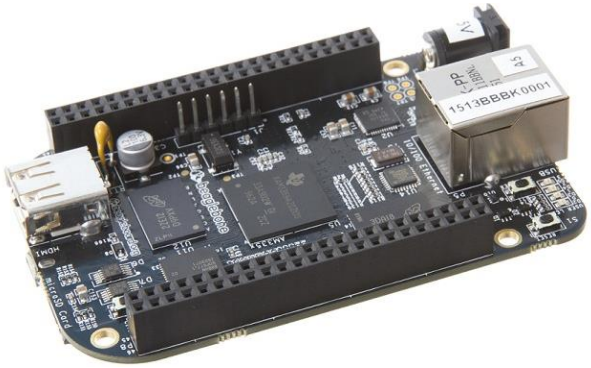


Figura 6: Mini PC BeagleBone Black. Recuperada de: http://elinux.org/Beagleboard:BeagleBoneBlack

- **Pine64:** se caracteriza por integrar un procesador de 64 bits Quad-Core ARM de 1.2 GHz. Además, cuenta con una memoria RAM de hasta 2 GB, tarjeta de red Gigabit Ethernet, 2 puertos USB 2.0, puerto HDMI 1.4 y WIFI y Bluetooth integrados.

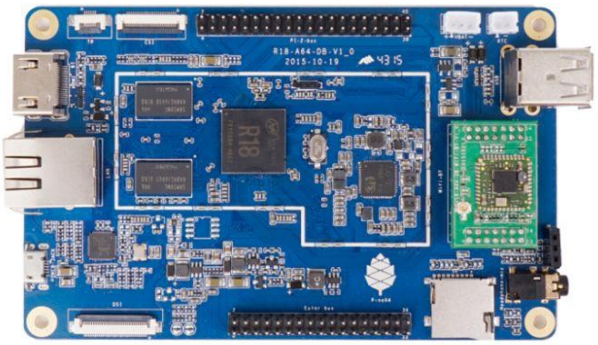


Figura 7: Mini PC Pine64. Recuperada de: http://www.techradar.com/news/computing/pc/pine-64-is-raspberry-pi-3-that-does-4k-but-you-ll-have-to-wait-to-get-one-1317136

Estas son sólo algunas de las alternativas a Raspberry Pi. A continuación se muestra una tabla comparativa con las características más relevantes de estos productos:

Modelo	Raspberry Pi 3 Model B	Banana Pi Pro	Odroid-C2	Jaguarboard	Beaglebone Black	Pine 64
CPU	ARMv8 64-bit Quad-Core de 1,2 GHz	ARM Cortex-A7 Dual-core de 1 GHz	Quad-Core Amlogic ARM Cortex A53 de 1,5 GHz	Intel Atom Z3735G de 1,83 GHz	ARM AM335x de 1GHz	Quad-Core ARM Cortex A53 de 64-bit a 1,2 GHz
Memoria RAM	1 GB	1 GB	2 GB	1 GB	512 MB	0,5/1/2 GB
Puertos	4 USB 2.0 1 HDMI 2.0	2 USB 2.0 1 USB OTG 1 HDMI 1.4	4 USB 2.0 y 1 USB OTG 1 HDMI 2.0	3 USB 2.0 1 HDMI 1.4	1 USB 2.0 1 HDMI 1.4	2 USB 2.0 1 HDMI 1.4
Almacenamiento	Micro SD	Micro SD	Micro SD	Interno de 16 GB	Interno de 4 GB	Micro SD
Tamaño	85,6 x 56 mm	92 x 60 mm	85 x 56 mm	101,9 x 64,5 mm	86,4 x 53,3 mm	12,7 x 7,9 cm
Precio	36,72€	48,50€	71,49€	74,88€	56,65€	119,11€

Tabla 7: Comparativa entre Raspberry Pi y sus alternativas.

Aunque existan alternativas más potentes que Raspberry Pi, la gran comunidad existente a su alrededor hace que sea la opción más adecuada a la hora de realizar un proyecto como el presente. Es el SBC más utilizado actualmente, por lo que existe una gran cantidad de información al respecto, lo que hace más probable que los errores comunes hayan sido identificados y solucionados, ya sea de modo oficial o mediante respuestas en foros por otros miembros de la comunidad.

2.2.2. Sistema operativo

Una vez escogida la plataforma que dará soporte al proyecto, toca elegir el sistema operativo sobre el que trabajar. Como finalmente se utilizará una Raspberry Pi, se ha de seleccionar el que mejor se adapte a los objetivos. Los más comunes son[8]:

- **ArchLinux:** parte de la familia GNU/Linux, se trata de un sistema operativo de simple entendimiento, pensado para la ejecución rápida de pequeños programas y usuarios sin grandes conocimientos. Este sistema queda descartado debido a su escasa optimización.

- **Rasplex:** sistema basado en XBMC, que permite conectar a un mismo dispositivo "servidor" múltiples dispositivos "cliente" a la vez. Este sistema operativo está pensado para utilizar la Raspberry Pi como servidor.

- **OpenELEC:** se trata de una distribución Linux, está creado para utilizar la Raspberry Pi como Media Center, idóneo para la reproducción de música, películas e imágenes de un disco local. Como el objetivo del proyecto es diferente, queda descartado.

- **RaspBMC:** al igual que openELEC, se trata de un sistema creado para la reproducción de archivos multimedia.

- **Pidora**: se trata de una distribución de Linux. Junto a ArchLinux y Raspbian, es uno de los mejores sistemas operativos presentes, aunque con peores resultados con respecto a Raspbian.

- **Raspbian:** distribución del sistema operativo GNU/Linux basado en Debian, este sistema operativo está diseñado para el procesador CPU de Raspberry Pi. Posee versiones con un entorno gráfico simple y fácil de usar, así como distribuciones más complejas cuyos resultados son mejores.

Debido a su facilidad de uso, y a que es el sistema operativo más optimizado para Raspberry Pi, finalmente **Raspbian**[9] **será el que se use a lo largo del proyecto**.

Figura 8: Logo Raspbian. Recuperado de: http://wallpapercraft.net/debian-logo/

2.2.3. Lenguaje de programación

Una vez elegido el sistema operativo, toca escoger el lenguaje de programación que se usará para dar formato al proyecto. Se puede optar por lenguajes clásicos, de fácil compresión, u otros más exhaustivos que permitan obtener el máximo potencial al sistema. Algunos de ellos pueden ser[10]:

- **Java:** lenguaje multiplataforma aunque poco extendido en el uso de Raspberry Pi, por lo que se ha descartado.

- **Javascript:** lenguaje de programación idóneo para trabajar con páginas web y servidores, con buen rendimiento y una comunidad amplia y activa. Como el proyecto no necesita de ello, se descarta.

- **Ensamblador:** se trata de un lenguaje de bajo nivel, que permite obtener unos resultados excelentes, pero que a su vez necesita conocimientos amplios para poder manejarlo.

- **C / C++:** es un lenguaje apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Disponible para los dispositivos señalados anteriormente, es un lenguaje bastante usado en Raspberry Pi.

- **Python**[11]: este lenguaje de programación es el "estándar" para Raspberry Pi. Se trata de un lenguaje interpretado, que puede ser ejecutado a través de distintos precompiladores, donde un "intérprete" va leyendo las instrucciones de ficheros denominados "Scripts" y ejecutándolas en tiempo real. Todo ello, unido a que es el lenguaje que mejor se adapta para interactuar con los terminales GPIO de la Raspberry Pi y que existe una gran comunidad entorno a él sobre la que apoyarse, hace que sea **el elegido para llevar a cabo el proyecto.**

Figura 9: Logo Python. Recuperada de: https://www.seeklogo.net/technology-logos/python-11832.html

2.2.4. Sensor de corriente

Un punto importante para diseñar el sistema es la selección del sensor de corriente a través del que captar la potencia del dispositivo a medir. En este caso, no ha sido necesario escoger ya que la universidad ha proporcionado diez sensores de corriente SCT-013-030. Aún así, existen otras opciones que pueden servir para captar la energía existente por un cable. Existen dos tipos de sensores:

1) **Sondas compactas**[12]**:** se trata de aparatos compactos, preparados para instalarse en una tarjeta electrónica (PCB). Tienen como principal desventaja que no se pueden abrir, por lo que habría que partir el cable para introducirlo en la sonda y poder monitorizar la energía (Fig.8).

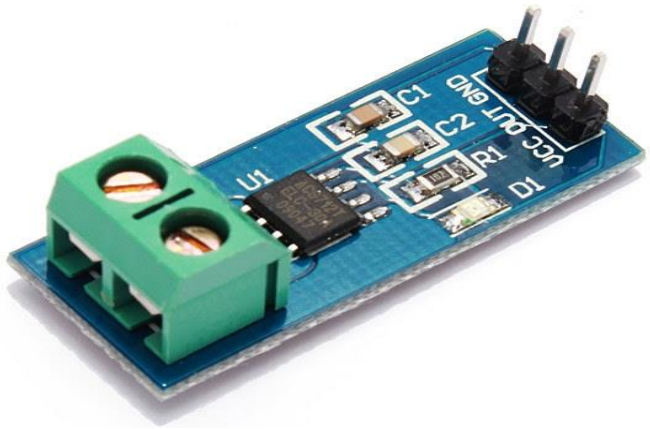


Figura 10: Sonda ACS712. Recuperada de: https://www.cnx-software.com/2016/01/23/acs712-module-measures-currents-30a-1-dollar/

2) **Sondas desmontables**[13]**:** se trata de sondas que miden la corriente alterna que circula por un cable sin necesidad de abrirlo. Basta con introducir el cable dentro de ella, cuya medición se obtendrá a través de un cable de conexión Jack como salida.

   Este sensor trabaja como un transformador, donde se genera una corriente en el devanado de salida en función de la que fluye por el cable.

Entre ellas existen dos modelos muy comunes:

- Sensor de corriente AC no invasivo SCT-013-000, que permite mediciones de hasta 100A.
- Sensor de corriente AC no invasivo SCT-013-030, que permite mediciones de hasta 30A.

La diferencia entre los modelos SCT tiene lugar por la cantidad de espiras que contienen, ya que estas representan la relación entre la corriente que circula por el cable y la que el sensor ofrece a su salida.

Figura 11: Sensor SCT-013-000.
Recuperada de:

https://hacktronics.co.in/current-measuring-sensor/100a-sct-013-000-non-invasive-ac-current-clamp-sensor

Figura 12: Sensor SCT-013-030.
Recuperada de:

https://ktechnics.com/shop/30a-sct-013-030-non-invasive-ac-current-sensor-split-core-current-transformer-al/

Como se ha comentado anteriormente, de entre los modelos presentados se usará el sensor de corriente **SCT-013-030**, ya que los 30A que permite medir son suficientes para recoger la energía contenida en cualquier aparato de una instalación domótica.

Este sensor se diferencia en que la corriente recogida pasa pasa por una resistencia interna, por lo que en su salida se obtendrá una tensión proporcional a la corriente obtenida anteriormente.

2.3. Productos similares

Antes de realizar el sistema se han analizado una serie de soluciones existentes actualmente en el mercado[14], de forma que sus especificaciones y funcionalidades han servido de ayuda para la realización de este proyecto.

1) Soluciones Efergy

La empresa Efergy ha desarrollado un producto llamado **Engage Hub Kit,** el cual recoge la señal emitida por los monitores de energía instalados por la vivienda, y envía la información a un portal web vía Internet. Dentro del mismo, los usuarios pueden acceder para consultar la actividad de su instalación sin tener que encontrarse cerca del dispositivo. Además, permite monitorizar simultáneamente hasta 5 circuitos eléctricos. Precio: 86,90€.

Figura 13: Engage Hub Kit. Recuperada de: http://www.blauden.com/kit-efergy-engage-hub-controlador-de-consumo-electrico-line

Además del anterior, la empresa ha creado otro producto llamado **ego smart Wi-Fi socket**, que consiste en un adaptador de enchufe que permite tanto extraer datos acerca del consumo del dispositivo conectado, como ofrecer al usuario opciones para manejar dispositivos de forma remota. Precio: 46,99€.

Figura 14: Ego smart Wi-Fi socket. Recuperada de: https://greenmagazine.com.au/product/efergy/ego-smart-wi-fi-socket/

Esto constituye una mejora de las funcionalidades, puesto que el sistema permite monitorizar cada aparato de forma independiente, consiguiendo así una información más detallada y precisa.

Figura 15: Interfaz web Efergy. Recuperada de: http://efergy.com/eu/pack-elitehub3phase

2) OpenEnergyMonitor

Es un proyecto de código y electrónica abierta, creado por una comunidad de desarrolladores a través de Internet con el objetivo de crear métodos de monitorización de energía gestionados a través de una aplicación web. Como es un sistema libre, en su página web se puede encontrar multitud de información para que cualquiera que desee un sistema de este tipo pueda construirlo.

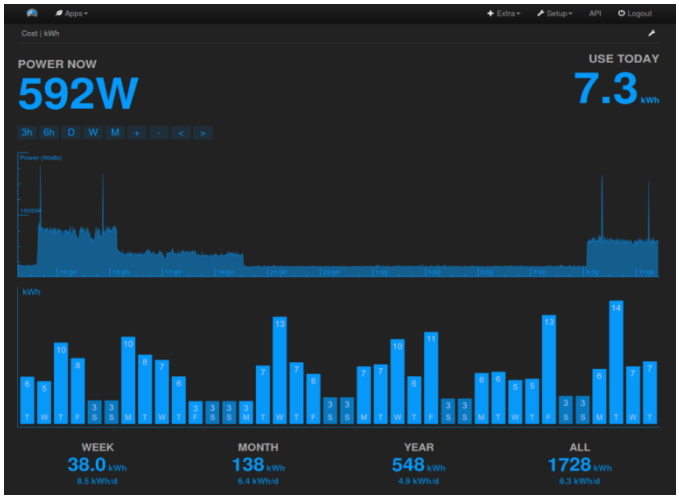


Figura 16: Interfaz web OpenEnergyMonitor. Recuperada de: http://www.nodocast.com/open-energy-monitor-arduino-emoncms-emontx-emonpi/

Para controlar el consumo de energía, el sistema se compone de cuatro partes principales:

- **emonTx/emonPi:** módulo principal de conexión de sensores y procesamiento de datos.
- **emonGLCD:** módulo de pantalla gráfica LCD inalámbrica.
- **emonBase:** módulo de recepción de datos de la monitorización de energía y envío de datos a un servidor remoto.
- **emoncms:** software para el procesamiento y visualización de los datos almacenados en el servidor.

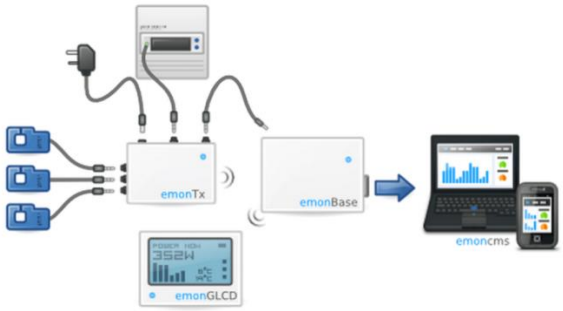


Figura 17: Componentes OpenEnergyMonitor. Recuperada de:
https://ricveal.com/blog/open-energy-monitor/

Existen dos módulos principales, como se acaba de comentar. **EmonTx** es un sistema electrónico formado por un microcontrolador que procesa las señales de hasta cuatro sensores de corriente para, más tarde, enviar los datos obtenidos hacia un módulo receptor, en este caso una Raspberry Pi. La información recibida es enviada más tarde hacia la plataforma web a través de la conexión a Internet con un router. Precio del módulo sin sensores: 51,00€. Precio con 4 sensores: 71,20€.

Figura 18: Módulo emonTx. Recuperada de:
https://guide.openenergymonitor.org/setup/

El segundo módulo es **emonPi**. Se trata de un sistema más compacto y autónomo que el anterior, compuesto por un procesador con dos sensores de corriente y una Raspberry Pi con conexión inalámbrica bajo una misma carcasa. De este modo, únicamente es necesario configurar el aparato para conectarlo con una red WiFi, sin necesidad de utilizar aparatos intermedios.

Figura 19: Módulo emonPi. Recuperada de:
https://guide.openenergymonitor.org/setup/

3) Wattio. Pack Wattio Energy

Otro de los productos destacados en el mercado es el **Pack Wattio Energy**[15]**.** Este sistema está compuesto por una serie de dispositivos de recogida de datos, gestionados a través de una consola domótica con pantalla táctil o bien desde cualquier dispositivo que disponga de conexión a internet.

Figura 20: Pack Wattio Energy. Recuperada de:
https://es.pinterest.com/pin/358951032778993625/?lp=true

Se trata de un sistema muy completo, que se compone de cuatro partes:

- **GATE:** es la central de control, desde la cual se pueden manejar los dispositivos conectados en la instalación del usuario. Cuenta con una pantalla táctil que permite manejarse por los menús del sistema y configurar el consumo eléctrico, alarmas, gestión térmica, salud, etc. Precio: 120,96€.

Figura 21: Wattio GATE. Recuperada de: https://nergiza.com/sistema-wattio-lo-revisamos/

- **BAT:** pequeño aparato donde se enchufan las pinzas amperimétricas, de forma que es posible monitorizar un máximo de tres circuitos a la vez en un cuadro eléctrico. Para instalar las pinzas se tendrá que abrir el cuadro y localizar las líneas que interesan. Precio: 54,99€.

Figura 22: Wattio BAT. Recuperada de: https://nergiza.com/sistema-wattio-lo-revisamos/

- **POD:** su función consiste en medir la energía consumida por el dispositivo enchufado y, a su vez, ofrecer la posibilidad de conectar/desconectar ese dispositivo. Resulta útil para activar aparatos a distancia desde el móvil. Precio: 49,95€.

Figura 23: Wattio POD. Recuperada de: https://nergiza.com/sistema-wattio-lo-revisamos/

- **THERMIC:** es el termostato del sistema Wattio. Posee un contacto conmutado que permite al usuario accionar su caldera o sistema de refrigeración a distancia. Precio: 188,87€.

Figura 24: Wattio THERMIC. Recuperada de: https://nergiza.com/sistema-wattio-lo-revisamos/

Para poder obtener un impacto interesante en el ahorro energético, el usuario deberá analizar los datos para detectar patrones de uso durante los primeros meses, de modo que aprovechando la información extraída de ellos se modifiquen los usos de ciertos aparatos.

En definitiva, el producto es una solución eficaz y económica. Dispone de aplicación para Android y iPhone e interfaz web, por lo que Wattio es una opción más que interesante para gestionar el sistema energético de cualquier instalación.

4) FIBARO

Se trata de un sistema domótico que permite la automatización de la vivienda de forma personalizada por los usuarios. La gran ventaja que presenta es que no necesita modificar la instalación eléctrica, ya que la información se transmite de forma inalámbrica.

Este sistema contiene un nodo central, el **FIBARO Home Center**, que es quien interactúa con los sensores y actuadores instalados por la vivienda utilizando la comunicación por radio, y quien gestiona los datos recogidos para mostrárselos al usuario a través de la interfaz web, ofreciendo de esta forma al usuario un control total sobre los dispositivos. Precio: 216,82€.

Dentro del mismo, los aparatos eléctricos son gestionados a través del **FIBARO Wall Plug,** adaptador de corriente que contiene un anillo luminoso que cambia de color según la potencia medida en cada momento. Precio: 50,99€.

Figura 25: FIBARO Wall Plug. Recuperada de: http://www.vesternet.com/z-wave-fibaro-wall-plug-schuko-gen5

Estos enchufes envían de forma inalámbrica los datos registrados por cada aparato al ordenador central, el cual gestiona toda la información para que el usuario pueda conocer su consumo. Por ejemplo, se puede consultar la actividad por tipo de electrodoméstico, lo que ayudaría al usuario a establecer nuevos hábitos de consumo para aumentar el ahorro energético.

Figura 26: Interfaz web FIBARO. Recuperada de: http://www.148apps.com/app/670919021/

Tras observar los productos similares que existen actualmente en el mercado, se ha realizado una comparación con el sistema creado en el presente proyecto, que se ve reflejada en la siguiente tabla:

Producto	Nº de sensores	Precio
Engage Hub Kit	5	86,90€
Ego smart Wi-Fi socket	1	45,99€
OpenEnergyMonitor: emonTx	4	71,20€
OpenEnergyMonitor: emonPi	2	175,13€
Wattio Energy GATE + BAT	3	175,95€
FIBARO Home Center + FIBARO Wall Plug	1	267,81€
Proyecto diseñado	10	149,49€

Tabla 8: Comparativa entre el proyecto y productos similares.

Como se observa en la tabla, el proyecto ofrece la posibilidad de instalar un sistema de monitorización en un cuadro eléctrico con un mayor número de sensores de corriente que las demás opciones. Además, el precio final del sistema es mucho más bajo si comparamos con respecto a otras opciones como FIBARO o Wattio Energy, aunque no disponga de interfaz web ni de otras funcionalidades más allá de la recogida de datos.

Algunas de las opciones planteadas, como emonTx o Engage Hub Kit, podrían presentarse como alternativas económicas ya que poseen una buena funcionalidad a bajo precio, aunque si realmente se quisiera monitorizar la instalación eléctrica de una vivienda, 4 y 5 sensores son escasos para poder hacerlo.

Por tanto, se concluye que el presente proyecto es una gran alternativa a los productos existentes, ya que aunque es un sistema menos compacto y con un acabado menos vistoso, permite extraer datos de diez líneas eléctricas a un precio muy reducido, y por tanto sería una buena opción para monitorizar una vivienda.

3. Raspberry Pi

3.1. Introducción

Raspberry Pi es un ordenador de placa reducida de bajo coste desarrollado en Reino Unido por la fundación Raspberry Pi con el objetivo principal de estimular la enseñanza de las ciencias de la computación a niños en sus colegios y a personas de países subdesarollados.

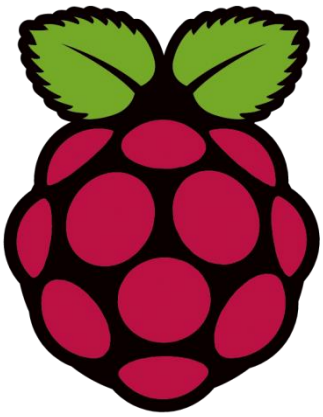


Figura 27: Logo Raspberry Pi. Recuperada de:
https://www.raspberrypi.org/blog/logo-competition-we-have-a-winner/

Se podría considerar como un ordenador de muy pequeño tamaño, comparable con el de una tarjeta de crédito (placa de 85 x 54 mm).

Todo modelo contiene al menos: un puerto de salida de video HDMI, otro de tipo RCA, minijack para conectar audio y un puerto USB 2.0 (la versión 3 Model B dispone de 4 USB) donde conectar un monitor y un ratón con los que manejarse por el sistema como si fuera un ordenador convencional, además de un puerto Ethernet.

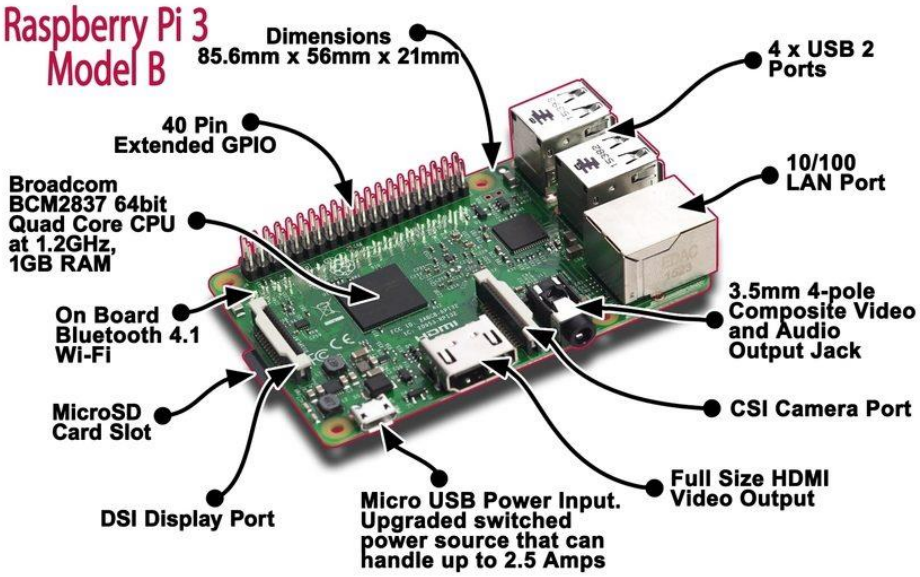


Figura 28: Raspberry Pi 3 Model B. Recuperada de: https://comohacer.eu/wp-content/uploads/2016/05/partes-raspberry-pi-830x524.jpg

A continuación, se detallan los componentes existentes[16] en esta versión:

1) **Puertos USB:** la plataforma posee cuatro puertos USB del estándar 2.0 a los que se suelen conectar dispositivos externos para manejar la Raspberry, como un teclado o monitor.

2) **Pines GPIO:** estos pines sirven de interfaz entre la Raspberry Pi y el exterior. De los 40 pines que contiene la Raspberry Pi, entre los que se encuentran pines del tipo UART, I2C o SPI para manejar dispositivos como luces LED, sensores, etc.

3) **Conector DSI:** permite la conexión de pequeñas pantallas LCD al dispositivo.

4) **Conector CSI:** conector tipo bus de 15 pines que permite añadir un dispositivo compatible con la interfaz CSI-2, y mediante el cual se conecta la cámara.

5) **Almacenamiento:** el aparato contiene un lector de tarjetas de memoria microSD. El arranque del sistema se hará desde la propia tarjeta, por lo que es conveniente que su tamaño sea de al menos 16GB de capacidad.

6) **Puerto Ethernet:** dispone de un puerto Ethernet para enchufar un cable RJ-45 directamente al router, aunque en la última versión no es necesario ya que tiene integrada conexión WiFi 802.11n y Bluetooth 4.1.

7) **Alimentación:** integra un conector micro USB, con el que la mayoría de cargadores para smartphone son compatibles (mayor de 700mA).

8) **Salidas de audio:** posee un conector de audio Jack de 3,5mm, además de la posibilidad de usar el propio HDMI.

9) **Conector HDMI:** permite la conexión de dispositivos como monitores para la extracción de vídeo y ser usado como salida de audio.

10) **Broadcom BCM2837:** el diseño incluye un procesador central ARMv8 de 64 bit a 1.2GHz, una GPU VideoCore IV y 1 GB de memoria RAM.

3.2. Terminales GPIO

La principal diferencia entre la Raspberry Pi y los demás SBC es la incorporación de los pines GPIO[17], que actúan como una interfaz física entre ella y el mundo exterior, y que pueden configurarse como entradas o salidas para controlar sensores, dispositivos externos, etc. Entre los terminales GPIO se encuentran otros dos tipos de pines: los SPI[18] y los I2C[19].

El bus SPI es un estándar de comunicaciones usado para la transferencia de información entre circuitos integrados en equipos electrónicos. Tiene una arquitectura de tipo maestro-esclavo, donde el dispositivo maestro inicia la comunicación con el esclavo, para así enviar o recibir datos de él.

Incluye una línea de reloj, dato entrante, dato saliente y un pin CS, que conecta o desconecta la operación del dispositivo con el que uno desea comunicarse.

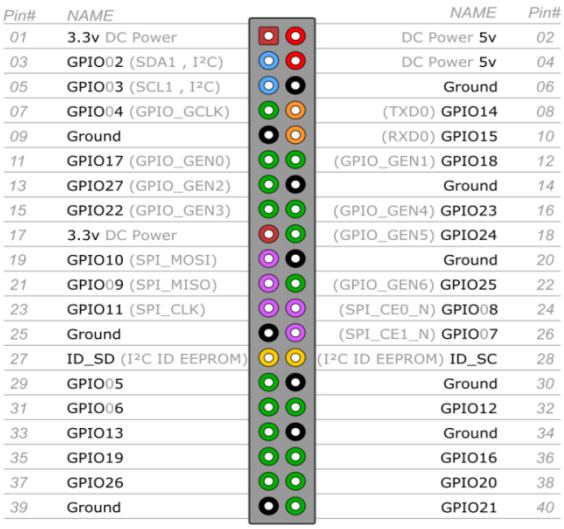
Por tanto, la sincronización y transmisión de datos requieren 4 líneas:

- **SCLK:** señal de reloj enviada por el maestro, es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
- **MOSI:** salida de datos del maestro y entrada de datos al esclavo.
- **MISO:** salida de datos del esclavo y entrada al maestro.
- **SS:** sirve para seleccionar el dispositivo con el que se va a realizar la comunicación.

Por otra parte, los I2C son buses de comunicación en serie diseñados para conectar diferentes partes de un circuito, por ejemplo entre un controlador y circuitos periféricos integrados.

Su principal característica es que utiliza dos líneas para transmitir la información, una para los datos (SDA) y otra para la señal de reloj (SCL), además de una tercera línea de referencia (GND).

A continuación se muestra la configuración de los pines de la Raspberry Pi utilizada en este proyecto (Raspberry Pi 3 Model B):



Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figura 29: Pines Raspberry Pi 3 Model B. Recuperada de: https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout

3.3. Configuración inicial

La Raspberry Pi es un pequeño ordenador que necesita un sistema operativo para arrancar todos los procesos necesarios para que las capacidades de hardware y de computación estén disponibles para las aplicaciones que se deseen ejecutar. Como no posee unidad de almacenamiento interna, se debe grabar ese sistema operativo en una tarjeta micro-SD[20]. Como se ha comentado anteriormente, el sistema operativo escogido para este proyecto es Raspbian, el cual se puede descargar de la página *raspberry.org*.

Una vez en ella, se debe descargar el documento ZIP correspondiente y descomprimir la imagen, que es un conjunto de sistema operativo y aplicaciones que han sido preparados en un único archivo con toda la información necesaria para que una plataforma (Raspberry Pi en este caso) arranque cuando sea encendida y cargue los programas necesarios automáticamente.

Para poder grabar esta imagen en la tarjeta, es necesario otro software llamado Win32 Disk Imager, que hay que descargar e instalar en el equipo. Una vez listo, basta con seleccionar la imagen que se desea grabar y el medio de almacenamiento donde guardarla.

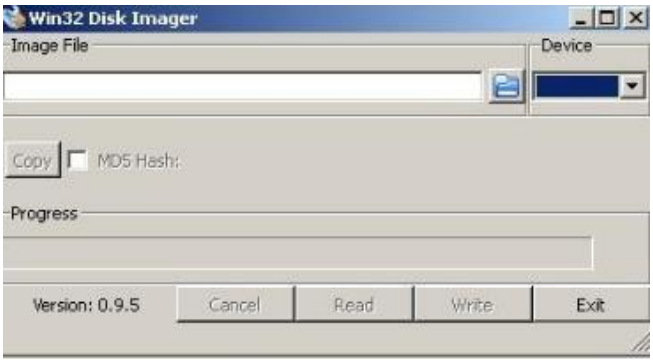


Figura 30: Win32 Disk Imager.

Tras grabar el sistema operativo en la tarjeta se debe insertar en la Raspberry Pi, conectar un monitor y teclado para poder manejarla, y enchufarle un cable Ethernet para conectarla a Internet. En su primer encendido aparece un menú de configuración[21], donde se puede acceder mediante el comando *raspi-config*, y en el que se pueden realizar cambios como la contraseña, control de periféricos, etc.

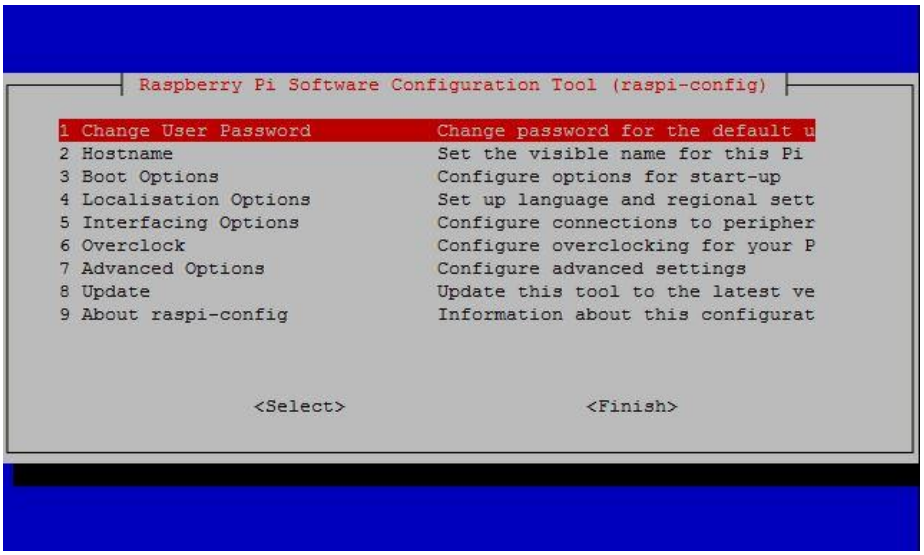


Figura 31: Menú de configuración Raspberry Pi.

Lo primero que se debe hacer dentro de éste menú es pulsar la opción 8, Update, para que la Raspberry Pi descargue la última versión software existente y se configure con ella de forma automática. Además, para poder conectarse a ella de manera remota, se seleccionará la opción 5, Interfacing Options.

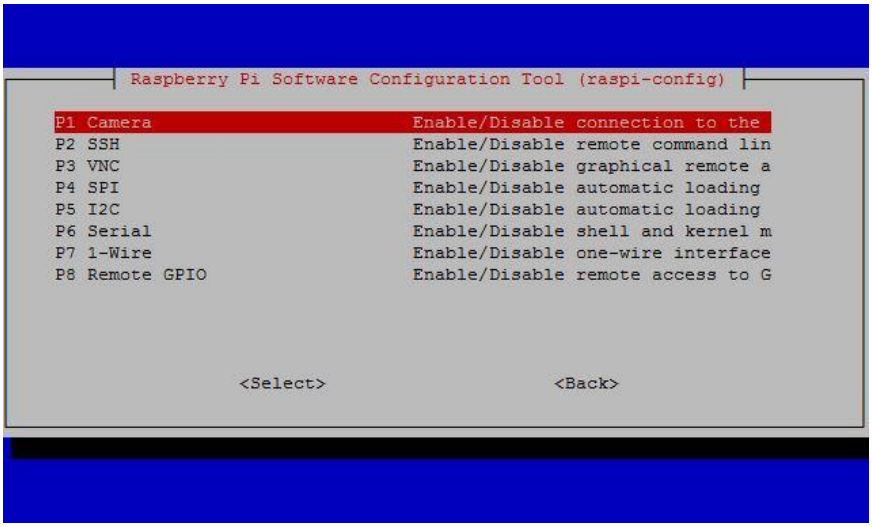


Figura 32: Interfaces Raspberry Pi.

En este panel, se habilitarán las opciones SSH (para conectarse de forma remota desde el ordenador), VNC (para conseguir una interfaz gráfica sobre la que navegar) y SPI / Remote GPIO (para permitir la conexión desde los puertos GPIO).

Tras terminar la configuración, se reiniciará la Raspberry Pi, y al volver a iniciarse pedirá un nombre de usuario y contraseña, que por defecto son:

- Usuario**:** pi
- Contraseña**:** raspberry

Una vez realizado, la Raspberry Pi está lista para utilizarse. Al iniciarse aparecerá la línea de terminal, donde se ejecutan los comandos usuales en Linux.

Algunos de los más utilizados[22] son:

- **apt-get update**: actualiza de la lista de paquetes.
- **apt-get upgrade**: instala los paquetes de la lista anterior.
- **cd:** cambiar de directorio.
- **ls**: lista de archivos y carpetas del directorio.
- **mkdir**: crear un directorio.
- **nano**: editor de textos para crear/modificar un archivo.
- **poweroff**: apagar el sistema.
- **raspi-config:** acceder al menú de configuración.
- **reboot**: reiniciar el sistema.
- **rm**: eliminar archivos.
- **sudo**: ejecución de comandos como superusuario.

3.4. Conexión a Internet y acceso remoto al PC

El acceso a Internet de la Raspberry Pi puede llevarse a cabo de dos formas: a través de un cable Ethernet para conectarla al router, o utilizar la conexión Wi-Fi que integra el aparato. Con el objetivo de evitar fallos o cortes de conexión, se ha escogido la opción del cable de red, aunque aún así se mostrará el procedimiento a seguir si se desea utilizar la conexión Wi-Fi[23].

Para realizar la conexión a una red Wi-Fi desde terminal de comandos es necesario seguir los siguientes pasos:

1. Escanear las redes Wi-Fi disponibles con el comando *sudo iwlist wlan0 scan*

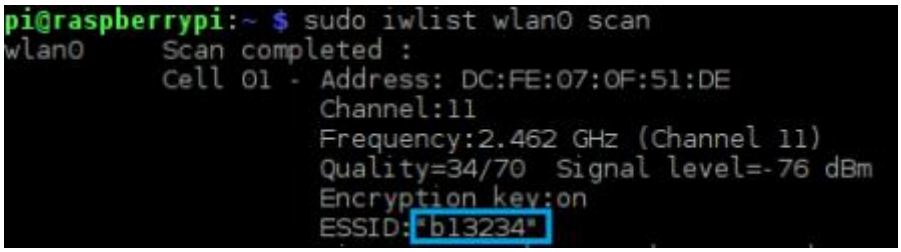


Figura 33: Configuración WiFi en Raspberry Pi. Recuperada de: http://bot-boss.com/configuracion-wifi-bluetooth-raspberry-pi-3/

2. Encontrar el nombre de la red a conectarse, que se ubica después de ESSID.

3. Agregar el nombre de red y contraseña en el archivo wpa_suplicant, con el comando *sudo nano /etc/wpa_supplicant/wpa_supplicant.conf*

4. Introducir las siguientes líneas al final del archivo:

   network={
   ssid="Nombre de ESSID"
   psk="Contraseña"
   }

5. Guardar cambios pulsando *CTRL+X*, después *Y*, *enter* para confirmar.
6. La Raspberry Pi se conectará automáticamente a la red Wi-Fi. En caso contrario, es posible que se necesite un reinicio introduciendo *sudo reboot.*

Aunque típicamente se utiliza un teclado, ratón y monitor externos para manejar el aparato, en este proyecto se ha trabajado de una forma diferente por mayor comodidad: a través de un ordenador, con el que se accede de forma remota al dispositivo gracias a SSH[24].

Puesto que ya se ha habilitado anteriormente la interfaz SSH, el siguiente paso para acceder de forma remota al PC es instalar en el mismo un cliente SSH (sólo para sistemas Windows), como por ejemplo PuTTY[25].

Una vez instalado, es el momento de hacer la conexión con la Raspberry Pi. Para ello, se ejecuta el programa, y se abrirá una ventana como la siguiente:

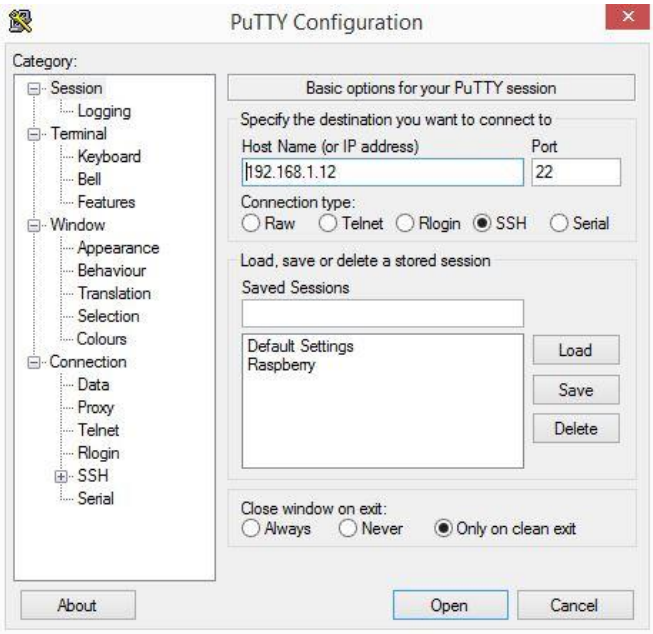


Figura 34: Pantalla principal de PuTTY.

Para iniciar la conexión, basta con introducir el puerto por el que se realizará (22) y la IP del dispositivo.

Es recomendable además dar un nombre a la conexión y hacer clic en “Save”, de forma que no habrá que volver a completar los datos cada vez que el usuario quiera conectarse, ya que bastará con hacer click en el nombre de la conexión y luego en “Load” para conectarse a la Raspberry Pi. Si el programa PuTTY establece la conexión, se abrirá una nueva ventana donde habrá que introducir los datos de usuario habituales, por lo que ya se podrá usar la Raspberry Pi de forma remota.

Por último, es posible trabajar con una interfaz gráfica en vez de únicamente a través del terminal. Para poder acceder a ella, es necesario realizar una conexión VNC[26]. Como su acceso ya está habilitado, se debe introducir el comando *sudo apt-get install tightvncserver* para instalar el paquete VNCServer en la Raspberry. El siguiente paso será poner en marcha el servidor VNC con el comando *tightserver*, donde habrá que introducir una contraseña para poder iniciar la conexión VNC (el usuario elegirá la que desee).

Para concluir, queda descargarse el programa VNC Viewer, aplicación que permitirá visualizar el escritorio de la Raspberry Pi de forma gráfica en vez de únicamente trabajar en línea de comandos. Una vez instalado, habrá que introducir el comando *tightvncserver –geometry 1280x640 –depth 16,* donde "1280x640" es la resolución correspondiente a la pantalla del ordenador del usuario. De esta forma, se iniciará una nueva sesión.

```
New 'X' desktop is raspberrypi:1

Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log
```

Figura 35: Acceso por VNC.

Tras ello, bastará con introducir en VNC Viewer la IP de la Raspberry Pi seguido del número de sesión (p. ej: 192.168.1.1:1), así como la contraseña *raspberry* para que, finalmente, aparezca la interfaz gráfica sobre la que se desarrolla la parte software del sistema de monitorización.

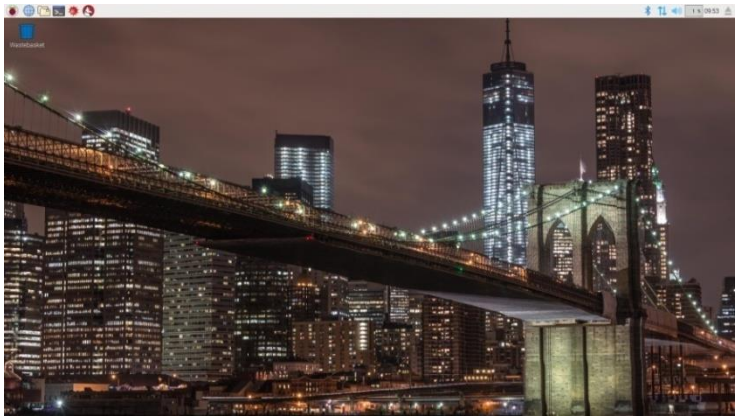


Figura 36: Escritorio VNC Viewer.

4. Programación básica del sistema

4.1. Instalación de Python

Como se ha comentado en los apartados anteriores el trabajo se desarrollará en el sistema operativo Python, así que lo primero que se debe hacer es instalar el paquete correspondiente[27].

En primer lugar, se deben introducir los siguientes comandos en la consola:

- sudo apt-get update
- sudo apt-get upgrade
- sudo reboot

Con ello, la Raspberry Pi descargará y actualizará las últimas versiones de los paquetes del repositorio del usuario, además de reiniciar el dispositivo para su correcto funcionamiento.

Tras ello, se instalará el paquete de Python con el módulo SPI, para permitir de esta forma utilizar los pines GPIO:

- sudo apt-get install python-dev
- mkdir python-spi
- cd python-spi
- wget https://github.com/JoBergs/RaspiContent/raw/master/spidev/setup.py
- wget https://github.com/JoBergs/RaspiContent/raw/master/spidev/spidev_module.c
- sudo python setup.py install

En este instante, la Raspberry Pi ya está preparada para ejecutar ficheros con la extensión de Python (.py), sobre los que se desarollará el sistema.

4.2. Configuración de demonio de arranque automático

Tras la instalación de Python toca configurar la Raspberry Pi para que, al encenderse, arranque automáticamente el programa creado en segundo plano.

El primer paso es crear un fichero que se encargue de arrancar el software de forma automática. Para ello, es necesario introducir el comando *sudo nano /etc/init.d/nombre_fichero,* donde *nombre_fichero* se ha denominado ***contador***. Dentro del mismo se copiará el siguiente código[28]:

```
#! /bin/sh
# /etc/init.d/contador

case "$1" in
 start)
  echo "Arrancando programa contador"
  #En este apartado hay que escribir el programa que script que se desea arrancar
  /usr/bin/python  /home/pi/python-spi/contador.py
  ;;
 stop)
  echo "Programa detenido"
  ;;
 *)
  echo "Modo de uso: /etc/init.d/contador {start|stop}"
  exit 1
  ;;
esac
exit 0
```

Tras ello, toca hacer el fichero ejecutable introduciendo el comando *sudo chmod 755 /etc/init.d/contador.* Por último, para activar el arranque automático es necesario introducir el comando *sudo update-rc.d contador defaults.* Una vez reiniciada la Raspberry Pi, el programa *contador.py* comenzará a ejecutarse de forma automática en segundo plano.

Como sugerencia, es una buena práctica comprobar el funcionamiento del mismo a través del comando *ps aux | grep "contador.py",* que es el programa que pone en marcha el script. Si por el contrario se desea desinstalar el demonio de arranque automático, el comando a introducir es *sudo update-rc.d –f /etc/init.d/contador remove*

5. Desarrollo de la solución: parte hardware

5.1. Conversor analógico-digital

Una vez mostrado el funcionamiento de la Raspberry Pi y su inicialización, se debe diseñar un circuito de acondicionamiento de la señal medida por los sensores para que los datos extraídos por los mismos puedan ser llevados hasta la Raspberry Pi. Como la entrada por los pines GPIO de la Raspberry es digital y el circuito de acondicionamiento proporcionará una señal analógica, es necesario transformar la información a digital a través de la utilización de un conversor A/D.

Las características principales que han de tenerse en cuenta a la hora de escoger el conversor: su resolución, tiempo de muestreo y número de canales de entrada, cuya forma de calcular se explicará a continuación y está basada en los comentarios del trabajo: *Medidor de consumo eléctrico de un hogar: procesado de datos mediante Raspberry-pi* [16].

Como la relación de calibración del sensor es de 30A/V y la Raspberry Pi estará alimentada con una tensión de 3.3V, el conversor debe tener una resolución mínima de 10 bits para obtener un margen de error pequeño entre cada muestra:

$$2^{10} \text{ bits} = 1024 \text{ valores posibles}$$

$$\text{Resolución} = \frac{3.3V}{1024\ valores} x\ 30\ \frac{A}{V} =\ 0.0967\ \frac{A}{valor\ digital}$$

Esta resolución indica que cada variación del valor digital obtenido varía 0.0967A en corriente, por lo que una resolución alta como en este caso permite tener una valor de A/valor_digital más pequeño, y por tanto una mejor precisión al medir.

Se ha programado el sistema de tal forma que se recogerán muestras cada 2 ms al inicio de cada interrupción, por lo que el tiempo de muestreo del conversor elegido debe ser menor. Entre los conversores A/D más utilizados para Raspberry Pi se encuentra el MCP3008. Dispone de 8 canales de entrada, resolución de 10 bits, y puede ser alimentado por una tensión de entre 2.7V y 5.5V. Según sea mayor o menor, el número de muestras obtenidas/s será diferente, extrayendo 200000 muestras/s para una tensión de 5.5V, y 75000 muestras/s para 2.7 V.

Figura 37: Conversor MCP3008. Recuperada de:
http://www.microchip.com/wwwproducts/en/MCP3008

De esta forma, en el peor de los casos el tiempo de muestreo es:

$$TM_{mayor} = \frac{1}{75000\ muestras/s} = 1.33\ x\ 10^{-5}\ s$$

Como el tiempo de muestreo utilizado en los sensores es menor que el calculado, se puede alimentar el conversor a la tensión deseada, dentro del rango permitido según su hoja de características. Como la Raspberry Pi incorpora un pin de 3.3V, se ha decidido que sea la tensión a la que alimentar el circuito.

Finalmente, tras todos los cálculos el conversor elegido ha sido el MCP3008[29]. Además de cumplir las anteriores especificaciones, contiene ocho canales de entrada, por lo que es más eficiente su uso que el de la otra opción disponible, el MCP3004, que únicamente incorpora cuatro canales de entrada.

Puesto que el objetivo final es monitorizar una instalación a través de diez sensores, es necesario incluir en el sistema dos conversores MCP3008, entre los que se repartirán cinco sensores para cada uno de ellos (se usarán los canales del 0 al 4 como entradas en ambos).

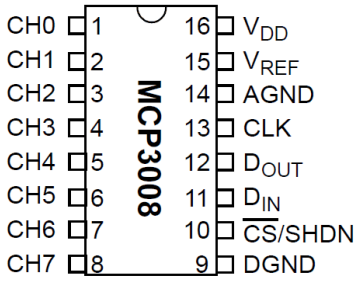


Figura 38: Terminales del conversor MCP3008. Recuperada de:
https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008

5.2. Diseño del circuito

Como se ha comentado anteriormente, para poder recibir correctamente el valor de la corriente obtenido por los sensores en la Raspberry Pi es necesario realizar un circuito de acondicionamiento de la señal medida.

Para desarrollarlo, hay que adaptar la señal de salida de los sensores de corriente AC, que permiten captar un máximo de 30A. Estos sensores contienen una resistencia interna, por lo que a la salida lo que realmente se consigue es una señal de tensión proporcional a la corriente alterna que fluye por el cable a medir, y cuyo rango se encontrará entre 0-1V.

Como la Raspberry Pi únicamente admite valores entre 0-3.3V, y la señal medida tomará valores negativos debido a su componente alterna (rango -1 a 1), se ha decidido rectificar la parte negativa de la señal del sensor y trabajar con la parte positiva, puesto que la parte negativa podría malograr la Raspberry Pi[13]. De este modo se puede aprovechar el rango completo de la lectura. Para ello, tras conectar cada sensor de corriente se ha realizado un seguidor de tensión con un amplificador operacional TL084, alimentado entre 0-3.3V, de forma que la señal de salida sea equivalente a la obtenida en el sensor, pero con las partes negativas eliminadas (valor 0).

Puesto que cada TL084 contiene 4 amplificadores en su interior, sólo ha sido necesario instalar 3 en el circuito completo. Por tanto, el circuito equivalente para adaptar cada sensor de corriente es el siguiente:

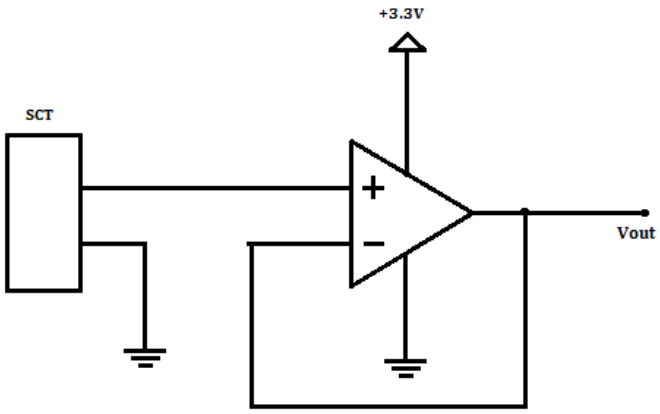


Figura 39: Rectificado de parte negativa de la señal del sensor.

Para conectarlos a la protoboard, los sensores se enchufarán a un conector Jack hembra cada uno, de forma que el cable de tierra del Jack estará conectado a GND, mientras que la parte activa del cable se conectará hacia la entrada positiva del amplificador. Una vez realizado este proceso se ha cableado cada salida del sensor con su entrada correspondiente al conversor, los cinco primeros sensores con los canales 0 al 4 del primer MCP, y los cinco siguientes con los canales 0 al 4 del segundo.

Por tanto, el circuito completo queda configurado de la siguiente forma:

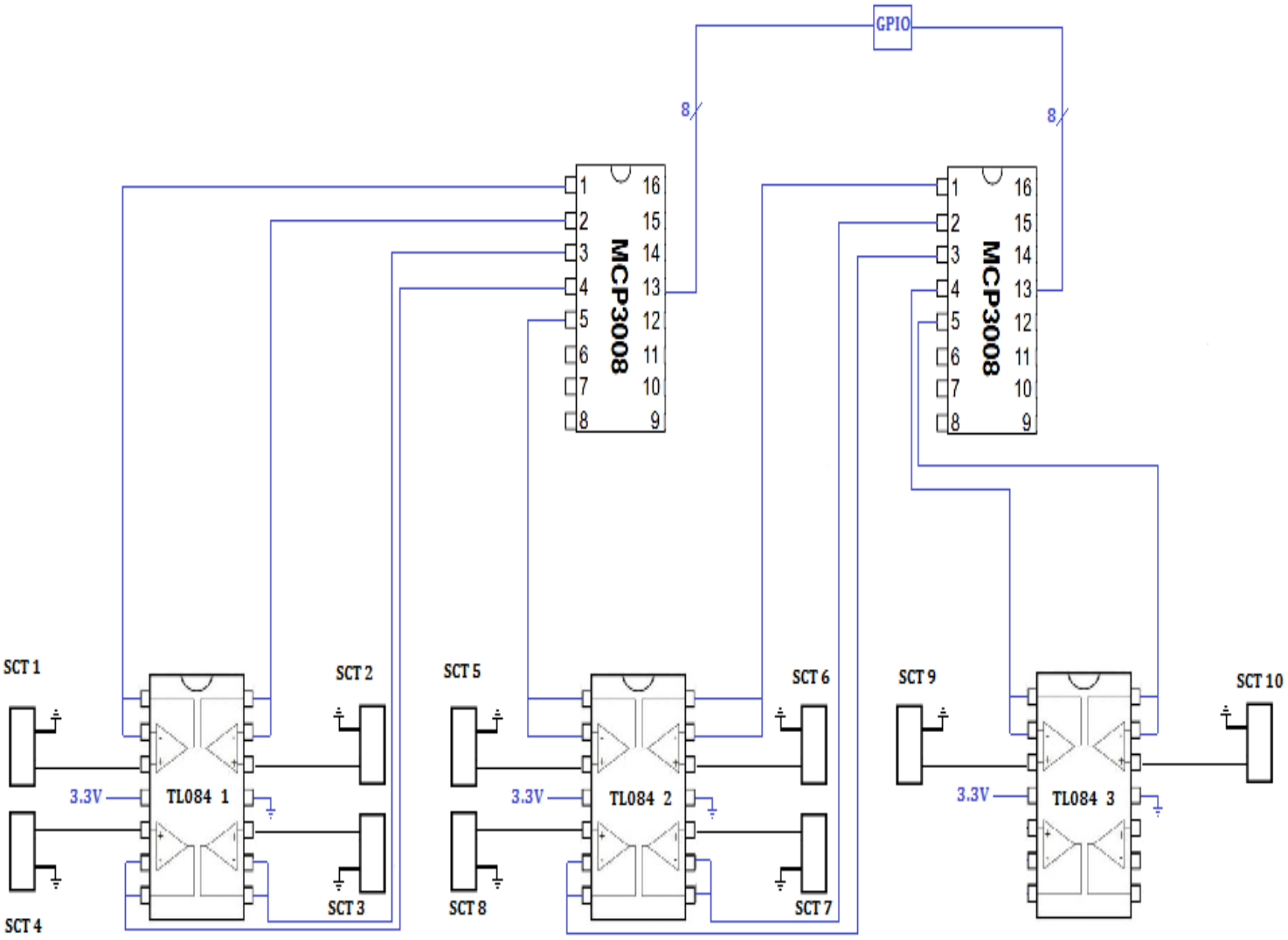


Figura 40: Diseño del circuito completo.

5.3. Conexión del circuito a la Raspberry Pi

En este apartado se mostrará la conexión realizada entre los terminales de los conversores analógico/digital y los pines de la Raspberry Pi.

La siguiente tabla muestra los pines conectados entre las dos partes:

Terminales MCP3008	Terminales GPIO Raspberry Pi
V_{DD} (Pin 16)	3.3V (Pin 1)
V_{REF} (Pin 15)	3.3V (Pin 1)
AGND (Pin 14)	GND (Pin 9)
CLK (Pin 13)	SPIO_SCLK (Pin 23)
D_{OUT} (Pin 12)	SPIO_MISO (Pin 21)
D_{IN} (Pin 11)	SPIO_MOSI (Pin 19)
CS0 (Pin 10) / CS1 (Pin 10)	SPIO_CS0 (Pin 24) / SPIO_CS1 (Pin 26)
DGND (Pin 9)	GND (Pin 39)

Tabla 9: Conexión de terminales entre conversor y Raspberry Pi.

Nota: Como son necesarios dos conversores MCP3008, el primero de ellos tendrá el pin 10 conectado al pin 24 de la Raspberry Pi (Chip Select 0, CS0), mientras que el segundo de ellos tendrá el pin 10 conectado al pin 26 de la Raspberry Pi (Chip Select 1, CS1).

Para una mejor interpretación, se muestra una imagen con las conexiones realizadas:

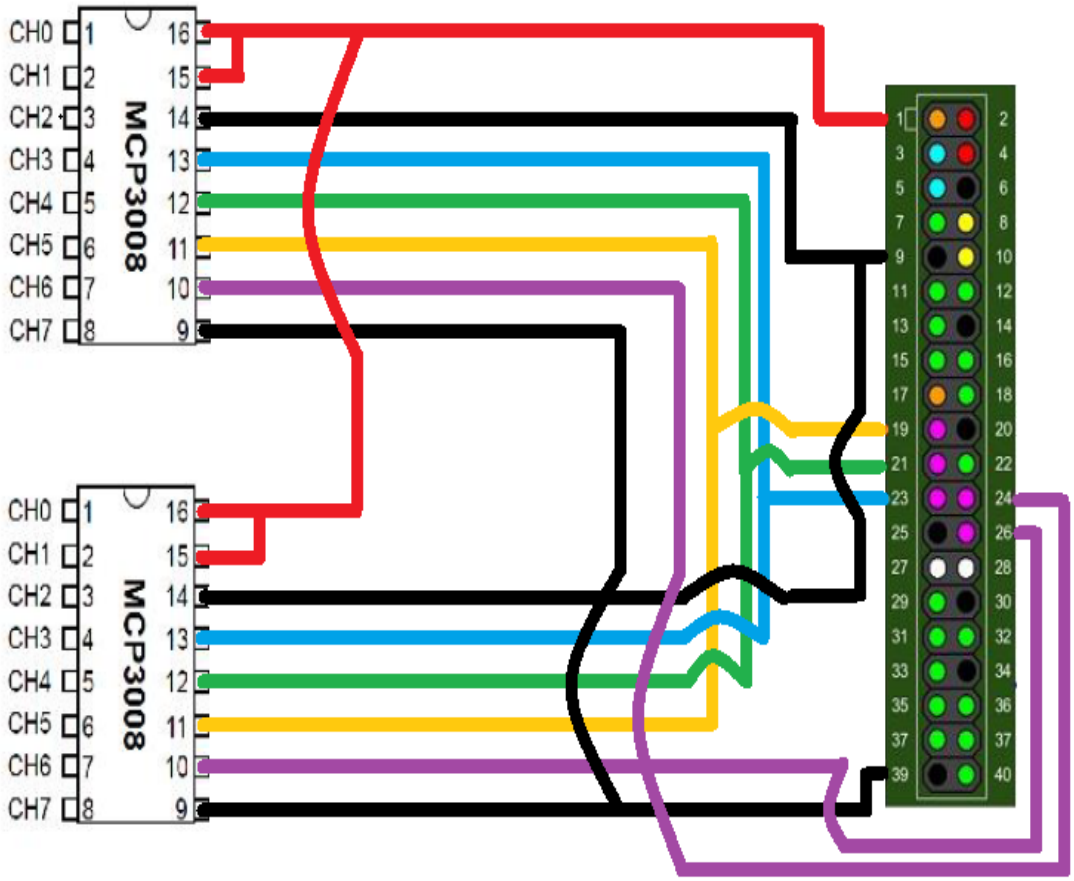


Figura 41: Conexión entre conversor y Raspberry Pi.

5.4. Montaje completo

Una vez diseñado el circuito completo y la conexión entre el mismo y la Raspberry Pi, toca realizarlo de forma física. Para ello, se han utilizado los materiales descritos en el apartado 1.3, quedando implementado de la siguiente forma:

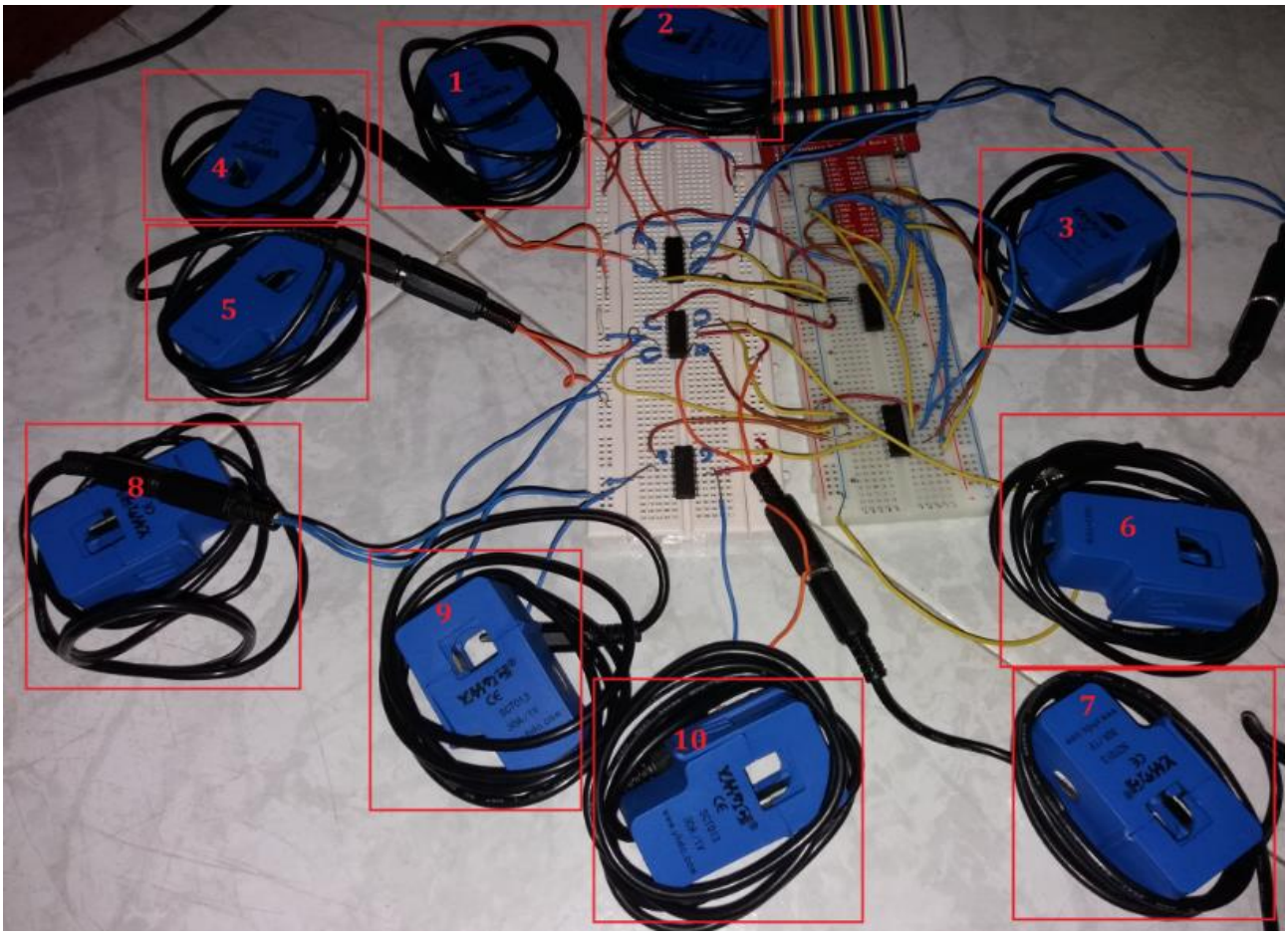


Figura 42: Montaje completo del circuito.

Dentro del mismo, se detallan los componentes incluidos en las protoboard en la siguiente imagen:

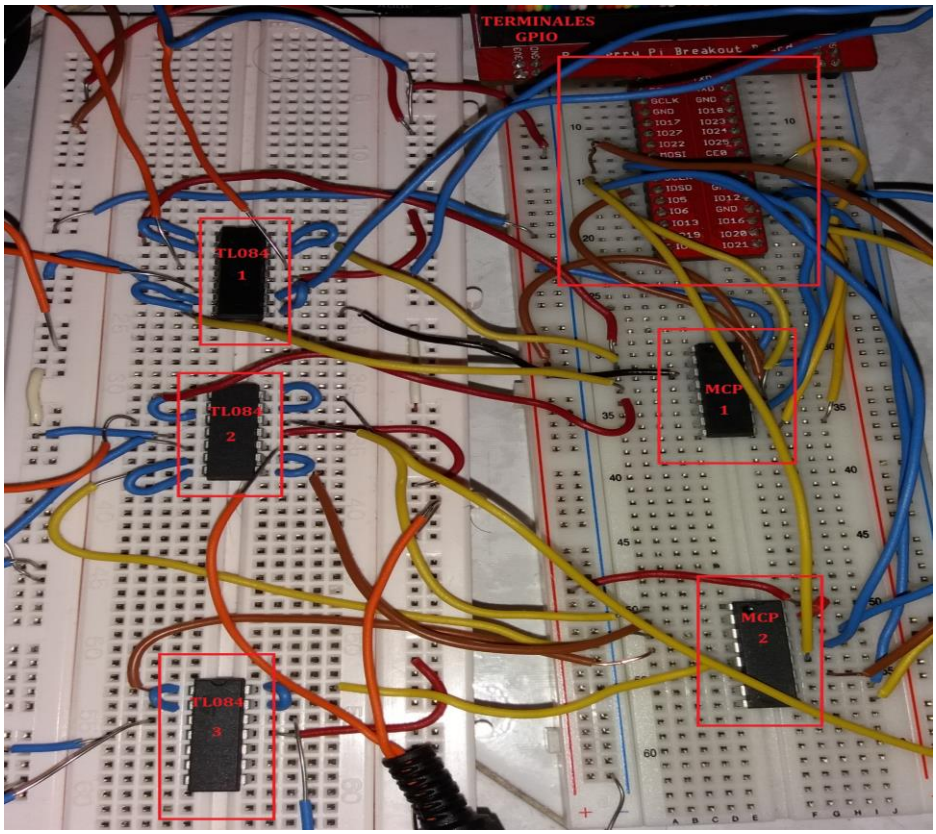


Figura 43: Montaje placas protoboard.

6. Desarrollo de la solución: parte software

6.1. Programa principal

El programa principal del sistema ejecutado por el demonio de arranque automático consta de un script denominado contador.py, dentro del cual se han implementado las funciones necesarias para la extracción de datos a través de los sensores. Para configurarlo, es necesario importar al comienzo del script las librerías necesarias a lo largo del fichero.

```
import spidev
import time
import os
import sys
import threading
import math
```

A continuación se debe realizar la apertura del bus SPI para poder controlar los dispositivos externos. Puesto que es necesario incluir en el circuito dos conversores A/D, deben crearse dos objetos SpiDev, para de esta forma habilitar en uno de ellos el CS0, y en el otro el CS1, que son los pines a los que se conectarán los conversores para transmitir los datos hacia la Raspberry Pi.

```
#Habilitar ChipSelect 0 y 1
spi = spidev.SpiDev()
spi.open(0,0)
spi2 = spidev.SpiDev()
spi2.open(0,1)
```

Acto seguido, deben definirse las funciones que permiten extraer los datos de los sensores. La primera de ellas es ***LecturaCanal***, que dado un canal de entrada permite extraer la información analógica contenida en el canal introducido, y convertirla a digital.

Como para cada conversor se usarán los cinco primeros canales, esto se tiene en cuenta a la hora de crear la función restando 5 cuando corresponde al segundo conversor, ya que el canal introducido por parámetro siempre será un número entre el 0 y 9, pero los canales para ambos conversores estarán comprendidos entre el 0 y 4.

```
def LecturaCanal(canal):
    if canal < 5:
        adc = spi.xfer2([1,(8+canal)<<4,0])
    else:
        ch = canal - 5
        adc = spi2.xfer2([1,(8+ch)<<4,0])
    dato = ((adc[1]&3) << 8) + adc[2]
    return dato
```

La función ***ConversionVoltaje*** obtiene el voltaje correspondiente al dato digital extraído por el sensor. Para ello, el dato es dividido por 1023 puesto que es el rango de valores que puede tomar, y más tarde multiplicado por 3.3, que es el voltaje de referencia usado en la Raspberry Pi. Además del dato digital, es necesario pasar por parámetros el número de decimales con el que expresar el voltaje.

```
def ConversionVoltaje(dato, decimal):
        voltaje = (dato*3.3)/float(1023)
        voltaje = round(voltaje, decimal)
    return voltaje
```

La función ***ConversionCorriente*** obtiene la corriente instantánea a partir del voltaje del dato extraído. Para ello, únicamente es necesario multiplicar el voltaje por el término de calibración dado por el sensor de corriente. Puesto que se han utilizado sensores SCT-013-030, este valor es 30 A/V.

```
def ConversionCorriente(voltaje, decimal):
        corriente = voltaje*30
        corriente = round(corriente, decimal)
    return corriente
```

Otra función implementada es ***CorrienteEficaz,*** que permite hallar la corriente eficaz dada una suma de corriente y un número de muestras. Como la señal de corriente que capta el sensor es alterna, si esta es medida obteniendo muestras cada x tiempo, lo único que se obtendría son muestras aleatorias de la señal alterna; por tanto, lo que se ha calculado en esta función para hallar la energía consumida es la corriente eficaz que fluye por el cable[13].

La corriente RMS o eficaz es aquella capaz de producir el mismo trabajo que su valor en corriente continua. Con ella se puede calcular fácilmente la potencia RMS o real, que representa el valor al que opera el sensor durante un período de tiempo.

Para calcular el valor de la corriente eficaz, es necesario aplicar la siguiente fórmula: $i = \sqrt{\frac{1}{N}\sum_{n=0}^{N} i_n^2}$, donde N es la cantidad de muestras tomadas, e i_n las muestras de corriente instantánea recogidas.

```
def CorrienteEficaz(sumaCorriente2, muestras, decimas):
        corrienteEficaz = math.sqrt(sumaCorriente2/muestras)
        corrienteEficaz = round(corrienteEficaz, decimas)
        return corrienteEficaz
```

Además, se ha creado la función ***ConversionPotencia***, que halla la potencia equivalente a la corriente eficaz que se pasa por parámetros.

```
def ConversionPotencia(corrienteEficaz, decimas):
        potencia = corrienteEficaz*230
        potencia = round(potencia, decimas)
        return potencia
```

La siguiente función implementada en el script es ***ini_delay,*** que inicializa el fichero de tiempos entre la recogida de muestras para cada canal a 1 segundo.

```
def ini_delay():
    archivo = open("/home/pi/Desktop/fichero_conf.txt", 'w')
    canal = 1
    for canal in range(1, 11):
        archivo.write("Canal " + str(canal) + ": 1\n")
    archivo.close()
```

Otra función implementada es ***get_delay,*** que dado un canal introducido por parámetros, se obtiene el tiempo existente entre recogida de muestras para ese canal. Para ello, se lee el fichero de tiempos fichero_conf.txt, y una vez se llega al canal deseado, se extrae el tiempo correspondiente.

```
def get_delay(canal):
    archivo = open("/home/pi/Desktop/fichero_conf.txt", 'r')
    cont = 1
    for linea in archivo.readlines():
        if cont == canal:
            delay = linea.split(" ")[2]
            delay = delay.split("\n")[0]
        cont = cont + 1
    archivo.close()
    return delay
```

Una vez definidas estas funciones, ha sido necesario crear una en la que se utilicen, denominada ***contador***, de forma que se obtenga finalmente la energía consumida para un canal introducido por parámetros. En ella se ha realizado un bucle en el que siempre que se vaya a extraer datos del sensor, primero se obtengan muestras (en este caso se han extraído 100), una cada 2ms, y después se calcule la suma de los cuadrados de las muestras de corriente instantánea, que más tarde servirá para calcular el valor eficaz de la corriente, la potencia, y con ello la energía consumida.

Por otra parte, al introducir un delay el sistema se ralentizará 100 veces durante 2ms, por lo que para que la recogida de muestras se realice con los tiempos exactos, se ha captado el tiempo que tarda realmente en terminar.

```
def contador(canal):
    delay_s = get_delay(canal)
    delay = 0.002
    muestras = 100
    sumaCorriente = 0
    #Tiempo de inicio del bucle
    start_time = time.time()

    for i in range (muestras):
        dato_gpio = LecturaCanal(canal-1)
        voltaje = ConversionVoltaje(dato_gpio, 3)
        corriente = ConversionCorriente(voltaje,3)
        corriente2 = corriente**2
        sumaCorriente = sumaCorriente + corriente2
        time.sleep(delay)

    #Calculo del tiempo de ejecucion del bucle
    elapsed_time = time.time() - start_time

   #Como se han eliminado los semiciclos negativos, es necesario tener en cuenta esa parte anulada duplicando el valor del sumatorio
    sumaCorriente = sumaCorriente*2

    #Calculo de la corriente eficaz que fluye por el cable
    corrienteEf = CorrienteEficaz(sumaCorriente, muestras, 3)
```

Una vez calculado el valor eficaz de la corriente ya es posible obtener la potencia, y a su vez la energía consumida. Esta energía es un valor expresado en kWh, por lo que a la hora de escribirlo en el fichero se ha expresado el valor de la energía consumida por los aparatos para una hora de uso.

```
potencia = ConversionPotencia(corrienteEf, 3)
energía = potencia/float(1000)
```

Una vez hallados los valores, sólo queda escribirlos en el fichero junto a los demás atributos (fecha, hora, canal y corriente).

```
fecha_actual = time.strftime("%d/%m/%y %H:%M:%S",
time.localtime(time.time()))
fichero = open('/home/pi/Desktop/datos_canal' + str(canal) + '.txt', 'a')
fichero.write("{}  Canal {}: {}A {}W {}kWh \n".format(fecha_actual, canal,
corrienteEf, potencia, energia))
fichero.close()
```

Para finalizar la función, se devolverá el tiempo recogido en el fichero de muestras para ese canal menos el tiempo de ejecución del bucle.

```
tiempo_restante = float(delay_s) - elapsed_time
return tiempo_restante
```

Por último, para conseguir que los sensores extraigan datos de forma continua y cada uno a un tiempo diferente, ha sido necesario realizarlo mediante interrupciones. Por tanto, se ha creado una función específica para cada sensor, de forma que cada uno funcione en paralelo a los demás y a un tiempo determinado.

Dentro de cada uno de ellos se llama a la función contador, y se obtiene el tiempo calculado para esperar los segundos establecidos entre la recogida de muestras para el canal correspondiente a cada interrupción. Además, para que la interrupción se repita constantemente en el tiempo es necesario llamarse a sí misma dentro de la función realizada para cada canal.

```
def timer1():
    delay1 = contador(1)
    t1 = threading.Timer(delay1, timer1)
    t1.start()

def timer2():
    delay2 = contador(2)
    t2 = threading.Timer(delay2, timer2)
    t2.start()

def timer3():
    delay3 = contador(3)
    t3 = threading.Timer(delay3, timer3)
    t3.start()

def timer4():
    delay4 = contador(4)
    t4 = threading.Timer(delay4, timer4)
    t4.start()

def timer5():
    delay5 = contador(5)
    t5 = threading.Timer(delay5, timer5)
    t5.start()

def timer6():
    delay6 = contador(6)
    t6 = threading.Timer(delay6, timer6)
    t6.start()

def timer7():
    delay7 = contador(7)
    t7 = threading.Timer(delay7, timer7)
    t7.start()
```

```
def timer8():
    delay8 = contador(8)
    t8 = threading.Timer(delay8, timer8)
    t8.start()

def timer9():
    delay9 = contador(9)
    t9 = threading.Timer(delay9, timer9)
    t9.start()

def timer10():
    delay10 = contador(10)
    t10 = threading.Timer(delay10, timer10)
    t10.start()
```

Una vez han sido definidas todas las funciones del programa, se ha inicializado el fichero de tiempos fichero_conf.txt y se ha llamado a las interrupciones creadas para los diez sensores, de forma que al iniciarse la Raspberry, el programa automáticamente comience a recoger muestras y guardarlas en los ficheros correspondientes a cada canal.

```
ini_delay()
timer1()
timer2()
timer3()
timer4()
timer5()
timer6()
timer7()
timer8()
timer9()
timer10()
```

6.2. Opciones adicionales

6.2.1. Cambio del tiempo entre muestras de un canal

Además de la función principal del programa, que es la recogida de datos de forma automática, el sistema debe poder llevar a cabo dos opciones complementarias. Una de ellas es la posibilidad de cambiar el tiempo que existe entre la extracción de muestras de un canal determinado.

En primer lugar, se ha creado un script denominado *script_conf.py,* que será el que se lance por consola para activar el cambio de tiempos a través del comando *python /home/pi/python-spi/script_conf.py canal tiempo*, donde el canal debe ser un número comprendido entre el 1 y 10, y el tiempo un número expresado en s.

Para conseguir estas restricciones ha sido necesario crear dentro del script dos funciones: ***lee_canal,*** que activa un mensaje de error siempre que el primer argumento introducido en la consola no sea un número entre 1 y 10, y ***lee_delay,*** que obliga a que el segundo argumento sea un número entero.

```
import os
import sys


def lee_canal():
    valor = ""
    while isinstance(valor, int) != True or valor<=1 or valor>10:
        valor = raw_input("\nIngrese un numero entero comprendido entre 1 y 10: ")
        print ""
        try:
            valor = int(valor)
            if (valor>0 and valor<=10):
                return valor
            else:
                print("Error al introducir canal")
        except ValueError:
            print("Error al introducir canal")
```

```
def lee_delay():
    valor = “”
    while isinstance(valor, int) != True:
        valor = raw_input(“\nIngrese el tiempo deseado entre cada muestra (en ms): “)
        print “”
        try:
            valor = int(valor)
            return valor
        except ValueError:
            print(“Error al introducir delay”)
```

Una vez configuradas las funciones, han sido utilizadas en el fichero de forma que aparezca un mensaje de confirmación por pantalla siempre y cuando el usuario haya introducido valores correctos; en caso contrario, aparecerá un mensaje de error mientras éstos no sean válidos.

```
if len(sys.argv) ¡= 3:
    print(“El numero de argumentos introducidos debe ser 2”)
else:
    #LECTURA DEL CANAL
    canal = sys.argv[1]
    try:
        canal = int(canal)
        if (canal<0 or canal>10):
            print(“Error al introducir canal”)
            canal = lee_canal()
    except ValueError:
        print(“Error al introducir canal”)
        canal = lee_canal()
```

```
    #LECTURA DEL DELAY
    delay = sys.argv[2]
    try:
        delay = int(delay)
    except ValueError:
        print("Error al introducir delay")
        delay = lee_delay()

print("\n-----------------------------------------------")
print "\nCanal: " + str(canal) + ", Delay: " + str(delay) + " ms"
print ""
```

A continuación, se leerá el fichero de configuración donde se guardan los tiempos entre muestras para cada canal, denominado *fichero_conf.txt,* y se creará un fichero auxiliar (*aux.txt*) sobre el que se irán almacenando los valores de los tiempos para cada canal.

```
archivo = open("/home/pi/Desktop/fichero_conf.txt", "r")
aux = open("/home/pi/Desktop/aux.txt", "w")
```

El programa utilizará un bucle que lee las líneas del fichero de configuración una a una, guardando los mismos valores de tiempos en el nuevo fichero auxiliar siempre que el canal no coincida con el introducido. Cuando se llega al canal que se desea modificar, se crea una nueva línea con el tiempo introducido, y se almacena en el fichero auxiliar. De este modo, el fichero *aux.txt* contendrá todos los tiempos entre muestras, incluido el modificado.

```
cont=1
#MODIFICACION DEL FICHERO DE CONFIGURACION
for linea in archivo.readlines():
    if cont == canal:
        linea_nueva = "Canal " + str(canal) + ": " + str(delay)
        aux.write(linea_nueva + "\n")
```

```
    else:
        aux.write(linea)
    cont = cont + 1
```

Para finalizar, el fichero *aux.txt* será renombrado como *fichero_conf.txt* otra vez, de forma que en la siguiente vez que se lance el script el proceso sea el mismo. Además, antes de terminar se cerrarán los ambos ficheros.

```
os.rename("/home/pi/Desktop/aux.txt", "/home/pi/Desktop/fichero_conf.txt")
aux.close()
archivo.close()
```

6.2.2. Obtención de datos a partir de canal, fecha y hora

Otra de las opciones que el sistema debe poder usar es la obtención de datos a partir de un canal, fecha y hora concretos, que se extraerán del fichero correspondiente al canal introducido (datos_canalX.txt), y serán guardados en uno nuevo denominado datos_fecha.txt.

Se ha creado un script denominado *get_data.py*, que será el que se lance por consola para activar la obtención de datos. Para ello, es necesario introducir el comando *python /home/pi/python-spi/get_data.py canal fecha hora*, donde se debe introducir como argumentos: un canal entre el 1 y 10, y fecha y hora a partir de los cuales se desea obtener datos (anteriores a la fecha en que se utilice el sistema).

Estas restricciones se han conseguido creando cinco funciones dentro del script: ***validarFecha***, que verifica si la fecha introducida está en el formato adecuado (dd/mm/aa), ***validarHora***, que verifica si la hora introducida está en el formato adecuado (hh:mm:ss), ***compararFecha***, que sirve para comprobar dos fechas y saber cuál es posterior a la otra, ***lee_canal,*** que tiene el mismo funcionamiento que en el apartado anterior, y ***lee_fecha***, que imprime por pantalla un mensaje de error en caso de que la fecha u hora no sean válidas, y obliga al usuario a introducirlos de forma correcta. Además de ello, se han importado las librerías correspondientes:

```
#! /usr/bin/python
import sys
import os
import time
#DEFINICION DE FUNCIONES
def validarFecha(fecha):
    try:
        result = time.strptime(fecha, "%d/%m/%y")
        return True
    except:
        return False
def validarHora(hora):
    try:
        result = time.strptime(hora, "%H:%M:%S")
        return True
    except:
        return False

def compararFecha(fecha1, fecha2, hora1, hora2):
    dia1 = fecha1.split("/")[0]
    dia2 = fecha2.split("/")[0]
    mes1 = fecha1.split("/")[1]
    mes2 = fecha2.split("/")[1]
    year1 = fecha1.split("/")[2]
    year2 = fecha2.split("/")[2]
    if year1 > year2:
        return 1
    elif year1 < year2:
        return 2
```

```
    else:
        if mes1 > mes2:
            return 1
        elif mes1 < mes2:
            return 2
        else:
            if dia1 > dia2:
                return 1
            elif dia1 < dia2:
                return 2
    if dia1 == dia2 and mes1 == mes2 and year1 == year2:
        h1 = hora1.split(":")[0]
        h2 = hora2.split(":")[0]
        m1 = hora1.split(":")[1]
        m2 = hora2.split(":")[1]
        s1 = hora1.split(":")[2]
        s2 = hora2.split(":")[2]

        if h1 > h2:
            return 1
        elif h1 < h2:
            return 2
        else:
            if m1 > m2:
                return 1
            elif m1 < m2:
                return 2
            else:
                if s1 > s2:
```

```
                return 1
            elif s1 < s2:
                return 2
            else:
                return 1
 def lee_canal():
    valor = ""
    while isinstance(valor, int) != True or valor<=1 or valor>10:
        valor = raw_input("\nIngrese un numero entero comprendido entre 1 y 10: ")
        print ""
        try:
            valor = int(valor)
            if (valor>0 and valor<=10):
                return valor
            else:
                print("Error al introducir canal")
        except ValueError:
            print("Error al introducir canal")


def lee_fecha():
    fecha = ""
    hora = ""
    cont = 0
    fecha_actual = time.strftime("%d/%m/%y")
    hora_actual = time.strftime("%H:%M:%S")


    while ((validarFecha(fecha) != True) or (validarHora(hora) != True) or
(compararFecha(fecha_actual, fecha, hora_actual, hora) == 2)):
        fecha = ""
```

```
            hora = ""
            if cont == 1:
                print "\nError al introducir datos. Introduzca una fecha menor a la actual\n"

            while validarFecha(fecha) != True:
                fecha = raw_input("\nIngrese fecha a partir de la que desea obtener datos (dd/mm/aa): ")
            print ""

            while validarHora(hora) != True:
                hora = raw_input("\nIngrese hora a partir de la que desea obtener datos (hh:mm:ss): ")
            cont = 1

    return fecha, hora
```

Una vez definidas las funciones, han sido utilizadas en el fichero de forma que aparezca un mensaje de confirmación por pantalla siempre y cuando el usuario haya introducido valores correctos por argumentos; en caso contrario, aparecerá un mensaje de error siempre que no sean válidos. Además, se realiza la comprobación de que la fecha y horas introducidas sean anteriores a la fecha de uso del sistema:

```
if len(sys.argv) != 4:
    print("\nEl numero de argumentos introducidos debe ser 3\n")
else:
    canal = sys.argv[1]
    fecha = sys.argv[2]
    hora = sys.argv[3]
```

```
    #LECTURA DEL CANAL
    try:
        canal = int(canal)
        if (canal<0 or canal>10):
            print("Error al introducir canal")
            canal = lee_canal()
    except ValueError:
        print("Error al introducir canal")
        canal = lee_canal()

    #LECTURA DE FECHA Y HORA
    fecha_actual = time.strftime("%d/%m/%y")
    hora_actual = time.strftime("%H:%M:%S")

    if validarFecha(fecha) and validarHora(hora) and (compararFecha(fecha_actual,
fecha, hora_actual, hora) == 1):
        print "\nFecha y hora correctas"

    else:
        print "\nError al introducir fecha y hora"
        aux = lee_fecha()
        fecha = aux[0]
        hora = aux[1]

    print("\n-----------------------------------------------")
    print "\nObteniendo datos del canal " + str(canal) + " a partir de: " + str(fecha) + " "
+ str(hora)
    print ""
```

A continuación, se lee el fichero donde se han guardado los datos correspondientes al canal introducido, datos_canalX.txt, y se crea un nuevo fichero denominado datos_fecha.txt en el que se guardarán los datos extraídos a partir de la fecha y hora introducidos por argumentos:

```
archivo = open("/home/pi/Desktop/datos_canal" + str(canal) + ".txt", "r")
datos_fecha = open("/home/pi/Desktop/datos_fecha.txt", "w")
```

Por último, el programa utiliza un bucle que lee las líneas del fichero datos_canalX.txt una a una, y la escribirá en el nuevo fichero datos_fecha.txt siempre que la fecha y hora del dato leído sean posteriores a las introducidas por línea de comandos:

```
#RELLENAR DATOS EN EL FICHERO A PARTIR DE LA FECHA INDICADA
for linea in archivo.readlines():
    fecha_linea = linea.split(" ")[0]
    hora_linea = linea.split(" ")[1]

    if compararFecha(fecha_linea, fecha, hora_linea, hora) == True:
        datos_fecha.write(linea)

datos_fecha.close()
archivo.close()
```

7. Valoración final

7.1. Pruebas realizadas

Una vez programada la Raspberry Pi y finalizado el montaje completo del sistema, se han llevado a cabo diversas pruebas para confirmar que el sistema funciona de manera correcta. Para ello se ha utilizado una regleta, y se han ido conectando los diferentes sensores sobre su cable de red para comprobar que cada canal funciona.

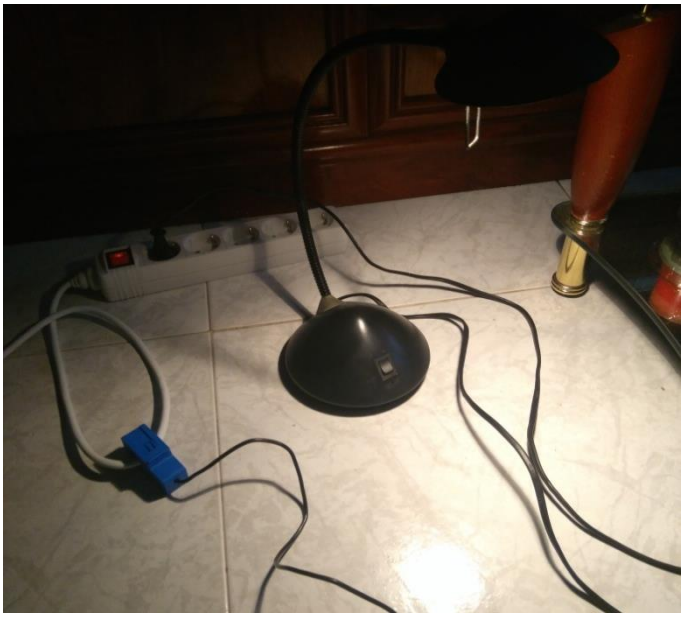


Figura 44: Medición de la regleta a través del sensor.

En primer lugar, se ha realizado una medición de la regleta encendida y enchufada a la corriente con el sensor nº1, sin ningún aparato conectado a ella. Esto se ha hecho para conocer el gasto que supone tener la regleta encendida aunque no haya dispositivos enchufados.

```
23/09/17 21:41:19  Canal 1: 0.003A 0.69W 0.00069kWh
23/09/17 21:41:20  Canal 1: 0.001A 0.23W 0.00023kWh
23/09/17 21:41:21  Canal 1: 0.005A 1.15W 0.00115kWh
23/09/17 21:41:22  Canal 1: 0.007A 1.61W 0.00161kWh
23/09/17 21:41:23  Canal 1: 0.002A 0.46W 0.00046kWh
23/09/17 21:41:24  Canal 1: 0.001A 0.23W 0.00023kWh
23/09/17 21:41:25  Canal 1: 0.003A 0.69W 0.00069kWh
23/09/17 21:41:26  Canal 1: 0.005A 1.15W 0.00115kWh
23/09/17 21:41:27  Canal 1: 0.003A 0.69W 0.00069kWh
23/09/17 21:41:28  Canal 1: 0.008A 1.84W 0.00184kWh
23/09/17 21:41:29  Canal 1: 0.001A 0.23W 0.00023kWh
23/09/17 21:41:30  Canal 1: 0.009A 2.07W 0.00207kWh
23/09/17 21:41:31  Canal 1: 0.005A 1.15W 0.00115kWh
23/09/17 21:41:32  Canal 1: 0.005A 1.15W 0.00115kWh
```

Figura 45: Gasto de regleta encendida sin aparatos conectados.

Como se puede comprobar, el gasto energético al medir la regleta encendida no es 0, puesto que la bombilla encendida consume, aunque sea muy poco. Además, el valor obtenido no es exacto, ya que existe error de ruido (unos 10-15mA) en la toma de corriente.

Por otra parte, se ha llevado a cabo la medición de dos aparatos eléctricos. Uno de ellos es una lámpara de escritorio, cuya potencia teórica es de 20W. Esta se ha realizado con el sensor nº9, que corresponde al segundo conversor. La medición por este canal se ha realizado para comprobar que se recogen datos correctamente a través de los dos conversores.

```
23/09/17 21:52:45  Canal 9: 0.087A 20.01W 0.02001kWh
23/09/17 21:52:46  Canal 9: 0.097A 22.31W 0.02231kWh
23/09/17 21:52:47  Canal 9: 0.091A 20.93W 0.02093kWh
23/09/17 21:52:48  Canal 9: 0.088A 20.24W 0.02024kWh
23/09/17 21:52:49  Canal 9: 0.093A 21.39W 0.02139kWh
23/09/17 21:52:50  Canal 9: 0.079A 18.17W 0.01817kWh
23/09/17 21:52:51  Canal 9: 0.094A 21.62W 0.02162kWh
23/09/17 21:52:52  Canal 9: 0.087A 20.01W 0.02001kWh
23/09/17 21:52:53  Canal 9: 0.089A 20.47W 0.02047kWh
23/09/17 21:52:54  Canal 9: 0.088A 20.24W 0.02024kWh
23/09/17 21:52:55  Canal 9: 0.094A 21.62W 0.02162kWh
23/09/17 21:52:56  Canal 9: 0.083A 19.09W 0.01909kWh
23/09/17 21:52:57  Canal 9: 0.083A 19.09W 0.01909kWh
23/09/17 21:52:58  Canal 9: 0.086A 19.78W 0.01978kWh
23/09/17 21:52:59  Canal 9: 0.089A 20.47W 0.02047kWh
23/09/17 21:53:00  Canal 9: 0.092A 21.16W 0.02116kWh
23/09/17 21:53:01  Canal 9: 0.088A 20.24W 0.02024kWh
```

Figura 46: Gasto energético lámpara de escritorio 20W.

Como se observa en la figura anterior, los datos recogidos muestran que su potencia oscila alrededor de 20W (gasto energético en torno a 0.02kWh para una hora de conexión).

Por último, la otra medición realizada es de otra lámpara de escritorio, en este caso de una potencia de 35W. Esta se ha realizado a través del sensor nº4, para verificar que también se pueden recoger datos por ese canal.

```
23/09/17 22:01:06  Canal 4: 0.155A 35.65W 0.03565kWh
23/09/17 22:01:07  Canal 4: 0.153A 35.19W 0.03519kWh
23/09/17 22:01:08  Canal 4: 0.152A 34.96W 0.03496kWh
23/09/17 22:01:09  Canal 4: 0.154A 35.42W 0.03542kWh
23/09/17 22:01:10  Canal 4: 0.152A 34.96W 0.03496kWh
23/09/17 22:01:11  Canal 4: 0.163A 37.49W 0.03749kWh
23/09/17 22:01:12  Canal 4: 0.151A 34.73W 0.03473kWh
23/09/17 22:01:13  Canal 4: 0.155A 35.65W 0.03565kWh
23/09/17 22:01:14  Canal 4: 0.159A 36.57W 0.03657kWh
23/09/17 22:01:15  Canal 4: 0.157A 36.11W 0.03611kWh
23/09/17 22:01:16  Canal 4: 0.162A 37.26W 0.03726kWh
23/09/17 22:01:17  Canal 4: 0.163A 37.49W 0.03749kWh
23/09/17 22:01:18  Canal 4: 0.156A 35.88W 0.03588kWh
23/09/17 22:01:19  Canal 4: 0.158A 36.34W 0.03634kWh
23/09/17 22:01:20  Canal 4: 0.149A 34.27W 0.03427kWh
23/09/17 22:01:21  Canal 4: 0.154A 35.42W 0.03542kWh
23/09/17 22:01:22  Canal 4: 0.161A 37.03W 0.03703kWh
23/09/17 22:01:23  Canal 4: 0.158A 36.34W 0.03634kWh
23/09/17 22:01:24  Canal 4: 0.154A 35.42W 0.03542kWh
```

Figura 47: Gasto energético lámpara de escritorio 35W.

7.2. Problemas encontrados

A lo largo del proyecto han aparecido diversos problemas que han ralentizado su desarrollo.

El primer problema grave apareció a la hora de realizar los cambios de intervalos entre muestras, puesto que la idea inicial era detener el proceso del programa principal configurado con el demonio de arranque automático, introducir los nuevos tiempos desde la consola y volver a lanzar el proceso, pero esa idea se desechó porque no se consiguió que se actuara en segundo plano y al introducir los nuevos tiempos, se mantenía la consola ocupada. Finalmente, tras unos días haciendo pruebas se solucionó programando un fichero .txt con los tiempos de cada canal, que se lee automáticamente cada vez que se desea recoger datos de un sensor.

Por otra parte, el problema más importante tuvo lugar al probar la obtención de datos. En un principio, la idea era extraer al inicio de cada intervalo (el establecido para cada canal) varias muestras de corriente instantánea cada 2 ms, y con ellas hallar el valor de corriente de pico de entre las muestras recogidas. Tras realizar varias pruebas con diferentes aparatos, se comprobó que el valor de potencia obtenido no era el adecuado. Por ello se investigó a través de Internet, y se llegó a la conclusión de que la corriente a hallar debía ser la eficaz, cuya modificación en el sistema hizo que finalmente los datos se obtuvieran de forma correcta.

7.3. Líneas futuras

Una vez finalizado el sistema de monitorización de energía consumida, se han tenido en cuenta una serie de posibles mejoras y nuevas implementaciones en el mismo para ofrecerle una mayor funcionalidad.

La idea principal es que una vez se ha desarrollado el proyecto, este sea optimizado de forma que sea impreso en una placa en vez de usar una protoboard, y se integre en el cuadro de electricidad de una vivienda. De este modo, se podría acceder a los datos del cuadro eléctrico incluyendo en el circuito un teclado matricial en el que introducir la línea a medir, y visualizarlos a través de un display digital.

Otra idea sería la de crear una interfaz web que gestionase la monitorización de cada línea del cuadro, de forma que el sistema tuviera conectividad a Internet y la información extraída por cada sensor se cargara automáticamente cada x tiempo en una base de datos. Esta sería tratada de forma que se pudieran obtener estadísticas gráficas sobre qué aparatos consumen más, de qué tipo son, en qué franjas horarias ocurre y cómo afecta ello en el precio final de la factura de luz.

Otra posible mejora podría ser diseñar una aplicación móvil, de forma que la información gestionada en la interfaz web también pudiera ser vista desde cualquier smartphone o tablet. Además, podría elaborarse un sistema para recibir alertas y notificaciones acerca del consumo de la instalación, que avise en caso de fallos o comportamientos extraños.

7.4. Conclusiones

Se ha desarrollado un sistema que permita monitorizar la energía consumida para una instalación domótica, de forma que se puede recoger información de diez sensores de corriente a la vez.

También se ha logrado que la extracción de datos se realice de forma automática al encender la Raspberry Pi, y el programa permite llevar a cabo las dos opciones adicionales pensadas, tanto el cambio de tiempos entre la recogida de muestras como la obtención de datos a partir de un canal, fecha y hora concretos.

Además, el objetivo es que éste sistema fuera sencillo, de bajo coste y de fácil instalación, lo que tras los resultados obtenidos se certifica que se ha podido conseguir.

Durante el desarrollo de este proyecto se ha realizado un análisis de las tecnologías relacionadas con el mismo, lo que ha permitido concluir que el IoT avanza muy rápidamente y que es de mucha utilidad, ya que facilita en gran medida la vida de sus usuarios. Además, tras una comparación con los productos similares que existen actualmente en el mercado, se ha verificado que la elaboración de este sistema permite monitorizar una instalación eléctrica con más líneas que con los productos existentes, y a un precio más bajo.

Personalmente, el proyecto me ha permitido tener la experiencia de llevar a cabo un trabajo completo de forma autónoma. Se ha conseguido establecer una gran relación entre las competencias adquiridas a lo largo de la carrera y la realización de un sistema real con unas especificaciones concretas como el realizado, ya que para desarrollarlo ha sido necesario utilizar conocimientos tanto de programación como de diseño de circuitos.

Por tanto, el presente proyecto ha supuesto todo un reto académico, ya que aunque han surgido problemas estos han podido ser resueltos por medio de la investigación autónoma y la ayuda del tutor, por lo que su consecución supone un éxito tanto personal como profesional.

8. Bibliografía

[1] Dotuel, F. (2012). "Conoce a la placa que quiere revolucionar tu mundo digital: Raspberry Pi a fondo (I)". *Componentes*, 20 de noviembre. Disponible en: https://www.xataka.com/componentes/conoce-a-la-placa-que-quiere-revolucionar-tu-mundo-digital-raspberry-pi-a-fondo [Consulta: 11 de marzo de 2017].

[2] Rivera, N. (2015). *Qué es el Internet of Things y cómo cambiará nuestra vida en el futuro*. Disponible en: https://hipertextual.com/2015/06/internet-of-things [Consulta: 11 de marzo de 2017].

[3] Canive, T. (2017). "¿Qué es un diagrama de Gantt y para qué sirve?". *Blog de gestión de proyectos*, 21 de abril. Disponible en: https://www.sinnaps.com/blog-gestion-proyectos/diagrama-gantt-sirve [Consulta: 4 de septiembre de 2017].

[4] *Ingeniería domótica a favor del medio ambiente* (2016). Disponible en: http://www.axialstructural.com/ingenieria-domotica-favor-del-medio-ambiente [Consulta: 28 de agosto de 2017].

[5] *A fondo: normativa domótica* (2011). Disponible en: http://www.domodesk.com/a-fondo-normativa-domotica [Consulta: 29 de agosto de 2017]

[6] Ruiz Sánchez, B. (2016). *Exposición de las principales normas y disposiciones legales aprobadas a nivel nacional, europeo e internacional para sistemas domóticos e inmóticos*. Disponible en:

https://www.casadomo.com/comunicaciones/exposicion-principales-normas-disosiciones-legales-aprobadas-nivel-nacional-europeo-internacional-para-sistemas-domoticos-inmoticos [Consulta: 1 de septiembre de 2017].

[7] Velasco, R. (2017). *6 alternativas a Raspberry Pi para comprar este 2017*. Disponible en: https://www.redeszone.net/2017/01/18/6-alternativas-al-raspberry-pi-comprar-este-2017/ [Consulta: 11 de marzo de 2017].

[8] Pajares Redondo, J. (2015). *Diseño e implementación de una ayuda electrónica para pacientes con baja visión periférica*. TFG. Universidad Carlos III, Madrid. Disponible en: https://e-archivo.uc3m.es/handle/10016/15439 [Consulta: 16 de marzo de 2017]

[9] *Raspbian*. (s.d.). Disponible en: https://www.raspberrypi.org/downloads/raspbian/ [Consulta: 7 de marzo de 2017].

[10] *¿En qué lenguaje se programará el Internet de las Cosas?* (2015) Disponible en: https://bbvaopen4u.com/es/actualidad/en-que-lenguaje-se-programara-el-internet-de-las-cosas [Consulta: 16 de marzo de 2017].

[11] *Python*. (s.d.). Disponible en: https://www.raspberrypi.org/documentation/usage/python/ [Consulta: 28 de marzo de 2017].

[12] "Tutorial sensor de corriente ACS712". *Sensores*, 26 de octubre de 2016. Disponible en: http://www.naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html [Consulta: 12 de julio de 2017].

[13] "Tutorial sensor de corriente AC no invasivo SCT-013". *Sensores*, 2 de julio de 2016. Disponible en: http://www.naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html [Consulta: 14 de julio de 2017].

[14] Ruiz Carreterio, D. (2015). *Sistema de Monitorización y Control de Consumo eléctrico a través de Internet.* PFC. Universidad de Sevilla.

[15] C. (2014). *Sistema Wattio: lo revisamos*. Disponible en: https://nergiza.com/sistema-wattio-lo-revisamos/ [Consulta: 12 de marzo de 2017].

[16] González Domínguez, C (2015). *Aplicaciones orientadas a la domótica con Raspberry Pi.* TFG. Universidad Carlos III, Madrid. Disponible en: https://e-archivo.uc3m.es/handle/10016/15439 [Consulta: 9 de agosto de 2017].

[17] Revuelta Irisarri, J. (2015). *Monitorización del consumo eléctrico de un hogar: sensado y acondicionamiento de la corriente*. TFG. Universidad Pública de Navarra.

[18] *Serial Peripheral Interface* (s.d.). Disponible en: https://raspberrypi-aa.github.io/session3/spi.html [Consulta: 29 de marzo de 2017].

[19] *Bus I2C de Raspberry Pi I* (2013). Disponible en: http://www.cmdearcos.es/bus-i2c-de-raspberry-pi-1/ [Consulta: 29 de marzo 2017].

[20] Doutel, F. (2014). "Cómo grabar un archivo .img en una SD de forma sencilla para tu Raspberry Pi". *Trucos y bricolaje smart*, 4 de abril. Disponible en: https://www.xatakahome.com/trucos-y-bricolaje-smart/grabar-la-sd-para-tu-raspberry-pi-no-puede-ser-mas-sencillo-con-estas-aplicaciones [Consulta: 20 de marzo de 2017].

[21] Flores, J. *Raspberry Pi – Primer inicio de Raspberry y configuración*. (15 de marzo de 2014). Disponible en: https://www.youtube.com/watch?v=ctr4ZGSh9Lg [Consulta: 20 de marzo de 2017].

[22] Escayola Calvo, J. (2016). *Colección de comandos útiles para Raspbian*. Disponible en: http://www.raspberrypizaragoza.es/coleccion-de-comandos-utiles-para-raspbian-actualizable/ [Consulta: 20 de marzo de 2017].

[23] Irineo Luis, R. (2016). *Configuración de Wifi y Bluetooth en Raspberry Pi 3*. Disponible en: http://bot-boss.com/configuracion-wifi-bluetooth-raspberry-pi-3/ [Consulta: 20 de agosto de 2017]

[24] Bejarano, M. (2013). *Tutorial 5 – Conexión remota al Raspberry Pi usando SSH.* Disponible en: http://www.frambuesapi.co/2013/09/25/tutorial-5-conexion-remota-al-raspberry-pi-usando-ssh/ . [Consulta: 22 de marzo de 2017].

[25] *¿Qué es PuTTY y para qué sirve?* (2014). Disponible en: http://www.vozidea.com/que-es-putty-y-para-que-sirve [Consulta: 23 de marzo de 2017].

[26] Bejarano, M. (2013). *Tutorial 6 – Conexión remota al Raspberry Pi usando VNC*. Disponible en: http://www.frambuesapi.co/2013/10/07/conexion-remota-al-raspberry-pi-usando-vnc/ [Consulta: 23 de marzo de 2017]

[27] *Tutorial de Raspberry Pi GPIO y Python* (2014). Disponible en: https://robologs.net/2014/04/12/tutorial-de-raspberry-pi-gpio-y-python-i/ [Consulta: 28 de marzo de 2017].

[28] "Cómo ejecutar un programa automáticamente al arrancar la Raspberry Pi". 20 de diciembre 2013. [Consulta: 27 de marzo 2017]. Disponible en: https://nideaderedes.urlansoft.com/2013/12/20/como-ejecutar-un-programa-automaticamente-al-arrancar-la-raspberry-pi/

[29] *Analogue Sensors On The Raspberry Pi Using An MCP3008* (2013). Disponible en: http://www.raspberrypi-spy.co.uk/2013/10/analogue-sensors-on-the-raspberry-pi-using-an-mcp3008/ [Consulta: 13 de julio de 2017].

[30] Legarrea Oyarzun, A. (2015). *Medidor de consumo eléctrico de un hogar: procesado de datos mediante Raspberry-pi*. TFG. Universidad Pública de Navarra.

A. Anexos

A.1. Extended abstract

Introduction

The huge irruption of the TIC sector has originated the development of techniques which search to transform people life. One of them is IoT, whose goal is to connect objects which are around them with the purpose to get data to optimize electronic devices and improve the user's life quality.

One of the most important applications is based on energy saving, because it allows creating intelligent houses capable of optimizing in an autonomous way each electrical device of an installation, thanks to data collected by sensors which are installed in each device.

Goals

The current project is framing inside IoT field. Specifically, it's based in an application which favors energy saving thanks to monitoring its consumed energy with a Raspberry Pi, from a domotics system.

This should be low-priced, and it should allow getting data through 10 sensors at a time, where each of them collects information corresponding to a line of the electricity box where it will be installed

Inside the system development can be separated two parts:

- On the one hand, it's necessary to carry out the design and the assembly of the signal sensing circuit, which is the responsible of carrying the obtained data by the electrical current sensors towards the Raspberry PI to be gestionated correctly.

- On the other hand, it's necessary to learn how to configure the Raspberry Pi and how it works, as well as carrying out the necessary programming to read and process obtained data through each sensor, which will be saved in different files called 'datos_canalX.txt', where X is the number of channel/sensor.

  In addition to the main program, it is necessary to configurate an auto start demon with the goal to system collects data in background when Raspberry Pi was turned on, without executing any instruction by the command console.

  Finally, the system can be able, introducing a concrete command in each option, to:

  - Modifying the time between the pickup of each sample for a specific channel, which will be saved in a file of times to be read continuously.

  - Picking-up data based on a channel, date and hour, in order to the 'datos_canalX.txt' file related to the introduced channel will be read, and subsequent data than introduced date and hour will be saved in another file.

Art state

To contextualize this project, it is necessary to talk about IoT. This is a new scenario where any device can have Internet connection, and because of that, having the ability to provide data about its behavior. The importance of these devices is based on the sensors they have, which are used to extract data about their activity, which will be treated to obtain innumerable advantages.

The platform provided by the university has been a raspberry Pi 3 Model B, a SBC which presents the basic functions of a computer. The operating system used to program the monitoring system has been Raspbian because it's the most optimized system for Raspberry Pi. The programming language chosen has been Python, considering that it's best one to interact with the GPIO terminals.

The type of current sensor used has been a detachable sounding line (SCT-013-030), in spite of it exists other models such as SCT-013-000 or the compact sounding line ACS712, which needs to break the cable to perform the measurement. Finally, a study has been carried out to search similar products in the market, among which we can found: Efergy solutions, OpenEnergyMonitor, Pack Wattio Energy or FIBARO.

Raspberry Pi

It's a low-cost computer with a small motherboard, which it's developed with the aim of stimulating the computer science teaching in schools. The main difference between Raspberry Pi and other SBC is the incorporation of GPIO pins, which act as an interface between it and the outside world, and that can be configured to control sensors, external devices, etc.

With the goal to work in an easier way, I have worked in a remote way from my PC. To do that, it has been necessary to allow SSH interfaces (to remote connection) and VNC (to get the graphic interface), as well as make the connections.

Basic system programming

The work will be developed in Python OS, so the first thing to do is to install the package with the SPI module which allows using GPIO pines.

After installing the OS, it´s time to configure Raspberry PI so that when it started, the created program will execute automatically in background. To do that, it's necessary to create a file in "/etc/init.d/" folder and introduce the corresponding code. Moreover, he file must be made executable with "sudo chmod 755 /etc/init.d/file" command and to activate the boot run with "sudo update-rc.d contador defaults" command.

System development: hardware part

It's necessary to create a conditioning circuit of the signal measured by the sensors. As input data to the Raspberry are digital and the circuit provides an analog signal, it must be transformed through an analog/digital converter, which has been decided after a study that was MCP3008.

This device only supports eight input channels and we need to be able to include ten sensors, so it has been necessary to install two more converters, using channels 0 to 4 of each one.

As Raspberry Pi only accepts values between 0-3.3V, and the measured signal will take negative values due to its alternating component (range -1 to 1), it has been decided to rectify the negative part of the sensor signal and work with positive part, so that it could take advantage of the full range of reading. After that, for each current sensor connected to the protoboard, a voltage follower has been made with an operational amplifier TL084, powered between 0-3.3V, so output signal is equivalent to that obtained in sensor, but with negative parts eliminated (value 0).

After this process, each sensor output is wired with its input corresponding to the converter; first five sensors with channels 0 to 4 of the first MCP, and the next five with channels 0 to 4 of the second.

System development: software part

- **Main program**

The main program of the system executed by the automatic boot daemon consists of a script called "contador.py", in which the necessary functions for the extraction of data through the sensors have been implemented.
The first one is "**LecturaCanal**", which consists in extract the analogical information contained in an input given channel and converts it to digital.
The "***ConversionCorriente***" function obtains the instantaneous current from the converter´s extracted data variable. In addition, the ***ConversionPotencia*** function has been created, which finds the power equivalent to the current that is passed through parameters.

The next function implemented in the script is "***ini_delay***", which initializes the time file between the sample collections for each channel to 1 second.
Another function implemented is "***get_delay"***, which given a channel introduced by parameters, gives us the time between a collection of samples for that channel.

Another implemented function is "***CorrienteEficaz"***, which allows finding the current given a current sum and a number of samples. Since the current signal flowing through the sensor is alternating, if this is measured by taking samples every x time, all that would be obtained are random samples of the alternating signal; therefore, what has been calculated in this function to find the energy consumed is the effective current flowing through the wire.

Once these functions have been defined, it has been necessary to create one in which they are used, called "***contador"***. A loop has been made in which, whenever data is to be extracted from the sensor, first samples are obtained (in this case 100 have been extracted), one each 2 ms, the sum of the squares of the instantaneous current samples has been calculated, which later will serve to calculate the effective value of the current, the power, and with it the consumed energy.

Finally, to get the sensors to extract data continuously and each at a different time, it has been necessary to do this through interruptions. Therefore, a specific function has been created for each sensor, so that each one operates in parallel to the rest with the others and at a certain time.

- **Additional options**

One of them is the possibility to change the time between the extractions of channel´s samples. A script named "***script_conf.py***" has been created, which will be launched by console to activate the change of times through the command "*python /home/pi/python-spi/script_conf.py canal tiempo*", where the channel must be a number between 1 and 10, and time must be a number expressed in s.
In order to achieve these restrictions it has been necessary to create two functions within the script: "***lee_canal***", which forces the channel to be a number between 1 and 10, and "***lee_delay***", which forces the time to be an integer.

The program uses a loop that reads the lines of the configuration file called "file_conf.txt" one by one, saving the same time values in an auxiliary file until the channel to be modified is reached.

At that time, a new line is created with the time entered by arguments, being stored in the auxiliary file. Later, this file will be renamed as "*fichero_conf.txt*".
Another option that is possible to accomplish is to obtain data given a channel, date and time, which are extracted from the file corresponding to the channel entered (datos_canalX.txt), being stored in a new one (datos_fecha.txt).

The "***get_data.py***" script has been created, which will be launched by console to enable to pick up data with the command "*python /home/pi/python-spi/get_data.py canal fecha hora"*, where the channel must be a number between 1 and 10, and the date and time must be the moment from which you want to obtain data.

To achieve these restrictions, five functions have been created within the script:

- ***validateDate***: Checks whether the date entered is in the format "dd/mm/aa"
- ***validateTime: C***hecks if the entered time is in the format (hh: mm: ss)
- ***compararFecha*** : Compares two dates and shows which one is after the other
- ***lee_canal:*** Forces the channel entered to be a number between 1 and 10
- ***lee_fecha:*** Forces the user to enter the date and time correctly if he has entered some invalid value

The program uses a loop which reads the lines of the file "*datos_canalX.txt*" one by one, and will write the line in the new file "*datos_fecha.txt*", as long as the date and time of the read data were later than the entered data.

Tests performed

Several tests have been carried out to confirm that the system works correctly. For this purpose, a powerstrip has been used, and the different sensors have been connected to their network cable to verify that each channel works.

First of all, a measurement of a power powerstrip and connected to the current with the sensor number 1 has been carried out without any devices connected to it. This has been done to know the expense of having the powerstrip running even if there are no devices plugged in.

Moreover, the measurement of two electrical appliances has been carried out. One of them is a desk lamp, whose theoretical power is 20W. This has been done with sensor number 9, to verify that data is correctly collected through the two converters. Collected data show that its power oscillates around 20W (energy expenditure around 0.02kWh for an hour of connection), although the measurements are not totally accurate, it varies due to a noise of about 10-15 mA.

Finally, the other measurement was made in another desk lamp, whose power in this case was of 35W. This has been done through sensor number 4 to verify that data can also be collected by that channel, and obtained values oscillate around 35W (energy expenditure around 0.035kWh for an hour of connection), although with a small margin of error between collected samples.

Future lines

Possible new implementations to provide greater functionality are:

- **System optimization**, in a way that it's printed on a plate instead of using a protoboard, and is integrated into the electricity panel of a house. In this way, the data could be accessed including in the circuit a matrix keyboard where to introduce the line to measure, and to visualize them through a digital display.

- **Creation of a web interface** which manages the monitoring of each line of the frame, so the system has internet connectivity and the information extracted by each sensor will be automatically uploaded every x time in a database.This would be treated to get graphical statistics on which appliances consume the most, what kind they are, what time zone occur in and how it affects in the final price of the light bill.

- **Create a mobile application,** so that the information managed in the web interface could also be viewed from any smartphone or tablet. In addition, a system could be developed to receive alerts and notifications about the facility's consumption.

Conclusions

A system has been developed to monitoring the consumed energy in a home automation installation, in order to information could be collected from up to 10 sensors at the same time. It has also been achieved that the data extraction was done automatically when Raspberry Pi turned on. Moreover, the program allows carrying out the two additional options, both the change of times between the collection of samples and the obtaining of data based on a channel, date and time.

In addition, another goal is that this system was simple, low cost and easy to install, which after the results is certified that it has been possible to achieve.

During the development of the project, an analysis of the related technologies has been carried out, concluding that the IoT is progressing very quickly, so it facilitates in a huge way users life. After a comparison with similar products, it has been verified that development of this system allows to monitor an electrical installation with more lines than with them, and at a lower price.

A.2. Datasheets

A.2.1. Raspberry Pi 3 Model B

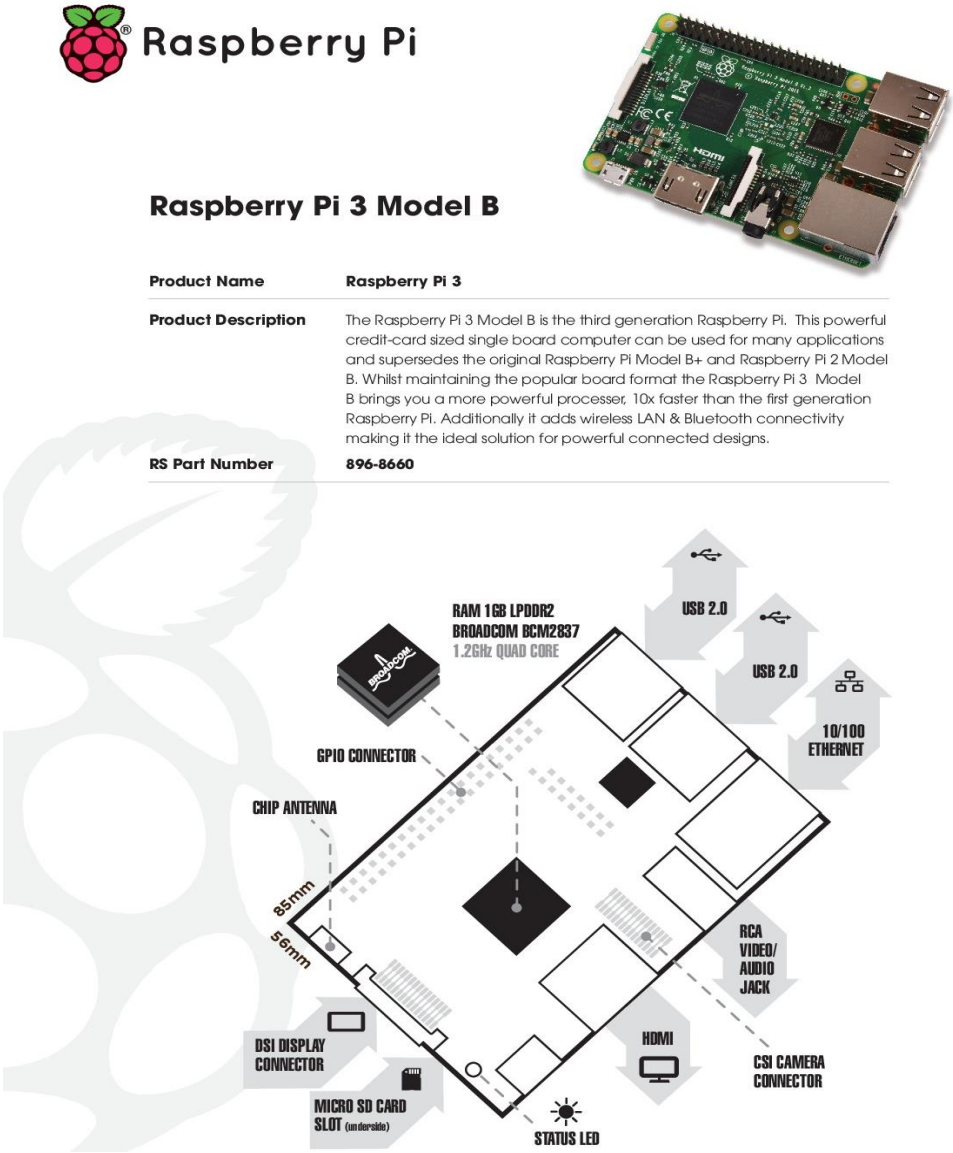


Figura 48: Datasheet Raspberry Pi 3 Model B (1). Recuperada de: http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

Raspberry Pi 3 Model B

Specifications

Processor	Broadcom BCM2387 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V1, 2.5A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4 Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

Key Benefits

- Low cost
- 10x faster processing
- Consistent board format
- Added connectivity

Key Applications

- Low cost PC/tablet/laptop
- Media centre
- Industrial/Home automation
- Print server
- Web camera
- Wireless access point
- IoT applications
- Robotics
- Server/cloud server
- Security monitoring
- Gaming
- Environmental sensing/monitoring (e.g. weather station)

Figura 49: Datasheet Raspberry Pi 3 Model B (2). Recuperada de: http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

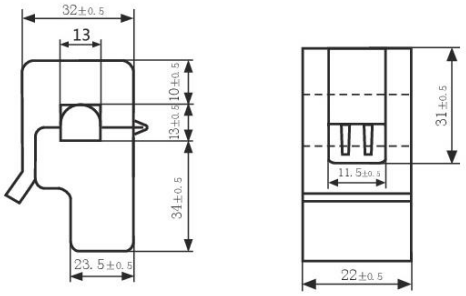
A.2.2. Sensor de corriente alterna SCT-013-030

SPECIFICATION

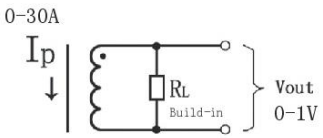
Customer Title : XiDi Technology
Product Name: Splilt-core current transformer
Manufacture Model : SCT-013-030

Charateristics: open size:13mm×13mm
1m leading wire
Core material:Ferrite
Fire resistance property:in accordance with UL 94-V0
Dielectric strength: 1500V AC/1min 5mA (between shell and output)

Outline size diagram:(in mm)



Front View Side View



Voltage output type

Schematic Diagram

Typical table of technical parameters:

input current	output voltage	non-linearity	build-in sampling resistance (RL)
0-30A	0-1V	±1%	62Ω
turn ratio	resistance grade	work temperature	dielectric strength(between shell and output)
1800:1	Grade B	-25℃～+70℃	1500V AC/1min 5mA

Customer Sign:

Beijing YaoHuadechang Electronic Co.,Ltd
Phone: 0355-7929499-803
Cell: 13693334514
Contact Name: Engineer Chen

Approve Sign:Chenjianping 2011-7-21

Figura 50: Datasheet sensor de corriente SCT-013-030. Recuperada de: http://webmeteobox.ru/docs/SCT013-030V.pdf

A.2.3. Conversor A/D MCP3008

Únicamente se han incluido las páginas más determinantes debido a la gran capacidad del archivo (40 hojas).

MICROCHIP

MCP3004/3008

2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface

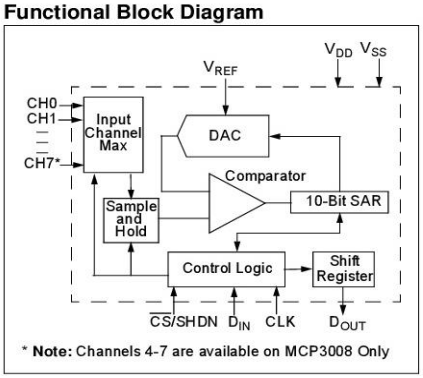
Features

- 10-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL
- 4 (MCP3004) or 8 (MCP3008) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 200 ksps max. sampling rate at V_{DD} = 5V
- 75 ksps max. sampling rate at V_{DD} = 2.7V
- Low power CMOS technology
- 5 nA typical standby current, 2 µA max.
- 500 µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Functional Block Diagram



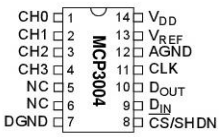
* **Note:** Channels 4-7 are available on MCP3008 Only

Description

The Microchip Technology Inc. MCP3004/3008 devices are successive approximation 10-bit Analog-to-Digital (A/D) converters with on-board sample and hold circuitry. The MCP3004 is programmable to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3008 is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. Differential Nonlinearity (DNL) and Integral Nonlinearity (INL) are specified at ±1 LSB. Communication with the devices is accomplished using a simple serial interface compatible with the SPI protocol. The devices are capable of conversion rates of up to 200 ksps. The MCP3004/3008 devices operate over a broad voltage range (2.7V - 5.5V). Low-current design permits operation with typical standby currents of only 5 nA and typical active currents of 320 µA. The MCP3004 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages, while the MCP3008 is offered in 16-pin PDIP and SOIC packages.

Package Types

PDIP, SOIC, TSSOP



PDIP, SOIC

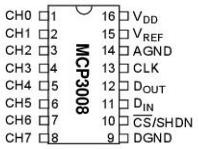


Figura 51: Datasheet MCP3008 (1). Recuperada de: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD} 7.0V
All Inputs and Outputs w.r.t. V_{SS} .....– 0.6V to V_{DD} + 0.6V
Storage Temperature .....–65°C to +150°C
Ambient temperature with power applied.....–65°C to +150°C
Soldering temperature of leads (10 seconds) ..... +300°C
ESD Protection On All Pins (HBM) ..... ≥ 4 kV

† Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 200 ksps and f_{CLK} = 18*f_{SAMPLE}. Unless otherwise noted, typical values apply for V_{DD} = 5V, T_A = +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Conversion Rate						
Conversion Time	t_{CONV}	—	—	10	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}	—	—	200 75	ksps ksps	V_{DD} = V_{REF} = 5V V_{DD} = V_{REF} = 2.7V
DC Accuracy						
Resolution			10		bits	
Integral Nonlinearity	INL	—	±0.5	±1	LSB	
Differential Nonlinearity	DNL	—	±0.25	±1	LSB	No missing codes over temperature
Offset Error		—	—	±1.5	LSB	
Gain Error		—	—	±1.0	LSB	
Dynamic Performance						
Total Harmonic Distortion		—	-76		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Signal-to-Noise and Distortion (SINAD)		—	61		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Spurious Free Dynamic Range		—	78		dB	V_{IN} = 0.1V to 4.9V@1 kHz
Reference Input						
Voltage Range		0.25	—	V_{DD}	V	Note 2
Current Drain		—	100 0.001	150 3	µA µA	 $\overline{CS}$ = V_{DD} = 5V
Analog Inputs						
Input Voltage Range for CH0 or CH1 in Single-Ended Mode		V_{SS}	—	V_{REF}	V	
Input Voltage Range for IN+ in pseudo-differential mode		IN-	—	V_{REF}+IN-		
Input Voltage Range for IN- in pseudo-differential mode		V_{SS}-100	—	V_{SS}+100	mV	

Note 1: This parameter is established by characterization and not 100% tested.
2: See graphs that relate linearity performance to V_{REF} levels.
3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for more information.

Figura 52: Datasheet MCP3008 (2). Recuperada de: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

MCP3004/3008

ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 200 ksps and f_{CLK} = 18*f_{SAMPLE}. Unless otherwise noted, typical values apply for V_{DD} = 5V, T_A = +25°C.

Parameter	Sym	Min	Typ	Max	Units	Conditions
Leakage Current		—	0.001	±1	µA	
Switch Resistance		—	1000	—	Ω	See Figure 4-1
Sample Capacitor		—	20	—	pF	See Figure 4-1
Digital Input/Output						
Data Coding Format		Straight Binary				
High Level Input Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low Level Input Voltage	V_{IL}		—	0.3 V_{DD}	V	
High Level Output Voltage	V_{OH}	4.1	—	—	V	I_{OH} = -1 mA, V_{DD} = 4.5V
Low Level Output Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 1 mA, V_{DD} = 4.5V
Input Leakage Current	I_{LI}	-10	—	10	µA	V_{IN} = V_{SS} or V_{DD}
Output Leakage Current	I_{LO}	-10	—	10	µA	V_{OUT} = V_{SS} or V_{DD}
Pin Capacitance (All Inputs/Outputs)	C_{IN}, C_{OUT}	—	—	10	pF	V_{DD} = 5.0V (**Note 1**) T_A = 25°C, f = 1 MHz
Timing Parameters						
Clock Frequency	f_{CLK}	—	—	3.6 1.35	MHz MHz	V_{DD} = 5V (**Note 3**) V_{DD} = 2.7V (**Note 3**)
Clock High Time	t_{HI}	125	—	—	ns	
Clock Low Time	t_{LO}	125	—	—	ns	
$\overline{CS}$ Fall To First Rising CLK Edge	t_{SUCS}	100	—	—	ns	
$\overline{CS}$ Fall To Falling CLK Edge	t_{CSD}	—	—	0	ns	
Data Input Setup Time	t_{SU}	50	—	—	ns	
Data Input Hold Time	t_{HD}	50	—	—	ns	
CLK Fall To Output Data Valid	t_{DO}	—	—	125 200	ns ns	V_{DD} = 5V, See Figure 1-2 V_{DD} = 2.7V, See Figure 1-2
CLK Fall To Output Enable	t_{EN}	—	—	125 200	ns ns	V_{DD} = 5V, See Figure 1-2 V_{DD} = 2.7V, See Figure 1-2
$\overline{CS}$ Rise To Output Disable	t_{DIS}	—	—	100	ns	See Test Circuits, Figure 1-2
$\overline{CS}$ Disable Time	t_{CSH}	270	—	—	ns	
D_{OUT} Rise Time	t_R	—	—	100	ns	See Test Circuits, Figure 1-2 (**Note 1**)
D_{OUT} Fall Time	t_F	—	—	100	ns	See Test Circuits, Figure 1-2 (**Note 1**)

Note 1: This parameter is established by characterization and not 100% tested.
2: See graphs that relate linearity performance to V_{REF} levels.
3: Because the sample cap will eventually lose charge, effective clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for more information.

Figura 53: Datasheet MCP3008 (3). Recuperada de: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

3.0 PIN DESCRIPTIONS

The descriptions of the pins are listed in Table 3-1. Additional descriptions of the device pins follows.

TABLE 3-1: PIN FUNCTION TABLE

MCP3004	MCP3008	Symbol	Description
PDIP, SOIC, TSSOP	PDIP, SOIC		
1	1	CH0	Analog Input
2	2	CH1	Analog Input
3	3	CH2	Analog Input
4	4	CH3	Analog Input
–	5	CH4	Analog Input
–	6	CH5	Analog Input
–	7	CH6	Analog Input
–	8	CH7	Analog Input
7	9	DGND	Digital Ground
8	10	$\overline{CS}$/SHDN	Chip Select/Shutdown Input
9	11	D_{IN}	Serial Data In
10	12	D_{OUT}	Serial Data Out
11	13	CLK	Serial Clock
12	14	AGND	Analog Ground
13	15	V_{REF}	Reference Voltage Input
14	16	V_{DD}	+2.7V to 5.5V Power Supply
5,6	–	NC	No Connection

3.1 Digital Ground (DGND)

Digital ground connection to internal digital circuitry.

3.2 Analog Ground (AGND)

Analog ground connection to internal analog circuitry.

3.3 Analog inputs (CH0 - CH7)

Analog inputs for channels 0 - 7, respectively, for the multiplexed inputs. Each pair of channels can be programmed to be used as two independent channels in single-ended mode or as a single pseudo-differential input where one channel is IN+ and one channel is IN-. See **Section 4.1 "Analog Inputs"**, "Analog Inputs", and **Section 5.0 "Serial Communication"**, "Serial Communication", for information on programming the channel configuration.

3.4 Serial Clock (CLK)

The SPI clock pin is used to initiate a conversion and clock out each bit of the conversion as it takes place. See **Section 6.2 "Maintaining Minimum Clock Speed"**, "Maintaining Minimum Clock Speed", for constraints on clock speed.

3.5 Serial Data Input (D_{IN})

The SPI port serial data input pin is used to load channel configuration data into the device.

3.6 Serial Data Output (D_{OUT})

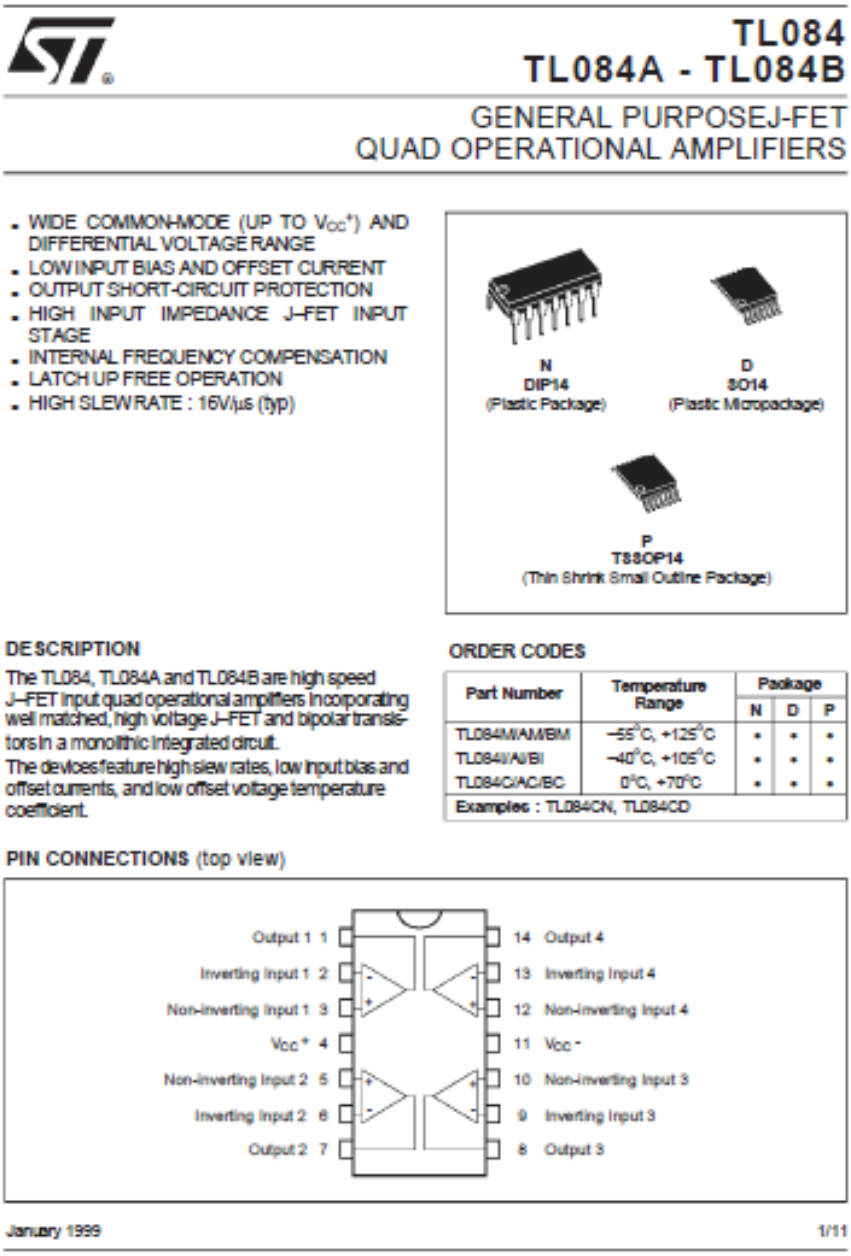
The SPI serial data output pin is used to shift out the results of the A/D conversion. Data will always change on the falling edge of each clock as the conversion takes place.

3.7 Chip Select/Shutdown ($\overline{CS}$/SHDN)

The $\overline{CS}$/SHDN pin is used to initiate communication with the device when pulled low. When pulled high, it will end a conversion and put the device in low-power standby. The $\overline{CS}$/SHDN pin must be pulled high between conversions.

Figura 54: Datasheet MCP3008 (4). Recuperada de: https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf

A.2.4. Amplificador TL084



TL084
TL084A - TL084B

GENERAL PURPOSEJ-FET
QUAD OPERATIONAL AMPLIFIERS

- WIDE COMMON-MODE (UP TO V_{CC}^+) AND DIFFERENTIAL VOLTAGE RANGE
- LOW INPUT BIAS AND OFFSET CURRENT
- OUTPUT SHORT-CIRCUIT PROTECTION
- HIGH INPUT IMPEDANCE J-FET INPUT STAGE
- INTERNAL FREQUENCY COMPENSATION
- LATCH UP FREE OPERATION
- HIGH SLEW RATE : 16V/µs (typ)

N
DIP14
(Plastic Package)

D
SO14
(Plastic Micropackage)

P
TSSOP14
(Thin Shrink Small Outline Package)

DESCRIPTION

The TL084, TL084A and TL084B are high speed J-FET input quad operational amplifiers incorporating well matched, high voltage J-FET and bipolar transistors in a monolithic integrated circuit.
The devices feature high slew rates, low input bias and offset currents, and low offset voltage temperature coefficient.

ORDER CODES

Part Number	Temperature Range	Package		
		N	D	P
TL084M/AM/BM	−55°C, +125°C	•	•	•
TL084I/AI/BI	−40°C, +105°C	•	•	•
TL084C/AC/BC	0°C, +70°C	•	•	•

Examples : TL084CN, TL084CD

PIN CONNECTIONS (top view)

January 1999 1/11

Figura 55: Datasheet TL084. Recuperada de: http://www.datasheet.hk/view_download.php?id=1093268&file=0031\tl084_249680.pdf

A.3. Manual de usuario

Tras la realización del sistema completo, en este apartado se mostrará el procedimiento a seguir para utilizarlo y obtener así información de cualquiera de los sensores.

En primer lugar, hay que cerciorarse de que todo el hardware está conectado a la Raspberry Pi: tarjeta microSD, fuente de alimentación, cable de red Ethernet y bus GPIO con el circuito completo.

A continuación se activará el control remoto desde el ordenador a través del programa PuTTY, introduciendo la IP de la Raspberry Pi, así como los datos de usuario: pi / raspberry. Tras ello, se activará la interfaz gráfica con el comando tight*vncserver –geometry 1280x640 -depth 16*, y dentro del programa VNC Viewer se introducirá la IP anterior.

Una vez que el usuario se encuentre en el escritorio principal, existen dos opciones para manejar el sistema: por un lado, se puede cambiar el tiempo entre muestras para la recogida de datos en un canal específico, mediante el comando *python /home/pi/python-spi/script_conf.py canal tiempo,* donde se debe introducir como argumentos un canal entre el 1 y 10, y el tiempo de recogida entre cada muestra en ms (tiempo). De esta forma, el fichero en el que se encuentran los tiempos para cada canal (fichero_conf.txt) cambia a los valores introducidos por línea de comandos siempre que los valores introducidos sean válidos.

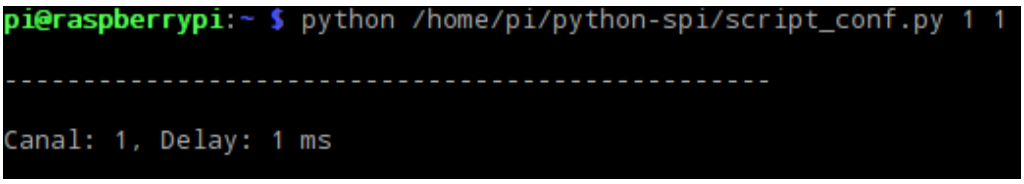


Figura 56: Cambio de tiempo entre muestras.

```
Canal 1: 1
Canal 2: 10
Canal 3: 10
Canal 4: 10
Canal 5: 10
Canal 6: 10
Canal 7: 10
Canal 8: 10
Canal 9: 10
Canal 10: 10
```

Figura 57: fichero_conf.txt modificado.

Por otra parte, es posible obtener los datos para un canal específico a partir de una fecha y hora determinada con el comando *python /home/pi/python-spi/get_data.py canal fecha hora*, donde es necesario introducir por argumentos el canal (entre el 1 y 10), y fecha y hora a partir de los cuales se desea obtener datos (deben ser anteriores a la fecha de uso). Si los argumentos introducidos son válidos, aparecerá un mensaje de confirmación.

```
pi@raspberrypi:~ $ python /home/pi/python-spi/get_data.py 1 23/09/17 21:48:00
Fecha y hora correctas
------------------------------------------------
Obteniendo datos del canal 1 a partir de: 23/09/17 21:48:00
```

Figura 58: Obtención de datos a partir de un canal, fecha y hora.

Por último, los datos que se desea obtener serán guardados en el fichero datos_fecha.txt

```
23/09/17 21:48:00  Canal 1: 0.096A 22.08W 0.02208kWh
23/09/17 21:48:01  Canal 1: 0.102A 23.46W 0.02346kWh
23/09/17 21:48:02  Canal 1: 0.086A 19.78W 0.01978kWh
23/09/17 21:48:03  Canal 1: 0.091A 20.93W 0.02093kWh
23/09/17 21:48:04  Canal 1: 0.089A 20.47W 0.02047kWh
23/09/17 21:48:05  Canal 1: 0.089A 20.47W 0.02047kWh
23/09/17 21:48:06  Canal 1: 0.095A 21.85W 0.02185kWh
23/09/17 21:48:07  Canal 1: 0.093A 21.39W 0.02139kWh
23/09/17 21:48:08  Canal 1: 0.091A 20.39W 0.02039kWh
23/09/17 21:48:09  Canal 1: 0.091A 20.39W 0.02039kWh
23/09/17 21:48:10  Canal 1: 0.088A 20.24W 0.02024kWh
23/09/17 21:48:11  Canal 1: 0.091A 20.39W 0.02039kWh
23/09/17 21:48:12  Canal 1: 0.095A 21.85W 0.02185kWh
23/09/17 21:48:13  Canal 1: 0.092A 21.16W 0.02116kWh
23/09/17 21:41:14  Canal 1: 0.089A 20.47W 0.02047kWh
23/09/17 21:41:15  Canal 1: 0.095A 21.85W 0.02185kWh
23/09/17 21:41:16  Canal 1: 0.092A 21.16W 0.02116kWh
```

Figura 59: Datos extraídos en fichero datos_fecha.txt.