Trabajo Fin de Grado
Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de un sistema de monitorización remoto de parámetros de la red eléctrica, basado en web y herramientas open source

Autor: Araceli García Granados

Tutor: Juan Antonio Sánchez Segura

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Ingeniería de las Tecnologías de Telecomunicación

Diseño y desarrollo de un sistema de monitorización remoto de parámetros de la red eléctrica, basado en web y herramientas open source

Autor:
Araceli García Granados

Tutor:
Juan Antonio Sánchez Segura
Profesor contratado

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Agradecimientos

En primer lugar desearía agradecer al profesor Juan Antonio Sánchez Segura por su disposición a dirigirme este trabajo, cuya idea surgió a raíz de mi colaboración como becaria en el departamento de calidad de la red eléctrica de Endesa.

En segundo lugar a mi familia por su apoyo incondicional en los momentos difíciles tanto en la elaboración de este trabajo como durante toda mi carrera.

En tercer lugar a todos los profesores de esta escuela por los conocimientos que me han transmitido.

Abstract

The target of this project is the design and implementation of a monophasic power network analyzer for home use. It gets the waves of tension, current and power. From these values it calculates the most important parameters of these waves as well as the power factor and armonics of the voltage and the current.

The device is based on an Open Hardware controller (Raspberry pi and Arduino) and control system will be programmed by using an Open Source code. The prototype will be configured as a web server, available anywhere on the Internet (be it a private home IP, or a public IP). Therefore, the home power network parameters will be thus available remotely. As a measurement instrument, it will be subjected to testing to know the precision and measurement error.

Índice general

Índice de figuras

Capítulo 0: Introducción

Introducción

La energía eléctrica la utilizamos a diario en multitud de actividades formando parte de nuestra vida diaria, siendo una forma de energía que se manifiesta cuando dos puntos, a diferente potencia, se unen mediante un material conductor estableciéndose una corriente eléctrica, y pudiéndose transformar en cualquier otro tipo de energía. Por tanto en este proceso interviene el movimiento de electrones en el interior de materiales conductores. Para posibilitar este traslado de cargas se requieren varios elementos, desde el material conductor por donde se moverán las cargas, hasta un generador que hará circular los electrones.

La ley de la conservación de la energía afirma que la energía no se crea ni se destruye, sólo cambia de una forma a otra, siendo la energía eléctrica una manifestación de la misma. La transformación se realiza en las centrales de centrales eléctricas siendo la energía primaria, materia prima, cualquier otra manifestación de la energía. Tradicionalmente se ha utilizado como materia prima las energías no renovables, como los combustibles fósiles, aunque cada vez son más habituales tipos de energías sostenibles.

El elemento que realiza la transformación de energía a nivel industrial es el generador eléctrico. Una vez generada la energía eléctrica debemos de trasladarla mediante líneas de alta tensión hasta los puntos de distribución (subestaciones eléctricas), siendo estos los que la acercan hasta los puntos de consumo, es decir, hogares, oficinas, industrias, etc.

La forma en que se manifiesta la corriente eléctrica puede ser continua, generada por ejemplo en las baterías, y alterna, producida por ejemplo en los generadores de las centrales eléctricas. La onda que generan las empresas eléctricas en Europa es idealmente sinusoidal con un periodo de 50 hercios y una amplitud nominal de 230V, con un margen del ± 10%, es decir de 207V a 253V.

Los usuarios residenciales, comerciales e industriales disponen de una gran cantidad de equipos eléctricos no lineales, como hornos microondas, computadoras, TV, etc., que contribuyen a la generación de armónicos, distorsionando la onda de tensión dejando de ser perfectamente sinusoidal, tal como se muestra en la siguiente figura.

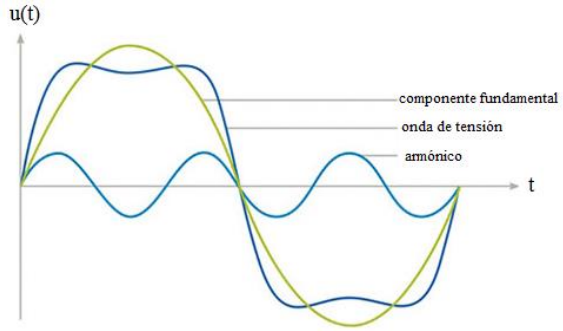


Figura 1. Ondas ideal y distorsionada

Objetivo del trabajo

Este trabajo tiene como objetivo el diseño y ejecución de un dispositivo capaz de realizar mediciones remotas de los principales parámetros de la red eléctrica monofásica de baja tensión (230VAC), usando elementos hardware y software de tipo "open source".

Especificación de requisitos que debe cumplir el prototipo:

1. *Visualización de las siguientes gráficas:*

   Tensión de la red donde se conecte la carga.
   Corriente que circula por la carga.
   Potencia instantánea consumida por la carga.
   Componente fundamental de la onda de tensión.
   Componente fundamental de la onda de corriente.

2. *Cálculo de los siguientes parámetros:*

   Componentes armónicos de la onda de tensión.
   Componentes armónicos de la onda de corriente.
   Distorsión armónica de la onda de tensión.
   Distorsión armónica de la onda de corriente.
   Potencias activa, reactiva, de distorsión y aparente.
   Factor de potencia y $\cos\varphi$ *.*

3. *Visualización de resultados vía web.*

4. *Acceso a Internet vía 3G.*

5. *Medición de cargas de hasta 1300W, escalable a mayores potencias.*

6. *Conexión y desconexión remota de la carga.*

7. *Error en las medidas inferior al 1%.*

8. *Coeficiente de variación de las medidas inferior al 1%.*

9. *Coste en materiales del prototipo inferior a 150€.*

En este trabajo se describirá cada uno de los elementos hardware y software que componen el dispositivo, así como su interconexión y configuración detallada, desde que se adquieren en el mercado hasta que quedan completamente integrados en el sistema.

Estructura del documento

Este documento se estructura en seis capítulos:

Capítulo 1: Base teórica

Se detallan los fundamentos teóricos sobre los que se apoya este trabajo.

Capítulo 2: Diseño

Se describe la construcción del dispositivo y su puesta en funcionamiento.

Capítulo 3: Manual de uso y calibración

En este capítulo se explicará la forma de utilizar el dispositivo.

Capítulo 4: Medidas

En este capítulo se realizarán diversos ensayos destinados a verificar la exactitud y precisión del dispositivo.

Capítulo 5: Conclusiones y mejoras

Se determinará si el dispositivo cumple los objetivos que nos planteamos así como los puntos de mejora.

Capítulo 6: Presupuesto

En este capítulo se realizará un cálculo del coste del dispositivo.

Bibliografía.

Anexo.

Se incluye el código de los programas C++ para el Arduino, Raspberry, y Matlab

Capítulo 1: Base teórica

Introducción

Este capítulo contiene la base teórica sobre la que se apoya el cálculo de los parámetros de una red eléctrica. Concretamente explicaremos cómo vamos a obtener, a partir de las muestras de las ondas de tensión y de corriente, tanto la potencia media, factor de potencia, $\cos\varphi$, como las componentes armónicas de dichas señales y sus distorsiones THD.

Cálculo de potencias

Sea el circuito mostrado en la figura, donde conectamos una fuente de tensión a una determinada carga:

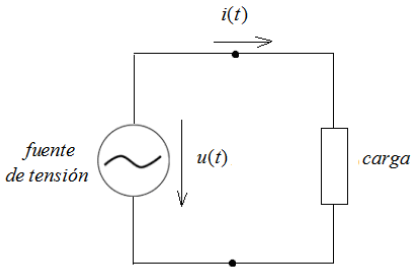


Figura 2. Circuito eléctrico

Se denomina potencia (instantánea) a la variación de la energía en la unidad de tiempo:

$$p(t) = \frac{dE(t)}{dt}$$

La tensión aplicada, $v(t)$, y la corriente de consumo, $i(t)$, se miden en los puntos indicados.

La potencia instantánea también la podemos determinar mediante el producto de ambas variables:

$$p(t) = v(t)i(t)$$

La carga absorberá potencia cuando $p(t) > 0$ y la cederá a la fuente cuando $p(t) < 0$.
Si $p(t)$ es periódica, se define la potencia activa como el valor medio de la potencia instantánea en un periodo, es decir:

$$P = \frac{1}{T}\int_{t_o}^{t_o+T} v(t)i(t)dt \text{ , para cualquier } t_o$$

Cargas lineales

Cuando se aplica una tensión sinusoidal directamente a cargas como resistencias, inductancias o condensadores (o una combinación de ellos), se genera una corriente eléctrica también sinusoidal. A estas cargas se les denomina lineales.

$$v(t) = \sqrt{2}V\text{seno}(\omega t + \varphi_v)$$
$$i(t) = \sqrt{2}I\text{seno}(\omega t + \varphi_i)$$

El valor eficaz de la tensión será el valor cuadrático medio de dicha tensión:

$$V_{ef} = \sqrt{\frac{1}{T}\int_{t_o}^{t_o+T} v^2(t)dt} = \sqrt{\frac{1}{T}\int_{t_o}^{t_o+T} 2V^2 sen^2(\omega t)dt} = \sqrt{\frac{2V^2}{T}\int_{t_o}^{t_o+T} \frac{1-\cos(2\omega t)}{2}dt} = \sqrt{\frac{V^2}{T}\left[\int_{t_o}^{t_o+T} dt - \int_{t_o}^{t_o+T} \cos(2\omega t)dt\right]} =$$

$$= \sqrt{\frac{V^2}{T}\left[T - \left|\frac{sen(2\omega t)}{2\omega}\right|_{t_o}^{t_o+T}\right]} = \sqrt{\frac{V^2}{T}\left[T - \frac{sen(2\omega t_o + 2\omega T)}{2\omega} + \frac{sen(2\omega t_o)}{2\omega}\right]} =$$

$$= \sqrt{\frac{V^2}{T}\left[T - \frac{sen(2\omega t_o + 4\pi)}{2\omega} + \frac{sen(2\omega t_o)}{2\omega}\right]} = \sqrt{\frac{V^2}{T}\left[T - \frac{sen(2\omega t_o)}{2\omega} + \frac{sen(2\omega t_o)}{2\omega}\right]} = \sqrt{\frac{V^2}{T}T} = V$$

Igualmente obtendríamos que $I_{ef} = I$

La corriente instantánea es:

$$p(t) = v(t)i(t) = 2VIseno(\omega t + \varphi_v)seno(\omega t + \varphi_i) = VI\left[\cos(\varphi_v - \varphi_i) - \cos(2\omega t + \varphi_v + \varphi_i)\right]$$

Observamos que la potencia instantánea oscila con una frecuencia doble del de la tensión. Hallemos su valor medio, es decir, la potencia activa:

$$P = \frac{1}{T}\int_{t_o}^{t_o+T} VI\left[\cos(\varphi_v - \varphi_i) - \cos(2\omega t + \varphi_v + \varphi_i)\right]dt = \frac{VI}{T}\int_{t_o}^{t_o+T} \cos(\varphi_v - \varphi_i)dt - \frac{VI}{T}\int_{t_o}^{t_o+T} \cos(2\omega t + \varphi_v + \varphi_i)dt =$$

$$= VI\cos(\varphi_v - \varphi_i) - \frac{VI}{T}\left|\frac{sen(2\omega t + \varphi_v + \varphi_i)}{2\omega}\right|_{t_o}^{t_o+T} = VI\cos(\varphi_v - \varphi_i) -$$

$$-\frac{VI}{2\omega T}\left[sen(2\omega(t_o + T) + \varphi_v + \varphi_i) - sen(2\omega t_o + \varphi_v + \varphi_i)\right] = VI\cos(\varphi_v - \varphi_i) -$$

$$-\frac{VI}{2\omega T}\left[sen(2\omega t_o + \varphi_v + \varphi_i + 4\pi) - sen(2\omega t_o + \varphi_v + \varphi_i)\right] = VI\cos(\varphi_v - \varphi_i) = VI\cos\varphi$$

, siendo φ *el desfase entre la tensión y la corriente.*

Por tanto obtenemos que la potencia activa toma el valor de $P = VI\cos\varphi$

$$\boxed{\cos\varphi = \frac{P}{VI}}$$

La potencia aparente se define como $\boxed{S = VI}$

La potencia reactiva la determinamos mediante el siguiente triángulo de potencias:

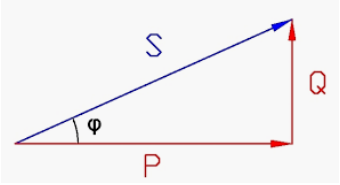


Figura 3. Triángulo de potencias

$$Q = VIseno\varphi = \sqrt{S^2 - P^2}$$

Al ratio entre la potencia activa y la aparente se denomina factor de potencia, y para el caso que estamos viendo de señales sinusoidales , se cumple:

$\frac{P}{S} = \frac{VI\cos\varphi}{VI} = \cos\varphi$, *es decir, el factor de potencia coincide con el valor* $\cos\varphi$. *Veremos posteriormente que esto no ocurre cuando las cargas no son lineales.*

El factor de potencia se utiliza para evaluar la conversión de energía y se encuentra entre los valores de 0 y 1. Para un valor de $\cos\varphi = 0$ *no se transfiera potencia activa. Si el factor de potencia vale 1 solo se suministra potencia activa a la carga.*

Cargas no lineales

Consideremos que la tensión sigue siendo aproximadamente sinusoidal, es decir:

$$v(t) = \sqrt{2}Vseno(\omega t + \varphi_v)$$

, pero ahora la corriente aún siendo periódica presenta una componente continua además de una serie de armónicos, es decir, componentes cuya frecuencia son un múltiplo entero de la frecuencia fundamental. Según el teorema de las series de Fourier, la onda de corriente se puede descomponer en una suma de señales de frecuencia múltiplo de la fundamental, es decir:

$$i(t) = I_o + \sqrt{2}I_1 seno(\omega t + \varphi_1) + \sqrt{2}I_2 seno(2\omega t + \varphi_2) + \sqrt{2}I_3 seno(3\omega t + \varphi_3) + ...$$

Hallemos la corriente eficaz, la raíz cuadrática media de $i(t)$ *:*

$$I^2 = \frac{1}{T}\int_{t_o}^{t_o+T} i^2(t)dt = \frac{1}{T}\int_{t_o}^{t_o+T} \left[I_o + \sqrt{2}I_1 seno(\omega t + \varphi_1) + \sqrt{2}I_2 seno(2\omega t + \varphi_2) + \sqrt{2}I_3 seno(3\omega t + \varphi_3) + ...\right]^2 dt$$

, donde $\int_{t_o}^{t_o+T} I_o\sqrt{2}I_i seno(i\omega t + \varphi_i)dt = \sqrt{2}I_o I_i \left|\frac{-\cos(i\omega t + \varphi_i)}{i\omega}\right|_{t=t_o}^{t=t_o+T} = 0$

Determinemos el valor de $\int_{t_o}^{t_o+T} seno(i\omega t + \varphi_i)seno(j\omega t + \varphi_j)dt$ *cuando* $i \neq j$ *:*

$$seno(i\omega t + \varphi_i)seno(j\omega t + \varphi_j) = \frac{1}{2}\cos[(i-j)\omega t + \varphi_i - \varphi_i] - \frac{1}{2}\cos[(i+j)\omega t + \varphi_i + \varphi_i] \rightarrow$$

$$\int_{t_o}^{t_o+T} seno(i\omega t + \varphi_i)seno(j\omega t + \varphi_j)dt = \frac{1}{2}\int_{t_o}^{t_o+T} \cos[(i-j)\omega t + \varphi_i - \varphi_i]dt - \frac{1}{2}\int_{t_o}^{t_o+T} \cos[(i+j)\omega t + \varphi_i + \varphi_i]dt$$

, teniendo en cuenta que $\frac{1}{2}\int_{t_o}^{t_o+T} \cos[n\omega t + \varphi_i - \varphi_i]dt = 0$ *para* $n \neq 0$*, entonces:*

$$\int_{t_o}^{t_o+T} seno(i\omega t+\varphi_i)seno(j\omega t+\varphi_j)dt=0$$

, por tanto,

$$I^2=\frac{1}{T}\int_{t_o}^{t_o+T} i^2(t)dt=\frac{1}{T}\int_{t_o}^{t_o+T}[I_0^2+2I_1^2seno^2(\omega t+\varphi_1)+2I_2^2seno^2(2\omega t+\varphi_2)+2I_3^2seno^2(3\omega t+\varphi_3)+...]dt=$$

$$=I_o^2+\frac{2I_1^2}{T}\int_{t_o}^{t_o+T} seno^2(\omega t+\varphi_1)dt+\frac{2I_2^2}{T}\int_{t_o}^{t_o+T} seno^2(2\omega t+\varphi_2)dt+\frac{2I_3^2}{T}\int_{t_o}^{t_o+T} seno^2(3\omega t+\varphi_3)dt+...=$$

$$=I_o^2+\frac{2I_1^2}{T}\int_{t_o}^{t_o+T}\frac{1-\cos(2\omega t+2\varphi_1)}{2}dt+\frac{2I_2^2}{T}\int_{t_o}^{t_o+T}\frac{1-\cos(4\omega t+2\varphi_2)}{2}dt+\frac{2I_3^2}{T}\int_{t_o}^{t_o+T}\frac{1-\cos(6\omega t+2\varphi_3)}{2}dt+...=$$

$$=I_o^2+\frac{2I_1^2}{T}\frac{T}{2}+\frac{2I_2^2}{T}\frac{T}{2}+\frac{2I_3^2}{T}\frac{T}{2}+...=I_o^2+I_1^2+I_2^2+I_3^2+...$$

$$\boxed{I=\sqrt{I_o^2+I_1^2+I_2^2+I_3^2+...}}$$

La potencia aparente es, por definición $S=VI$ *. Hallemos su valor:*

$$S=VI=V\sqrt{I_o^2+I_1^2+I_2^2+I_3^2+...}=\sqrt{(VI_o)^2+(VI_1)^2+(VI_2)^2+(VI_3)^2+...}=\sqrt{S_0^2+S_1^2+S_2^2+S_3^2+...}$$

, donde $S_i=VI_i$

La potencia activa es $P=\frac{1}{T}\int_{t_o}^{t_o+T} v(t)i(t)dt=$

$$=\frac{1}{T}\int_{t_o}^{t_o+T}\sqrt{2}Vseno(\omega t+\varphi_v)\left[I_o+\sqrt{2}I_1seno(\omega t+\varphi_1)+\sqrt{2}I_2seno(2\omega t+\varphi_2)+\sqrt{2}I_3seno(3\omega t+\varphi_3)+..\right]dt=$$

$$=\int_{t_o}^{t_o+T}\sqrt{2}VI_o seno(\omega t+\varphi_v)dt+\frac{2VI_1}{T}\int_{t_o}^{t_o+T} seno(\omega t+\varphi_v)seno(\omega t+\varphi_1)dt+\frac{2VI_2}{T}\int_{t_o}^{t_o+T} seno(\omega t+\varphi_v)seno(2\omega t+\varphi_2)dt+$$

$$+\frac{2VI_3}{T}\int_{t_o}^{t_o+T} seno(\omega t+\varphi_v)seno(3\omega t+\varphi_3)dt+...=$$

$$=0+\frac{VI_1}{T}\int_{t_o}^{t_o+T}[\cos(\varphi_v-\varphi_1)-\cos(2\omega t+\varphi_v+\varphi_1)]dt+\frac{VI_2}{T}\int_{t_o}^{t_o+T}[\cos(\omega t+\varphi_2-\varphi_v)-\cos(3\omega t+\varphi_v+\varphi_2)]dt+$$

$$+\frac{VI_3}{T}\int_{t_o}^{t_o+T}[\cos(2\omega t+\varphi_3-\varphi_v)-\cos(4\omega t+\varphi_v+\varphi_3)]dt+...=\frac{VI_1}{T}\int_{t_o}^{t_o+T}\cos(\varphi_v-\varphi_1)dt=VI_1\cos(\varphi_v-\varphi_1)=P_1$$

, donde hemos tenido en cuenta que $\int_{t_o}^{t_o+T}\cos(n\omega t-\beta)dt=0\,,\ \forall n>0$

$$\boxed{P=VI_1\cos(\varphi_v-\varphi_1)=VI_1\cos\varphi_1}$$

$\boxed{\varphi = \varphi_v - \varphi_1}$ *es el desfase entre la componente fundamental de la tensión (que coincide con la tensión al ser esta sinusoidal) y la componente fundamental de la corriente.*

El factor de potencia se define como la relación entre la potencia activa y la potencia aparente:

$$\boxed{FP = \frac{P}{S}} \rightarrow FP = \frac{VI_1 \cos\varphi}{V\sqrt{I_o^2 + I_1^2 + I_2^2 + I_3^2 + \ldots}} = \frac{I_1}{\sqrt{I_o^2 + I_1^2 + I_2^2 + I_3^2 + \ldots}} \cos\varphi < \cos\varphi$$

, por tanto el factor de potencia es inferior a $\cos\varphi$

Sólo en ausencia de armónicos (y de componente continua) se cumple: $FP = \cos\varphi$

La potencia reactiva se especifica como el producto de las componentes fundamentales de la tensión y la corriente, y el seno del ángulo que forman ambas componentes. En nuestro caso $V_1 = V$ *:*

$$\boxed{Q = V_1 I_1 sen(\varphi_v - \varphi_1) = VI_1 sen\varphi},$$

, por tanto, $P^2 + Q^2 = (VI_1 \cos\varphi)^2 + (VI_1 sen\varphi)^2 = (VI_1)^2 = S_1^2$

$$S^2 = S_0^2 + S_1^2 + S_2^2 + S_3^2 + \ldots = P^2 + Q^2 + S_0^2 + S_2^2 + S_3^2 + \ldots$$

Se denomina potencia de distorsión, D *, a* $\sqrt{S_0^2 + S_2^2 + S_3^2 + \ldots}$

, obteniendo finalmente que $\boxed{S^2 = P^2 + Q^2 + D^2}$

Estas cuatro potencias P, Q, D, S *están relacionadas ortogonalmente por lo que podemos representarlas mediante un rectángulo sólido:*

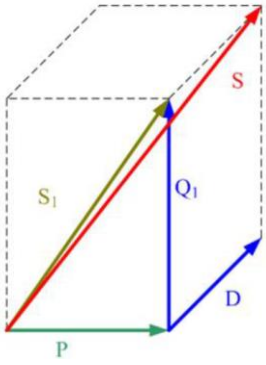


Figura 4.Paralelepipedo rectangular de potencias

Armónicos

En general los armónicos son debidos a cargas no lineales, lo que significa que su impedancia no es constante, es decir, depende de la tensión aplicada a la misma, y aunque la tensión sea sinusoidal, se obtiene una intensidad que no es sinusoidal. Podemos por tanto considerar que las cargas no lineales se comportan como fuentes de intensidad inyectando armónicos en la red eléctrica. Como ejemplo de cargas no lineales tenemos los variadores de velocidad, rectificadores, convertidores, reactancias saturables, equipos de soldadura, hornos de arco, impedancias variables con la frecuencia, filtros electrónicos, microondas, computadoras, TV, equipos de música, lámparas fluorescentes, etc.

El método más usado para caracterizar la distorsión armónica en un sistema de potencia es la distorsión armónica total, THD, pudiéndose calcular para la onda de tensión o para la onda de corriente, dependiendo de dónde se quiera medir la distorsión. En nuestro dispositivo lo mediremos tanto sobre la onda de tensión como la de corriente.

El teorema de Fourier dice que toda onda periódica $f(t)$ *de periodo* $T = \frac{1}{f} = \frac{2\pi}{\omega}$ *puede ser descompuesta como la suma de ondas sinusoidales de frecuencia "nf" mediante la aplicación de la serie de Fourier, siempre y cuando se cumplan las siguientes condiciones:*

1. *Que las integrales* $\int_0^T f(t)sen(n\omega t)dt$ *y* $\int_0^T f(t)\cos(n\omega t)dt$ *tengan valores finitos*
2. *Que la función* $f(t)$ *posea un número finito de discontinuidades en un periodo*
3. *Que la función* $f(t)$ *posea un número finito de máximos y mínimos en un periodo*

, condiciones que se cumplen tanto para la tensión como para la corriente de una red eléctrica.

Vamos a aplicar este teorema para determinar los valores eficaces de los armónicos.

Descomponiendo la tensión $v(t)$ *en series de Fourier obtenemos:*

$$v(t) = V_o + \sqrt{2}V_1 sen(\omega t + \varphi_1) + \sqrt{2}V_2 sen(2\omega t + \varphi_2) + \sqrt{2}V_3 sen(3\omega t + \varphi_3) + ...$$

, donde V_i *es la tensión eficaz de la componente armónica "i", y* V_o *la tensión media,* $\boxed{V_o = \frac{1}{T}\int_0^T v(t)dt}$

Desarrollando $v(t)$ *tendremos:*

$$v(t) = V_o + \sqrt{2}V_1 \cos\varphi_1 seno(\omega t) + \sqrt{2}V_1 sen\varphi_1 \cos(\omega t) + \sqrt{2}V_2 \cos\varphi_2 seno(2\omega t) + \sqrt{2}V_2 sen\varphi_2 \cos(2\omega t) +$$
$$+ \sqrt{2}V_3 \cos\varphi_3 seno(3\omega t) + \sqrt{2}V_3 sen\varphi_3 \cos(3\omega t) + ... =$$
$$= V_o + \sqrt{2}\left[A_1 sen(\omega t) + B_1 \cos(\omega t)\right] + \sqrt{2}\left[A_2 sen(2\omega t) + B_2 \cos(2\omega t)\right] + \sqrt{2}\left[A_3 sen(3\omega t) + B_3 \cos(3\omega t)\right] + ...$$

, donde hemos llamado

$A_n = V_n \cos\varphi_n$, y $B_n = V_n sen\varphi_n$

, por tanto,

$$A_n^2 + B_n^2 = V_n^2 \cos^2\varphi_n + V_n^2 sen^2\varphi_n = V_n^2 \rightarrow \boxed{V_n = \sqrt{A_n^2 + B_n^2}}$$

$$\frac{B_n}{A_n}=\frac{V_n sen\varphi_n}{V_n\cos\varphi_n}\rightarrow \boxed{tg\varphi_n=\frac{B_n}{A_n}}$$

Según Fourier, los coeficientes A_n *y* B_n *se determinan de la siguiente forma:*

$$\sqrt{2}A_n=\frac{2}{T}\int_0^T v(t)sen(n\omega t)dt\rightarrow \boxed{A_n=\frac{\sqrt{2}}{T}\int_0^T v(t)sen(n\omega t)dt}$$

$$\sqrt{2}B_n=\frac{2}{T}\int_0^T v(t)\cos(n\omega t)dt\rightarrow \boxed{B_n=\frac{\sqrt{2}}{T}\int_0^T v(t)\cos(n\omega t)dt}$$

Casos particulares:

1. *Para una función par, es decir, sí* $v(t)=v(-t)$ *, entonces:*

$$A_n=\frac{\sqrt{2}}{T}\int_0^T v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{T/2} v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)sen(n\omega t)dt$$

Si en $\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt$ *hacemos el cambio de variable* $u=-t$ *, entonces:*

$$A_n=\frac{\sqrt{2}}{T}\int_{T/2}^{0} v(-u)sen(-n\omega u)(-du)+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)sen(n\omega t)dt=-\frac{\sqrt{2}}{T}\int_0^{T/2} v(u)sen(n\omega u)du+$$

$$+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)sen(n\omega t)dt=0\rightarrow \boxed{A_n=0}$$

2. *Para una función impar, es decir, sí* $v(t)=-v(-t)$ *, entonces:*

$$B_n=\frac{\sqrt{2}}{T}\int_0^T v(t)\cos(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{T/2} v(t)\cos(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)\cos(n\omega t)dt+$$

$$+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)\cos(n\omega t)dt$$

Si en $\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)\cos(n\omega t)dt$ *hacemos el cambio de variable* $u=-t$ *, entonces:*

$$B_n=\frac{\sqrt{2}}{T}\int_{T/2}^{0} v(-u)\cos(-n\omega u)(-du)+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)\cos(n\omega t)dt=-\frac{\sqrt{2}}{T}\int_0^{T/2} v(u)\cos(n\omega u)du+$$

$$+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)\cos(n\omega t)dt=0\rightarrow \boxed{B_n=0}$$

$$A_n=\frac{\sqrt{2}}{T}\int_0^T v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{T/2} v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt+$$

$$+\frac{\sqrt{2}}{T}\int_0^{T/2} v(t)sen(n\omega t)dt$$

Si en $\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt$ *hacemos el cambio de variable* $u=-t$, *entonces:*

$$A_n=\frac{\sqrt{2}}{T}\int_{T/2}^{0} v(-u)sen(-n\omega u)(-du)+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)\cos(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(u)\cos(n\omega u)du+$$

$$+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt \rightarrow \boxed{A_n=\frac{2\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt}$$

Para n=par se cumplirá que $A_n=0$, *y por tanto,* $V_n=\sqrt{A_n^2+B_n^2}=0$, *es decir, los armónicos pares de una función periódica impar son iguales a cero. Como posteriormente veremos, la tensión de la red eléctrica que nos proporciona la empresa suministradora es prácticamente una función periódica impar.*

3. *Simetría de media onda, es decir,* $v(t)=-v(t-\frac{T}{2})$ *:*

$$A_n=\frac{\sqrt{2}}{T}\int_{0}^{T} v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{T/2} v(t)sen(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt+$$

$$+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt$$

Si en $\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)sen(n\omega t)dt$ *hacemos el cambio de variable* $u=t+\frac{T}{2}$, *entonces:*

$$A_n=\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(u-\frac{T}{2})sen\left[n\omega\left(u-\frac{T}{2}\right)\right]du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt=$$

$$=\frac{\sqrt{2}}{T}\int_{0}^{T/2}[-v(u)]sen(n\omega u-n\pi)du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt\text{ , y si n es par, entonces,}$$

$$A_{n\_par}=-\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(u)sen(n\omega u)du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)sen(n\omega t)dt=0 \rightarrow \boxed{A_{n\_par}=0}$$

$$B_n=\frac{\sqrt{2}}{T}\int_{0}^{T} v(t)\cos(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{T/2} v(t)\cos(n\omega t)dt=\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)\cos(n\omega t)dt+$$

$$+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)\cos(n\omega t)dt$$

Si en $\frac{\sqrt{2}}{T}\int_{-T/2}^{0} v(t)\cos(n\omega t)dt$ *hacemos el cambio de variable* $u=t+\frac{T}{2}$, *entonces:*

$$B_n=\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(u-\frac{T}{2})\cos\left[n\omega\left(u-\frac{T}{2}\right)\right]du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)\cos(n\omega t)dt=$$

$$=\frac{\sqrt{2}}{T}\int_{0}^{T/2}[-v(u)]\cos(n\omega u-n\pi)du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)\cos(n\omega t)dt\text{ , y si n es par, entonces,}$$

$$B_{n\_par}=-\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(u)\cos(n\omega u)du+\frac{\sqrt{2}}{T}\int_{0}^{T/2} v(t)\cos(n\omega t)dt=0 \rightarrow \boxed{B_{n\_par}=0}$$

En el apartado anterior vimos que el valor eficaz de una onda periódica con armónicos de valores eficaces $V_1, V_2, V_3, \ldots$ y valor medio V_o, toma el valor:

$$V = \sqrt{V_0^2 + V_1^2 + V_2^2 + V_3^2 + \ldots}$$

Se define la distorsión armónica de orden i-ésimo, HD_i, con $i > 1$, al cociente entre los niveles de tensión del armónico i-ésimo y el de la componente fundamental, en valor absoluto:

$$\boxed{HD_i = \left|\frac{V_i}{V_1}\right|}$$

Se define la distorsión armónica total, THD, como el cociente entre la tensión de la señal y la de la componente fundamental, en valor absoluto:

$$THD = \left|\frac{V}{V_1}\right| = \sqrt{\frac{V^2}{V_1^2}} = \sqrt{\frac{V_2^2 + V_3^2 + V_4^2 + \ldots}{V_1^2}} = \sqrt{\left(\frac{V_2}{V_1}\right)^2 + \left(\frac{V_3}{V_1}\right)^2 + \left(\frac{V_4}{V_1}\right)^2 + \ldots} \rightarrow$$

$$\boxed{THD = \sqrt{HD_2^2 + HD_3^2 + HD_4^2 + \ldots}}$$

Cálculo numérico de la distorsión armónica

En este apartado trataremos de resolver numéricamente las integrales definidas de los términos del desarrollo de Fourier. Primero vamos a determinar el área, en un periodo T, de una función continua $f(t)$. Para conseguir una aproximación del área dividiremos el intervalo T en "m"subintervalos, donde $\Delta t = \frac{T}{m}$, obteniendo una partición equiespaciada del periodo T.

Si $m \to \infty$, entonces se cumplirá que $\text{Área} = \int_0^T f(t)dt = \sum_{n=0}^{m-1} f(n\Delta t)\Delta t = \sum_{n=0}^{m-1} f\left(\frac{nT}{m}\right)\frac{T}{m} = \frac{T}{m}\sum_{n=0}^{m-1} f\left(\frac{nT}{m}\right)$

En general, $v(t)$ al ser periódica, podemos expresarla según el teorema de las series de Fourier:

$$v(t) = V_o + \left[\sqrt{2}A_{v1}sen(\omega t) + \sqrt{2}B_{v1}\cos(\omega t)\right] + \left[\sqrt{2}A_{v2}sen(2\omega t) + \sqrt{2}B_{v2}\cos(2\omega t)\right] + \ldots$$

, o también:

$$v(t) = V_o + \sqrt{2}V_1 sen(\omega t + \varphi_{v1}) + \sqrt{2}V_2 sen(2\omega t + \varphi_{v2}) + \sqrt{2}V_3 sen(3\omega t + \varphi_{v3}) + \ldots$$

, donde:

$$A_{vn} = \frac{\sqrt{2}}{T}\int_0^T v(t)sen(n\omega t)dt$$

$$B_{vn} = \frac{\sqrt{2}}{T}\int_0^T v(t)\cos(n\omega t)dt$$

Aproximando la integral por un sumatorio tendremos:

$$A_{vn} \approx \frac{\sqrt{2}}{T}\frac{T}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})sen\left(n\omega\frac{nT}{m}\right)$$
$$B_{vn} \approx \frac{\sqrt{2}}{T}\frac{T}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})\cos\left(n\omega\frac{nT}{m}\right)$$

$v(\frac{nT}{m})$ *es el valor de la tensión en el tiempo* $t = \frac{nT}{m}$, *es decir, es el valor de la tensión de la muestra "n", que representaremos como* $v(n)$. *Queda por tanto:*

$$\boxed{A_{vn} \approx \frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(n)sen\left(n\omega\frac{nT}{m}\right)}$$
$$\boxed{B_{vn} \approx \frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(n)\cos\left(n\omega\frac{nT}{m}\right)}$$

El valor eficaz del armónico "n" lo podremos hallar según $\boxed{V_n = \sqrt{A_{vn}^2 + B_{vn}^2}}$, *y su fase* $\boxed{tg\varphi_{vn} = \frac{B_{vn}}{A_{vn}}}$

La distorsión armónica de orden i-ésimo de la tensión será: $\boxed{HD_{vi} = \left|\frac{V_i}{V_1}\right|}$

, y la distorsión armónica total de la tensión : $\boxed{THD_v = \sqrt{HD_{v2}^2 + HD_{v3}^2 + HD_{v4}^2 + ...}}$

Igualmente haríamos con la corriente, obteniendo:

$$i(t) = V_o + \left[\sqrt{2}A_{c1}sen(\omega t) + \sqrt{2}B_{c1}\cos(\omega t)\right] + \left[\sqrt{2}A_{c2}sen(2\omega t) + \sqrt{2}B_{c2}\cos(2\omega t)\right] + ...$$
$$i(t) = I_o + \sqrt{2}I_1 sen(\omega t + \varphi_{c1}) + \sqrt{2}I_2 sen(2\omega t + \varphi_{c2}) + \sqrt{2}I_3 sen(3\omega t + \varphi_{c3}) + ...$$

$$\boxed{A_{cn} \approx \frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} i(n)sen\left(n\omega\frac{nT}{m}\right)}$$
$$\boxed{B_{cn} \approx \frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} i(n)\cos\left(n\omega\frac{nT}{m}\right)}$$
$$\boxed{I_n = \sqrt{A_{cn}^2 + B_{cn}^2}}$$
$$\boxed{tg\varphi_{cn} = \frac{B_{cn}}{A_{cn}}}$$

$$HD_{ic} = \left|\frac{I_i}{I_1}\right|$$

$$THD_c = \sqrt{HD_{c2}^2 + HD_{c3}^2 + HD_{c4}^2 + ...}$$

Cálculo numérico del factor de potencia

Aplicando el mismo concepto de integral definido en el apartado anterior, podemos calcular la potencia activa como:

Potencia activa, $P = \frac{1}{T}\int_t^{t+T} v(t)i(t)dt = \frac{1}{T}\int_t^{t+T} p(t)dt \approx \frac{1}{T}\frac{T}{m}\sum_{n=0}^{m-1} p\left(\frac{nT}{m}\right) \rightarrow \boxed{P = \frac{1}{m}\sum_{n=0}^{m-1} p(n)}$

, siendo $p\left(\frac{nT}{m}\right)$ *la potencia en el instante* $\frac{nT}{m}$*, es decir, la potencia de la muestra "n", que representaremos como* $p(n)$

Corriente eficaz, $I_{ef} = \sqrt{\frac{1}{T}\int_t^{t+T} i^2(t)dt} \approx \sqrt{\frac{1}{T}\frac{T}{m}\sum_{n=0}^{m-1} i^2\left(\frac{nT}{m}\right)} \rightarrow \boxed{I_{ef} = \sqrt{\frac{1}{m}\sum_{n=0}^{m-1} i^2(n)}}$

Tensión eficaz, $V_{ef} = \sqrt{\frac{1}{T}\int_t^{t+T} v^2(t)dt} \approx \sqrt{\frac{1}{T}\sum_{n=0}^{m-1} v^2\left(\frac{nT}{m}\right)\frac{T}{m}} \rightarrow \boxed{V_{ef} = \sqrt{\frac{1}{m}\sum_{n=0}^{m-1} v^2(n)}}$

El desfase entre las componentes fundamentales de la tensión y la corriente es:

$$\varphi = \varphi_{v1} - \varphi_{c1}$$

$$\cos\varphi = \cos(\varphi_{v1} - \varphi_{c1})$$

Potencia aparente, $S = V_{ef} I_{ef}$

Potencia aparente de la componente fundamental, $S_1 = V_{ef} I_{c1}$

Potencia activa, $P = P_1 = S_1 \cos\varphi$

Potencia reactiva, $Q = Q_1 = \sqrt{S_1^2 - P^2}$

Potencia reactiva de distorsión, $D = \sqrt{S^2 - P^2 - Q^2}$

Factor de potencia, $FDP = \frac{P}{S}$

Capítulo 2: Diseño


UNIVERSIDAD DE SEVILLA

Introducción

En este apartado se enumeran los elementos que componen el dispositivo, y en los siguientes se realiza un análisis detallado de cada uno de ellos.

El prototipo se compone de:

1. *Un transformador con una relación 20.4:1, es decir, que reduzca la tensión alterna de red:*

   $$v_i(t) = 230\sqrt{2}seno(2\pi 50t)$$

   , a una tensión de salida:

   $$v_o(t) = \frac{v_i(t)}{20.4} = 11.3\sqrt{2}seno(2\pi 50t)$$

   Figura 5: Transformador reductor

   Un circuito que acondicione la salida del transformador:

   $$11.3\sqrt{2}seno(2\pi 50t)$$

   , a una tensión compatible con las lecturas analógicas de la placa Arduino DUE:

   $$1.65 + 1.35seno(2\pi 50t)$$

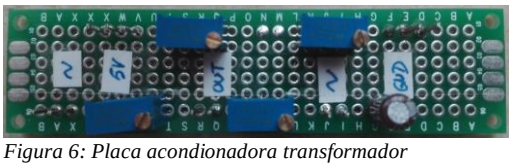
   

   Figura 6: Placa acondionadora transformador

   El transformador junto con el circuito acondicionador constituirá el sensor de tensión.

2. *Un sensor de corriente, que capte la intensidad que circula por el circuito. Utilizaremos el sensor de corriente LEM LTS 6-NP:*

Figura 7: Sensor de corriente LEM LTS 6-NP

También se requiere un circuito acondicionador pues la salida de este sensor está en el rango de 0.5 a 4.5V y el rango compatible en las entradas del Arduino DUE va de 0 a 3.3V. Se trata simplemente de un divisor de tensión.

3. *Un relé de estado sólido que nos permita conectar y desconectar la carga:*

Figura 8: Módulo del relé SSR

4. *Una placa microcontroladora Arduino DUE que tome el mayor número posible de lecturas, tanto de la onda de tensión como la de corriente:*

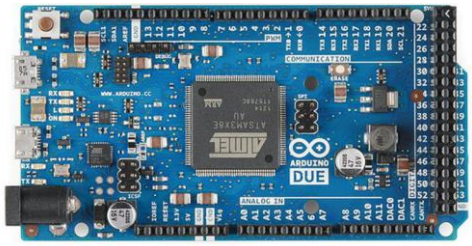


Figura 9: Arduino DUE

5. *Una placa microcomputadora Raspberry Pi 3 modelo B como servidor de aplicaciones que aloje nuestro proyecto y genere la página http (código HTML) que presentará los datos:*

Figura 10: Raspberry Pi 3 modelo B

6. *Un router 3G que realice la función de Gateway para la Raspberry permitiendo el acceso a Internet mediante una interfaz 3G. Como indicábamos en los objetivos de este trabajo, hemos de considerar la posibilidad de que en la ubicación donde se sitúa la carga no tengamos acceso a Internet por red fija, por lo que necesitamos acceso a la red móvil. Una posibilidad es conectar un router wifi 3G:*

Figura 11: Router 3G

Podemos prescindir del este router conectando el módem 3G directamente en la Raspberry, previa configuración de esta última placa.

7. *Un módem 3G que permita la conexión a Internet de la Raspberry:*

Figura 12: Módem 3G

Como hemos indicado, este módem irá conectado al router 3G, o directamente en la Raspberry.

En los apartados siguientes se realizará un análisis más detallado de cada uno de estos elementos. Se expondrán distintas versiones de sensores, circuitos de acondicionamiento de señal, y relés, de los que elegiremos, por los motivos que iremos exponiendo, el sensor de tensión compuesto por el transformador reductor y acondicionamiento mediante componentes pasivos, el LEM LTS 6-NP y divisor de tensión como sensor de corriente, y el relé de estado sólido como interruptor de conexión y desconexión de la carga.

Raspberry PI 3 modelo B

La Raspberry Pi es una placa microcomputadora de bajo coste y del tamaño de una tarjeta de crédito desarrollada en Reino Unido por la Fundación Raspberry Pi, Universidad de Cambridge, en 2011 con un objetivo educacional, siendo esta una de las herramienta más importantes para el aprendizaje de la electrónica y la programación.

Las características más destacables de la Raspberry Pi 3 modelo B son:

- *Procesador Broadcom BCM2387 con 4 núcleos ARM Cortex-A53 a 1.2GHz*
- *LAN inalámbrica (Wifi) 802.11 bgn, que usaremos en caso de utilizar el router wifi 3G.*
- *Memoria RAM de 1GB para poder correr las aplicaciones*
- *Procesador a 64 bits*
- *4 puertos USB, aunque sólo usaremos uno como interfaz del protocolo UART.*
- *Un puerto Ethernet 10/100, que no usaremos*
- *Un puerto micro SD para cargar el sistema operativo Raspbian y almacenar datos*
- *Un conector micro USB para alimentación de la Raspberry*
- *4O pines GPIO para control de dispositivos externos, del que sólo usaremos uno.*

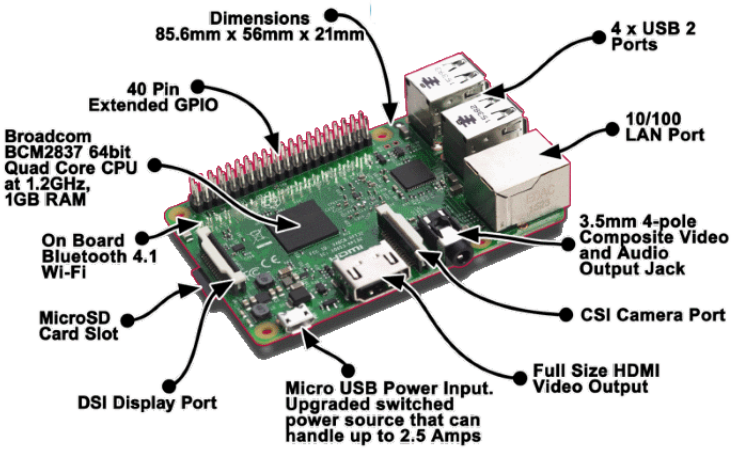


Figura 13: Componentes de la placa Raspberry

La Raspberry soporta una gran cantidad de sistemas operativos y distribuciones ligeras aunque normalmente se usan sistemas operativos de Linux, y más concretamente una versión creada expresamente para la placa Raspberry, llamada Raspbian (versión basada en Debian).

En nuestro trabajo instalaremos en la Raspberry un servidor de aplicaciones Web dinámica montando en ella nuestro proyecto. El servidor que hemos elegido es Apache Tomcat al ser el más utilizado en entornos no empresariales.

Es importante destacar que en la placa Raspberry todos los pines de entrada/salida son digitales, es decir, no dispone de entradas analógicas para poder leer las lecturas de tensión y corriente en nuestro proyecto. Una posible solución a la falta de entradas analógicas habría sido conectarle un convertidor analógico-digital, por ejemplo el ADS1115.

Figura 14: Convertidor analógico-digital ADS1115

El ADS1115 tiene 4 canales y 16 bits de resolución (15 de medición y 1 de signo) y comunicación I2C. Según se especifica en el datasheets, el máximo número de muestras que puede tomar en 1 segundo (SPS) es de 860, es decir, que el mínimo tiempo entre muestras es de 1,16ms, por tanto, en un periodo de 20ms (periodo de la onda eléctrica) podríamos tomar un máximo de 17 muestras. Este valor es muy bajo si queremos tener cierta calidad de muestreo.

La solución adoptada ha sido utilizar un microcontrolador que lleve incorporado los canales ADC, es decir, que disponga de entradas analógicas. El muestreo se realizará en dos fases:

1. *El microcontrolador tomará el mayor número de muestras posible guardándolas internamente en memoria, sin perder tiempo en enviarlas a la Raspberry.*

2. *Una vez tomadas todas las muestras, se enviarán a la Raspberry.*

El microcontrolador que hemos elegido es el Arduino DUE ya que es el más económico que hemos encontrado (unos 11€) y con la velocidad suficiente como para capturar las muestras que necesitamos.

Arduino DUE

Arduino DUE es una tarjeta microcontroladora basada en el procesador ARM Cortex-M3, de 32 bits. Dispone de 54 pines digitales de entrada/salida (de los cuales 12 pueden ser utilizados para salidas PWM), 12 entradas analógicas, 2 salidas analógicas, un cristal a 84MHz, un conector USB-OTG, un conector para alimentación, un conector JTAG, un conector ICSP (protocolo SPI), y un botón de reset.

Características más destacables del Arduino DUE:

- *Microcontrolador: AT91SAM3X8E.*
- *Voltaje de operación: 3.3V.*
- *Voltaje recomendado de alimentación (pin Vin): 7-12V.*
- *Pines de entrada y salida digitales: 54 pines I/O, de los cuales 12 proveen salida PWM.*
- *Pines de entrada analógicos: 12.*
- *Pines de salida analógicos: 2.*
- *Corriente de salida total en los pines I/O: 130mA.*
- *Corriente DC máxima en el pin de 3.3V: 800mA.*
- *Corriente DC máxima en el pin de 5V: 800mA.*
- *Memoria Flash: 512 KB toda disponible para aplicaciones del usuario.*
- *SRAM: 96 KB (en dos bancos de: 64KB y 32KB).*

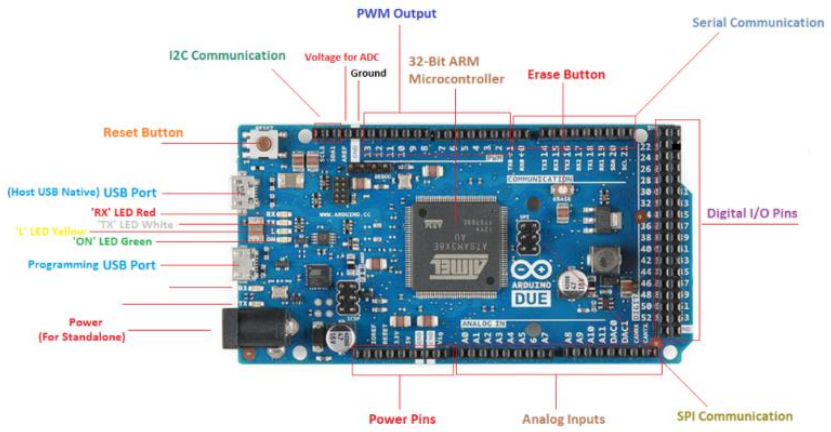


Figura 15: Componente de la placa Arduino DUE

Como ya indicamos, la función de la placa Arduino va a ser exclusivamente tomar el mayor número posible de muestras, tanto de la tensión como de la corriente, durante un periodo de 20ms correspondiente a la frecuencia de 50Hz de la onda de tensión de la red eléctrica. De las pruebas realizadas se ha comprobado que el tiempo mínimo que transcurre entre dos lecturas es de 4.35 μs , por lo que la frecuencia máxima de muestreo que podemos conseguir con el Arduino DUE es de unos $\frac{1}{4.35x10^{-6}} \approx 230\,KHz$ *. El hecho de elegir esta placa de Arduino es precisamente esta alta velocidad de muestreo. Como comparación, con placa Arduino UNO obtendríamos una frecuencia de muestreo de 9kHz.*

Con la placa DUE, el máximo número de muestras que podríamos adquirir en 20ms (periodo de la onda de tensión eléctrica), es de $\frac{20x10^{-3}}{4.35x10^{-6}} = 4602$ *. Como el elemento "canvas" de HTML donde vamos a*

dibujar tiene una anchura de 800 pixeles, no nos interesa disponer de un número de muestras mucho mayor a 800 ya que de otra forma tendríamos muchos puntos (x,y) con la misma coordenada "x" y distinta coordenada "y", produciéndose un efecto de dientes de sierra en las gráficas. Para disminuir el número de muestras sin perder calidad, éstas se hallarán promediando 6 lecturas, con lo que llegaremos a $\frac{4602}{6} = 767$ *muestras. Por tanto serán 767 el número de muestras que el Arduino enviará a la Raspberry tanto para la onda de tensión como para la de la corriente.*

Hay que destacar que las muestras de tensión y de corriente se obtendrán en periodos distintos aunque en la misma fase. Con ello conseguimos maximizar el número de muestras.

Un problema que tenemos que evitar con las placas Arduino son los reinicios automáticos (de unos 500ms) que se producen cuando se inicializa una conexión serie (UART). En concreto, se ignoran los primeros bytes recibidos si se comienza la transmisión desde la Raspberry justo al inicializar el puerto serie, lo que hará a que nuestra aplicación no funcione correctamente. En el Arduino DUE una forma de solucionar estos reinicios es conectar una resistencia de $1k\Omega$ *entre los pines RESET y 3.3V. Esta resistencia es necesario retirarla únicamente para cargar los programas en la placa.*

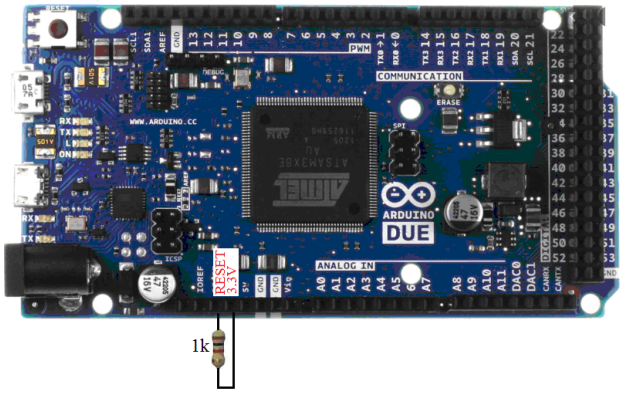


Figura 16: Resistencia de 1k para evitar reinicios en el Arduino DUE

Router wifi 3G

Es el elemento que dará conexión Internet a la Raspberry. El router elegido es el ZTE H108N que dispone de un puerto USB para la conexión del módem 3G.

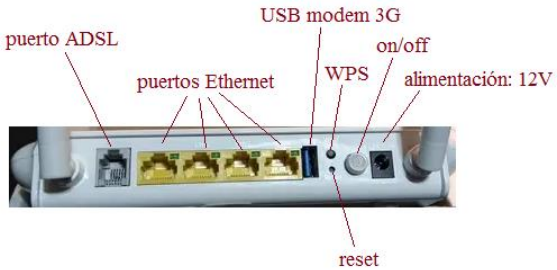


Figura 17: Puertos del router ZTE H108N

La conexión Raspberry-router se realiza vía wifi mediante el AP que incorpora internamente el router.

Una vez introducido el módem 3G en el puerto USB tendremos que esperar a que el éste quede con su led fijo (es decir, sin parpadeo), estado que indicará la conexión correcta a la red 3G.

Módem 3G ZTE-MF190

Este módem 3G es un dispositivo de comunicaciones que se conecta al puerto USB de un equipo y le permite acceder a Internet a través de la red móvil 3G. La elección de módem ZTE MF-190 se debe a que es uno de los más económicos y fiables que se pueden encontrar en el mercado. Sus principales características son:

- *Conexión por USB*
- *Bandas soportadas:*
  - *HSPA/UMTS 900/2100MHz*
  - *GPRS/EDGE 850/900/1800/1900 MHz*
  - *HSPA hasta 7.2Mbps*
  - *HSUPA hasta 1.4Mbps*
- *Dispone de un conector para tarjetas de memoria micro SD de hasta 4GB, que no usaremos.*

Figura 18: Puerto USB del módem 3G

Este módem ya viene configurado para conectarse a la red de nuestro operador de red móvil por lo que lo único que tenemos que realizar es la inserción de una tarjeta SIM de dicho operador, tal como se aprecia en el la figura siguiente:

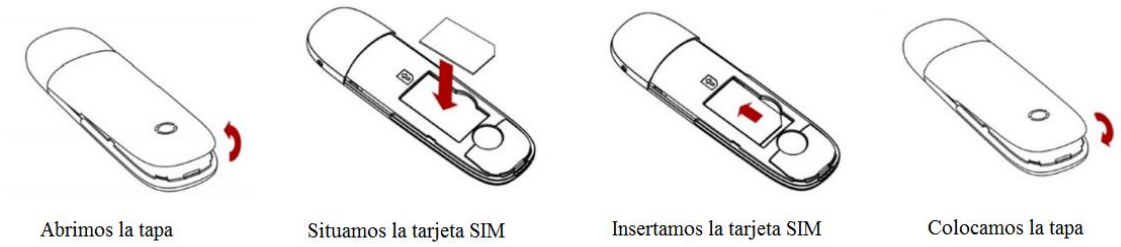


Figura 19: Inserción de la SIM en el módem 3G

Sensor de tensión con transformador y componentes pasivos

Este sensor de tensión está compuesto de un transformador reductor y un circuito adaptador de tensiones. El transformador tiene relación de transformación 20.4:1, por lo que si a su entrada le conectamos la tensión de red:

$$v_i(t) = 230\sqrt{2}seno(2\pi 50t)$$

, la señal a su salida será

$$v_o(t) = \frac{v_i(t)}{20.4} = 11.3\sqrt{2}seno(2\pi 50t) = 16seno(2\pi 50t)$$

, es decir, que la tensión oscilará entre $16V$ *y* $-16V$

Las entradas analógicas en el Arduino DUE admiten un rango de tensión de $0V$ *a* $3.3V$ *por lo que necesitaremos un circuito acondicionador que adapte estas tensiones. El circuito que vamos a utilizar es el siguiente, compuesto únicamente por elementos pasivos:*

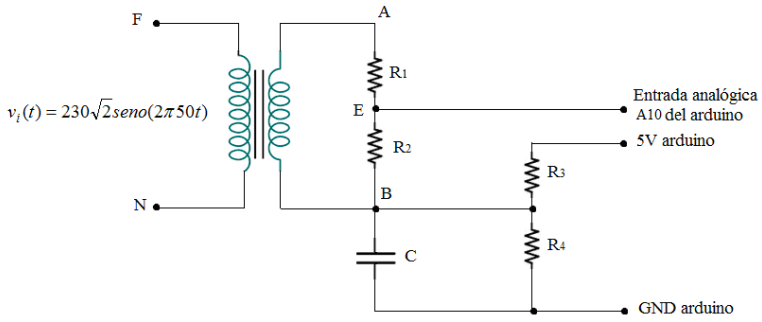


Figura 20: Sensor de tensión con transformador y componentes pasivos

Al tratarse de un circuito lineal podemos aplicar la propiedad aditiva. El circuito anterior dispone de dos fuentes de tensión, una variable con el tiempo $v_i(t)$*, y otra continua de 5V. La tensión de salida* $v_E(t)$ *será la suma de las respuestas individuales de cada fuente.*

Con la fuente de 5V y $v_i(t) = 0$ *tendremos que* $v_{AB} = 0$ *y el circuito equivalente será:*

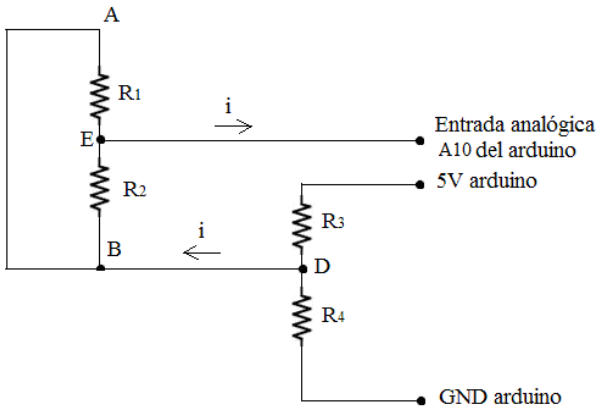


Figura 21: Circuito equivalente del sensor de tensión en corriente continua

Como la impedancia de entrada del pin analógico del Arduino es muy alta (unos $100M\Omega$*) consumirá una corriente despreciable (unos* $10nA$*) por lo que,* $i \approx 0 \rightarrow v_D = 5\frac{R_4}{R_3 + R_4} = v_B = v_E \rightarrow$

$$\boxed{v_E = 5\frac{R_4}{R_3 + R_4}}$$

Sin la fuente de 5V y con $v_i(t) = 230\sqrt{2}seno(100\pi t)$ *tendremos el siguiente circuito equivalente:*

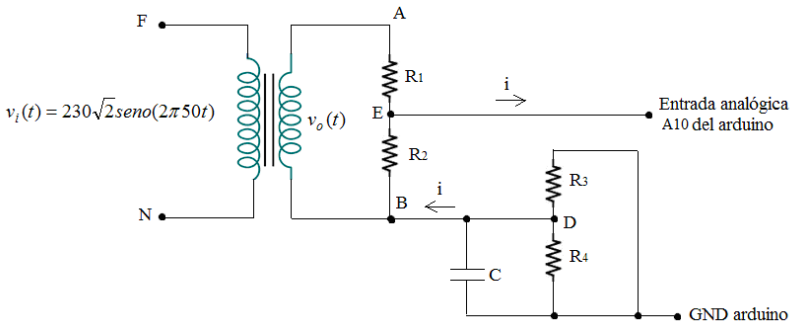


Figura 22: Circuito equivalente del sensor de tensión en corriente alterna

Como hemos visto, $i \approx 0 \rightarrow v_B = 0 \rightarrow v_A(t) = v_o(t) = \frac{v_i(t)}{20.4} = 11.3\sqrt{2}seno(2\pi 50t)$

, y como $v_E(t) = v_A(t)\frac{R_2}{R_1 + R_2}$ *, tendremos que* $v_E(t) = \frac{v_i(t)}{20.4}\frac{R_2}{R_1 + R_2} = 11.3\sqrt{2}\frac{R_2}{R_1 + R_2}seno(2\pi 50t)$

La tensión a la salida, debida a ambas fuentes será la suma de sus contribuciones:

$$v_E(t) = 5\frac{R_4}{R_3 + R_4} + \frac{v_i(t)}{20.4}\frac{R_2}{R_1 + R_2} = 5\frac{R_4}{R_3 + R_4} + 11.3\sqrt{2}\frac{R_2}{R_1 + R_2}seno(2\pi 50t)$$

, y la corriente que circula por R_1 *es* $i_{AB}(t) = \frac{11.3\sqrt{2}}{R_1 + R_2}seno(2\pi 50t)$

El valor del condensador no es determinante, de hecho podemos prescindir de el, pero lo mantendremos para disminuir el rizado que pueda introducir la fuente de 5V. Tomaremos una $\boxed{C = 10\mu F}$

En rango de valores que queremos conseguir de $v_E(t)$ *debe ir de* $0V$ *a* $3.3V$ *pues es el admitido para una entrada analógica en el Arduino DUE, aunque para no dañarlo con una posible sobretensión en el pin de entrada, vamos a asignar un rango de tensión admisible de* $0.3V \le v_C(t) \le 3V$ *. El valor medio es* $1.65V$ *, por tanto:*

$$5\frac{R_4}{R_3 + R_4} = 1.65 \rightarrow 5R_4 = 1.65R_3 + 1.65R_4 \rightarrow R_3 = 2.03R_4$$

Tomaremos un valor alto para R_3 *y* R_4 *para reducir el consumo, por ejemplo* $\boxed{R_3 = 500k\Omega}$, $\boxed{R_4 = 246k\Omega}$

$$11.3\sqrt{2}\frac{R_2}{R_1 + R_2} = 3 - 1.65 \rightarrow 11.3\sqrt{2}R_2 = 1.35R_1 + 1.35R_2 \rightarrow R_1 = 10.84R_2$$

Fijando por ejemplo $\boxed{R_1 = 500k\Omega}$*, obtenemos* $\rightarrow \boxed{R_2 = 46k\Omega}$

Queda finalmente $v_E(t) = 1.65 + \frac{v_i(t)}{241} = 1.65 + 1.35seno(2\pi 50t)$

La impedancia de entrada del sensor de tensión (incluido el transformador) es de:

$$R_i = 20.4^2 (R_1 + R_2) = 227 M\Omega$$

, lo que significa que la corriente en el primario es de $I_p = \frac{230}{R_i} = 1\mu A$ *, suficientemente pequeña para garantizar que el transformador no entrará en saturación.*

Simulando el circuito con Proteus-ISIS obtenemos:

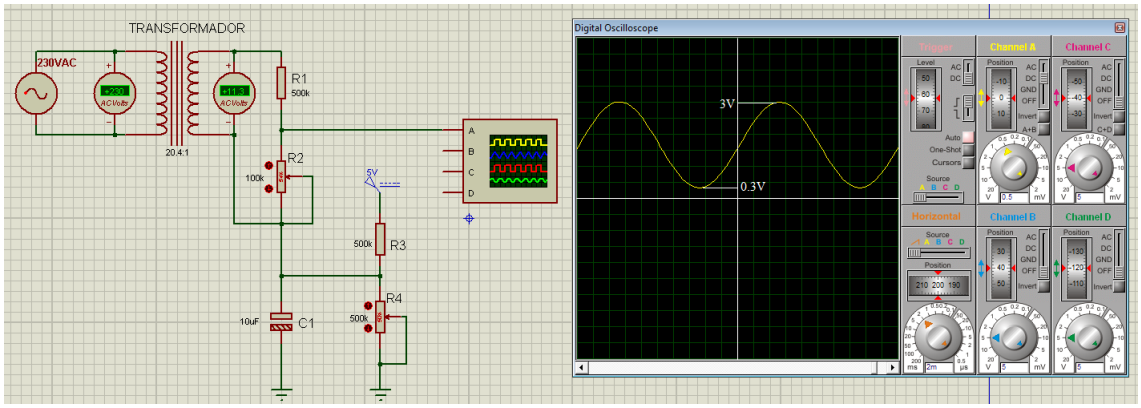


Figura 23: Simulación del sensor de tensión con Proteus (ISIS)

También lo podemos ver realizando una simulación TRAN con LTspice:

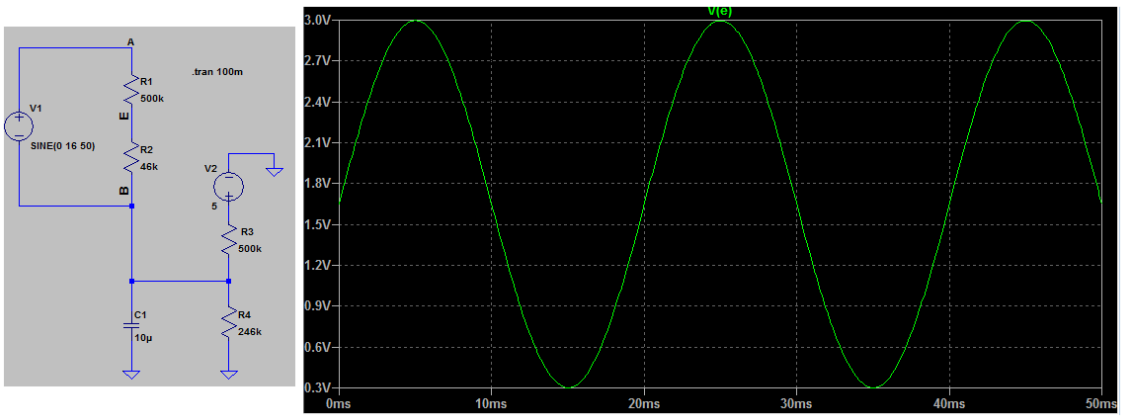


Figura 24: Simulación del sensor de tensión con LTSpice

La potencia consumida en R_1 *es:*

$$P_{R_1} = \frac{1}{0.02}\int_{t=0}^{0.02} R_1 \left[\frac{11.3\sqrt{2}}{R_1 + R_2} seno(2\pi 50t)\right]^2 dt = \frac{1}{0.02}\frac{255.38 R_1}{(R_1 + R_2)^2}\int_{t=0}^{0.02} \frac{1-\cos(4\pi 50t)}{2} dt =$$

$$= \frac{1}{0.02}\frac{255.38 R_1}{(R_1 + R_2)^2}\frac{1}{2}0.02 = \frac{255.38 R_1}{2(R_1 + R_2)^2} = \frac{255.38 x 500 x10^3}{2(546 x10^3)^2} = 214\,\mu W$$

La potencia consumida en R_2 *es:*

$$P_{R_2} = \frac{1}{0.02}\int_{t=0}^{0.02} R_2\left[\frac{11.3\sqrt{2}}{R_1+R_2}\,seno(2\pi 50t)\right]^2 dt = \frac{1}{0.02}\frac{255.38R_2}{(R_1+R_2)^2}\int_{t=0}^{0.02}\frac{1-\cos(4\pi 50t)}{2}dt =$$

$$= \frac{1}{0.02}\frac{255.38R_2}{(R_1+R_2)^2}\frac{1}{2}0.02 = \frac{255.38R_2}{2(R_1+R_2)^2} = \frac{255.38x46x10^3}{2\left(546x10^3\right)^2} = 20\mu W$$

La potencia consumida en R_3 *es* $P_{R_3} = R_3\left(\frac{5}{R_3+R_4}\right)^2 = 500x10^3\left(\frac{5}{746x10^3}\right)^2 = 22.5\mu W$

La potencia consumida en R_4 *es* $P_{R_4} = R_4\left(\frac{5}{R_3+R_4}\right)^2 = 246x10^3\left(\frac{5}{746x10^3}\right)^2 = 11.1\mu W$

La función de transferencia entre la tensión de red y la tensión de entrada al pin del Arduino será:

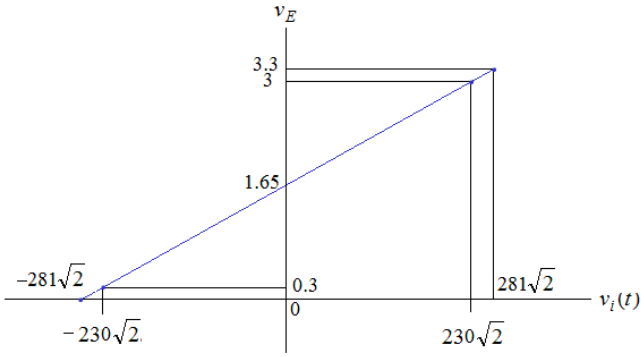


Figura 25: Función de transferencia del sensor de tensión

La tensión eficaz máxima de red que nos puede dar la empresa suministradora es $230+10\%=253V$ *, que es inferior a los 281V que admite el dispositivo (*$3.3V$ *).*

Por defecto, los convertidores ADC que dispone el Arduino DUE son de 10 bits aunque pueden configurarse a 12bits. En principio parecería deseable configurarlo a 12bits pero veamos que esto no es así. El elemento "canvas" de HTML donde vamos a dibujar las gráficas tiene un ancho de 800 pixeles y un alto de 500 pixeles. El mínimo incremento de tensión que se va a poder apreciar en el gráfico es de $\frac{2x230\sqrt{2}}{500}=1.3V$ *. Con el convertidor analógico-digital configurado a 10 bits, y para un rango de* 0 *a* $3.3V$ *a su entrada, el menor incremento de tensión que es capaz de detectar el Arduino es de* $\frac{3.3}{2^{10}}=3.22mV$ *. Nuestro rango de entrada va de 0.3V a 3V, donde 0.3V corresponde a una tensión de red de* $-230\sqrt{2}V$ *, y 3V a* $230\sqrt{2}V$ *. Por tanto, la menor tensión de red que vamos a poder detectar con el Arduino es* $3.22x10^{-3}\frac{2x230\sqrt{2}}{3-0.3}\approx 0.7V$ *, inferior a los 1.3V que podemos diferenciar en el gráfico, por lo que la resolución de 10bits del ADC del Arduino es suficiente, no siendo necesario configurarlo a 12bits, que lo haría más lento que a 10 bits obteniendo un número menor de muestras en los 20ms.*

Sensor de tensión con transformador y operacional LM358

Podemos usar como sensor de tensión el siguiente circuito:

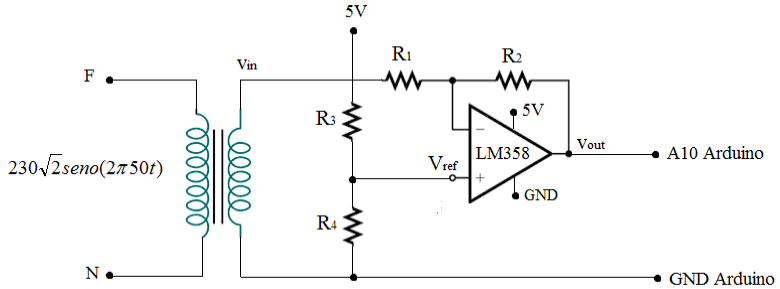


Figura 26: Sensor de tensión con transformador y operacional LM358

Determinemos la tensión a la salida, $v_{out}(t)$:

$$\frac{V_{in}-V_{ref}}{R_1}=\frac{V_{ref}-V_{out}}{R_2}\rightarrow\frac{V_{out}}{R_2}=\frac{V_{ref}}{R_2}+\frac{V_{ref}}{R_1}-\frac{V_{in}}{R_1}\rightarrow V_{out}=\left(1+\frac{R_2}{R_1}\right)V_{ref}-\frac{R_2}{R_1}V_{in}$$

$$v_{in}(t)=11.3\sqrt{2}seno(2\pi 50t)\rightarrow v_{out}(t)=\left(1+\frac{R_2}{R_1}\right)V_{ref}-11.3\sqrt{2}\frac{R_2}{R_1}seno(2\pi 50t)$$

Con alimentación no simétrica del operacional (es decir, ente 5V y GND) la tensión mínima del mismo en zona lineal (sin saturación) es de aproximadamente 600mV, y la máxima no debe sobrepasar los 3V para no dañar a la placa Arduino DUE en caso de sobretensiones, por tanto:

$$0.6V\le v_{out}(t)\le 3V\rightarrow v_{out}(t)=1.8+1.2seno(2\pi 50t)$$

, es decir:

$$1.8+1.2seno(2\pi 50t)=\left(1+\frac{R_2}{R_1}\right)V_{ref}+11.3\sqrt{2}\frac{R_2}{R_1}seno(2\pi 50t):$$

$$11.3\sqrt{2}\frac{R_2}{R_1}=1.2\rightarrow\frac{R_1}{R_2}=13.32$$

Tomando $\boxed{R_2=10k\Omega}\rightarrow\boxed{R_1=133.2k\Omega}$

$$\left(1+\frac{R_2}{R_1}\right)V_{ref}=1.8\rightarrow\left(1+\frac{1}{13.32}\right)V_{ref}=1.8\rightarrow V_{ref}=1.67V$$

$$V_{ref}=\frac{5}{R_3+R_4}R_4\rightarrow 1.67=\frac{5}{R_3+R_4}R_4\rightarrow\frac{R_3+R_4}{R_4}=3\rightarrow R_3=2R_4$$

Tomando $\boxed{R_4=10k\Omega}\rightarrow\boxed{R_3=20k\Omega}$

Función de transferencia:

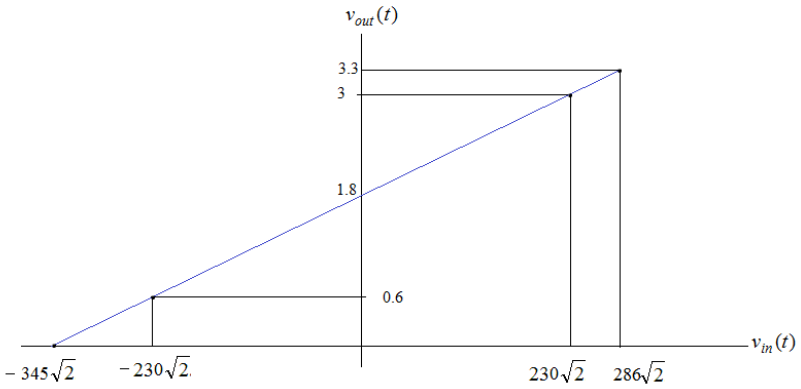


Figura 27: Función de transferencia con el operacional LM358

Vamos a simular el circuito con Ltspice:

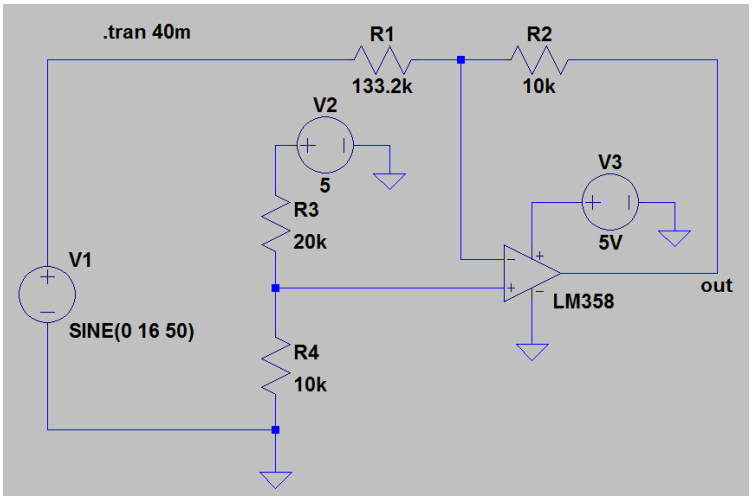


Figura 28: Circuito LTspice sensor de tensión con el operacional LM358

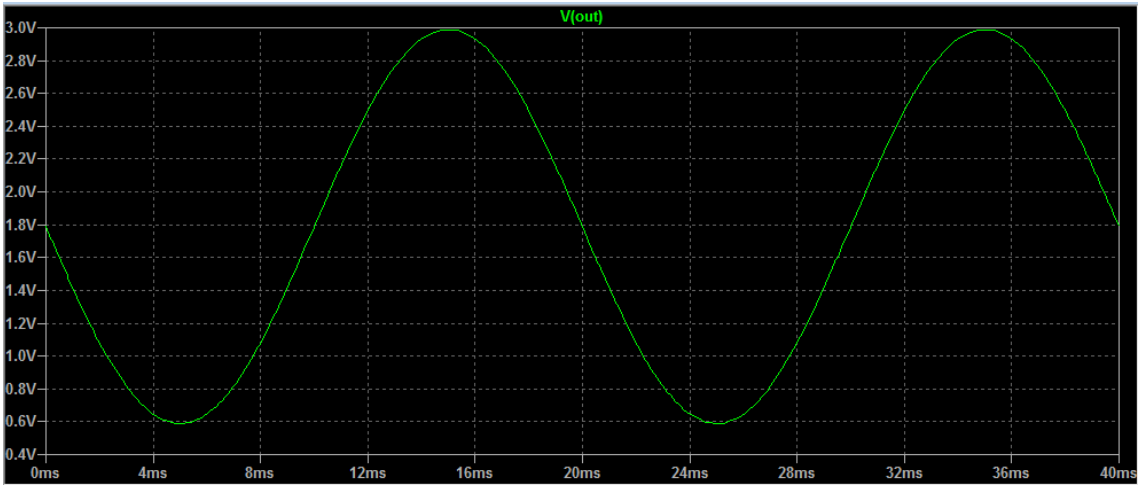


Figura 29: Respuesta circuito LTspice sensor de tensión con el operacional LM358

El menor incremento de tensión que es capaz de detectar el Arduino es de $\frac{3.3}{2^{10}} = 3.22mV$ *. Este incremento corresponde a una variación de tensión en la entrada de* $3.22\frac{2x230\sqrt{2}}{3-0.6} \approx 0.87V$ *. Por tanto la resolución que vamos a obtener con este circuito es de* $0.87V$ *, inferior al hallado para el circuito pasivo descrito en el apartado anterior.*

Podríamos haber realizado una alimentación simétrica del operacional (de $\pm 5V$ *) eliminando la restricción de los 600mV con lo que el rango de tensiones sería:*

$$0.3V \le v_{out}(t) \le 3V \rightarrow v_{out}(t) = 1.65 + 1.35seno(2\pi 50t)$$

, donde hemos establecido una tensión mínima de 0.3V para evitar tensiones negativas en el pin de entrada del Arduino en caso de sobretensiones. La función de transferencia sería similar al del sensor con el transformador reductor y circuito pasivo con la desventaja de tener que disponer de una alimentación adicional de -5V (por ejemplo con el integrado 7660S).

Sensor de tensión con el LEM LV 25-P

Este transductor permite medir tensiones de entrada de entre 10V y 500V. La precisión es de $\pm 0.9\%$ *de la tensión nominal de entrada cuando la corriente en la entrada es de unos 10mA. Su precio es de 62.24€.*

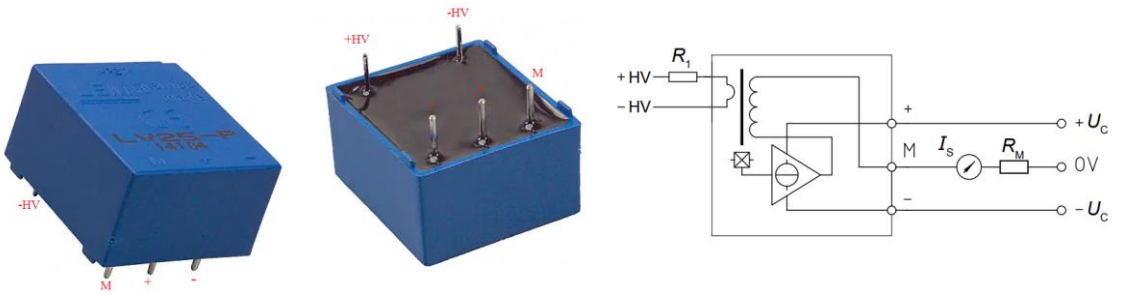


Figura 30: LEM LV 25-P

Según el datasheet de este componente, la resistencia de entrada R_1 *debe calcularse para que en el circuito primario circule una intensidad aproximada de 10mA a la tensión nominal de entrada. La tensión de alimentación* U_c *debe estar en el rango de entre 12 y 15V. La resistencia* R_M *debe estar en el rango de* 30Ω *a* 190Ω *cuando* $U_c = 12V$

En nuestro caso:

$R_1 = \frac{230}{0.01} = 23k\Omega \rightarrow R_1 = 23k\Omega$ *. Tomaremos el valor normalizado de* $\boxed{R_1 = 22k\Omega}$ *por lo que la corriente en el primario será de* $I_1 = \frac{230}{22000} = 10.5mA$ *(valor eficaz)*

La potencia consumida en esta resistencia:

$$P_{R_1} = R_1 I_i^2 = 22000\left(\frac{10.5}{1000}\right)^2 = 2.43W$$

Si no encontráramos una resistencia de $22k\Omega$ *podríamos utilizar dos resistencias en serie, de* $R_{11}=12k\Omega$ *y* $R_{12}=10k\Omega$. *Las potencias consumidas en estas resistencias serían*

$$P_{R_{11}}=12000\left(\frac{10.5}{1000}\right)^2=1.32W \rightarrow \textit{elegiríamos una resistencia de 2W de potencia}$$

$$P_{R_{12}}=10000\left(\frac{10.5}{1000}\right)^2=1.10W \rightarrow \textit{elegiríamos una resistencia de 2W de potencia}$$

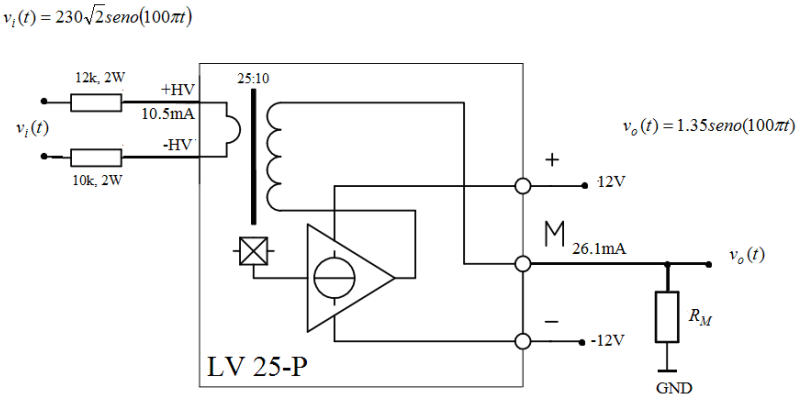


Figura 31: Resistencias de entrada y R_M *del LV 25-P*

Queremos obtener una tensión de salida $v_o(t)=1.35seno(100\pi t)$.

La corriente eficaz en el secundario es $10.5x\frac{25}{10}=26.1mA$ *pues la relación de transformación es 25:10 según se indica en el datasheet.*

La tensión eficaz de salida es $\frac{1.35}{\sqrt{2}}=26.1x10^{-3}R_M \rightarrow \boxed{R_M=36.5\Omega}$. *Utilizaremos un potenciómetro de* $100\,\Omega$ *para realizar ajustes, y calibrado en* 36.5Ω.

La potencia consumida en esta resistencia es de $P_{R_M}=36.5\left(\frac{26.1}{1000}\right)^2=25mW$, *por lo que sería suficiente con una resistencia de* 36.5Ω *y 1/8W de potencia.*

El circuito acondicionador consta de dos seguidores de tensión (alta impedancia de entrada) y un sumador no inversor:

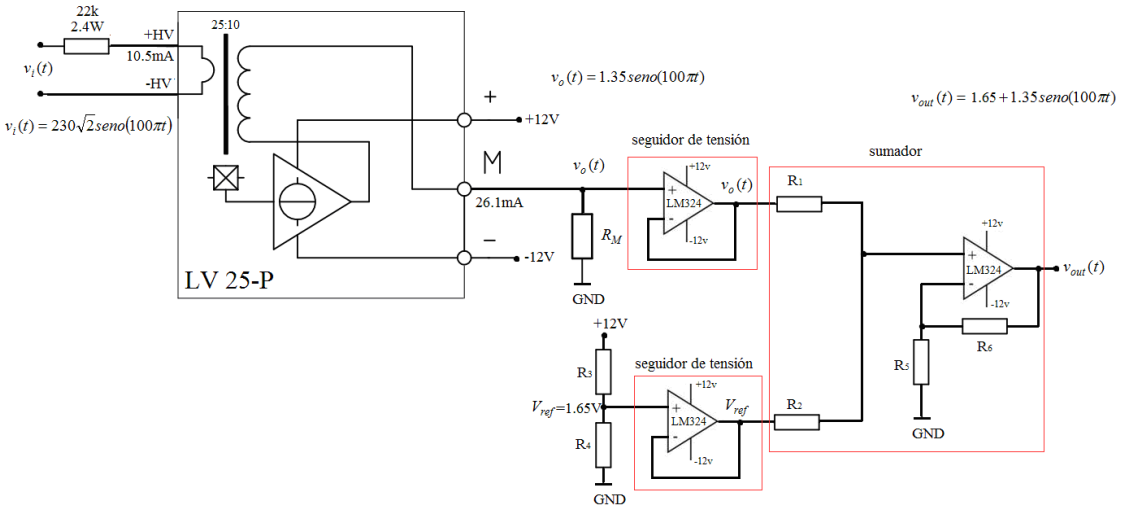


Figura 32: LV 25-P con circuito acondicionador

$$\frac{v_o(t)-V_+}{R_1}+\frac{V_{ref}-V_+}{R_2}=0 \rightarrow \frac{v_o(t)}{R_1}+\frac{V_{ref}}{R_2}=\left(\frac{1}{R_1}+\frac{1}{R_2}\right)V_+ \rightarrow V_+=\frac{R_1}{R_1+R_2}V_{ref}+\frac{R_2}{R_1+R_2}v_o(t)$$

$$V_-=\frac{R_5}{R_5+R_6}v_{out}(t) \rightarrow \frac{R_5}{R_5+R_6}v_{out}(t)=\frac{R_1}{R_1+R_2}V_{ref}+\frac{R_2}{R_1+R_2}v_o(t) \rightarrow$$

$$v_{out}(t)=\frac{R_5+R_6}{R_1+R_2}\left(\frac{R_1}{R_5}V_{ref}+\frac{R_2}{R_5}v_o(t)\right)=\frac{R_5+R_6}{R_1+R_2}\left(\frac{R_1}{R_5}V_{ref}+\frac{R_2}{R_5}1.35seno(100\pi t)\right)=$$

Queremos obtener $v_{out}(t)=1.65+1.35seno(100\pi t)$ *, entonces,*

, tomando $R_1=R_2$ *y* $R_5=R_6$ $\rightarrow v_{out}(t)=V_{ref}+1.35seno(100\pi t)$

$$V_{ref}=\frac{12}{R_3+R_4}R_4 \rightarrow 1.65=\frac{12}{R_3+R_4}R_4 \rightarrow \frac{R_3+R_4}{R_4}=7.27 \rightarrow R_3=6.27R_4$$

Tomando $R_3=10k\Omega \rightarrow R_4=1594\Omega$

Podemos elegir por ejemplo $\boxed{R_1=R_2=R_3=R_5=R_6=10k\Omega}$, $\boxed{R_4=1595\Omega}$

Las resistencias R_1 *y* R_4 *las implementaremos con potenciómetros de* $20k\Omega$ *(calibrado en* $10k\Omega$*) y* $2k\Omega$ *(calibrado en* 1594Ω*), respectivamente, para poder realizar ajustes. Con* R_1 *variaremos la amplitud de la señal de salida, y con* R_4 *variaremos su valor medio (tensión de referencia).*

La alimentación de -12V la podemos obtener con el integrado 7660S:

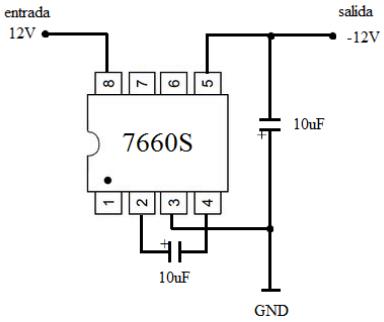


Figura 33: Integrado 7660S

Vamos a simular el circuito con Ltspice:

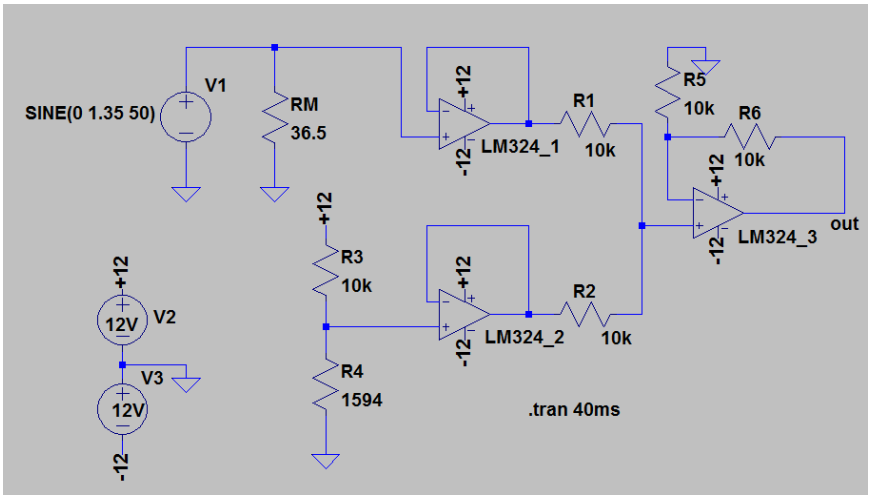


Figura 34: Circuito LTspice sensor de tensión con el operacional LM358

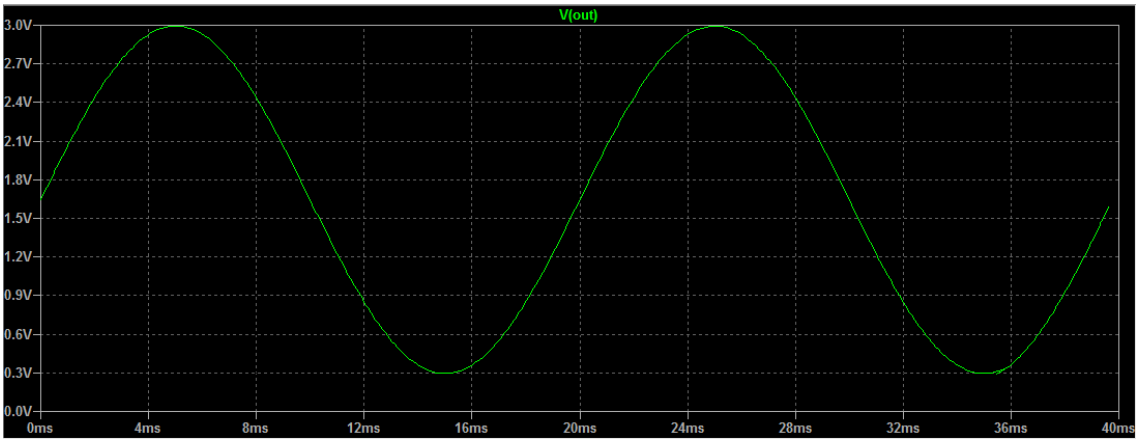


Figura 35: Respuesta circuito LTspice sensor de tensión con el operacional LM358

La resolución de este sensor es similar a la del circuito pasivo con los siguientes inconvenientes: el LV 25-P tiene un coste de unos 60€ (muy superior al coste de transformador), requiere mayor circuitería adicional, y mayor consumo de potencia debido a la resistencia de entrada de 2.4W. Como ventaja podemos destacar el menor espacio que ocupa el LV 25-P respecto al transformador.

Por los motivos expuestos, de los tres sensores de tensión descritos finalmente decidimos elegir el compuesto por elementos pasivos.

Sensor de corriente ACS712

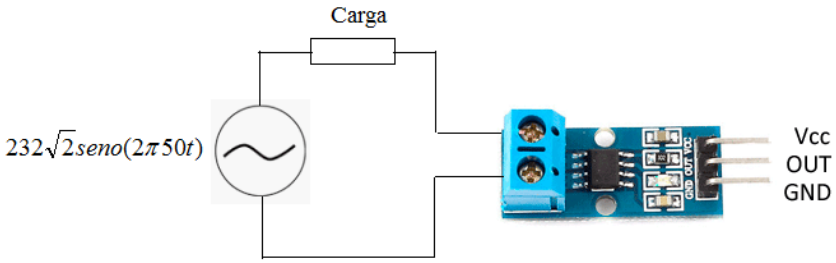


Figura 36: Medición de corriente con el sensor ACS712

Se trata de un sensor de corriente de efecto Hall. Este efecto hace que aparezca una tensión, denomina tensión Hall, en el interior de un conductor por el que transcurre una corriente con una velocidad de desplazamiento $\vec{v}$ *, en presencia de un campo magnético* $\vec{B}$*. Este campo generará, según la fuerza de Lorentz* $\vec{F} = q\,\vec{v} \wedge \vec{B}$*, un campo eléctrico de valor* $\vec{E} = \frac{\vec{F}}{q} = \vec{v} \wedge \vec{B}$*, que desplazará las cargas perpendicularmente apareciendo un potencial eléctrico en esta dirección (tensión Hall).*

La tensión que aparece por este movimiento de cargas es proporcional a la intensidad de la corriente que circula, fenómeno que se utiliza en los sensores de efecto Hall para medir la intensidad de la corriente eléctrica.

El sensor de corriente ACS712 es un sensor Hall de bastante precisión y bajo offset. La resistencia al paso de la corriente es de tan sólo $1.2m\Omega$*. En el capítulo de calibración describiremos la forma de eliminar la influencia del offset en los sensores de corriente.*

La salida del sensor es una tensión proporcional a la corriente y bastante independiente a la temperatura. Existen tres modelos para mediciones de corriente hasta $5A$*,* $20A$ *y* $30A$*.*

El ACS712 dispone de 3 pines:

- *Vcc: alimentación. Tensión máxima 8V. Nosotros podremos alimentarlo con los 5V del Arduino.*
- *OUT: salida analógica*
- *GND: tomada del Arduino*

Según el datasheet, la tensión de salida es:

$$u_{out}(t) = 2.5 + ki(t)$$

, donde k *es la sensibilidad, y depende del modelo elegido:*

Modelo de 5A: $k = 0.185V / A$

Modelo de 20A: $k = 0.1V / A$
Modelo de 30A: $k = 0.066V / A$

En el caso del modelo ACS712 de $30A$*, los valores extremos de tensión son:*

$V_{\min} = 2.5 + 0.066(-30) = 0.52V$ *, para una corriente de -30A*
$V_{\max} = 2.5 + 0.066x30 = 4.48V$ *, para una corriente de 30A*

La función de transferencia del sensor ACS712 de 30A es:

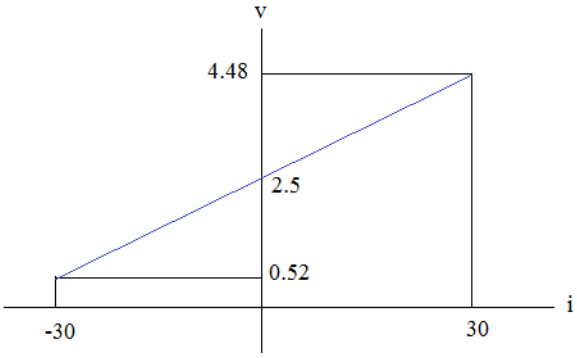


Figura 37: Función de transferencia del sensor ACS712 de 30A

Puesto que la Raspberry trabaja con tensiones de 3.3V tendremos que añadir un circuito acondicionador para adaptar las tensiones de salida del ACS712 al rango de 0 a 3.3V. Simplemente utilizaremos un divisor de tensión con dos resistencias (una de ellas variable por si requerimos pequeños ajustes):

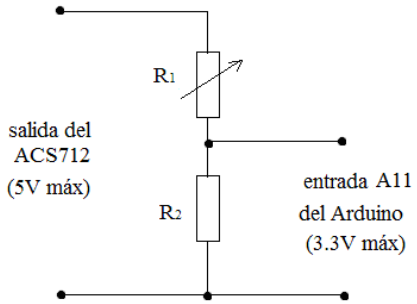


Figura 38: Circuito acondicionador para el ACS712

$$5\frac{R_2}{R_1 + R_2} = 3.3 \rightarrow R_1 = 0.52R_2$$

Si tomamos $\boxed{R_2 = 100k\Omega} \rightarrow \boxed{R_1 = 52k\Omega}$

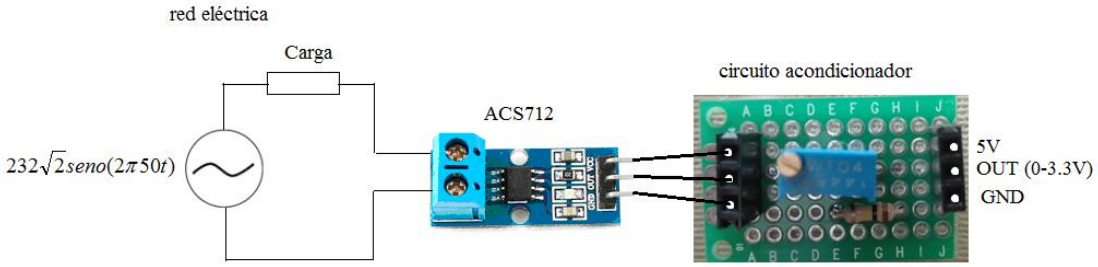


Figura 39: Implementación del circuito acondicionador para el ACS712

El convertidor ADC del Arduino DUE es de 10 bits, con un rango de 0 *a* $3.3V$ *, por lo que el menor incremento de tensión que es capaz de detectar el Arduino es de* $\frac{3.3}{2^{10}} = 3.22mV$ *. Este incremento corresponde a una corriente en el sensor ACS712 de 30A de* $\frac{3.22}{66} = 0.049 \rightarrow 49mA$ *. Por tanto la resolución que vamos a obtener con este sensor es de* $49mA$ *, es decir, vamos a poder detectar variaciones en la corriente de red de unos 49mA. En el caso de ACS712 de 5A tendremos* $\frac{3.22}{185} = 17.4mA$ *, mejorando la resolución a costa de limitar la corriente máxima.*

Sensor de corriente LEM LTS 6-NP

Es otro tipo de sensor de corriente de efecto Hall bastante preciso, lineal, poca dependencia de la temperatura, alta inmunidad a las interferencias externa. Mide corrientes de hasta 19.2A con tensión máxima en el primario de 600V y tensión de salida de entre 0 y 5V.

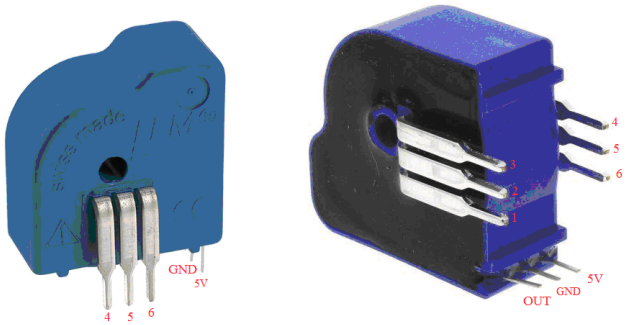


Figura 40: Sensor de corriente LEM LTS 6-NP

El circuito implementado en este sensor es el siguiente:

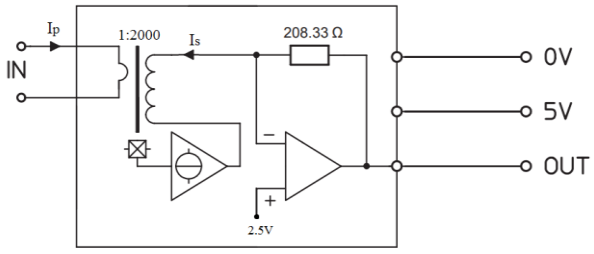


Figura 41: Circuito sensor de corriente LEM LTS 6-NP

Según el fabricante, disponemos de tres formas de conectar el circuito primario del LTS 6-NP, obteniendo en cada una ellas una sensibilidad y una corriente máxima distinta. Veámoslas dichas conexiones y las funciones de transferencia correspondientes:

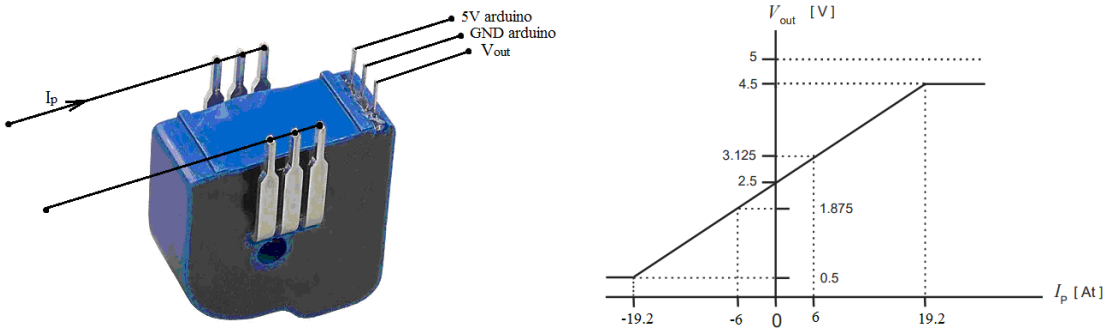


Figura 42: Sensor de corriente LEM LTS 6-NP con una vuelta en el primario

Corriente máxima: 19.2A

$$Sensibilidad = \frac{4.5 - 0.5}{19.2x2} = 104.17 mV / A$$

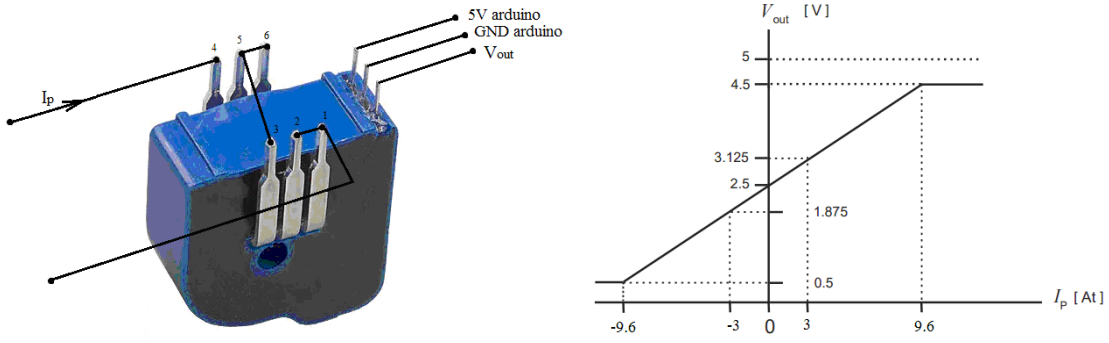


Figura 43: Sensor de corriente LEM LTS 6-NP con dos vueltas en el primario

Corriente máxima: 9.6A

$$Sensibilidad = \frac{4.5 - 0.5}{9.6x2} = 208.33 mV / A$$

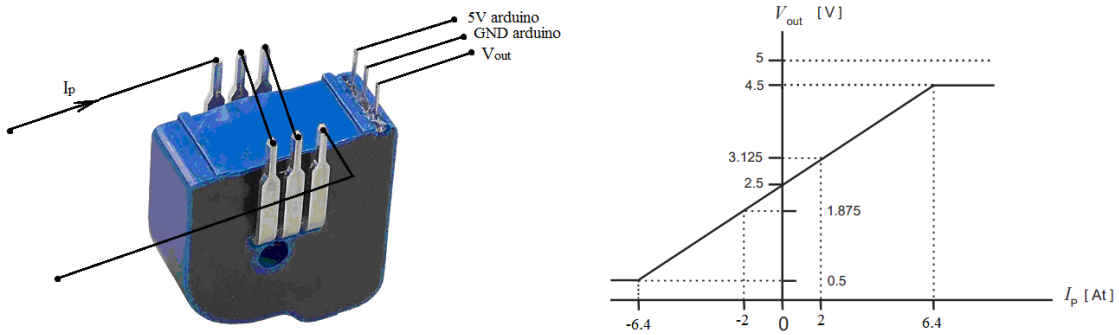


Figura 44: Sensor de corriente LEM LTS 6-NP con tres vueltas en el primario

Corriente máxima: 6.4A

Sensibilidad $= \frac{4.5-0.5}{6.4x2} = 312.5mV/A$

De los tres tipos de conexionado vistos anteriormente usaremos la primera, que aunque tenga menor sensibilidad nos permite medir corrientes de hasta 19.2A.

Según su circuito interno, para una corriente máxima de $I_p = 19.2A$ *la salida será*

$V_{out} = V_- + 208.22 I_s = V_- + 208.22 \frac{I_p}{2000} = V_+ + 208.22 \frac{I_p}{2000} = 2.5 + 208.33 \frac{19.2}{2000} = 4.5V$, *es decir, la máxima tensión que obtendremos a la salida del LTS 6-NP es de 4.5V*

Para una corriente primaria de $I_p = -19.2A$ *la salida será*

$V_{out} = V_- - 208.22 I_s = V_- - 208.22 \frac{I_p}{2000} = V_+ - 208.22 \frac{I_p}{2000} = 2.5 - 208.33 \frac{19.2}{2000} = 0.5V$, *es decir, la mínima que obtendremos a la salida del LTS 6-NP es de 0.5V*

Para conectar este sensor a la placa Arduino DUE necesitamos un circuito acondicionador, que puede un divisor de tensión similar al usado para el ACS712:

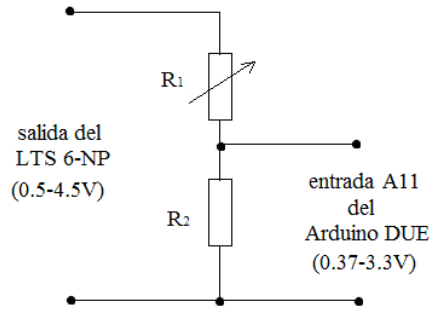


Figura 45: Circuito acondicionador para el sensor de corriente LEM LTS 6-NP

Cuando a la salida tengamos 4.5V, al Arduino le deben llegar 3.3V, por tanto:

$4.5 \frac{R_2}{R_1 + R_2} = 3.3 \rightarrow 4.5R_2 = 3.3R_1 + 3.3R_2 \rightarrow R_2 = 2.75R_1$

Si tomamos $\boxed{R_2 = 100k\Omega} \rightarrow \boxed{R_1 = 36.4k\Omega}$

La tensión mínima será de $0.5\frac{R_2}{R_1+R_2}=0.5\frac{100}{36.4+100}=0.37V$

La función de transferencia incluyendo en circuito acondicionador es:

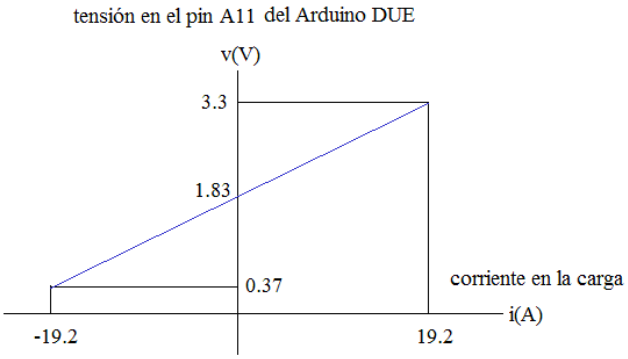


Figura 46: Función de transferencia del LEM LTS 6-NP con el circuito acondicionador

$$Sensibilidad=\frac{3.3-0.37}{2x19.2}=76.3mV/A$$

Aunque tiene mayor precio que el ACS712 finalmente elegiremos el LEM LTS 6-NP al disponer de mayor sensibilidad e inmunidad al ruido (menor influencia a campos electromagnéticos externos).

El circuito montado en placa es este:

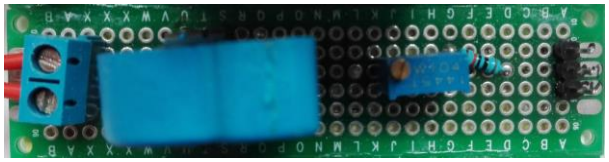


Figura 47: Módulo sensor de corriente LEM LTS 6-NP

Relé electromecánico (contactor)

El relé tiene la función de conectar y desconectar la carga de la red eléctrica en función de una señal eléctrica que enviaremos desde el Arduino DUE, en concreto desde el pin 53. El relé electromagnético (o contactor) usa el desplazamiento de una pieza móvil (contacto) debida a la generación de un campo magnético proveniente de una bobina.

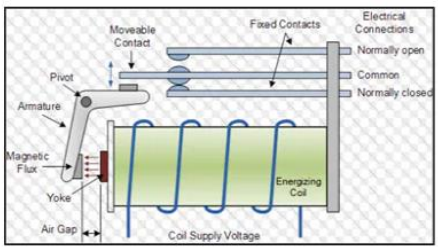


Figura 48: Relé electromecánico

Este es el módulo del relé que hemos construido:

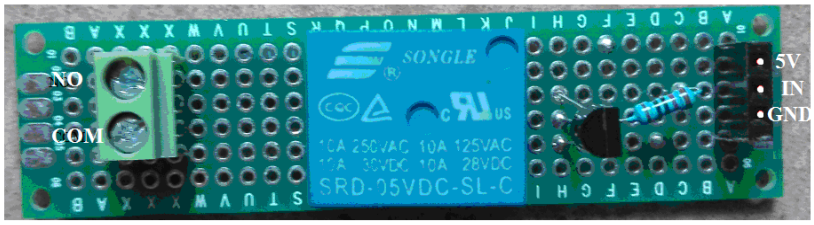


Figura 49: Módulo relé electromecánico

, y cuyo circuito es el siguiente:

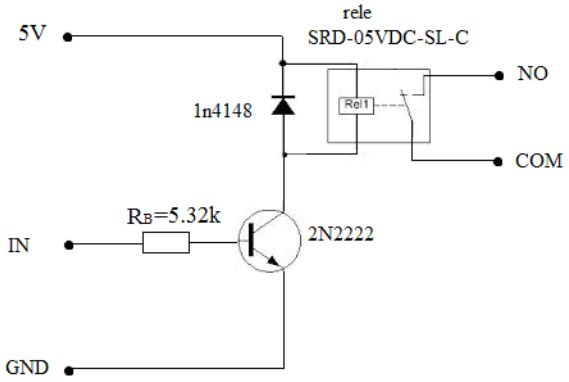


Figura 50: Circuito del módulo del relé electromecánico

Según el datasheet, la resistencia de la bobina del relé SRD-05VDC-SL-C es de 70Ω. *Cuando el transistor esté saturado,* $(V_{CEsat} \approx 0.2V)$ *circulará una corriente de colector de* $\frac{5-0.2}{70} = 68.6mA$. *Como la ganancia de este transistor es* $\beta \approx 150$, *para conseguir la saturación necesitamos una corriente de base* $I_B > \frac{I_C}{150} = \frac{68.6x10^{-3}}{150} = 457.1\mu A$.

En saturación $V_{BE(sat)} \approx 0.8V \rightarrow V_{IN} - 0.8 = R_B I_B \rightarrow R_B < \frac{3.3-0.8}{457.1x10^{-6}} = 5.47k\Omega$. *Tomaremos* $\boxed{R_B = 5.1k\Omega}$

Vamos a simular el punto de operación con LTspice:

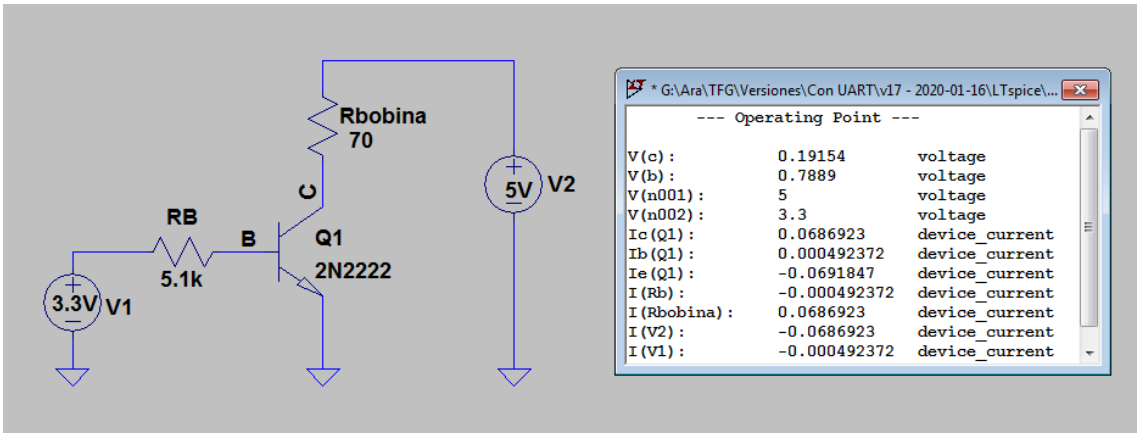


Figura 51: Simulación con LTspice del punto de operación (OP) del circuito relé electromecánico

El diodo en antiparalelo 1n4148 protege al transistor del transitorio de corriente que se produce cuando se desconecta la carga.

Este circuito dispone de los siguientes terminales:

VCC: alimentación de 5V (que le proporcionará el Arduino)
IN: con IN=3.3V el terminal COM se conecta a NO
con IN=0V el terminal COM se desconecta de NO
GND: 0V
COM: salida "común"
NO: salida "normalmente desconectada" de COM

Hay que tener en cuenta que el relé electromecánico genera un campo magnético que distorsionará la medida del sensor de corriente, motivo por el que descartamos este tipo de relé.

Relé de estado sólido (SSR)

Un relé de estado sólido es un interruptor electrónico, es decir, no dispone de contactos móviles ni de bobina, con lo que evitamos la generación de un campo magnético que interfiera con el sensor de corriente. Para nuestro proyecto vamos a construirlo con el acoplador óptico MOC3041, que protegerá al microcontrolador de la red eléctrica, y el TRIAC BTA16 como interruptor, tal como se muestra en la figura.

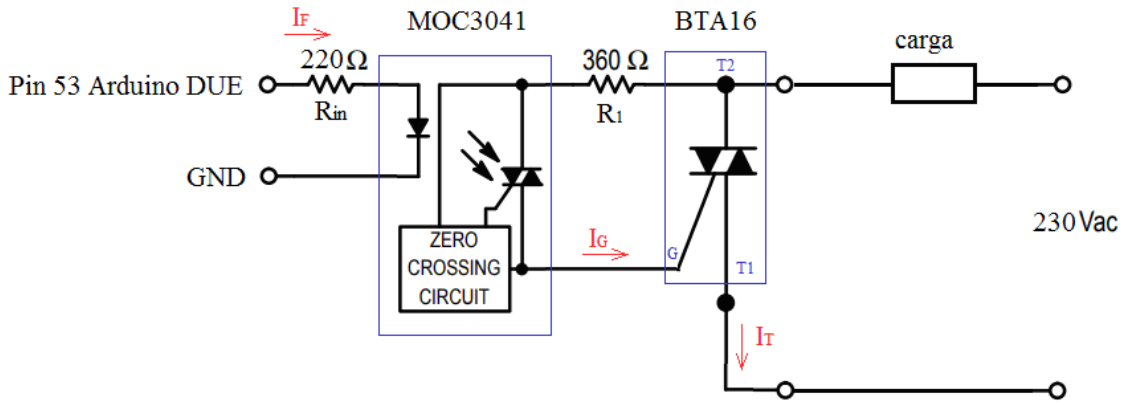


Figura 52: Relé de estado sólido (SSR)

El funcionamiento es el siguiente: cuando ponemos 3.3V en el pin 53 del Arduino DUE, el diodo del MOC3041 emite luz infrarroja que hace disparar al Triac del MOC3041 en el siguiente paso por cero de la tensión entre ánodos. Al dispararse este Triac se genera una corriente eléctrica I_G *a través de la resistencia* R_1 *que a su vez dispara el Triac BTA16, conectando la carga a la red eléctrica, y provocando el apagado del Triac del MOC3041 al bajar su corriente por debajo de la de mantenimiento.*

El datasheet del MOC3041 nos indica que la corriente en el diodo no debe superar $I_F = 15mA$*, por tanto,*

$$R_{in} \geq \frac{V_{pin53} - V_{diodo}}{I_F} = \frac{3.3 - 1.2}{0.015} = 140\Omega$$ *. Tomamos el valor comercial* $R_{in} = 220\Omega$

La resistencia R_1 *tiene la función de limitar la corriente de puerta de BTA16 de forma que* $I_G < 35mA$ *del BTA16, según indica el fabricante.*

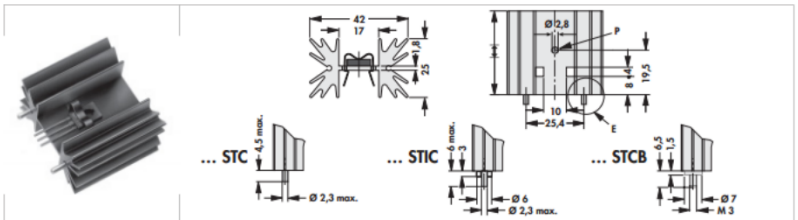
La corriente máxima que puede circular por el BTA16 es $I_T = 16A$*, según indica el datasheet, aunque también tenemos que tener en cuenta la potencia máxima que el Triac es capaz de disipar.*

Cálculo de la corriente máxima que puede soportar el BTA16, sin disipador:

Máxima temperatura de la unión, $T_J = 125^\circ C$ *(según datasheet)*
Temperatura ambiente: $T_a = 30^\circ C$
Resistencia térmica unión-ambiente, $R_{\theta JA} = 60^\circ C/W$ *(según datasheet)*
Potencia máxima que puede disipar el BTA16, $P_D = \frac{T_j - T_a}{R_{\theta JA}} = \frac{125 - 30}{60} = 1.6W$
Caída de tensión máxima entre terminales T2-T1, $V_T = 1.55V$ *(según datasheet)*
Corriente máxima, $I_T = \frac{1.6}{1.55} \approx 1A$

, por tanto, para corrientes superiores a $I_T = 1A$*, como es nuestro caso, es necesario acoplarle un disipador.*

Cálculo de la corriente máxima I_T *en el BTA16 con el disipador SK12925,4 de Fischer Elektronik:*



art. no.	[mm]	R_{th} [K/W]	
SK 129 25,4 ...	25.4	7.8	TO 220
SK 129 38,1 ...	38.1	6.5	TO 220
SK 129 50,8 ...	50.8	5.3	TO 220
SK 129 63,5 ...	63.5	4.5	TO 220
please indicate:	... mouting method STC = with solder pin STIC = with solder pin and insulating washer STCB = with threaded bolt M3, brass		
surface:	black anodised		

Figura 53: Disipador Fischer Elektronic

Máxima temperatura de la unión, $T_J = 125^{o}C$ *(según datasheet)*

Resistencia térmica unión-carcasa, $R_{\theta jC} = 2.3^{o}C/W$ *(según datasheet)*

Resistencia térmica carcasa-disipador, $R_{\theta CS}$ *:*

La pasta HY710 térmica tiene una resistencia de $0.067^{o}C - in^2/W$

Superficie de contacto carcasa-disipador $\approx 105 mm^2 \approx 0.163 in^2$

$$R_{\theta CS} = \frac{0.067}{0.163} = 0.41^{o}C/W$$

Resistencia térmica disipador-ambiente, $R_{\theta Sa} = 7.8^{o}C/W$ *(según fabricante del disipador)*

Potencia máxima que se puede disipar, $$P_D = \frac{T_j - T_a}{R_{\theta jC} + R_{\theta CS} + R_{\theta Sa}} = \frac{125 - 30}{2.3 + 0.41 + 7.8} = 9.04W$$

Caída de tensión máxima entre terminales T2-T1, $V_T = 1.55V$ *(según datasheet)*

Corriente máxima, $$I_T = \frac{9.04}{1.55} \approx 5.83A$$

El módulo del relé es el siguiente:

Figura 54: Circuito relé estado sólido

Para corrientes superiores a 5.83A debemos usar un disipador mayor (menor resistencia térmica), o sustituir el Triac BTA16 por uno de mayor potencia (BTA24, BTA41,…), o usar un relé electromecánico con el consiguiente error en la medida del sensor de corriente provocado por el campo magnético generado por su bobina.

Los SSR tienen la ventaja respecto a los contactores de que su vida no está limitada por el número de ciclos de conexión-desconexión.

Relé de estado sólido SSR-20DA de FOTEK

En el mercado ya existen relés SSR con su circuitería integrada, por ejemplo el SSR-20 DA de Fotek, cuyas características son las siguientes:

- *Corriente máxima (nominal) en la carga: 20A*
- *Tensión de red admitida en la carga: 24-380VAC*
- *Tensión en la entrada para disparar con seguridad el relé: 3-32VDC*
- *Resistencia de entrada: 1.2kΩ aprox.*
- *Corriente consumida a la entrada: 4-20mA*
- *Tiempo de encendido inferior a 10ms*
- *Tiempo de apagado inferior a 10ms*
- *Caída de tensión a 20ºC: 1.6V*

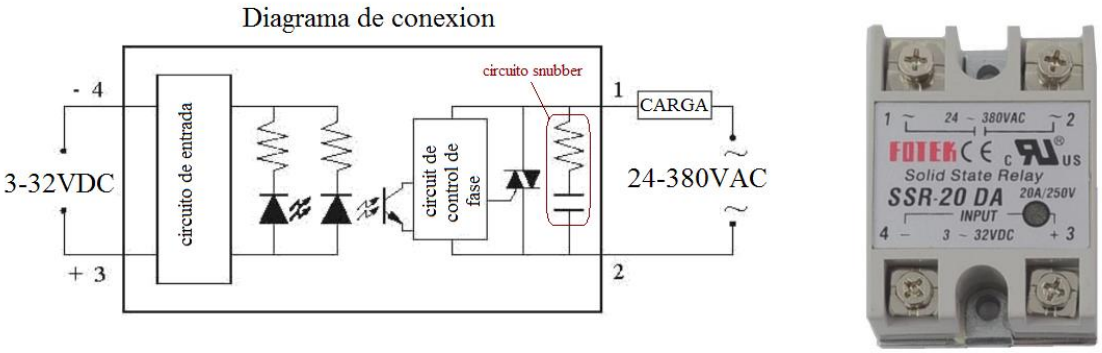


Figura 55: SSR-20DA de FOTEK

Como vemos lleva incorporado un circuito snubber compuesto por resistencia y condensador. Dicho circuito tiene la función de suavizar los picos de tensión transitorios que se producen en cargas inductivas.

Este dispositivo tiene la desventaja de que es más caro que el circuito electrónico anterior implementado por nosotros.

Instalación del sistema operativo Raspbian en la Raspberry

Raspbian es un sistema operativo gratuito recomendado para la Raspberry al estar optimizado para su hardware. Se basa en una distribución GNU/Linux llamada Debian.

El proceso que seguiremos para la instalación de Raspbian en la Raspberry es el siguiente:

1. *Formatearemos una tarjeta micro SD de al menos 16GB con una aplicación al efecto, por ejemplo "SD Card Formatter".*

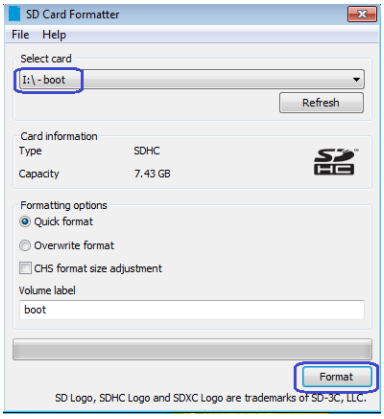


Figura 56: Aplicación SD Card Formatter

2. *De la página Web https://www.raspberrypi.org/downloads/raspbian/ descargamos el sistema operativo "Raspbian Buster Lite". "Buster Lite" es el código de desarrollo de la versión 10 de Debian, no dispone de interfaz gráfica, se maneja únicamente mediante comandos, y es la distribución que requiere menos recursos (memoria y consumo de potencia).*

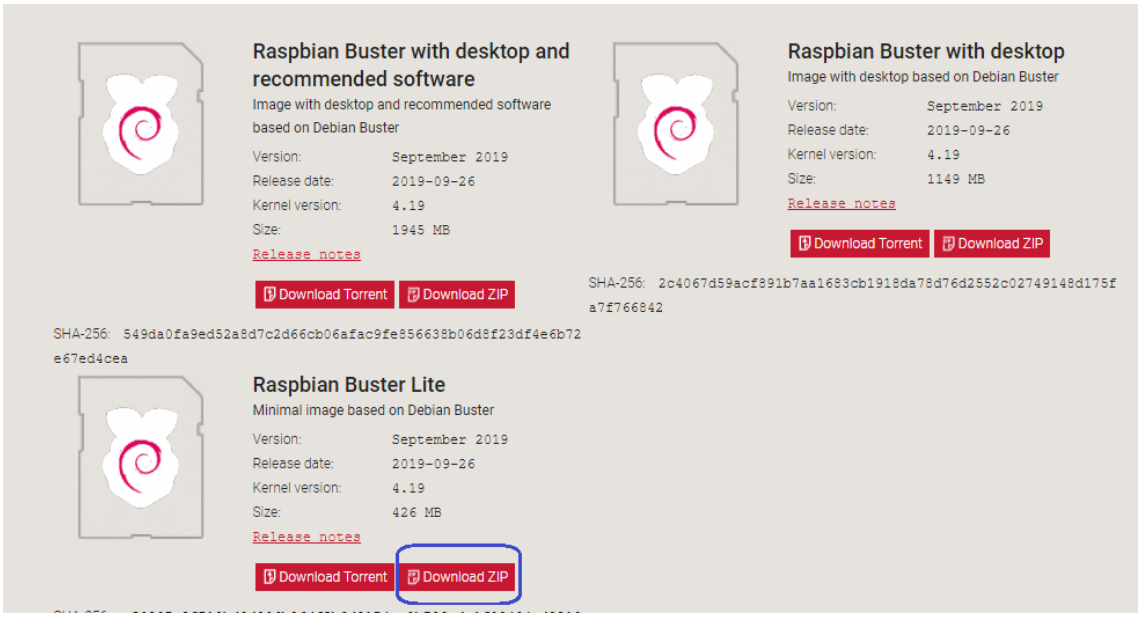


Figura 57: Página de descarga de Raspbian

3. *Descomprimimos el fichero "zip" descargado.*
4. *Abrimos la aplicación "Win32DiskImager" para copiar el fichero imagen descomprimido en la tarjeta micro SD:*

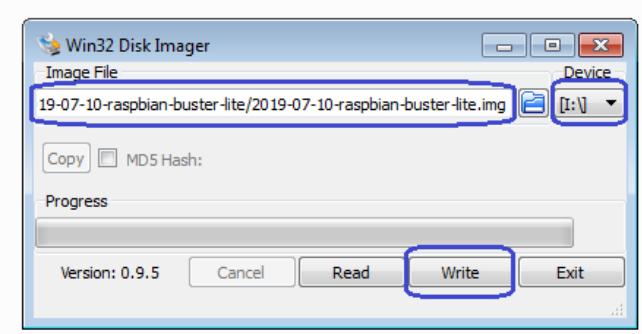


Figura 58: Aplicación Win32DiskImager

5. *Con el explorador de archivos nos situamos en la carpeta de la tarjeta SD, y con el "Bloc de notas" añadimos un fichero sin contenido al que llamaremos "ssh", sin ninguna extensión*

6. *Expulsamos la tarjeta SD de nuestro PC y la insertamos en la Raspberry:*

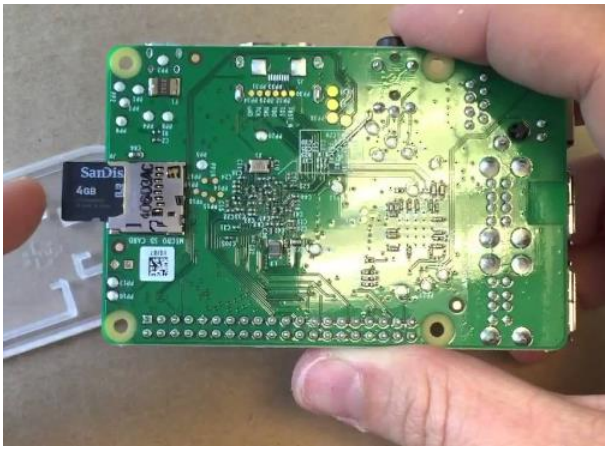


Figura 59: Introducción de la tarjeta SD en la Raspberry

Primera conexión de la Raspberry

1. *Con la tarjeta micro SD ya insertada, alimentamos la Raspberry.*
2. *Conectamos la Raspberry mediante cable Ethernet (conector RJ45) a nuestro router Wifi.*
3. *Abrimos en nuestro PC un programa que nos permita visualizar los equipos conectados a nuestra red, por ejemplo, "Wireless Network Watcher"*
4. *Al cabo de un rato nos debe aparecer la dirección IP que el router le ha asignado a nuestra Raspberry.*

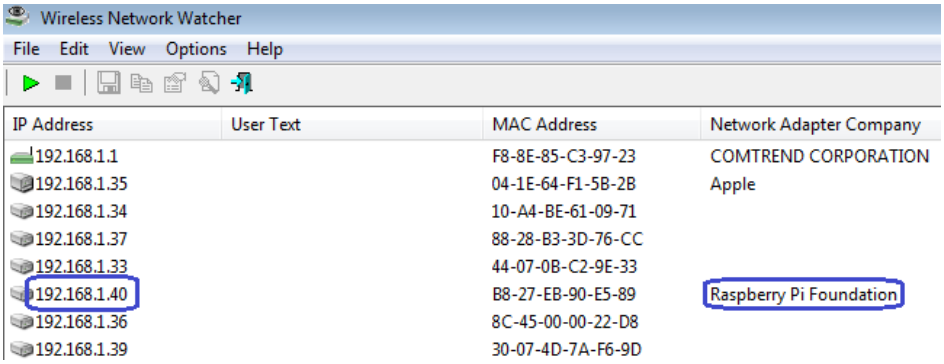


Figura 60: Exploración de equipos con la aplicación Wireless Network Watcher

1. *Abrimos en nuestro PC cualquier aplicación cliente SSH, por ejemplo "Putty", introducimos la dirección IP asignada a la Raspberry, y seleccionamos el protocolo SSH (puerto 22):*

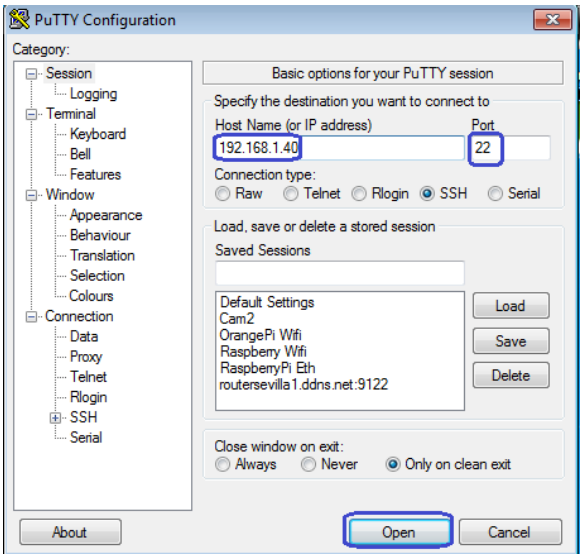


Figura 61: Aplicación Putty para la conexión con la Raspberry

2. *Nos dará este aviso de seguridad. Pulsamos en "Sí":*

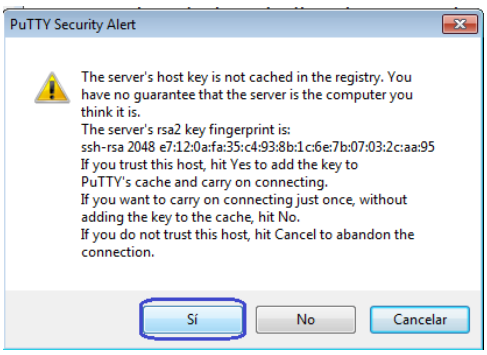


Figura 62: Aplicación Putty. Aviso de primera conexión

3. Nos pedirá usuario y password para validarnos. Introducimos como usuario "pi" y como contraseña "raspberry".

4. Por seguridad cambiaremos la contraseña del usuario "pi" con este comando Linux:

   pi@raspberrypi:~ $ passwd pi

5. Ahora cambiaremos el idioma del sistema operativo. Introducimos el comando:

   pi@raspberrypi:~ $ sudo raspi-config

   Seleccionamos pulsando con la barra espaciadora: "Localisation Options" → "Change Locale" y marcamos es_ES.UTF-8 UTF-8

6. Configuramos la zona horaria de nuestra ubicación:

   pi@raspberrypi:~ $ raspi-config

   Seleccionamos "Localisation Options" → "Change Timezone" → Europe → Madrid

Conexión wifi de la Raspberry y asignación de IP fija

Nuestra Raspberry se va a conectar al router mediante Wifi por lo que es necesaria la configuración de la interfaz inalámbrica en la RPi. Además, debemos asignar una dirección IP fija a la Raspberry de forma que, cuando el router reciba peticiones a la aplicación Web (puerto 8080 de tomcat), las reenvíe a la dirección IP de nuestra Raspberry. Primero veremos cómo se configura la interfaz Wifi en la Raspberry y luego le asignaremos una dirección IP fija tanto a la interfaz Eth como a la interfaz Wifi.

Los comandos que siguen necesitan permiso de superusuario para poderse ejecutar por lo que inicialmente debemos asignar una password al usuario "root".

pi@raspberrypi:~ $ sudo passwd root

Nos responderá con "New password" e introduciremos la contraseña que queramos asignarle al usuario "root". Tendremos que confirmar la clave cuando nos indique "Retype new password".

Ahora ya podremos cambiarnos del usuario "pi" al usuario "root":

pi@raspberrypi:~ $ su

, nos solicitará que introduzcamos la password de "root".

Ahora tenemos que editar el fichero de configuración /etc/wpa_supplicant/wpa_supplicant.conf :

root@raspberrypi:/home/pi# nano /etc/wpa_supplicant/wpa_supplicant.conf

, e introducimos el nombre y contraseña de las redes wifi a las que queremos que se conecte la Raspberry. Podemos configurar más de una asignándole prioridad, teniendo en cuenta que cuando más alto sea el valor de "priority" mayor prioridad. Dejamos el fichero por ejemplo así:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network={
        ssid=" nombre_de_la_red_wifi_3"
        psk=" clave_de_la_red_wifi_3"
        id_str=" red_de_mayor_prioridad "
        priority=3
}

network={
        ssid=" nombre_de_la_red_wifi_2"
        psk=" clave_de_la_red_wifi_2"
        id_str="red_de_prioridad_intermedia"
        priority=2
}

network={
        ssid="nombre_de_la_red_wifi_1"
        psk="clave_de_la_red_wifi_1"
        id_str="red_de_menor_prioridad"
        priority=1
}
```

Ahora vamos ahora a asignarle una dirección IP fija (privada) a las interfaces de nuestra Raspberry. Editamos el fichero de configuración /etc/dhcpcd.conf

```
root@raspberrypi:/home/pi# nano /etc/dhcpcd.conf
```

y al final del fichero le añadimos:

```
interface eth0
static ip_address=192.168.1.99/24
static routers=192.168.1.1
static domain_name_servers=80.58.61.250 80.58.61.254

interface wlan0
static ip_address=192.168.1.100/24  #Dirección IP deseada
static routers=192.168.1.1      #IP de nuestro router
static domain_name_servers= 80.58.61.250 80.58.61.254
```

Con esto hemos asignado la dirección IP 192.168.1.99 a la interfaz Ethernet de la Raspberry, y la dirección IP 192.168.1.100 a la interfaz Wifi.

Reseteamos la Raspberry:

```
root@raspberrypi:/home/pi# reboot
```

Ya podemos desconectar el cable Ethernet de la Raspberry con el router y acceder vía Wifi desde nuestro PC con la aplicación cliente "Putty".

Configuración del servicio SSH de la Raspberry

SSH es un servicio que nos permite acceder remotamente a un equipo distante (en nuestro caso, la Raspberry) como si estuviéramos directamente conectados. Este servicio ya está instalado en el sistema operativo Raspbian que hemos descargado pero vamos a modificar el puerto de escucha (por defecto, el 22). Esto lo hacemos debido a existen operadores de red (por ejemplo MoviStar) que no permiten abrir el puerto 22 en el router, además de evitar accesos no autorizados al desconocerse el puerto en que se ha configurado el servicio SSH.

En el mismo fichero /etc/ssh/sshd_config cambiamos esta línea:

#Port 22

, por esta obra:

Port 9122

Reseteamos la Raspberry:

root@raspberrypi:/home/pi# reboot

Ya podemos acceder a la Raspberry a través de SSH por el puerto 9122, con el usuario "pi":

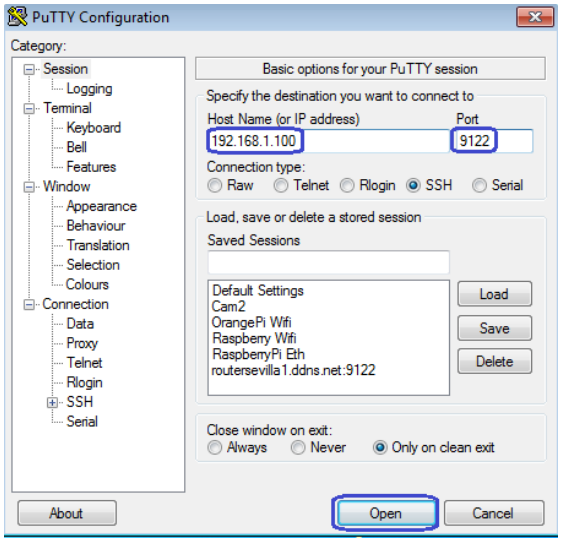


Figura 63: Acceso a la Raspberry con Putty por el puerto 9122

También por seguridad únicamente vamos a permitir al usuario "pi" el acceso al servicio SSH:

1. *Editamos el fichero de configuración SSH*

   root@raspberrypi:/home/pi# nano /etc/ssh/sshd_config

2. *Al final del fichero añadimos*

AllowUsers pi

Con el siguiente comando podremos conocer los intentos fallidos de accesos :

root@raspberrypi:/home/pi# cat /var/log/auth.log | grep 'invalid user\|Failed'

Instalación y configuración del servicio DDNS en la Raspberry

DDNS son las iniciales de Dynamic Domain Name System, es decir, sistema dinámico de nombres de dominio, y se utiliza para la resolver los nombres de URLs de equipos tienen una dirección IP dinámica. La URL que le vamos a asignar a nuestra aplicación es:

http://analizador.ddns.net:8080/TFG

Esta dirección URL debe estar relacionada con la dirección IP que nuestro proveedor de Internet haya asignado a nuestro router. El problema lo tendremos cuando esa dirección IP cambie (es decir, no sea fija) ya que entonces fallará la resolución DNS de ese nombre. La solución pasa por darse de alta en un servidor Dynamic DNS e instalar una aplicación DDNS en nuestra Raspberry, de forma que cada cierto tiempo la Raspberry informe al servidor DDNS de la dirección IP asignada, pudiendo entonces resolver la URL mediante consulta del DNS del peticionario al servidor DDNS.

Un servicio DDNS gratuito se ofrece, por ejemplo, en www.no-ip.com mediante el previo registro.

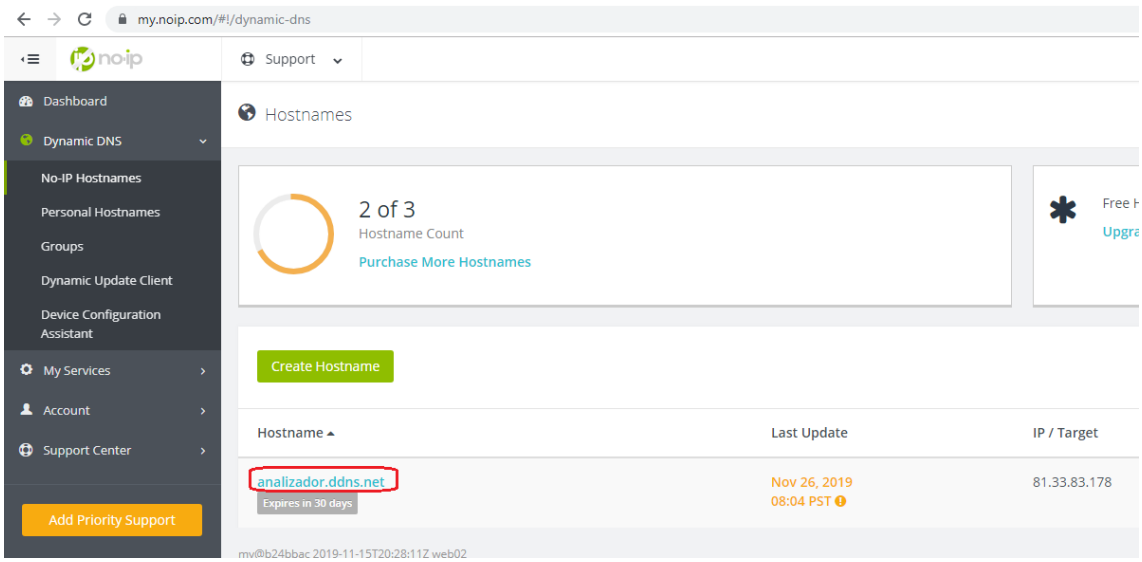


Figura 64: Página Web www.noip.com

Una vez que nos hemos dado de alta, tenemos que instalar la aplicación cliente DDNS de "no-ip" en nuestra Raspberry. Procedemos de la siguiente forma:

1. *Nos cambiamos al directorio personal del usuario "pi":*

   root@raspberrypi:~# cd /home/pi

2. *Descargamos el fichero noip-duc-linux.tar.gz ubicado en la dirección http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz:*

   root@raspberrypi:/home/pi# wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz

3. *Descomprimimos el fichero descargado:*

   root@raspberrypi:/home/pi# tar -zxvf noip-duc-linux.tar.gz

4. *Nos cambiamos al directorio /home/pi/no-ip-2.1.9-1:*

   root@raspberrypi:/home/pi# cd noip-2.1.9-1

5. *Ejecutamos el comando "make":*

   root@raspberrypi:/home/pi/noip-2.1.9-1# make

6. *Ejecutamos el comando "make install":*

   root@raspberrypi:/home/pi/noip-2.1.9-1# make install

7. *Nos pide que introduzcamos el usuario y la clave con el que nos hemos registrado en www.no-ip.com:*

   Please enter the login/email string for no-ip.com _____
   Please enter the password for user 'xxxxxx' _____

   , y nos responderá con nuestra URL si es que únicamente tenemos un nombre registrado en el servidor no-ip:

   *Only one host **analizador.ddns.net** is registered to this account.*
   It wil be used.

   Si tuviéramos más de un nombre registrado nos preguntaría si queremos actualizar todos ellos:

   Do you wish to have them all updated?[N] (y/N)

   , a lo que responderíamos "N". Después nos iría preguntando uno por uno cual es el nombre que queremos mantener actualizado:

   Do you wish to have host [analizador.ddns.net] updated?[N] (y/N)

8. *Ahora nos pedirá que introduzcamos el intervalo de actualización, es decir, cada cuánto tiempo (en minutos) la Raspberry informará al servidor DDNS de la dirección IP actual. Dejamos por ejemplo 5 minutos:*

   *Please enter an update interval: [30] **5***

9. *Después nos preguntará si queremos ejecutar algún "script" cada vez que se actualice la dirección IP. Contestamos "N":*

   *Do you wish to run something at successful update?[N] (y/N) **N***

10. *Ahora tenemos que configurar el cliente DDNS para que se inicie con el arranque de la Raspberry. Creamos el fichero script /etc/init.d/noip2*

root@raspberrypi:/home/pi/noip-2.1.9-1# nano /etc/init.d/noip2

11. *Copiamos este texto en el fichero anterior:*

```
#! /bin/bash
### BEGIN INIT INFO
# Provides: blabla
# Required-Start: $syslog
# Required-Stop: $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: blabla
# Description:
#
### END INIT INFO
sudo /usr/local/bin/noip2
```

Salimos del fichero con CTRL-X

12. *Damos al script los permisos de ejecución:*

root@raspberrypi:/home/pi/noip-2.1.9-1# chmod +x /etc/init.d/noip2

13. *Ponemos el script la cola de ejecución para que se inicie cuando arranquemos la Raspberry:*

root@raspberrypi:/home/pi/noip-2.1.9-1# update-rc.d noip2 defaults

14. *Iniciamos el servicio:*

root@raspberrypi:/home/pi/noip-2.1.9-1# /usr/local/bin/noip2

Ya tenemos configurado en la Raspberry el servicio DDNS.

Port Forwarding en el router ZTE H108N

También denominado "redireccionamiento" o "apertura de puertos" en el router. Cuando el router reciba peticiones desde el exterior de la red para un servicio determinado éste debe conocer la dirección IP privada del equipo de su red que la va a gestionar. Para ello tenemos que asociar en el router cada servicio (puerto) con la dirección IP privada de la máquina de su red que lo va a atender.

En nuestro caso la Raspberry va a gestionar dos servicios:

SSH (configurado en el puerto 9122) para poder introducir comandos a la Raspberry de forma remota.

HHTP (configuraremos Tomcat en el puerto 8080) para atender las peticiones web.

El redireccionamiento en el router ZTE H108N se realiza mediante el siguiente proceso:

1. *Desde el navegador de nuestro PC conectado a la misma red introducimos la dirección IP privada del router. En nuestro caso: http://192.168.1.1*

2. *Pulsamos en "Configurar aplicaciones y puertos". Nos pedirá una clave, que por defecto para equipos de Telefónica es 1234.*

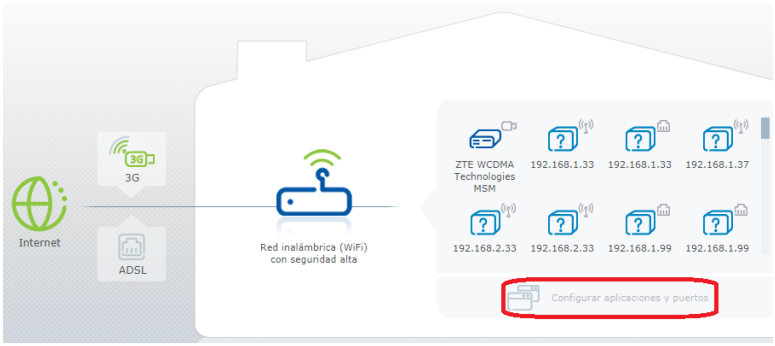


Figura 65: Web router ZTE H108N

3. *Abrimos el puerto 9122 del protocolo TCP asignándolo a la dirección IP privada de nuestra Raspberry. Pulsamos en "Guardar cambios".*

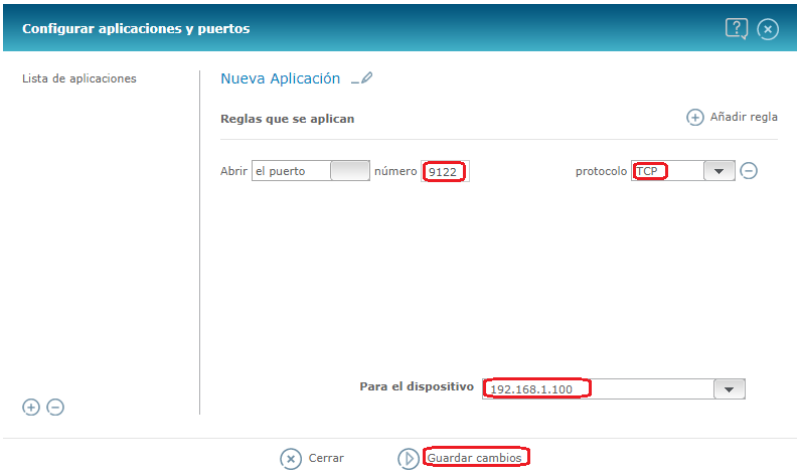


Figura 66: Apertura del puerto 9122 en el router ZTE H108N

4. *Pulsamos sobre "Añadir regla".*

5. *De la misma forma haremos para abrir el puerto 8080. Este puerto también lo atenderá la IP de la Raspberry:*

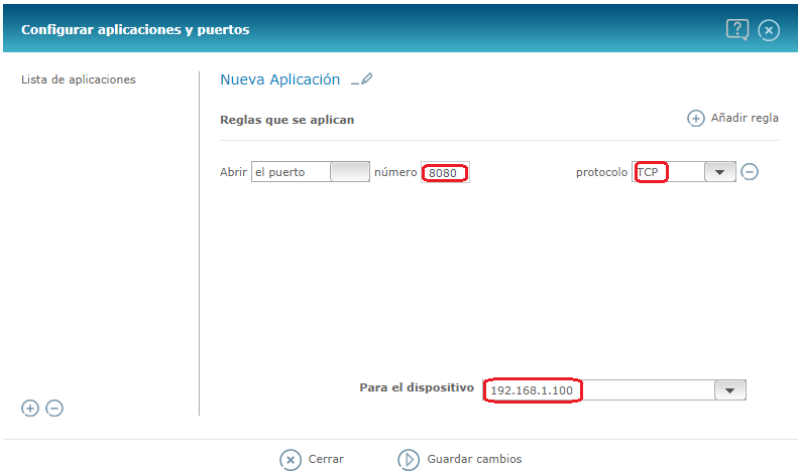


Figura 67: Apertura del puerto 8080 en el router ZTE H108N

6. *Pulsamos en "Cerrar"*

Ya tenemos atendidos los puertos 9122 y 8080 en nuestro router 3G.

Conexión directa del módem 3G ZTE MF190 en la Raspberry

Podemos prescindir del router 3G mediante la conexión directa del módem a la Raspberry. Para ello deberemos instalar estas aplicaciones:

- *ppp: para poder manejar el protocolo ppp*
- *Sakis3g: para configurar la interfaz 3G*
- *Umtskeeper: para permitir las reconexiones del módem 3G*

El procedimiento de instalación sería el siguiente:

1. *Conectamos el módem 3G a la Raspberry.*

2. *Nos situamos en una carpeta cualquiera de la Raspberry, por ejemplo, "home/pi":*

   root@raspberrypi:~# cd /home/pi

3. *Instalamos la aplicación "ppp" para poder establecer la conexión entre el nodo del operador móvil y nuestra interfaz 3G:*

   root@raspberrypi:/home/pi# apt-get install ppp

4. *Creamos la carpeta "umtskeeper"*

   root@raspberrypi:/home/pi# mkdir umtskeeper

5. *Nos situamos en dicho directorio:*

   root@raspberrypi:/home/pi# cd umtskeeper

6. *Descargamos la aplicación comprimida "umtskeeper" que nos permitirá realizar reconexiones del módem 3G:*

   root@raspberrypi:/home/pi/umtskeeper# wget http://mintakaconciencia.net/squares/umtskeeper/src/umtskeeper.tar.gz

7. *Descomprimimos el fichero descargado:*

   root@raspberrypi:/home/pi/umtskeeper# tar -xzvf umtskeeper.tar.gz

8. *Establecemos permisos de ejecución a estos dos ficheros:*

   root@raspberrypi:/home/pi/umtskeeper# chmod +x umtskeeper sakis3g

9. *Abrimos la aplicación sakis3g que se encuentra en la carpeta "umtskeeper":*

   root@raspberrypi:/home/pi/umtskeeper# ./sakis3g --interactive

10. *Seleccionamos consecutivamente:*

    "Connect with 3G"
    "USB device"
    "ZTE WCDMA Technologies MSM"
    "Interface #4"
    "Movistar (movistar.es)"

11. *Nos informará que el módem 3G ya está conectado a la red móvil. El led verde del módem dejará de está parpadeando quedándose fijo. Aceptamos y salimos de la aplicación sakis3g.*

12. *Ahora tenemos que asegurar que la conexión 3G se restablezca en caso de caídas de la red 3G o reconexiones del módem. Primero comprobamos el correcto funcionamiento del siguiente comando:*

    /home/pi/umtskeeper/umtskeeper --sakisoperators "USBINTERFACE='4' OTHER='USBMODEM' USBMODEM='19d2:0124' APN='movistar.es' CUSTOM APN='movistar.es' APN USER='MOVISTAR' APN PASS='MOVISTAR'" --sakisswitches "--sudo --console" --devicename 'ZTE' --log --nat 'no'

    El valor de USBINTERFACE es el que seleccionamos en el punto 10, valor del "interface".
    El valor de USBMODEM se puede encontrar en el campo ID al ejecutar el comando "lsusb".
    Los valores APN, CUSTOM, APN USER, APN PASS son los proporcionados por nuestro operador de red móvil.

    Si todo ha ido bien, el resultado del comando será "Success… we are online!"

13. *Pulsamos CTRL+C para salir de la ejecución del comando anterior. Podemos ya comprobar que al reconectar el módem 3G se vuelve a tener acceso a Internet pasados unos 40 segundos.*

14. *Para ejecutar el comando anterior con cada inicio de la Raspberry haremos lo siguiente. Editamos el fichero /etc/rc.local*

    root@raspberrypi:/home/pi/umtskeeper# nano /etc/rc.local

15. *Al final del fichero (antes del "exit 0") añadimos lo siguiente (ejecución en segundo plano del comando anterior):*

/home/pi/umtskeeper/umtskeeper --sakisoperators "USBINTERFACE='4' OTHER='USBMODEM' USBMODEM='19d2:0124' APN='movistar.es' CUSTOM APN='movistar.es' APN USER='MOVISTAR' APN PASS='MOVISTAR'" --sakisswitches "--sudo --console" --devicename 'ZTE' --log --silent --monthstart 8 --nat 'no' &

16. *Reiniciamos la Raspberry*

Hay que tener en cuenta que en este escenario (módem 3G conectado directamente a la Raspberry) nuestro operador (movistar) no nos va a permitir entrar por SSH a la dirección IP pública asignada. Una solución sería utilizar el "SSH inverso" a costa de implicar a una tercera máquina con acceso SSH. Obviaremos esto ya que en este trabajo no vamos a necesitar conectarnos a la Raspberry vía SSH desde fuera de nuestra red.

Instalación y configuración de Tomcat en la Raspberry

Un servidor de aplicaciones Web es un programa servidor que aloja uno o varios programas de aplicación con contenido http, en nuestro caso, la aplicación que informará de los parámetros eléctricos de la red. El servidor que hemos elegido para este trabajo es Apache-Tomcat.

En este apartado configuraremos el servidor de aplicaciones Web Apache-Tomcat. Apache fue lanzado en 1995 y es desarrollado y mantenido por una comunidad de usuarios con la supervisión de la Apache Software Fundation dentro del proyecto HTTP Server. Es uno de los servidores Web más usado en entornos no empresarias además de ser gratuito.

Proceso de instalación de Tomcat8:

1. *Actualizamos la lista de paquetes disponibles en los distintos servidores así como la versión en la que se encuentran:*

   root@raspberrypi:/home/pi/ # apt-get update

2. *Actualizamos los paquetes de nuestra Raspberry:*

   root@raspberrypi:/home/pi/ # apt-get upgrade

3. *Comprobamos si tenemos Java instalado ya que Tomcat lo necesita:*

   root@raspberrypi:/home/pi/ # java -version

   Si no tenemos java instalado nos contestará con "command not found". En caso de tenerlo instalado nos saltaríamos al punto 5.

4. *Instalamos Java:*

   root@raspberrypi:/home/pi/ # apt-get install default-jdk

5. *Instalamos Tomcat8:*

   root@raspberrypi:/home/pi/ # apt-get install tomcat8

6. *Instalamos Tomcat8-admin, que nos facilitará el despliegue de aplicaciones web:*

root@raspberrypi:/home/pi/ # apt-get install tomcat8-admin

7. *Creamos un usuario con el rol de manager para acceder a tomcat8-admin. Para ello editamos el fichero /var/lib/tomcat8/conf/tomcat-users.xml*

   root@raspberrypi~#: nano /var/lib/tomcat8/conf/tomcat-users.xml

   , y le añadimos justo antes de </tomcat-users> lo siguiente (debemos tener cuidado con las comillas si vamos a copiar el comando pues puede hacerlo erróneamente):

   <user username="usu" password="usu" roles="manager-gui,admin.-gui"/>

   Hemos creado el usuario de tomcat8-admin usu con password usu

8. *Reiniciamos Tomcat:*

   root@raspberrypi~#: service tomcat8 restart

9. *Comprobamos que Tomcat ha quedado correctamente instalado introduciendo la dirección IP de la Raspberry con el puerto 8080 (puerto de escucha por defecto del servidor Tomcat):*

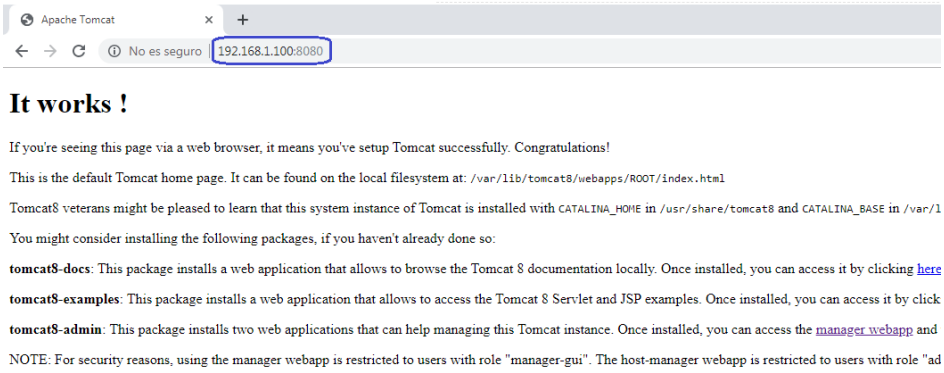


Figura 68: Página inicial de Tomcat

10. *Pulsamos en manager webapp para comprobar que tenemos acceso al "manager" de Tomcat, y nos debe aparecer una ventana en el que tendremos que introducir el usuario y clave del manager de Tomcat8 (en nuestro ejemplo usu/usu):*

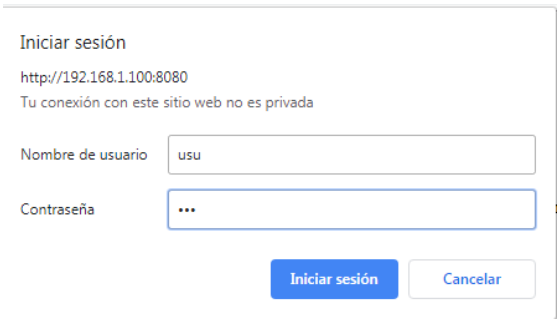


Figura 69: Validación acceso al "manager" de Tomcat

11. *Al pulsar en "Iniciar sesión" nos aparecerá el Gestor de Aplicaciones Web de Tomcat:*

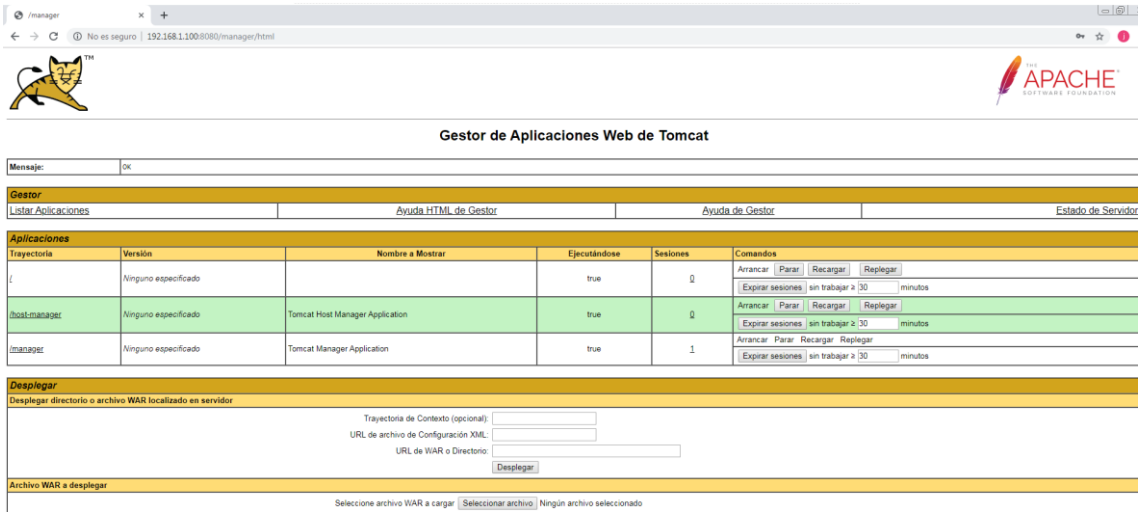


Figura 70: Gestor de aplicaciones de Tomcat

12. *Nuestra aplicación web usará la técnica CGI (Common Gateway Interface), concretamente programas ejecutables escritos en C++, por lo que es necesario marcar el contexto como "privilegiado" para que Tomcat pueda manejarlos. Para ello editamos el fichero /var/lib/tomcat8/conf/context.xml*

    root@raspberrypi~#: nano /var/lib/tomcat8/conf/context.xml

, y modificamos la etiqueta

    <Context>

, por:

    <Context privileged="true">

13. *Reiniciamos Tomcat:*

    root@raspberrypi~#: service tomcat8 restart

Protocolo UART

El protocolo UART es un protocolo de comunicación digital serie asíncrono en el que los bits se envían uno detrás de otro, y para ello se requiere la conversión de los bytes del bus de datos en un flujo de bits, y viceversa.

En el esquema general de los bloques de un sistema UART se pueden observar los registros de datos, tanto de recepción como de transmisión, así como sus correspondientes registros de desplazamiento.

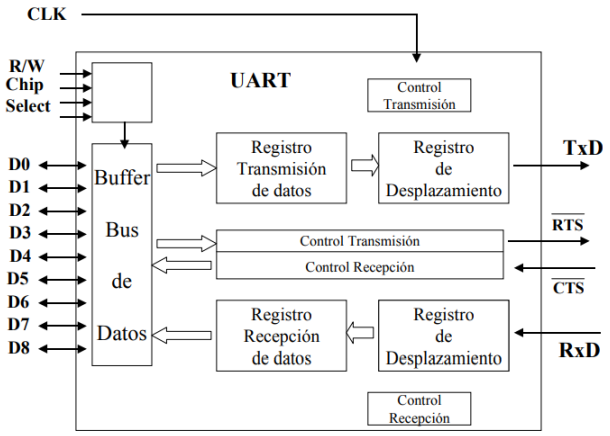


Figura 71: Esquema de bloques del protocolo UART

La sincronización en la transmisión de datos se inicia enviando un bit de comienzo (bit start), para después enviar los 8 bits de datos (entre 5 y 9 bits, dependiendo de la configuración) comenzando por el bit menos significativo (LSB). Por último se envía uno o dos bits de parada (dependiendo de la configuración). Este proceso se repite para los siguientes datos de información.

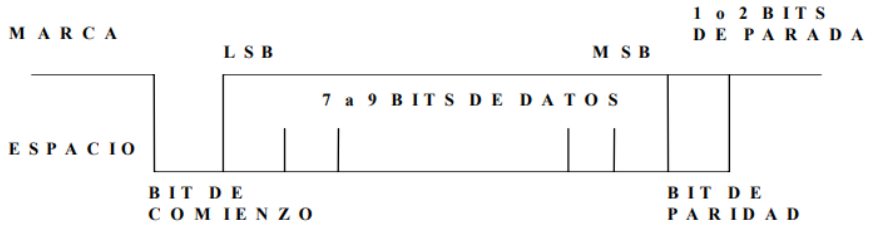


Figura 72: Sincronización protocolo UART

En nuestro proyecto este protocolo será el que utilizaremos para la comunicación entre el Arduino y la Raspberry. La Raspberry solicitará al Arduino el inicio de toma de lecturas de tensión y corriente enviando el carácter "", y el Arduino responderá a la Raspberry con las muestras tomadas transmitiendo 4 caracteres por muestra. Por ejemplo, la muestra que el Arduino tome con valor 923 se enviará a la Raspberry como "0923".*

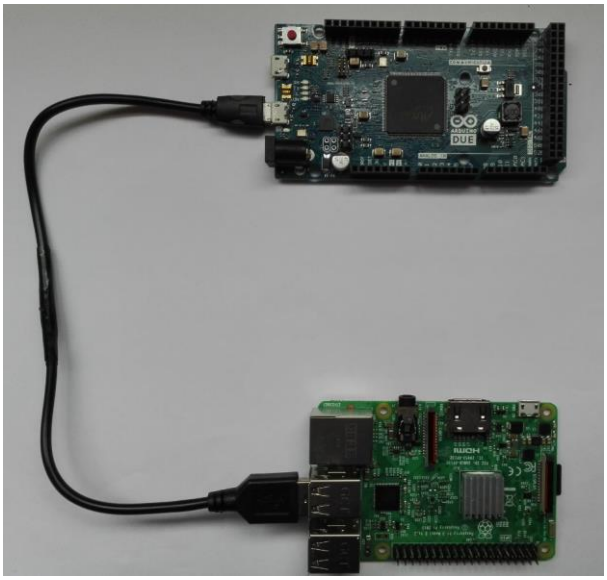


Figura 73: Conexión UART entre Raspberry y Arduino

En el Arduino el protocolo UART está habilitado desde el momento que se enciende la placa, pero en la Raspberry necesitamos realizar la configuración de dicho protocolo. Lo realizamos así:

1. *Abrimos el menú gráfico de configuración de la Raspberry:*

   root@raspberrypi:/var/lib/tomcat8/webapps/TFG#: raspi-config

2. *Seleccionamos "Interfacing Options" y pulsamos en "<Select>":*

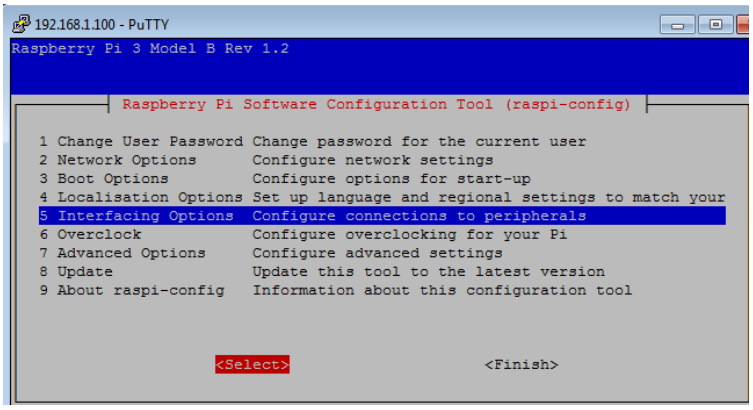


Figura 74: Menú de configuración de la Raspberry

3. *Pulsamos en "P6 Serial" y luego en "<Select>".*

4. *Nosotros vamos a usar el protocolo serie para comunicarnos con otro dispositivo (el Arduino), no para controlar a la Raspberry, por lo que a la siguiente pregunta contestamos "No":*

   Would you like a login shell to be accessible over serial?

5. *A la pregunta de que si queremos habilitar el hardware del protocolo serie contestamos "Sí":*

   Would you like the serial port hardware to be enable?

6. *Confirmamos pulsando en "Aceptar".*

7. *Pulsamos en "Finish".*

8. *Reseteamos la Raspberry contestando afirmativamente a la pregunta:*

   Would you like to reboot now?

9. *La conexión entre la Raspberry y el Arduino la realizaremos físicamente mediante los puertos USB por lo que necesario habilitar permisos de lectura, escritura y ejecución al usuario "tomcat8" sobre el fichero que está asociado a la interfaz USB, /dev/ttyACM0. Lo hacemos con el comando:*

   root@raspberrypi:/var/lib/tomcat8/webapps/TFG#: chmod 667 /dev/ttyACM0

10. *Para que los permisos del fichero /dev/ttyACM0 permanezcan tras reiniciar la Raspberry es necesario incluir el comando anterior en el fichero /etc/rc.local. Este fichero se ejecuta en cada arranque de la Raspberry.*

    Editamos el fichero /etc/rc.local :

    root@raspberrypi:~# nano /etc/rc.local

    , e incluimos el comando de cambio de permisos justo antes de la sentencia "exit 0":

    chmod 667 /dev/ttyACM0

Debemos tener en cuenta que, si estando la Raspberry encendida, reconectamos el cable USB, los permisos del fichero /dev/ttyACM0 se modifican a 660 (usuario "tomcat8" sin permisos) por lo que deberemos volver a introducir el comando chmod 667 /dev/ttyACM0 o reiniciar la Raspberry.

Instalación de la aplicación WiringPi en la Raspberry

Esta aplicación incorpora las librerías de C++ "wiringPi.h" y "wiringSerial.h" necesarias para el manejo de los puertos GPIO y del protocolo serie de la Raspberry. Este protocolo está implementado en los puertos GPIO 15 (Tx) y 16 (Rx) así como en cualquiera de sus interfaces USB:

Para instalar la librería introducimos el comando:

root@raspberrypi:/home/pi#: apt-get install wiringpi

Las funciones que vamos a usar en nuestra aplicación son:

wiringPiSetupGpio(): para configurar los puertos GPIO antes de usarlos.
digitalWrite(): para poner un puerto GPIO a "1" o a "0".
serialOpen(): para abrir el puerto uart
serialClose(): para cerrar el puerto uart
serialPutchar: para enviar un carácter al Arduino
serialDataAvail(): para detectar el evento de envío de un carácter por el Arduino.
serialGetchar(): para leer un carácter que nos haya enviado el Arduino.

Para permitir al usuario "tomcat8" el manejo de los puertos GPIO es necesario que pertenezca al grupo "gpio", y para ello introduciremos este comando Linux en la Raspberry:

```
root@raspberrypi:~# adduser tomcat8 gpio
```

El único puerto que vamos a usar, a parte de los puertos serie, es el GPIO17 para resetear la placa Arduino, uniendo el puerto GPIO17 de la Rasbperry con el Reset del Arduino de forma que al poner GPIO17 a "0" se resetee dicha placa.

El manejo de los puertos GPIO (excepto los correspondientes al puerto serie) requiere que previamente se solicite al sistema su uso, lo que se denomina "exportar", tarea que únicamente puede realizar el usuario "root". Dado que la aplicación web se ejecuta desde el usuario "tomcat8" es necesario que en el arranque de la Raspberry se exporten previamente los puertos que se vayan a utilizar. Para ello, en el fichero de arranque del sistema de la Raspberry, "/etc/rc.local", introduciremos este código justo antes de la sentencia "chmod 667 /dev/ttyACM0" (que ya incluimos en el apartado "protocolo UART"):

```
#Exportamos el pin GPIO17 (BCM) para poder usar por ejemplo digitalWrite()
echo 17 > /sys/class/gpio/export

#Configuramos el pin GPIO17 como de salida
echo out > /sys/class/gpio/gpio17/direction

#Ponemos el pin GPIO17 a 1:
echo 1 > /sys/class/gpio/gpio17/value
```

Con el comando "echo 17 > /sys/class/gpio/export" exportamos el puerto GPIO17.
Con "echo out > /sys/class/gpio/gpio17/direction" lo configuramos como de salida.
Con "echo 1 > /sys/class/gpio/gpio17/value" lo ponemos a "1" para que no se resetee la placa Arduino.

Instalación del IDE de Arduino en el PC

Un IDE es un Entorno de Desarrollo Integrado, es decir, una aplicación que nos permite desarrollar software sobre un dispositivo. La carga de programas en la placa Arduino la realizaremos desde nuestro PC con el IDE específico de Arduino. Los programas (también llamados "sketch") se escriben en un lenguaje de programación que está basado en C++.

Este es el proceso para instalar el entorno de desarrollo (IDE):

1. *Entramos en la web oficial de Arduino: https://www.arduino.cc/en/Main/Software*
2. *Pulsamos sobre "Windows ZIP file for non admin install"*

Figura 75: Página de descarga del software de Arduino

3. *Pulsamos en "JUST DOWNLOAD".*

4. *En nuestro PC creamos por ejemplo la carpeta C:/Program Files/Arduino y extraemos aquí el contenido del fichero ".zip" descargado.*

5. *Conectamos la placa Arduino DUE a nuestro PC a través del puerto USB Programming (de los dos que dispone, el más cercano al conector de alimentación según se muestra en la figura), retirando (si la tiene) la resistencia colocada entre los pines Reset y 3.3V Ω (más adelante hablaremos del motivo de esta resistencia):*

Figura 76: Conexión del Arduino DUE al puerto USB Programming

6. *Entramos en la aplicación Arduino que tendremos por defecto en C:/Program Files/Arduino/Arduino.exe, o pulsando sobre el icono del escritorio:*

Figura 77: Icono IDE Arduino

7. *Seleccionamos Herramientas→Placa y comprobamos si aparece "Arduino DUE (Programming port)". En caso afirmativo la seleccionamos y pasamos al punto 11.*

8. *Si no aparece la tarjeta Arduino DUE en el Gestor de Tarjetas, seleccionamos Herramientas→Placa→Gestor de tarjetas, y en "Arduino SAM Boards (32-bits ARM Cortex-M3) by Arduino", seleccionamos la última versión y pulsamos en "Instalar":*

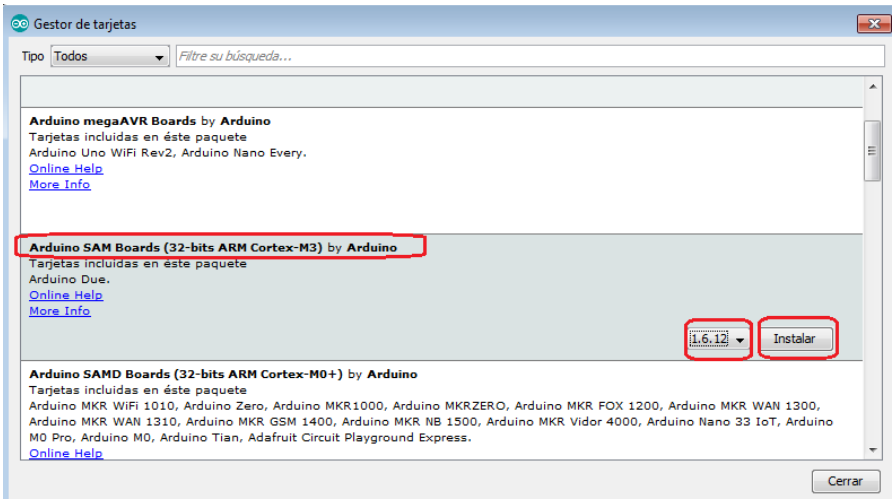


Figura 78: Gestor de tarjetas del IDE de Arduino

9. *Reiniciamos la aplicación IDE Arduino.*
10. *Pulsamos en Herramientas→Placa y seleccionamos "Arduino DUE (Programming port)".*
11. *En Herramientas→Puerto seleccionamos el puerto COM que nos aparezca.*
12. *Introduciremos el código (basado en C++) que aparece en el anexo de este documento.*
13. *Pulsamos en "Subir":*

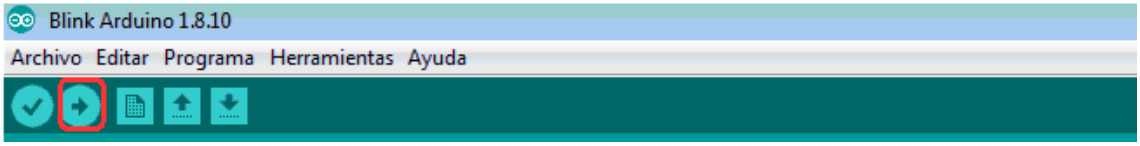


Figura 79: Botón "Subir" del IDE de Arduino

14. *Pulsamos el botón de "RESET" de la placa Arduino.*
15. *Ya tendremos corriendo el programa en la placa Arduino DUE*

Técnica CGI

Nuestra aplicación web, que se cargará en el servidor Tomcat de la Raspberry, se va a escribir siguiendo la técnica CGI. Esta es una técnica web que permite a un cliente (navegador web) solicitar datos a un programa ejecutable. Un programa CGI se puede escribir en cualquier lenguaje que genere un ejecutable. En nuestro caso usaremos C++.

El proceso de actuación de un CGI es el siguiente:

1. *El servidor (Tomcat en nuestro caso) recibe una petición del cliente que ha activado una URL correspondiente a un programa CGI*

2. *El servidor, al identificar que se trata de una URL correspondiente a un CGI, solicita la ejecución de dicho programa, que si está escrito en C++ se compone de sentencias "printf()" con etiquetas HTML como parámetros de dichas sentencias.*
3. *Al ejecutarse el programa CGI se genera un documento HTML en la salida estándar que recibe el servidor (Tomcat).*
4. *El servidor envía el documento HTML al cliente (navegador).*

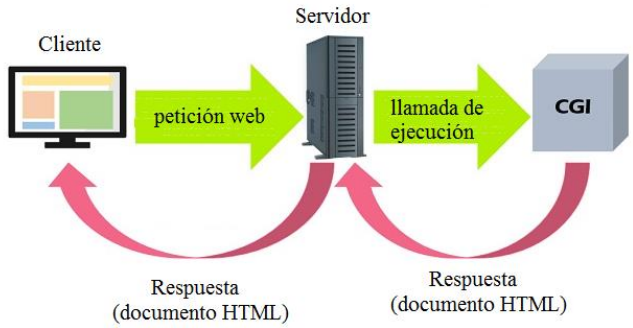


Figura 80: Técnica CGI

Seguridad. Instalación de cortafuegos

Un cortafuegos se utiliza fundamentalmente para evitar que los usuarios de Internet no autorizados tengan acceso a redes privadas conectadas a Internet. Todos los mensajes que entren o salgan de la red privada pasan a través del cortafuegos, que examina cada mensaje y bloquea aquellos que no cumplen los criterios de seguridad especificados.

*En nuestro caso vamos a proteger el acceso a la Raspberry de accesos no autorizados instalando en el mismo el software firewall "ufw". Este cortafuegos es uno de los más sencillos de configurar como indican sus iniciales, "**u**ncomplicated **f**ire**w**all".*

Para su instalación y configuración seguiremos este procedimiento:

1. *Instalamos el cortafuegos:*

   root@raspberrypi:/home/pi# apt-get install ufw

2. *Por defecto el cortafuegos ufw deniega todo el tráfico entrante y permite todo el tráfico saliente. Vamos a modificarlo permitiendo únicamente el acceso al servicio SSH (que tenemos configurado en el puerto 9122) y el acceso al servidor Tomcat (configurado por defecto en el puerto 8080):*

   root@raspberrypi:/home/pi# ufw allow 9122/tcp
   root@raspberrypi:/home/pi# ufw allow 8080/tcp

3. *Activamos el cortafuegos:*

   root@raspberrypi:/home/pi# ufw enable

4. *Para visualizar las reglas introducidas en el cortafuego introduciremos el comando:*

   root@raspberrypi:/home/pi# ufw status

5. *Si quisiéramos restablecer el cortafuegos a valores por defecto introduciremos el comando:*

   root@raspberrypi:/home/pi# ufw reset

Aplicación Web

En este apartado vamos a crear la estructura de la aplicación web en nuestro PC con la aplicación Eclipse y posteriormente la llevaremos a la Raspberry donde incluiremos los programas en C++.

1. *Descargamos de https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers la aplicación "Eclipse IDE for Java EE Developers" y la instalamos en nuestro PC. Una vez instalada, la abrimos pulsando sobre su icono:*

Figura 81: Icono aplicación Eclipse

2. *Seleccionamos File→New→Dynamic Web Project*

3. *Damos nombre a nuestra aplicación y pulsamos en "Finish":*

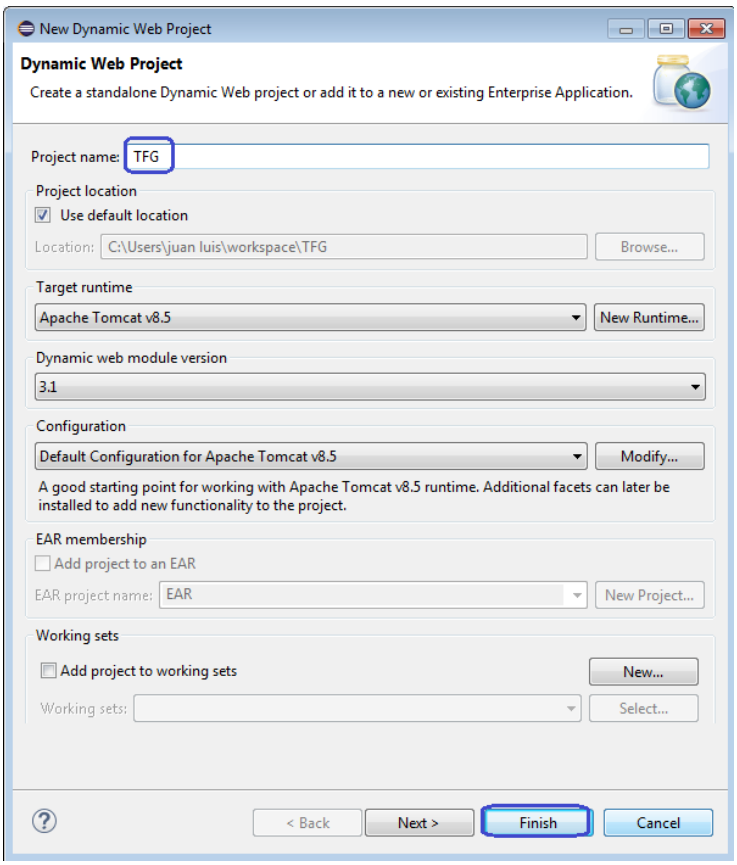


Figura 82: Creación de un nuevo proyecto en Eclipse

4. Ahora crearemos la carpeta que alojará los ficheros CGI. Con el botón derecho del ratón pulsamos en WEB-INF (ubicada en la carpeta WebContent de nuestro proyecto TFG) y seleccionamos New→Folder, e introducimos "cgi" como "Folder name". Pulsamos en "Finish".

5. Vamos a crear el descriptor de despliegue de nuestra aplicación. Con el botón derecho del ratón pulsamos en WEB-INF y seleccionamos New→File, e introducimos "web.xml" como "File name". Pulsamos en "Finish".

6. Hacemos doble clic sobre el fichero recién creado, e incluimos este texto:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
id="WebApp_ID" version="3.1">
  <display-name>TFG</display-name>
  <welcome-file-list>
    <welcome-file>cgi-bin/tfg</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>cgi</servlet-name>
    <servlet-class>org.apache.catalina.servlets.CGIServlet</servlet-class>
    <init-param>
      <param-name>cgiPathPrefix</param-name>
      <param-value>WEB-INF/cgi</param-value>
    </init-param>
    <init-param>
      <param-name>executable</param-name>
      <param-value></param-value>
    </init-param>
    <load-on-startup>5</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>cgi</servlet-name>
    <url-pattern>/cgi-bin/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Vamos a comentar el código anterior:

Primero declaramos el documento xml en versión 1.0 y con juego de caracteres UTF-8:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Este es el elemento raíz del documento XML:

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
id="WebApp_ID" version="3.1">
```

Damos un nombre a la aplicación, TFG:

```
  <display-name>TFG</display-name>
```

Le indicamos a Tomcat que en caso de que la URL que escribe el cliente indique únicamente el directorio de la aplicación (en nuestro caso …/TFG), el recurso que debe atenderlo se encuentra en …/TFG/cgi-bin/tfg:

```
  <welcome-file-list>
    <welcome-file>cgi-bin/tfg</welcome-file>
  </welcome-file-list>
```

Creamos un único servlet con el nombre "cgi":

```
<servlet>
  <servlet-name>cgi</servlet-name>
```

Identificamos la clase que va a atender a este servlet. Al tratarse de un CGI la clase es org.apache.catalina.servlets.CGIServlet:

```
  <servlet-class>org.apache.catalina.servlets.CGIServlet</servlet-class>
```

La búsqueda de los recursos cgi comenzará desde la carpeta WEB-INF/cgi:

```
  <init-param>
    <param-name>cgiPathPrefix</param-name>
    <param-value>WEB-INF/cgi</param-value>
  </init-param>
```

Informamos que el fichero (CGI) es un ejecutable:

```
  <init-param>
    <param-name>executable</param-name>
    <param-value></param-value>
  </init-param>
```

La prioridad en la carga es de 5:

```
  <load-on-startup>5</load-on-startup>
</servlet>
```

Las peticiones con URL …:8080/TFG/cgi-bin/… las atenderán los recursos …:8080/TFG/WEB-INF/cgi/…

```
<servlet-mapping>
  <servlet-name>cgi</servlet-name>
  <url-pattern>/cgi-bin/*</url-pattern>
</servlet-mapping>
</web-app>
```

7. *Ahora tendremos que llevarnos la aplicación de nuestro PC a la Raspberry. Con el botón derecho del ratón pulsamos en TFG y seleccionamos Export→WAR file, e introducimos un nombre al fichero WAR en el campo "Destination", por ejemplo "F:\TFG.war". Pulsamos en "Finish".*

8. *Una vez guardada la aplicación, abrimos el manager de Tomcat8 introduciendo en el navegador de nuestro PC la dirección http://192.168.1.100:8080/manager, siendo 192.168.1.100 la dirección IP de la Raspberry.*

9. *En el apartado "Desplegar" pulsamos en "Seleccionar archivo" y buscamos el fichero WAR anteriormente guardado. Pulsamos en el botón "Desplegar", que se encuentra justo debajo de "Seleccionar archivo".*

Figura 83: Despliegue de archivo WAR desde Tomcat

10. En la carpeta /var/lib/tomcat8/webapps de la Raspberry se habrá cargado nuestra aplicación TFG.

11. Nos cambiamos al directorio TFG/WEB-INF/cgi

    cd /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi

, y creamos los siguientes ficheros C++:

    /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi/tfg.c
    /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi/rele.c
    /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi/calibrar.c
    /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi/registro.c
    /var/lib/tomcat8/webapps/TFG/WEB-INF/cgi/resetArduino.c

, con el código que aparece en el anexo a este trabajo.

12. Compilamos los fichero anteriores:

    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# g++ -o tfg tfg.c -lwiringPi
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# g++ -o rele rele.c -lwiringPi
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# g++ -o calibrar calibrar.c -lwiringPi
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# g++ -o registro registro.c
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# g++ -o resetArduino resetArduino.c -lwiringPi

13. Damos permiso de ejecución al usuario "tomcat8" para los fichero anteriores:

    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# chmod +x tfg
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# chmod +x rele
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# chmod +x calibrar
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# chmod +x registro
    root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# chmod +x resetArduino

14. Creamos el fichero "datos.txt" en la carpeta /var/lib/tomcat8/webapps/TFG

    root@raspberrypi:/var/lib/tomcat8/webapps/TFG# nano datos.txt

    , con este contenido:

    968
    169
    523.5
    223.0

15. Damos permisos de escritura y lectura a "tomcat8" para el fichero anterior:

    root@raspberrypi:/var/lib/tomcat8/webapps/TFG# chmod 666 datos.txt

16. Creamos la carpeta /var/lib/tomcat8/webapps/TFG /imagenes

    root@raspberrypi:/var/lib/tomcat8/webapps/TFG# mkdir imagenes

17. Guardamos en la carpeta /var/lib/tomcat8/webapps/TFG/imagenes los ficheros de las imágenes:

    on.jpg

off.jpg
recargar.jpg
calibrar.jpg
us.png

18. *Cambiamos el usuario y el grupo a toda la aplicación TFG para que el usuario "tomcat8" pueda ejecutar la aplicación cuando se accede desde un navegador:*

    root@raspberrypi:/var/lib/tomcat8/webapps# chown -R tomcat8 TFG
    root@raspberrypi:/var/lib/tomcat8/webapps# chgrp -R tomcat8 TFG

Con esto ya tenemos preparada la aplicación en la Raspberry. Si también hemos cargado el programa de la placa Arduino (ver código en el anexo), sólo nos queda realizar las conexiones (ver apartado siguiente) y ejecutar la aplicación desde cualquier navegador introduciendo la dirección URL:

analizador.ddns.net:8080/TFG

, o también

192.168.1.100:8080/TFG

, si se accede desde un terminal conectado a la misma red wifi que la Raspberry.

Montaje y funcionamiento general

En la siguiente figura se muestra las conexiones de los distintos elementos del dispositivo.

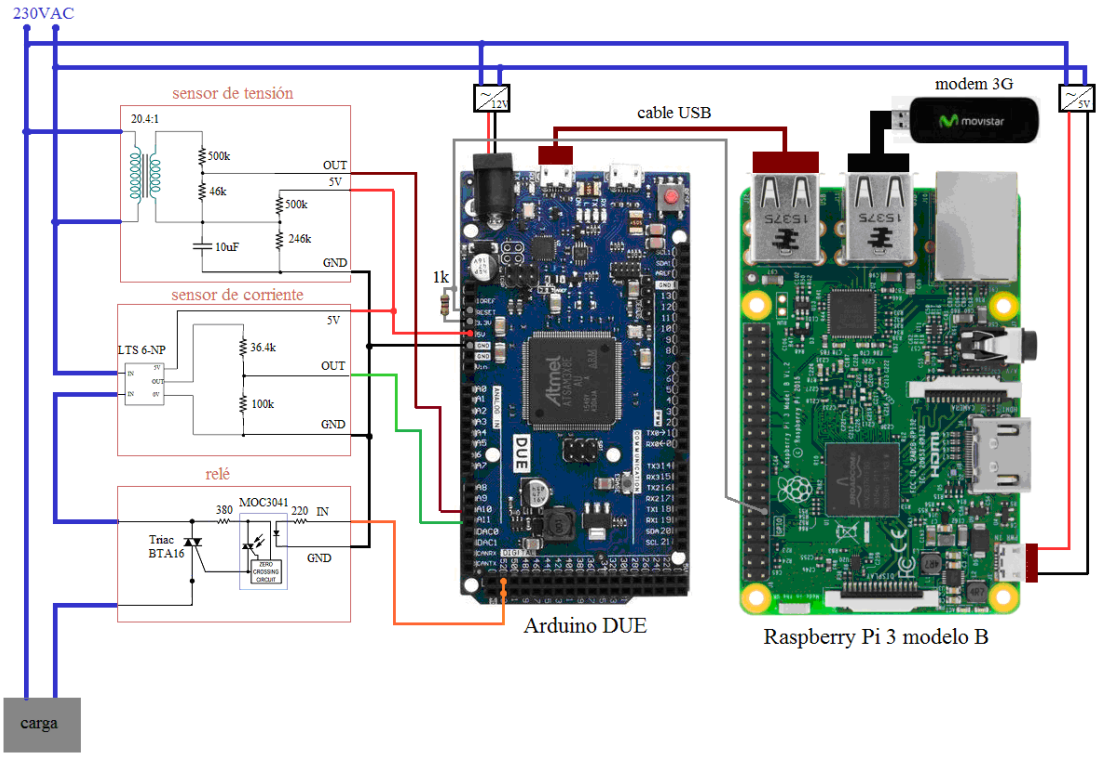


Figura 84: Conexionado del dispositivo

El montaje del dispositivo es el siguiente:

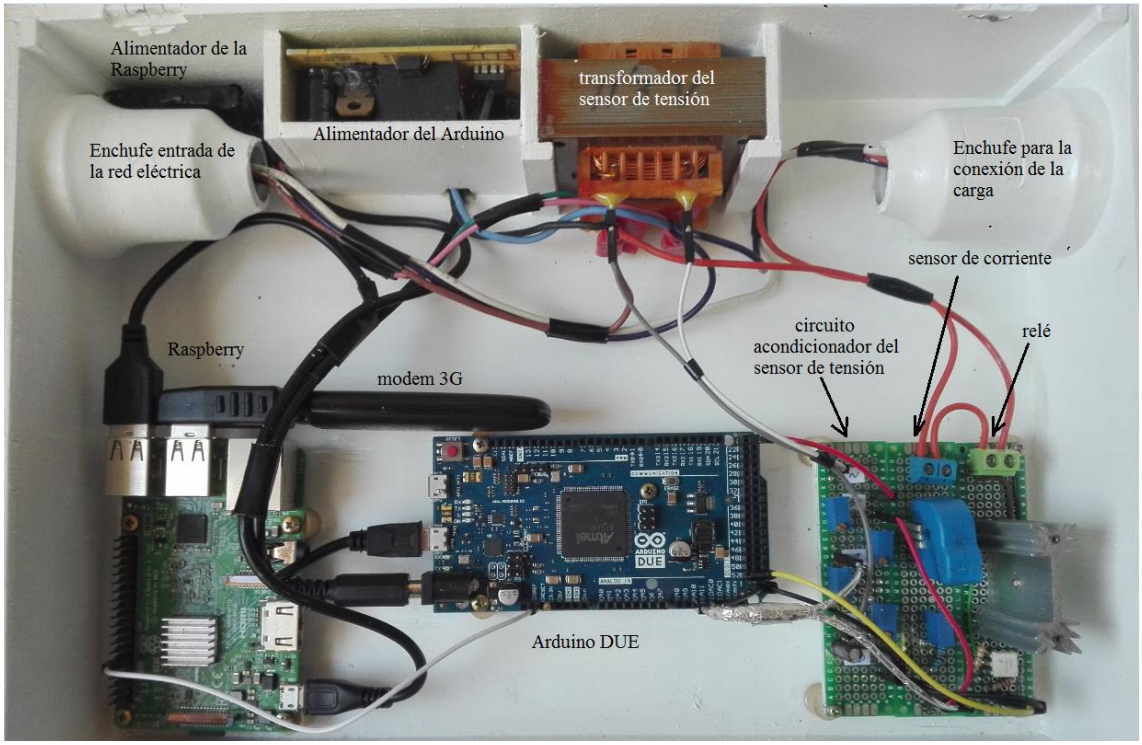


Figura 85: Montaje del dispositivo

El funcionamiento general del sistema se resume en los siguientes pasos:

1. *El transformador junto con el circuito de acondicionamiento asociado forman el sensor de tensión entregando en el pin A10 de la tarjeta Arduino una tensión proporcional a la de la red.*

2. *El sensor de corriente genera en el pin A11 de la tarjeta Arduino una tensión proporcional a la intensidad de la red.*

3. *El relé conecta y desconecta la carga de la red eléctrica en función del valor que se escriba en el pin 53 del Arduino. El estado del relé se puede determinar leyendo el pin 52 del Arduino.*

4. *La tarjeta Arduino toma por su puerto A10 4602 lecturas de tensión (valores de 0 a 1023) en un periodo de 20ms. Tras 40ms (2 periodos) desde el comienzo de la primera lectura de la tensión se procede a tomar por el puerto A11 otras 4602 lecturas de corriente (valores de 0 a 1023). Dado que las medidas se va a tomar transcurridos muchísimos periodos desde la conexión de la carga, y que la ésta es invariable en el tiempo, estamos en un régimen permanente por lo que es indiferente tomar la muestra de corriente en* t_o *o en* $t_o + kT$*, con* k *entero. Podríamos haber optado por repartir las 4602 lecturas para las tensiones y las corrientes capturándolas alternativamente (primero tensión y luego corriente), pero de esta forma haríamos que la carga fuera algo más capacitiva de lo real. Aparte, con capturas alternativas en un mismo periodo tomaríamos la mitad de muestras (2301) que con la solución adoptada por lo que la precisión sería inferior.*

   Con cada 6 lecturas de tensión o corriente el Arduino hace la media (número entero de 0 a 1023) por lo que se generan 767 muestras de tensión y otras 767 de corriente.

5. *La tarjeta Arduino envía a la Raspberry, mediante el protocolo UART, las 767 muestras de la tensión y posteriormente las 767 muestras de la corriente a una velocidad de 115200bps. Cada muestra es un número entero que está en el rango de 0 a 1023 y se remite a la Raspberry como 4 caracteres numéricos.*

6. *La Raspberry traduce estas muestras de tensión y corriente en voltios y amperios.*

7. *La Raspberry calcula la distorsión armónica de la onda de tensión y el factor de potencia de la carga.*

8. *La Raspberry genera un código HTML con las gráficas y cálculos anteriores mediante la técnica CGI.*

9. *Las Raspberry envía el código HTML al router wifi 3G mediante la pila de protocolos HTTP/TCP/IP usando la red Wifi como protocolo de enlace. Si conectáramos el módem 3G directamente en la Raspberry podríamos prescindir del router.*

10. *Enviamos por el interfaz 3G la página HTTP (código HTML) al navegador del cliente usando la red móvil de nuestro operador.*

Código C++ del Arduino

La programación en la tarjeta Arduino se realiza en un lenguaje basado en C++. Vamos a comentar el programa que hemos cargado en dicha tarjeta:

Inicializamos las variables. El pin A10 es el pin por donde leeremos las tensiones, y el pin A11 por donde leeremos la corriente. El pin 53 es un pin digital de salida por donde se activará o desactivará el relé. El pin 52 es un pin digital de entrada por donde leeremos el estado del relé (activado o desactivado). Hay que tener en cuenta que cuando nos conectamos al dispositivo desconocemos si la carga está conectada o no a la red. Este estado lo conoceremos leyendo el pin 52. Los vectores lecturaV y lecturaI almacenarán los valores de las muestras en valores enteros con rango 0 a 1023 pues el Arduino dispone de convertidores analógico-digitales de 10 bits. Los vectores muestraV y muestraI almacenarán las muestras que se enviarán a la Raspberry y se determinarán haciendo la media cada N=6 lecturas correspondientes de tensión y corriente, respectivamente. Por tanto, se toman 767x6 lecturas que corresponderán a 767 muestras.

```
const int N=6; //Número de lecturas que promediarán una muestra de corriente o de tensión
const int numeroMuestras=767;  //Número de muestras de tensiones y corrientes que el Arduino enviará a la Raspberry
const int numeroLecturas=numeroMuestras*N; //Número de lecturas de tensiones y corrientes que tomará en Arduino

const int pinLecturasTensiones=A10;    //Entrada analógica A10 por donde leeremos la tensión
const int pinLecturasCorrientes=A11;   //Entrada analógica A11 por donde leeremos la corriente
const int pinReleEscritura=53;  //Por el pin 53 actuaremos sobre el relé
const int pinReleLectura=52;    //Por el pin 52 leeremos el estado del relé

int lecturaV[1000*N];   //Lecturas de tensiones
int lecturaI[1000*N];   //Lecturas de corrientes
int muestraV[numeroMuestras]; //Muestras de tensiones
int muestraI[numeroMuestras]; //Muestras de de corriente
```

La función "setup" en el lenguaje Arduino se ejecuta una sola vez. Aquí vamos a inicializar el protocolo UART a 115200bps además de configurar los pines digitales que son de entrada y de salida. Los pines analógicos son de entrada en el Arduino que por defecto no hace falta configurarlos como tales:

```
void setup() {
  Serial.begin(115200);          // Inicializamos el puerto serie a esa velocidad
  pinMode(pinReleEscritura,OUTPUT); //Establecemos el pin 53 como de escritura (modificará el estado del relé)
```

```
 pinMode(pinReleLectura,INPUT);    //Establecemos el pin 52 como de lectura (leerá el estado del relé)
}
```

La función "loop" en el lenguaje Arduino se ejecuta indefinidamente.

```
void loop() {
```

Si en el puerto serie no hay datos por leer no hacemos nada, en caso contrario guardamos en la variable "carácter" el primer carácter que nos ha enviado la Raspberry::

```
 char caracter;
 if (Serial.available()>0){         //Si tenemos caracteres por leer en el puerto serie...
  caracter=Serial.read(); //Leemos un caracter del puerto serie
```

Si el carácter que ha enviado la Raspberry al Arduino es un "" le está pidiendo que comience el muestreo:*

```
  if (Serial.read()=='*'){         //Si hemos recibido el caracter '*'
```

Durante 20ms se toman lecturas de tensión para preparar el ADC del Arduino DUE:

```
   //Hacemos lecturas de tensiones de preparación durante 20ms
   long t=micros();  //Anotamos el momento en que se comienzan las lecturas de tensiones
   while (micros()-t<20000){
    analogRead(pinLecturasTensiones); //hacemos lecturas (sin guardarlas) durante 20ms
   }
```

El Arduino procede a tomar las muestras de las tensiones:

```
   //Tomamos las lecturas de tensiones
   t=micros();  //Anotamos el momento en que se comienzan las lecturas de tensiones
   for (int i=0;i<1000*N;i++){
    lecturaV[i]=analogRead(pinLecturasTensiones); //Tomamos las lecturas de tensiones
   }
```

Ahora hay que esperar a tomar la primera lectura de corriente cuando se encuentre en la misma fase que se tomó la primera lectura de tensión. Para ello tendremos que esperar un número entero de periodos. Hemos tomados 2 periodos:

```
   //Esperamos a que pasen 2 periodos desde la primera lectura de la tensión para que
   //las ondas de tensión y corriente estén en fase en caso de una carga resistiva
   while (micros()-t<40085){  //Realmente es algo mas de 2 periodos (40000us)
    //para hacerlo más inductivo tenemos que disminuir esa cantidad
    analogRead(pinLecturasCorrientes); //Lecturas de corrientes de preparación
   }
```

El Arduino procede a tomar las muestras de las corrientes:

```
   //Tomamos las lecturas de las corrientes
   for (int i=0;i<1000*N;i++){
    lecturaI[i]=analogRead(pinLecturasCorrientes); //Tomamos las lecturas de corrientes
   }
```

Ahora calculamos las muestras como la media de cada "N" lecturas:

```
   //Calculamos las muestras como media de cada N lecturas
   for (int i=0;i<numeroMuestras;i++){
    muestraV[i]=0;
    muestraI[i]=0;
    for (int j=0;j<N;j++){
     muestraV[i]=muestraV[i]+lecturaV[N*i+j];
     muestraI[i]=muestraI[i]+lecturaI[N*i+j];
    }
    muestraV[i]=round(muestraV[i]/N);
    muestraI[i]=round(muestraI[i]/N);
   }
```

Enviamos por el puerto serie (UART) a la Raspberry las muestras de las tensiones:

```
char cadena[5]; //Cada muestra se enviará como 4 caracteres mas 1 de fin de caracter, \0
//Enviamos a la Raspberry las muestras de las tensiones
for (int i=0;i<numeroMuestras;i++){
  sprintf(cadena,"%4i",muestraV[i]); //Convertimos la lectura en 4 caracteres
  Serial.print(cadena);
 }
```

Enviamos por el puerto serie (UART) a la Raspberry las muestras de las corrientes:

```
//Enviamos a la Raspberry las muestras de las corrientes
for (int i=0;i<numeroMuestras;i++){
  sprintf(cadena,"%4i",muestraI[i]); //Convertimos la lectura en 4 caracteres
  Serial.print(cadena);
 }
}
```

Si el carácter que ha enviado la Raspberry al Arduino es un "0" le está pidiendo que ponga su salida pin 53 a "0" para desconectar la carga de la red eléctrica:

```
if (caracter=='0'){ //Si es el caracter recibido es un '0'...
  digitalWrite(pinReleEscritura,LOW); //Ponemos el relé a '0'
 }
```

Si el carácter que ha enviado la Raspberry al Arduino es un "1" le está pidiendo que pidiendo que ponga su salida pin 53 a "1" para conectar la carga a la red eléctrica:

```
if (caracter=='1'){ //Si es el caracter recibido es un '1'...
  digitalWrite(pinReleEscritura,HIGH);  //Ponemos el relé a '1'
 }
```

Si el carácter que ha enviado la Raspberry al Arduino es un "2" le está pidiendo que le informe del estado del relé. Para ello el Arduino lee su pin 52 (que está conectado directamente a su pin 53). Hay que tener en cuenta que el pin 53 es de salida por lo que no podemos leerlo, de aquí que hayamos tenido que unir el pin 52 con el 53. Una vez que el Arduino ha leído el pin 52 envía un "1" o un "0" por el puerto serie a la Raspberry en función de que el relé tenga la carga conectada o no:

```
if (caracter=='2'){ //Si es el caracter recibido por el puerto serie es un '2'...
  int lectura=digitalRead(pinReleLectura);  //Leemos el estado del relé
  if (lectura==1) Serial.print("1");  //Si el relé está en '1' enviamos un '1'
  if (lectura==0) Serial.print("0");  //Si el relé está en '0' enviamos un '0'
 }
}
}
```

Código C++ TFG/cgi/tfg.c de la Raspberry

El código en la Raspberry lo hemos generado en C++ y ubicado en la carpeta TFG/cgi. El sistema operativo Raspbian que hemos descargado en la Raspberry ya trae el compilador de C++ por lo que nos es necesaria su instalación.

Una vez escrito el programa, lo compilaremos con el comando:

```
g++ -o tfg tfg.c -lwiringPi
```

Vamos a explicar el código C++ que hemos de instalar en la Raspberry.

Inicialmente declaramos las librerías que vamos a usar:

1. time.h: esta librería la necesitamos para poder usar la función time() y ctime()
2. stdio.h: para poder hacer uso de la función printf()
3. wiringSerial.h: para usar las funciones del protocolo serie
4. match.h: librería con las funciones sqrt(), pow(), sin(), cos(), acos()
5. unistd.h: esta librería es necesaria para poder usar la función sleep()
6. string.h: para poder usar strcpy()

```
//Librerias
#include <time.h>          //para poder usar la funcion time() y ctime()
#include <stdio.h>         //para poder usar la funcion printf
#include <wiringSerial.h>  //para poder usar las funciones del puerto serie
#include <math.h>          //para poder usar las funciones seno, coseno, potencia
#include <unistd.h>        //para poder usar la funcion usleep()
#include <string.h>        //para poder usar strcpy(), strchr()
```

Definimos las directivas (“define”) de los identificadores:

Número de muestras que la Raspberry va a recibir del Arduino tanto de la onda de tensión como de la onda de corriente:

```
#define numeroMuestras 767     //número de muestras que se van a tomar en un periodo de 20ms
```

Número de bits del convertidor ADC del Arduino:

```
#define NBits 10               //Número de bist del convertidor analógico-digital del Arduino
```

Constante pi:

```
#define pi 3.1415926
```

Velocidad del protocolo serie UART:

```
#define bpsUART 115200         //Velocidad del puerto serie UART
```

Sensibilidad del sensor de corriente utilizado (en voltios/amperio):

```
#define senSensorCorr 0.0736   //sensibilidad del sensor de corriente, medido en voltios/amperio (incluido el circuito
                               //de acondicionamiento), medido en voltios/amperio
```

Prototipo de las funciones:

```
void capturarLecturasDelArduino();   //La Raspberruy solicita las muestras al Arduino
void obtenerTensionesCorrientes();   //Traduce las muestras a voltios y amperios
void ordenarMuestras();              //Ordena las muestras para comenzar en fase 0 para la tensión
void obtenerArmonicosTension();      //Obtiene armónicos y distorsión de la tensión
void obtenerArmonicosCorriente();    //Obtiene armónicos y distorsión de la corriente
void obtenerFactorPotencia();        //Obtiene potencias y factor de potencia
void crearHTML();                    //Crea el código HTML
```

```
void ondasPrueba();                 //Genera ondas de prueba para la  tensión y para la corriente
void registro();                    //Registra la dirección IP y fecha de la máquina que ha accedido a la página
```

Definición de variables:

```
int muestraV[numeroMuestras];       //aqui guardamos las lecturas analógicas de tensiones del arduino
int muestraI[numeroMuestras];       //aqui guardamos las lecturas analógicas de corrientes del arduino
int muestraVmax;                    //mayor lectura analógica de tensión que lee el arduino
int muestraVmin;                    //menor lectura analógica de tensión que lee el arduino
float muestraImedia;                 //media de las lectura de corriente que lee el arduino

float tension[numeroMuestras];      //aqui guardamos las tensiones muestreadas (valores en voltios)
float corriente[numeroMuestras];    //aqui guardamos las corrientes muestreadas (valores en amperios)
float potencia[ numeroMuestras];    //aqui guardamos los lvalores de las potencias de cada muestra (valores en vatios)

float Vmax;          //mayor tensión muestreada (en voltios)
int nVmax;           //número de lectura en el que se ha tomado la mayor tensión
float Imax;          //mayor corriente muestreada (en amperios)
int nImax;           //número de lectura en el que se ha tomado la mayor corriente
float Pmax;          //mayor potencia de todas las muestras (en vatios)
int nPmax;           //número de lectura en el que se ha dado la mayor potencia

float Vef;           //tensión eficaz (en voltios)
float Ief;           //corriente eficaz (en amperios)
float Vm;            //tensión media (en voltios)
float Im;            //corriente media (en amperios)

float potenciaP;     //Potencia activa (en vatios)
float potenciaQ;     //Potencia reactiva (en var)
float potenciaS;     //Potencia aparente (en va)
float potenciaD;     //Potencia reactiva de distorsión (var)
float fp;            //Factor de Potencia
float fi_v1;         //fase de la componente fundamental de la tensión
float fi_c1;         //fase de la componente fundamental de la corriente
float cos_fi;        //cos de fi (desfase entre la tensión y la componente fundamental de la corriente)

float VmaxGrafica;       //Tensión máxima que se va a poder dibujar (en voltios)
float ImaxGrafica;       //Corriente máxima que se va a poder dibujar (en amperios)
float PmaxGrafica;       //Potencia máxima que se va a poder dibujar (en vatios)

float arm1t;         //valor eficaz de la componente fundamental de la tensión (en voltios)
float arm2t;         //valor eficaz del armónico 2 de la tensión (en voltios)
float arm3t;         //valor eficaz del armónico 3 de la tensión (en voltios)
float arm4t;         //valor eficaz del armónico 4 de la tensión (en voltios)
float arm5t;         //valor eficaz del armónico 5 de la tensión (en voltios)
float arm6t;         //valor eficaz del armónico 6 de la tensión (en voltios)
float arm7t;         //valor eficaz del armónico 7 de la tensión (en voltios)
float arm8t;         //valor eficaz del armónico 8 de la tensión (en voltios)

float thdt;          //Coeficiente de distorsión armónica total de la tensión
float th2t;          //Coeficiente de distorsión del armónico de la tensión
float th3t;          //Coeficiente de distorsión del armónico 3 de la tensión
float th4t;          //Coeficiente de distorsión del armónico 4 de la tensión
float th5t;          //Coeficiente de distorsión del armónico 5 de la tensión
float th6t;          //Coeficiente de distorsión del armónico 6 de la tensión
float th7t;          //Coeficiente de distorsión del armónico 7 de la tensión
float th8t;          //Coeficiente de distorsión del armónico 8 de la tensión

float arm1c;         //valor eficaz de la componente fundamental de la tensión (en voltios)
float arm2c;         //valor eficaz del armónico 2 de la tensión (en voltios)
float arm3c;         //valor eficaz del armónico 3 de la tensión (en voltios)
float arm4c;         //valor eficaz del armónico 4 de la tensión (en voltios)
float arm5c;         //valor eficaz del armónico 5 de la tensión (en voltios)
float arm6c;         //valor eficaz del armónico 6 de la tensión (en voltios)
float arm7c;         //valor eficaz del armónico 7 de la tensión (en voltios)
float arm8c;         //valor eficaz del armónico 8 de la tensión (en voltios)

float thdc;          //Coeficiente de distorsión armónica total de la tensión
float th2c;          //Coeficiente de distorsión del armónic0 2 de la tensión
float th3c;          //Coeficiente de distorsión del armónic0 3 de la tensión
```

```
float th4c;              //Coeficiente de distorsión del armónic0 4 de la tensión
float th5c;              //Coeficiente de distorsión del armónic0 5 de la tensión
float th6c;              //Coeficiente de distorsión del armónic0 6 de la tensión
float th7c;              //Coeficiente de distorsión del armónic0 7 de la tensión
float th8c;              //Coeficiente de distorsión del armónic0 8 de la tensión

char maquina[30];                //Va a contener la dirección IP de la máquina que solicita la página
```

La función "main" consta de varias llamadas a funciones que describiremos más adelante:

```
int main (int argc, char *argv[], char *env[]) {
 strcpy(maquina,env[2]);          //copiamos en la variable "maquina" la dirección IP de host remoto
 capturarLecturasDelArduino();
 obtenerTensionesCorrientes();
 ordenarMuestras();
 //ondasPrueba();
 obtenerArmonicosTension();
 obtenerArmonicosCorriente();
 obtenerFactorPotencia();
 crearHTML();
 registro();
 return 0;
}
```

La función "registro" obtiene la fecha del sistema y la dirección IP de la máquina que accede a la web y graba en el fichero "registro.txt" esta información así como la tensión, corriente y potencia :

```
void registro(){
```

Obtenemos la fecha del sistema en la variable "fecha":

```
  char fecha[25];//ctime devuelve 26 caracteres pero también se podría usar un puntero de char
  time_t current_time;
  current_time=time(NULL);
  ctime(&current_time);
  strcpy(fecha,ctime(&current_time));
```

Abrimos el fichero "registro.txt" como escritura al final del mismo:

```
  FILE *fp=fopen("../../registro.txt","a"); //abrimos /TFG/registro.txt como escritura al final del mismo
  if (fp==NULL) {
    printf("No se puede abrir el fichero registro.txt desde tfg\n");
  }
```

Sustituimos el carácter fin de línea (\n) que incorpora la variable fecha por un fin de cadena (\0) para que no nos aplique el salto de línea al imprimir "fecha":

```
  fecha[sizeof(fecha)-1]='\0'; //sustituimos el último carácter, \n, de fecha por un \0
```

Escribimos en el fichero la dirección IP remota, fecha, tensión, corriente, y potencia. La variable máquina tiene la forma "REMOTE_HOST=xxx.xxx.xxx.xxx", siendo "xxx.xxx.xxx.xxx" la dirección IP de host remoto. Para eliminar el texto "REMOTE_HOST" usamos la función "strchr" que se encuentra en la librería "string.h":

```
  fprintf(fp,"IP remoto:%s  Fecha:%s Vef:%.1fV Ief:%.3fA P:%.1fW\n",strchr(maquina,'=')+1,fecha,Vef,Ief,potenciaP);
```

Cerramos el fichero:

```
  fclose(fp);
  return;
}
```

La función "ondasPrueba" genera una onda de tensión de 230V (eficaces) y una onda de corriente de 5A (eficaces) desfasada 22.5º en atraso. Esta función la usamos para realizar comprobaciones de

parámetros. Si hacemos uso de esta función no debemos llamar a las funciones "capturarLecturasDelArduino" , "obtenerTensionesCorrientes" y "ordenarMuestras()":

```
//Genera ondas de prueba de tensión y corriente
//Para llamar a ondasPrueba() no debemos llamar a ninguna de estas funciones:
//              capturarLecturasDelArduino();
//              obtenerTensionesCorrientes();
//              ordenarMuestras();
void ondasPrueba(){
 for (int i=0;i<numeroMuestras;i++){
   tension[i]=230.0*sqrt(2)*sin(100*pi*0.02*i/(numeroMuestras-1));
   corriente[i]=5.0*sqrt(2)*sin(100*pi*0.02*i/(numeroMuestras-1)-pi/8); //carga inductiva
   potencia[i]=tension[i]*corriente[i];
 }
 ImaxGrafica=8.0;
 VmaxGrafica=230.0*sqrt(2);
 PmaxGrafica=5000.0;
}
```

La función "capturarLecturasDelArduino" tiene como objetivo recoger las muestras (números enteros en el rango 0 a 2^Nbits-1, siendo "Nbits" en el números de bits del ADC) tomadas por el Arduino y almacenarlas en los vectores muestraV[] y muestraI[], además de determinar la máxima muestra de tensión (muestraVmax) y la mínima (muestraVmin), corriente media (muestraImedia), así como los números de muestra en que se dan las muestras máximas de tensión y corriente (nVmax, nImax). Las muestras máximas y los instantes en que se dan los utilizaremos para mostrar en las gráficas de tensión y de corriente los valores máximos (ver función crearHTML()). Las valores muestraVmax, muestraVmin y muestraImedia los usaremos para convertir las muestras a voltios y amperios en la rutina "obtenerTensionesCorriente()".

```
//Recoge las lecturas tomadas por el arduino
void capturarLecturasDelArduino(){
```

Inicializamos el puerto serie (UART) a bpsUART=115200 bits por segundo como medio de comunicación con el Arduino:

```
 int fd;
 if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
   printf("No se puede abrir el dispositivo serie ttyACM0\n");
   return;
 }
```

Esperamos unos 150us para que se inicialice el puerto serie:

```
 usleep(150);
```

Enviamos al Arduino el carácter "" para ordenarle que comience a tomar lecturas:*

```
 int datoLeido;
 serialPutchar(fd,'*'); //Enviamos este carácter al arduino;
```

Cada muestra que obtiene el Arduino es un número entero que va del 0 al 2^Nbits-1, siendo "Nbits" el número de bits del ADC, y que remite a la Raspberry por el puerto serie como 4 caracteres, empezando por el más significativo. La función serialDataAvail() comprueba que la Raspberry tenga algún carácter por leer, y en caso afirmativo lo captura con la función serialGetchar(). En la variable "total" vamos acumulando las sumas parciales que corresponde cada dígito de cada muestra. Una vez terminado de totalizar los 4 dígitos de la primera muestra se continúa con el primer dígito de la segunda muestra, y así sucesivamente hasta completar el total de muestras:

```
 //Lectura de tensiones
 for (int i=0;i<numeroMuestras;i++){
   int total=0;
   for (int j=0;j<4;j++){
    while (serialDataAvail(fd)<0){};
     char c=serialGetchar(fd);
```

```
    if (c==32) datoLeido=0; //Un caracter en blanco lo consideramos como un 0
    else datoLeido=c-48; //Convertimos el caracter a un digito decimal
    if (datoLeido>=0 && datoLeido<=9){
      total=total+datoLeido*pow(10,3-j);
    }
   }
   muestraV[i]=total;
   //printf("%4i %i\n",i,muestraV[i]);
 }
```

El mismo proceso anterior se sigue para las muestras de corriente:

```
//Lectura de corrientes
  for (int i=0;i<numeroMuestras;i++){
    int total=0;
    for (int j=0;j<4;j++){
     while (serialDataAvail(fd)<0){};
      char c=serialGetchar(fd);
      if (c==32) datoLeido=0; //Un caracter en blanco lo consideramos como un 0
      else datoLeido=c-48; //Convertimos el caracter a un digito decimal
      if (datoLeido>=0 && datoLeido<=9){
        total=total+datoLeido*pow(10,3-j);
      }
    }
    muestraI[i]=total;
    //printf("%4i %i\n",i,muestra [i]);
  }
```

Cerramos el puerto serie:

```
serialClose(fd);
```

Una vez que tenemos todas las muestras de tensiones y corrientes (como números enteros que van del 0 al 2^Nbits-1) hallamos los valores máximos y mínimos de la tensión (muestraVmax, muestraVmin), valor medio de la corriente (muestraImedia), así como los instantes (número de muestra) en que se producen los valores máximos (nVmax, nImax):

```
  //Determinación de los valores máximos y mínimos de las lecturas
  muestraVmax=muestraV[0];
  muestraVmin=muestraV[0];
  muestraImedia=0;
  nImax=0;
  for (int i=0;i<numeroMuestras;i++){
    if (muestraV[i]>muestraVmax){
      muestraVmax=muestraV[i];
      nVmax=i;
    }
    if (muestraV[i]<muestraVmin) muestraVmin=muestraV[i];
    muestraImedia=muestraImedia+muestraI[i];
    if (muestraI[i]>muestraI[nImax]){
      nImax=i;
    }
  }
  muestraImedia=muestraImedia/numeroMuestras;
}
```

La función "obtenerTensionesCorrientes" convierte las muestras de tensión y de corriente que ha enviado el Arduino (valores enteros comprendidos entre 0 y 2^Nbits-1) en valores de voltios y amperios:

```
//Obtenemos los valores de las tensiones y corrientes a partir de las muestras  tomadas
void obtenerTensionesCorrientes(){
```

Abrimos el fichero "datos.txt" en modo lectura:

```
  //leemos del fichero "datos.txt" la tensión eficaz de la red
  FILE *fp=fopen("../../datos.txt","r"); //abrimos /TFG/datos.txt como lectura
  if (fp==NULL) {
```

```
  printf("No se puede abrir el fichero datos.txt desde tfg\n");
}
```

Leemos el primer dato y lo guardamos en la variable "datoVmax":

```
char cadena[5];
fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de tensión máxima
int datoVmax=atoi(cadena);
```

Leemos el segundo dato y lo guardamos en la variable "datoVmin":

```
fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de tensión mínima
int datoVmin=atof(cadena);
```

Leemos el tercer dato y lo guardamos en la variable "datoImedia":

```
fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de corriente media
float datoImedia=atof(cadena);
```

Leemos el cuarto dato y lo guardamos en la variable "datoVef":

```
fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la tensión eficaz
float datoVef=atof(cadena);
```

Cerramos el fichero "datos.txt":

```
fclose(fp);
```

Traducimos cada lectura de tensión y corriente del Arduino (0 a 2^Nbits-1) a voltios y amperios, respectivamente, según las fórmulas descritas en el capítulo de "calibración":

$$V = \sqrt{2}V_{ef}\left(1 - 2\frac{muestraV\max - muestra}{muestraV\max - muestraV\min}\right)$$

$$I = \frac{3.3}{2^{10}sensibilidad}\left(\text{muestra - muestraImedia}\right)$$

$$P = VI$$

```
Pmax=tension[0]*corriente[0];
/*
La tensión en cada instante se calcula en función de:
    muestraV[i]: muestra de tensión tomada por el Arduino en ese instante
          datoVef: valor eficaz de la tensión en la fase de calibración
    datoVmax: muestra máxima de la tensión tomada por el Arduino en la fase de calibración
    datoVmin: muestra mínima de la tensión tomada por el Arduino en la fase de calibración

La corriente en cada instante se calcula en función de:
    muestraI[i]: muestra de corriente tomada por el Arduino en ese instante
    senSensorCorr: sensibilidad del sensor de corriente (en V/A)
    NBits: número de bits del ADC del Arduino
    datoImedia: muestra media de la corriente tomada por el Arduino en la fase de calibración
*/
for (int i=0;i<numeroMuestras;i++){
  tension[i]=sqrt(2)*datoVef*(1.0-2.0*(datoVmax-muestraV[i])/(datoVmax-datoVmin));
  corriente[i]=3.3/(senSensorCorr*pow(2,NBits))*(muestraI[i]-datoImedia);
  potencia[i]=tension[i]*corriente[i];
```

Determinamos la potencia máxima y el número de muestra en la que se produce:

```
  if (potencia[i]>Pmax){
    Pmax=potencia[i];
    nPmax=i;
  }
  //printf("tension(%3i)=%.1f;\n",i+1,tension[i]); //para comparar con Matlab
}
```

Hallamos los valores máximos de las tensiones y corrientes:

```
 Vmax=tension[nVmax];
 Imax=corriente[nImax];
```

Determinamos los valores máximos, de tensión, corriente y potencia, y que usaremos como valores máximos a escalar en la gráfica:

```
 VmaxGrafica=Vmax;
 if (Imax<1) ImaxGrafica=2.0; else ImaxGrafica=Imax;
 if (Pmax<500.0) PmaxGrafica=500.0; else PmaxGrafica=Vmax*Imax;
}
```

La función "ordenarMuestras()" tiene como función ordenar las muestras de forma que la primera muestra tenga fase 0 para la tensión:

```
//Ordena las muestras de forma que la fase de la onda de tensión a dibujar sea 0
void ordenarMuestras(){
```

Buscamos el número "j" de la muestra cuya tensión más se acerque a 0 y esté en fase creciente:

```
//Ordena las muestras de forma que la fase de la onda de tensión a dibujar sea 0
void ordenarMuestras(){
 int j=0;
 int minimo=abs(tension[0]);
 //buscanmos la tensión mínima cuando la función crece
 for (int i=1;i<numeroMuestras;i++){
   if (tension[i-1]<tension[i] && abs(tension[i])<minimo){
     j=i;
     minimo=abs(tension[i]);
   }
 }
 j++;
 if (j==numeroMuestras) j=0;
 //if (j>=numeroMuestras) j=j-numeroMuestras;
 //El valor de j es la muestra con tensión 0 en fase creciente
 float mv[numeroMuestras];
 float mi[numeroMuestras];
```

Copiamos las muestras de tensión y corriente en nuevos vectores:

```
 //Copiamos las muestras de tensiones y corrientes
 for (int i=0;i<numeroMuestras;i++){
   mv[i]=tension[i];
   mi[i]=corriente[i];
 }
```

Las muestras de la "j" al final las ponemos las primeras:

```
 //Las muestras desde la j hasta el final las ponemos las primeras
 for (int i=j;i<numeroMuestras;i++){
    tension[i-j]=mv[i];
    corriente[i-j]=mi[i];
    potencia[i-j]=mv[i]*mi[i];
 }
```

Las muestras de la "0" a la "j-1"las ponemos las últimas:

```
 //Las muestras desde la 0 hasta la j-1 la ponemos las últimas
 for (int i=0;i<j;i++){
   tension[numeroMuestras-j+i]=mv[i];
   corriente[numeroMuestras-j+i]=mi[i];
   potencia[numeroMuestras-j+i]=mv[i]*mi[i];
 }
```

Al haber cambiado de orden las muestras, tenemos que recalcular los instantes en que se dan los máximos y mínimos de las tensiones, corrientes y potencias:

```
 //Actualizamos los instantes en que se producen los máximos
 if (nVmax>=j) nVmax=nVmax-j;
 else nVmax=numeroMuestras-j+nVmax;
 if (nImax>=j) nImax=nImax-j;
 else nImax=numeroMuestras-j+nImax;
 if (nPmax>=j) nPmax=nPmax-j;
 else nPmax=numeroMuestras-j+nPmax;
}
```

La función "obtenerArmonicosTensión" tiene como objetivo hallar el valor eficaz de la tensión (Vef), el valor medio de la tensión (Vm), el valor eficaz de la componente fundamental de la tensión (arm1), los valores eficaces de los armónicos 2 al 8 (arm2, arm3,…, arm8), la distorsión armónica total (thd), y las distorsiones de correspondientes a los armónicos 2 al 8 (th2, th3,…,th8)

La variable a_1 *va a ir almacenando el valor de* $\frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})sen\left(n\omega\frac{nT}{m}\right)$

La variable b_1 *va a ir almacenando el valor* $\frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})\cos\left(n\omega\frac{nT}{m}\right)$

La variable a_2 *va a ir almacenando el valor* $\frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})sen\left(2n\omega\frac{nT}{m}\right)$

La variable b_2 *va a ir almacenando el valor* $\frac{\sqrt{2}}{m}\sum_{n=0}^{m-1} v(\frac{nT}{m})\cos\left(2n\omega\frac{nT}{m}\right)$

, y así sucesivamente para a_3 , b_3 … a_8 , b_8

La variable V_{ef} *va a ir almacenando el valor* $V_{ef}=\sum_{n=0}^{m-1} v^2\left(\frac{nT}{m}\right)$ *a la que posteriormente dividiremos por* m *y le realizaremos la raíz cuadrada.*

La variable V_m *va a ir almacenando el valor* $V_m=\frac{1}{m}\sum_{n=0}^{m-1} v\left(\frac{nT}{m}\right)$*, es decir, el valor medio de la tensión.*

```
//Obtenemos información de los armónicos
void obtenerArmonicosTension(){
 float a1=0;
 float b1=0;
 float a2=0;
 float b2=0;
 float a3=0;
 float b3=0;
 float a4=0;
 float b4=0;
 float a5=0;
 float b5=0;
 float a6=0;
 float b6=0;
 float a7=0;
 float b7=0;
 float a8=0;
 float b8=0;
 Vef=0;
 Vm=0;
```

El sumatorio de cada una de las variables anteriores se implementa en el bucle "for" siguiente:

```
 for (int i=0;i<numeroMuestras;i++){
```

```
Vef=Vef+tension[i]*tension[i];
Vm=Vm+tension[i]/numeroMuestras;
a1=a1+sqrt(2)/numeroMuestras*tension[i]*sin(2*pi*50*i*0.02/numeroMuestras);
b1=b1+sqrt(2)/numeroMuestras*tension[i]*cos(2*pi*50*i*0.02/numeroMuestras);
a2=a2+sqrt(2)/numeroMuestras*tension[i]*sin(2*2*pi*50*i*0.02/numeroMuestras);
b2=b2+sqrt(2)/numeroMuestras*tension[i]*cos(2*2*pi*50*i*0.02/numeroMuestras);
a3=a3+sqrt(2)/numeroMuestras*tension[i]*sin(3*2*pi*50*i*0.02/numeroMuestras);
b3=b3+sqrt(2)/numeroMuestras*tension[i]*cos(3*2*pi*50*i*0.02/numeroMuestras);
a4=a4+sqrt(2)/numeroMuestras*tension[i]*sin(4*2*pi*50*i*0.02/numeroMuestras);
b4=b4+sqrt(2)/numeroMuestras*tension[i]*cos(4*2*pi*50*i*0.02/numeroMuestras);
a5=a5+sqrt(2)/numeroMuestras*tension[i]*sin(5*2*pi*50*i*0.02/numeroMuestras);
b5=b5+sqrt(2)/numeroMuestras*tension[i]*cos(5*2*pi*50*i*0.02/numeroMuestras);
a6=a6+sqrt(2)/numeroMuestras*tension[i]*sin(6*2*pi*50*i*0.02/numeroMuestras);
b6=b6+sqrt(2)/numeroMuestras*tension[i]*cos(6*2*pi*50*i*0.02/numeroMuestras);
a7=a7+sqrt(2)/numeroMuestras*tension[i]*sin(7*2*pi*50*i*0.02/numeroMuestras);
b7=b7+sqrt(2)/numeroMuestras*tension[i]*cos(7*2*pi*50*i*0.02/numeroMuestras);
a8=a8+sqrt(2)/numeroMuestras*tension[i]*sin(8*2*pi*50*i*0.02/numeroMuestras);
b8=b8+sqrt(2)/numeroMuestras*tension[i]*cos(8*2*pi*50*i*0.02/numeroMuestras);
}
```

Como indicamos anteriormente, al valor de V_{ef} hallado anteriormente tenemos que dividirlo por el número de muestras y aplicarle la raíz cuadrada para que represente realmente el valor eficaz:

```
Vef=sqrt(Vef/numeroMuestras);
```

Hallamos el valor eficaz de cada armónico:

$$arm_i = \sqrt{a_i^2 + b_i^2}$$

```
arm1t=sqrt(a1*a1+b1*b1);
arm2t=sqrt(a2*a2+b2*b2);
arm3t=sqrt(a3*a3+b3*b3);
arm4t=sqrt(a4*a4+b4*b4);
arm5t=sqrt(a5*a5+b5*b5);
arm6t=sqrt(a6*a6+b6*b6);
arm7t=sqrt(a7*a7+b7*b7);
arm8t=sqrt(a8*a8+b8*b8);
```

Hallamos los coeficientes armónicos:

$$TH_i = \left| \frac{arm_i}{arm_1} \right|$$

```
th2t=abs(arm2t/arm1t);
th3t=abs(arm3t/arm1t);
th4t=abs(arm4t/arm1t);
th5t=abs(arm5t/arm1t);
th6t=abs(arm6t/arm1t);
th7t=abs(arm7t/arm1t);
th8t=abs(arm8t/arm1t);
```

Determinamos la distorsión armónica total:

$$THD \approx \sqrt{HD_2^2 + HD_3^2 + HD_4^2 + ... + HD_8^2}$$

```
thdt=sqrt(th2t*th2t+th3t*th3t+th4t*th4t+th5t*th5t+th6t*th6t+th7t*th7t+th8t*th8t);
```

Hallamos el ángulo φ_{v1}, es decir, el desfase de la componente fundamentar de la onda de tensión:

```
if (a1>0) fi_v1=atan(b1/a1);
else fi_v1=atan(b1/a1)+pi;
```

Mostramos los resultados si descomentamos el código. Estos resultados únicamente se visualizarán si ejecutamos el programa C++ en local. En remoto (mediante navegador web) se verá en pantalla con el código HTML que se describe un poco más abajo:

```
 /*
 printf("Valor eficaz de la tensión: %f\n",Vef);
 printf("Valor eficaz de la componente fundamental de la tensión: %f\n",arm1t);
 printf("Valor eficaz del armónico 2 de la tensión: %f\n",arm2t);
 printf("Valor eficaz del armónico 3 de la tensión: %f\n",arm3t);
 printf("Valor eficaz del armónico 4 de la tensión: %f\n",arm4t);
 printf("Valor eficaz del armónico 5 de la tensión: %f\n",arm5t);
 printf("Valor eficaz del armónico 6 de la tensión: %f\n",arm6t);
 printf("Valor eficaz del armónico 7 de la tensión: %f\n",arm7t);
 printf("Valor eficaz del armónico 8 de la tensión: %f\n",arm8t);
 printf("Distorsión armónica de la onda de tensión: %f\n",thdt);
 */
 /*
 printf("Valor medio: %7.2f\n",Vm);
 printf("Valor eficaz: %7.2f\n",Vef);
 printf("Componente fundamental: %7.2f\n",arm1t);
 printf("Distorsión armónica total, THD(%): %7.2f\n",100*thdt);
 printf("Distorsión armónica de orden 2 (%): %7.2f\n",100*th2t);
 printf("Distorsión armónica de orden 3 (%): %7.2f\n",100*th3t);
 printf("Distorsión armónica de orden 4 (%): %7.2f\n",100*th4t);
 printf("Distorsión armónica de orden 5 (%): %7.2f\n",100*th5t);
 printf("Distorsión armónica de orden 6 (%): %7.2f\n",100*th6t);
 printf("Distorsión armónica de orden 7 (%): %7.2f\n",100*th7t);
 printf("Distorsión armónica de orden 8 (%): %7.2f\n",100*th8t);
 */
}
```

La función "obtenerArmonicosCorriente" tiene la función que "obtenerArmonicosTensión" pero ahora para la corriente:

```
void obtenerArmonicosCorriente(){
 float a1=0;
 float b1=0;
 float a2=0;
 float b2=0;
 float a3=0;
 float b3=0;
 float a4=0;
 float b4=0;
 float a5=0;
 float b5=0;
 float a6=0;
 float b6=0;
 float a7=0;
 float b7=0;
 float a8=0;
 float b8=0;
 for (int i=0;i<numeroMuestras;i++){
  Ief=Ief+corriente[i]*corriente[i];
  Im=Im+1/0.02*corriente[i]*(0.02/numeroMuestras);
  a1=a1+sqrt(2)/numeroMuestras*corriente[i]*sin(2*pi*50*i*0.02/numeroMuestras);
  b1=b1+sqrt(2)/numeroMuestras*corriente[i]*cos(2*pi*50*i*0.02/numeroMuestras);
  a2=a2+sqrt(2)/numeroMuestras*corriente[i]*sin(2*2*pi*50*i*0.02/numeroMuestras);
  b2=b2+sqrt(2)/numeroMuestras*corriente[i]*cos(2*2*pi*50*i*0.02/numeroMuestras);
  a3=a3+sqrt(2)/numeroMuestras*corriente[i]*sin(3*2*pi*50*i*0.02/numeroMuestras);
  b3=b3+sqrt(2)/numeroMuestras*corriente[i]*cos(3*2*pi*50*i*0.02/numeroMuestras);
  a4=a4+sqrt(2)/numeroMuestras*corriente[i]*sin(4*2*pi*50*i*0.02/numeroMuestras);
  b4=b4+sqrt(2)/numeroMuestras*corriente[i]*cos(4*2*pi*50*i*0.02/numeroMuestras);
  a5=a5+sqrt(2)/numeroMuestras*corriente[i]*sin(5*2*pi*50*i*0.02/numeroMuestras);
  b5=b5+sqrt(2)/numeroMuestras*corriente[i]*cos(5*2*pi*50*i*0.02/numeroMuestras);
  a6=a6+sqrt(2)/numeroMuestras*corriente[i]*sin(6*2*pi*50*i*0.02/numeroMuestras);
  b6=b6+sqrt(2)/numeroMuestras*corriente[i]*cos(6*2*pi*50*i*0.02/numeroMuestras);
  a7=a7+sqrt(2)/numeroMuestras*corriente[i]*sin(7*2*pi*50*i*0.02/numeroMuestras);
  b7=b7+sqrt(2)/numeroMuestras*corriente[i]*cos(7*2*pi*50*i*0.02/numeroMuestras);
  a8=a8+sqrt(2)/numeroMuestras*corriente[i]*sin(8*2*pi*50*i*0.02/numeroMuestras);
```

```
  b8=b8+sqrt(2)/numeroMuestras*corriente[i]*cos(8*2*pi*50*i*0.02/numeroMuestras);
 }
 Ief=sqrt(Ief/numeroMuestras);
 arm1c=sqrt(a1*a1+b1*b1);
 arm2c=sqrt(a2*a2+b2*b2);
 arm3c=sqrt(a3*a3+b3*b3);
 arm4c=sqrt(a4*a4+b4*b4);
 arm5c=sqrt(a5*a5+b5*b5);
 arm6c=sqrt(a6*a6+b6*b6);
 arm7c=sqrt(a7*a7+b7*b7);
 arm8c=sqrt(a8*a8+b8*b8);
 th2c=abs(arm2c/arm1c);
 th3c=abs(arm3c/arm1c);
 th4c=abs(arm4c/arm1c);
 th5c=abs(arm5c/arm1c);
 th6c=abs(arm6c/arm1c);
 th7c=abs(arm7c/arm1c);
 th8c=abs(arm8c/arm1c);
 thdc=sqrt(th2c*th2c+th3c*th3c+th4c*th4c+th5c*th5c+th6c*th6c+th7c*th7c+th8c*th8c);
 if (a1>0) fi_c1=atan(b1/a1);
 else fi_c1=atan(b1/a1)+pi;
 /*
 printf("Valor medio: %7.2f\n",Im);
 printf("Valor eficaz: %7.2f\n",Ief);
 printf("Componente fundamental: %7.2f\n",arm1c);
 printf("Distorsión armonica total, THD(%): %7.2f\n",100*thdc);
 printf("Distorsión armónica de orden 2 (%): %7.2f\n",100*th2c);
 printf("Distorsión armónica de orden 3 (%): %7.2f\n",100*th3c);
 printf("Distorsión armónica de orden 4 (%): %7.2f\n",100*th4c);
 printf("Distorsión armónica de orden 5 (%): %7.2f\n",100*th5c);
 printf("Distorsión armónica de orden 6 (%): %7.2f\n",100*th6c);
 printf("Distorsión armónica de orden 7 (%): %7.2f\n",100*th7c);
 printf("Distorsión armónica de orden 8 (%): %7.2f\n",100*th8c);
 */
}
```

En la función "obtenerFactorPotencia" calculamos los siguientes parámetros:

$$\cos\varphi = \cos\left(\varphi_{v1} - \varphi_{i1}\right)$$

$$S = VI$$

$$S_1 = VI_1$$

$$P = S_1 \cos\varphi$$

$$Q = \sqrt{S_1^2 - P^2}$$

$$D = \sqrt{S^2 - P^2 - Q^2}$$

$$FP = \frac{P}{S}$$

```
void obtenerFactorPotencia(){
 cos_fi=cos(fi_v1-fi_c1); //desfase entre la onda de tensión y la fundamental de la corriente
 potenciaS=Vef*Ief;  //Potencia aparente
 float potenciaS1=Vef*arm1c;  //Potencia del primr armónico
 potenciaP=potenciaS1*cos_fi;  //Potencia activa
 potenciaQ=sqrt(potenciaS1*potenciaS1-potenciaP*potenciaP);  //Potencia reactiva
 potenciaD=sqrt(potenciaS*potenciaS-potenciaP*potenciaP-potenciaQ*potenciaQ);    //Potencia reactiva de distorsión
 fp=potenciaP/potenciaS;
}
```

La función "crearHTML" tiene como objetivo crear el código HTML para una página web en el que se muestren las gráficas y parámetros calculados:

```
//Crea la página HTML con los datos obtenidos
void crearHTML(){
```

Indicamos que el tipo de contenido que enviará la Raspberry al navegador es tipo HTML:

```
printf("Content-Type:text/html\n\n");
```

Declaramos el tipo de documento, HTML5:

```
printf("<!DOCTYPE html>\n");
printf("<html>\n");
printf("<head>\n");
printf("<meta charset='utf-8'>\n");
```

Le damos un título a la página:

```
printf("<title>Analizador de red</title>\n");
```

Declaramos que la hoja de estilos está escrita en CSS:

```
printf("<style type='text/css'>\n");
```

Definimos los estilos (tamaño de letra, color y tipo de letra) para la etiqueta "h2":

```
printf("h2{font-size:24px;color:brown;font-style:italic}\n");
```

Declaramos las características de un botón, "botonRele", con un borde marrón, sin padding, y en la posición 65 pixeles hacia abajo y 820 pixeles hacia la derecha, con una dimensión de 140x140 pixeles. Este botón nos servirá para conectar y desconectar la carga:

Figura 86: Botón conectar/desconectar de la aplicación web

```
printf("#botonRele{border:2px solid brown;padding:0;position:absolute;top:65px;left:820px;width:140px;height:140px}\n");
```

Declaramos las características de otro botón, "botonRecarga", con un borde marrón, sin padding, y en la posición 235 pixeles hacia abajo y 820 pixeles hacia la derecha, con una dimensión de 140x140 pixeles. Este botón nos servirá para recargar la página:

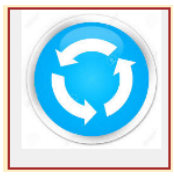


Figura 87: Botón actualizar de la aplicación web

```
printf("#botonRecarga{border:2px solid
brown;padding:0;position:absolute;top:235px;left:820px;width:140px;height:140px}\n");
```

Declaramos las características de otro botón, "botonCalibrar", con un borde marrón, sin padding, y en la posición 405 pixeles hacia abajo y 820 pixeles hacia la derecha, con una dimensión de 140x140 pixeles. Este botón nos servirá para calibrar el dispositivo:

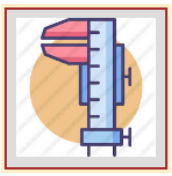


Figura 88: Botón calibrar de la aplicación web

```
printf("#botonCalibrar{border:2px solid
brown;padding:0;position:absolute;top:405px;left:820px;width:140px;height:140px}\n");
```

Definimos los estilos (color, tamaño de letra, tipo de letra y posición absoluta) para la etiqueta "p":

```
printf("p{color:brown;font-size:13px;font-style:italic;position:absolute;top:535px;left:820px;}\n");
```

Declaramos las características de una caja de texto, "cajaTexto", con texto en marrón, de tamaño 12 pixeles, letra en cursiva, y ubicada en la posición 550 px hacia abajo y 920 px hacia la derecha, con un ancho de 35 píxeles y un alto de 8 px:

```
printf("#cajaTexto{color:brown;font-size:12px;font-
style:italic;position:absolute;top:550px;left:920px;width:35px;height:8px}\n");
```

Tensión ef. (V): 223.5

Definimos los estilos para las etiquetas "pre" y "div":

```
printf("pre{color:brown;font-size:14px;font-style:italic}\n");
printf("div{color:brown;font-size:18px;font-style:italic;position:absolute;top:840px;left:0px}\n");
```

Declaramos las características de la imagen del escudo de la universidad de Sevilla, de dimensiones 120x120 pixeles:

Figura 89: Imagen icono de la Universidad de Sevilla

```
printf("img{width:120px;height:120px;absolute;position:absolute;top:0px;left:10px}\n");
```

Definimos los estilos para las etiquetas "pre" que se encuentra bajo la etiqueta "div":

```
printf("div > pre{position:absolute;top:10px;left:150px;}\n");
```

Terminamos la hoja de estilos y la cabecera HTML:

```
printf("</style>\n");
printf("</head>\n");
```

Damos un fondo marrón claro a la página:

```
printf("<body style='background-color:#FFF0C9;'>\n");
```

Escribimos el encabezado de la aplicación:

```
printf("<h2> Analizador de redes eléctricas</h2>\n");
```

Declaramos una etiqueta "canvas" con identificador "lienzo", donde dibujaremos las gráficas, de ancho 800 px y de alto 500 px:

```
//Aqui comienza el elemento "canvas" de 800x500
```

```
printf("<canvas id='lienzo' width='800' height='500'>\n");
```

Indicamos que el script que vamos a añadir está escrito en javascript:

```
printf("<script type='text/javascript'>\n");
```

Creamos una variable para guardar el "id" de la etiqueta canvas:

```
printf("var canvas=document.getElementById('lienzo');\n");
```

Creamos una variable para guardar el contexto de nuestro elemento "canvas":

```
printf("var ctx=canvas.getContext('2d');\n");
```

Seleccionamos como color de fondo del canvas el verde claro:

```
//Color de fondo del canvas
printf("ctx.fillStyle='#F8FBCD';\n");
```

Dibujamos un rectángulo con las dimensiones del canvas, con ello, aplicamos el color anterior al interior del canvas:

```
printf("ctx.fillRect(0, 0, canvas.width, canvas.height);\n");
```

Escribimos los textos "10ms" y "20ms":

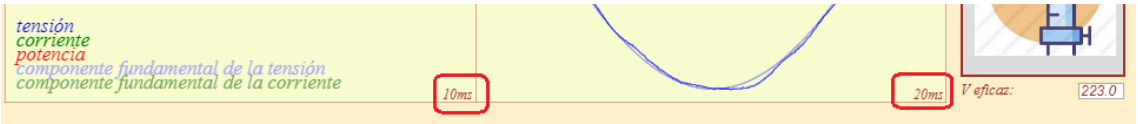


Figura 90: Marcas temporales de las gráficas de la aplicación web

```
//Marcas tenporales de 10ms y 20ms
printf("ctx.font = 'oblique 12px italic';\n");
printf("ctx.fillStyle = 'brown';\n");          //Color del texto
printf("ctx.fillText('10ms',372,497);\n");
printf("ctx.fillText('20ms',772,497);\n");
```

Dibujamos las indicaciones de las 5 gráficas (ver figura anterior):

```
//Indicación de las gráficas
printf("ctx.font = 'oblique 17px italic';\n");
printf("ctx.fillStyle = 'blue';\n");
printf("ctx.fillText('tensión',10,440);\n");
printf("ctx.fillStyle = 'green';\n");
printf("ctx.fillText('corriente',10,452);\n");
printf("ctx.fillStyle = 'red';\n");
printf("ctx.fillText('potencia',10,464);\n");
printf("ctx.fillStyle = '#A6A5F4';\n");
printf("ctx.fillText('componente fundamental de la tensión',10,476);\n");
printf("ctx.fillStyle = '#70A55A';\n");
printf("ctx.fillText('componente fundamental de la corriente',10,488);\n");
```

Comenzamos un camino:

```
//Líneas verticales y horizontales del gráfico (canvas)
printf("ctx.beginPath();\n"); //comienzo del camino
```

Definimos el trazado del camino de 0.4 pixeles::

```
printf("ctx.lineWidth = 0.4;\n"); //línea de espesor 0.4 pixeles
```

Definimos las siguientes líneas en el lienzo. Debemos tener en cuenta que para canvas las coordenadas (x,y) son de esta forma:

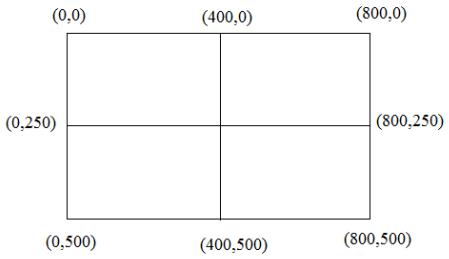


Figura 91: Coordenadas "canvas"

```
//Eje horizontal superior
printf("ctx.translate(0,0.5);\n"); //elimina el efecto difuminado de la línea
printf("ctx.moveTo(0,0);\n");
printf("ctx.lineTo(800,0);\n");
//Eje horizontal central
printf("ctx.moveTo(0,250);\n");
printf("ctx.lineTo(800,250);\n");
//Eje horizontal inferior
printf("ctx.moveTo(0,499);\n"); //Si lo ponemos en 500 no se ve
printf("ctx.lineTo(800,499);\n");
//Eje vertical izquierdo
printf("ctx.translate(0.5,0);\n"); //elimina el efecto difuminado de la línea
printf("ctx.moveTo(0,0);\n");
printf("ctx.lineTo(0,500);\n");
//Eje vertical central
printf("ctx.moveTo(400,0);\n");
printf("ctx.lineTo(400,500);\n");
//Eje vertical derecho
printf("ctx.moveTo(799,0);\n"); //Si lo ponemos en 800 no se ve
printf("ctx.lineTo(799,500);\n");
```

Seleccionamos el color marrón para las líneas definidas anteriormente:

```
printf("ctx.strokeStyle='brown';\n");
```

Dibujamos las líneas definidas anteriormente con lo que ya tenemos la cuadrícula

```
printf("ctx.stroke();\n");
```

Definimos la letra cursiva de 12 pixeles de tamaño y tipo de letra itálica:

```
//Letra cursiva de tamaño 12px y tipo itálica
printf("ctx.font = 'oblique 12px italic';\n");
```

Dibujamos la onda de tensión comenzando definiendo las coordenadas "x" e "y":

```
//Dibujamos la onda de tensión
float x;
float y;
```

Definimos el ancho del trazo en 0.8 pixeles:

```
printf("ctx.lineWidth = 0.8;\n"); //línea de espesor 0.9 pixeles
```

Comenzamos el camino:

```
printf("ctx.beginPath();\n");
```

Empezamos poniendo el cursor en la coordenada "x" igual a 0, y la coordenada "y" será tal que para una tensión de VmaxGrafica valga y=10, y para una tensión de – VmaxGrafica valga 490. Con esto nos aseguramos que cualquier tensión siempre se va a dibujar en el rango $-VmaxGrafica \leq y \leq VmaxGrafica$

```
x=0;
```

```
y=-(250-10)/VmaxGrafica*tension[0]+250; //10<=y<=490
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
```

Ahora entramos en un bucle para ir dibujando las rectas que unen los puntos correspondientes a cada muestra:

```
for (int i=1;i<numeroMuestras;i++){
```

Repartimos el ancho del dibujo entre el número de muestras que tenemos (menos 1) para que la coordenada "x" del canvas vaya recorriendo el dibujo:

```
  x=800.0/(numeroMuestras-1)*i;          //0<=x<=800
```

Para la coordenada "y" aplicamos la transformación descrita anteriormente:

```
  y=-(250-10)/VmaxGrafica*tension[i]+250; //10<=y<=490
```

Definimos el otro punto de la línea:

```
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
```

Aplicamos el color azul a todas las líneas definidas:

```
printf("ctx.strokeStyle='blue';\n");
```

Dibujamos las líneas (que corresponden a la tensión):

```
printf("ctx.stroke();\n");
```

Dibujamos en color azul el valor decimal correspondiente al valor máximo de la tensión, en la posición del gráfico donde se da este máximo:

```
printf("ctx.fillStyle = 'blue';\n");
printf("ctx.fillText('%.1fV',%.1f,%.1f);\n",Vmax,nVmax*800.0/numeroMuestras-10,-(250-10)/VmaxGrafica*Vmax+248);
```

Dibujamos ahora la gráfica de la corriente de la misma forma que antes dibujamos la de la tensión:

```
//Dibujamos la gráfica de la corriente:
printf("ctx.beginPath();\n");
x=0;
y=-(250-10)/ImaxGrafica*corriente[0]+250;
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=-(250-20)/ImaxGrafica*corriente[i]+250; //20<=y<=480
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
printf("ctx.strokeStyle='green';\n");
printf("ctx.stroke();\n");
printf("ctx.fillStyle = 'green';\n"); //El siguiente texto irá en verde
printf("ctx.fillText('%.1fA',%.1f,%.1f);\n",Imax,nImax*800.0/numeroMuestras-10,-(250-20)/ImaxGrafica*Imax+248);
```

Dibujamos ahora la gráfica de la potencia de la misma forma que antes dibujamos la de la tensión:

```
//Dibujamos la gráfica de la potencia:
printf("ctx.beginPath();\n");
x=0;
y=-(250-10)/PmaxGrafica*potencia[0]+250;
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=-(250-30)/PmaxGrafica*potencia[i]+250; //30<=y<=470
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
printf("ctx.strokeStyle='red';\n");
printf("ctx.stroke();\n");
```

```
printf("ctx.fillStyle = 'red';\n"); //El siguiente texto irá en rojo
printf("ctx.fillText('%.1fW',%.1f,%.1f);\n",Pmax,nPmax*800.0/numeroMuestras-10,-(250-30)/PmaxGrafica*Pmax+248);
```

Dibujamos la gráfica de la componente fundamental de la tensión:

```
//Dibujamos la gráfica de la componente fundamental de la tension:
printf("ctx.lineWidth = 0.4;\n"); //línea de espesor 0.4 pixeles
printf("ctx.beginPath();\n");
x=0;
y=sqrt(2)*arm1t*sin(fi_v1);
y=-(250-10)/VmaxGrafica*y+250; //10<=y<=490
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=sqrt(2)*arm1t*sin(100*pi*0.02*i/(numeroMuestras-1)+fi_v1);
  y=-(250-10)/VmaxGrafica*y+250; //ParaVPmaxGrafica-->10, para -VmaxGrafica-->490
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
printf("ctx.strokeStyle='blue';\n");
printf("ctx.stroke();\n");
```

Dibujamos la gráfica de la componente fundamental de la corriente:

```
//Dibujamos la gráfica de la componente fundamental de la corriente:
printf("ctx.beginPath();\n");
x=0;
y=sqrt(2)*arm1c*sin(fi_c1);
y=-(250-20)/ImaxGrafica*y+250;
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=sqrt(2)*arm1c*sin(100*pi*0.02*i/(numeroMuestras-1)+fi_c1);
  y=-(250-20)/ImaxGrafica*y+250; //Para ImaxGrafica-->20, para ImaxGrafica-->480
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
printf("ctx.strokeStyle='green';\n");
printf("ctx.stroke();\n");
```

Terminamos el "script" y el "canvas":

```
printf("</script>\n");
printf("</canvas>\n");
```

Mostramos los valores de potencias y armónicos de tensión y corriente:

```
//Mostramos los valores armónicos:
printf("<pre>\n");
printf("Potencia media,     P=%8.1fW                    Onda de tensión   Onda de corriente\n",potenciaP);
printf("Potencia reactiva,  Q=%6.1fVAr      Valor eficaz:          %10.1fV      %10.3fA\n",potenciaQ,Vef,Ief);
printf("Potencia de distorsión,D=%7.1fVA    Valor medio:           %10.1fV      %10.1fmA\n",potenciaD,Vm,1000*Im);
printf("Potencia aparente,  S=%7.1fVA       Distorsión arm. total, THD:  %10.1f&#37
%10.1f&#37\n",potenciaS,100*thdt,100*thdc);
printf("Factor de potencia, FP=%8.1f        Componente fundamental:      %10.1fV      %10.1fA\n",fp,arm1t,arm1c);
printf("                cos&#966=%6.1f      Componente arm. orden 2:     %10.1fV      %10.1fmA\n",cos_fi,arm2t,1000*arm2c);
printf("                &#966=%8.1f&deg     Componente arm. orden 3:     %10.1fV      %10.1fmA\n",180/pi*(fi_v1-
fi_c1),arm3t,1000*arm3c);
printf("                                    Componente arm. orden 4:     %10.1fV      %10.1fmA\n",arm4t,1000*arm4c);
printf("                                    Componente arm. orden 5:     %10.1fV      %10.1fmA\n",arm5t,1000*arm5c);
printf("                                    Componente arm. orden 6:     %10.1fV      %10.1fmA\n",arm6t,1000*arm6c);
printf("                                    Componente arm. orden 7:     %10.1fV      %10.1fmA\n",arm7t,1000*arm7c);
printf("                                    Componente arm. orden 8:     %10.1fV      %10.1fmA\n",arm8t,1000*arm8c);
```

Mostramos la dirección IP de la máquina remota (la que se conecta a la web):

```
printf("\nConexión realizada desde la máquina %s\n",strchr(maquina,'=')+1);
printf("</pre>\n");
```

Mostramos una etiqueta "p" que se situará bajo el botón "calibrar" y que nos permitirá introducir la tensión eficaz de la onda de tensión de referencia (proceso de calibración):

```
printf("<p>V eficaz:</p>\n");
```

Mostramos el icono de la universidad de Sevilla. Este fichero, "us.png" debe guardarse en la carpeta /TFG/imagenes de la aplicación.

```
printf("<div>\n");
printf("<img src='/TFG/imagenes/us.png'>\n");
printf("<pre>\n");
printf("Departamento de Electrónica\n");
printf("Ingeniería de las Tecnologías de Telecomunicación\n");
printf("Trabajo Fin de Grado\n");
printf("Araceli García Granados\n");
printf("</pre>\n");
printf("</div>\n");
```

Ahora vamos a definir una función Ajax que accederá al recurso /TFG/cgi-bin/rele (programa escrito en C++) para activar el relé y así conectar la carga a la red eléctrica. El tipo de petición utilizada es "Post" con un único parámetro de valor "data" de valor "uno". Esta función Ajax no va a recibir ningún dato de vuelta del recurso anterior. Al final de la función Ajax forzamos la recarga de la página para que se tomen nuevas lecturas y se actualice la imagen con la situación del relé:

```
printf("<script type='text/javascript'>\n");
printf("function encenderRele() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/rele';\n");
printf("var datosPost='data=uno';\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();"); //recargamos la página
printf("}\n");
```

Esta otra función Ajax accede al mismo recurso /TFG/cgi-bin/rele pero con "data" igual a "cero" para desconectar la carga de la red eléctrica:

```
printf("function apagarRele() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/rele';\n");
printf("var datosPost='data=cero';\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();");
printf("}\n");
```

Esta otra función Ajax accede al recurso /TFG/cgi-bin/calibrar enviándole por el método "Post" los valores "lecturaVmax", "lecturaVmin", "lecturaImedia" y el valor introducido en la caja de texto… Estos datos son los que se usan para traducir las lecturas del Arduino (valores de 0 a 2^(Nbits-1)) a voltios y amperios:

```
printf("function calibrar() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/calibrar';\n");
printf("var datosPost='%i&%i&%.2f&'+",muestraVmax,muestraVmin,muestraImedia);
printf("document.getElementById('cajaTexto').value;\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();");
printf("}\n");
```

Cerramos la etiqueta "script":

```
printf("</script>\n");
```

Iniciamos el protocolo serie y esperamos unos 150us para darle tiempo a la Raspberry:

```
int fd;
if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
  printf("No se puede abrir el dispositivo serie ttyACM0\n");
  return;
}
usleep(150);
```

Enviamos al Arduino el carácter "2" para que lea su pin 52 que nos indica la situación del relé (cerrado o no):

```
serialPutchar(fd,'2'); //Enviamos este caracter al arduino;
```

Esperamos a que el Arduino nos devuelva el estado del relé (un "1" si está conectado, o un "0" si está desconectado):

```
while (serialDataAvail(fd)<0){};
char c=serialGetchar(fd);
```

Si está conectado, mostramos un botón con la imagen de apagado por si posteriormente queremos apagarlo. El fichero "off.jpg" debemos ubicarlo en la carpeta /TFG/imagenes de nuestra aplicación.

```
if (c=='1') printf("<button id='botonRele' onclick='apagarRele();'><img src='/TFG/imagenes/off.jpg'></button>\n");
```

Si está desconectado, mostramos un botón con la imagen de encendido por si posteriormente queremos encenderlo. El fichero "on.jpg" debemos ubicarlo en la carpeta /TFG/imagenes de nuestra aplicación.

```
if (c=='0') printf("<button id='botonRele' onclick='encenderRele();'><img src='/TFG/imagenes/on.jpg'></button>\n");
```

Mostramos un botón de recarga con un evento "clic", de manera que cuando se pulse se recargue la página. La imagen de este botón, "recarga.jpg" debemos ubicarla en la carpeta /TFG/imagenes de nuestra aplicación Web:

```
printf("<button id='botonRecarga' onclick='location.reload();'><img src='/TFG/imagenes/recargar.jpg'></button>\n");
```

Mostramos un botón de calibrar con un evento "clic", de manera que cuando se pulse se ejecute la función "calibrar()" de javascript. La imagen de este botón, "calibrar.jpg" debemos ubicarla en la carpeta /TFG/imagenes de nuestra aplicación Web:

```
printf("<button id='botonCalibrar' onclick='calibrar();'><img src='/TFG/imagenes/calibrar.jpg'></button>\n");
```

Mostramos una caja de texto con identificador "cajaTexto" y valor predeterminado "223.5":

```
printf("<input id='cajaTexto' type='text' value='223.0'>\n");
+
```

Finalmente cerramos las etiquetas "body" y "html":

```
printf("</body>\n");
printf("</html>\n");
```

Código C++ TFG/cgi/rele.c de la Raspberry

Este es el programa que se llamará vía Ajax desde el código principal (/TFG/cgi-bin/tfg). Su función va a ser leer el dato que trae la llamada del tipo Post. Si es "data=uno" enviará al Arduino por el protocolo serie el carácter "1", y si es "data=cero" enviará al Arduino el carácter "0". El Arduino, en función del carácter recibido, activará o desactivará el relé a través de su pin 53, respectivamente.

Una vez escrito el programa, lo compilaremos con el comando:

```
g++ -o rele rele.c -lwiringPi
```

Explicamos el código.

Declaramos las librerías que vamos a necesitar:

```
//Librerías
#include <stdio.h>            //para poder usar la función printf()
#include <wiringSerial.h>      //para poder usar las funciones del puerto serie
#include <unistd.h>           //para poder usar la función usleep()
#include <stdlib.h>
#include <string.h>   //para poder usar la función strcmp()

#define bpsUART 115200

int main (void){
  char input[50];
  char *lenstr;
  long len;
```

Obtenemos la cadena que contiene longitud de la petición Post:

```
  lenstr = getenv("CONTENT_LENGTH");
```

Asignamos a la variable entera "len" la longitud de los datos enviados mediante Post:

```
  if(lenstr != NULL && sscanf(lenstr,"%ld",&len)==1 && len <= 50){
```

Inicializamos el protocolo serie a 115200bps:

```
    int fd;
    if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
      printf("No se puede abrir el dispositivo serie ttyACM0\n");
      return 0;
    }
    usleep(50);
```

Leemos los datos enviados mediante Post y los guardamos en la variable "input":

```
    fgets(input, len+1, stdin);
```

Si "input" es igual a "data=uno", enviamos por el puerto serie al Arduino el carácter "1" para que proceda a poner a "1" su pin 53 y así active el relé:

```
    if (strcmp(input,"data=uno")==0){
      serialPutchar(fd,'1');
    }
```

Si "input" es igual a "data=cero", enviamos por el puerto serie al Arduino el carácter "0" para que proceda a poner a "0" su pin 53 y así desactive el relé:

```
    else if (strcmp(input,"data=cero")==0){
      serialPutchar(fd,'0');
    }
```

```
  serialClose(fd);
 }
 return 0 ;
}
```

Código C++ TFG/cgi/calibrar.c de la Raspberry

Este es el programa que se llamará vía Ajax desde el código principal (/TFG/cgi-bin/tfg). Su función va a ser guardar en el fichero "datos.txt" estos cuatro números:

1. *muestra máxima de tensión (entero en el rango 0-2^Nbits-1).*
2. *muestra mínima de tensión (entero en el rango 0-2^Nbits-1).*
3. *muestra media de corriente (decimal en el rango 0-2^Nbits-1).*
4. *tensión eficaz (en voltios) introducida en la página web, apartado de calibración.*

Estos cuatro datos se los pasa el programa /TFG/cgi-bin/tfg a /TFG/cgi-bin/calibrar mediante el método "post" vía "Ajax".

Una vez escrito el programa, lo compilaremos con el comando:

```
g++ -o calibrar calibrar.c
```

Explicamos el código.

Incluimos las librerías necesarias:

```
//Librerías
#include <stdio.h>            //para poder usar la función printf
#include <string.h> //para poder usar strlen()
#include <math.h> //para poder usar la función pow()
```

Incluimos el prototipo de la función "leerNumero", que describiremos más abajo:

```
float leerNumero(char *cadena,int *n);

int main (void){
 char cadena[50];
 char *lenstr;
 long len;
```

Abrimos el fichero "datos.txt" en modo escritura. Debemos tener la precaución de cambiar los permisos de este fichero ya que se accederá a el vía web con el usuario "tomcat8" y si éste no tiene permiso de escritura sobre el fichero, no lo podrá modificar.

```
 FILE *fp=fopen("../../datos.txt","w"); //Ojo: chmod 777 datos.txt
 if (fp==NULL) {
   printf("No se puede abrir el fichero datos.txt desde calibrar\n");
 }
```

Obtenemos el valor de la variable de entorno "CONTENT_LENGTH" que nos dará la longitud de la cadena de texto recibida por el método "post" desde el recurso /TFG/cgi-bin/tfg:

```
 lenstr = getenv("CONTENT_LENGTH");
```

Comprobamos que dicha longitud sea válida:

```
 if(lenstr != NULL && sscanf(lenstr,"%ld",&len)==1 && len <= 50){
```

Obtenemos en la variable "cadena" el texto que nos envía el recurso /TFG/cgi-bin/tfg:

```
   fgets(cadena, len+1, stdin); //recogemos lo recibido por Post
```

Leemos el primero número (lecturaVmax) que contiene "cadena" y lo guardamos en el fichero "datos.txt:

```
  int i=0;
  float numero=leerNumero(cadena,&i);
  fprintf(fp,"%.0f\n",numero);
```

Leemos el segundo número (lecturaVmin) que contiene "cadena" y lo guardamos en el fichero "datos.txt:

```
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.0f\n",numero);
```

Leemos el tercer número (lecturaImedia) que contiene "cadena" y lo guardamos en el fichero "datos.txt:

```
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.1f\n",numero);
```

Leemos el cuarto número (Vef) que contiene "cadena" y lo guardamos en el fichero "datos.txt:

```
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.1f\n",numero);
 }
```

Cerramos el fichero "datos.txt" y terminamos:

```
 fclose(fp);
 return 0 ;
}
```

Rutina "leerNumero": devuelve el número que contiene la variable "cadena" desde la posición "n", y actualizando la variable "n" con la posición del siguiente número que aparece en "cadena". La variable cadena contiene 4 números separados por un "&":

```
//lee el número que hay en "cadena" desde la posición "n"
// Por ejemplo, cadena es "959&82&509.8&226.0"
float leerNumero(char *cadena,int *n){
 int i=*n;
 int j=0;
 float numero=0;
```

Mientras que el caracter leído sea un número…:

```
 while (cadena[i]>='0' && cadena[i]<='9'){
```

Acumulamos en la variable "número" el valor del carácter numérico con su peso correspondiente. Por ejemplo, si el número es 123.4 haremos el cálculo numero=1/10+2/100+3/1000+4/10000=0.1234

```
  numero=numero+(cadena[i]-'0')/pow(10,i-(*n)+1);
  i++;
  j++;
 }
```

Cuando el carácter sea un "." No lo tenemos en cuenta, incrementando el contador "i":

```
 if (cadena[i]=='.'){
  i++;
```

Seguimos incrementando el acumulador de la misma forma que antes de encontrar el ".":

```
  while (cadena[i]>='0' && cadena[i]<='9'){
   numero=numero+(cadena[i]-'0')/pow(10,i-(*n));
   i++;
  }
 }
```

Seguimos incrementando el acumulador de la misma forma que antes de encontrar el ".":

```
i++;
```

Actualizamos "n" para que apunte al siguiente número dado que ahora apunta a un "&":

```
*n=i;
```

La variable "j" contiene el número de cifras no decimales que tiene el número que hemos leído. En el caso del ejemplo, 123.4, son 3. Por tanto, el número leído es 0.1234x1000=123.4

```
numero=numero*pow(10,j);
```

Devolvemos el resultado:

```
return numero;
}
```

Código C++ TFG/cgi/registro.c de la Raspberry

Este programa permitirá que visualicemos vía web los accesos que se han realizado a la aplicación, en concreto se visualizarán, para cada acceso:

Dirección IP de la máquina que ha usado la aplicación
Fecha y hora en la que se ha accedido a la aplicación
Tensión eficaz medida
Corriente eficaz medida
Potencia media

Inicialmente declaramos las librerías que vamos a usar:

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

Iniciamos la página web con el título y declaración de estilos CSS:

```
int main(){
 printf("Content-Type:text/html\n\n");
 printf("<!DOCTYPE html>\n");
 printf("<html>\n");
 printf("<head>\n");
 printf("<meta charset='utf-8'>\n");
 printf("<title>Conexiones</title>\n");
 printf("<style type='text/css'>\n");
 printf("p{font-size:16px;color:brown;font-style:italic}\n");
 printf("</style>\n");
 printf("</head>\n");
 printf("<body style='background-color:#FFF0C9;'>\n");
```

Abrimos el fichero en modo lectura:

```
 FILE *fp=fopen("../../registro.txt","r"); //abrimos /TFG/datos.txt como lectura
 if (fp==NULL) {
   printf("No se puede abrir el fichero registro.txt desde tfg\n");
 }
```

Iniciamos un elemento "p":

```
 printf("<p>");
```

Escribimos en el elemento "p" cada carácter leído del fichero, hasta llegar al final del fichero:

```
 char caracter=getc(fp);
 while (feof(fp)==0) {
   if (caracter=='\n') printf("</br>");
   else printf("%c",caracter);
   caracter=getc(fp);
 }
```

Cerramos el elemento "p" y terminamos:

```
 printf("</p>");
 printf("</body>\n");
 printf("</html>\n");
 return 0;
```

```
}
```

El código se compilará con la instrucción:

```
g++ -o registro registro.c
```

Para ejecutar el programa vía web introduciremos en un navegador la dirección:

http://analizador.ddns.net:8080/TFG/cgi-bin/registro

Código C++ TFG/cgi/resetArduino.c de la Raspberry

Este programa nos permitirá resetear vía web la tarjeta Arduino en caso de que se quedara bloqueada. Hace uso del pin GPIO17 de la Raspberry, que lo tenemos conectado al pin Reset del Arduino DUE.

Veamos su código

Inicialmente declaramos las librerías que vamos a usar:

```
#include <stdio.h> //para poder usar la función printf()
#include <stdlib.h>  //para poder usar la función system()
#include <wiringPi.h> //para poder usar la función digitalWrite()
```

Iniciamos la página web con el título:

```
int main (void){
 printf("Content-Type:text/html\n\n");
 printf("<!DOCTYPE html>\n");
 printf("<html>\n");
 printf("<head>\n");
 printf("<title>Reset Arduino</title>\n");
 printf("</head>\n");
 printf("<body>\n");
```

Configuramos el puerto GPIO17 como de salida.

```
 system("echo out > /sys/class/gpio/gpio17/direction"); //configuramos el puerto GPI017 como de salida
```

Inicializamos el sistema wiringPi usando el sistema BCM para nombrar los puertos GPIO::

```
 wiringPiSetupGpio(); //este comando lo puede ejecutar un usuario normal
```

Ponemos a "0" el pin GPIO17 durante 1 segundo y luego lo volvemos a dejar a "1":

```
 digitalWrite(17, LOW); //pin GPIO BCM 17 ó pin GPIO wiringPi 0
 delay(1000);   //1s
 digitalWrite(17, HIGH); // pin GPIO BCM 17 ó pin GPIO wiringPi 0
```

Terminamos con:

```
 printf("Resetada la placa Arduino...\n");
 printf("</body>\n");
 printf("</html>\n");
 return 0 ;
}
```

El código se compilará con la instrucción:

```
g++ -o resetArduino resetArduino.c -lwiringPi
```

Para ejecutar el programa vía web introduciremos en un navegador la dirección:

http://analizador.ddns.net:8080/TFG/cgi-bin/resetArduino

Capítulo 3: Manual de uso y calibración


UNIVERSIDAD DE SEVILLA

Manual de uso

La presentación de resultados del dispositivo se realiza vía web desde cualquier terminal fijo o móvil con acceso a Internet. La dirección http de la página web del dispositivo es:

http://analizador.ddns.net:8080/TFG

Si el terminal desde el que accedemos está en la misma red (por Wifi o conectado a un puerto Ethernet) también podríamos acceder mediante la dirección local:

http://192.168.1.00:8080/TFG

La página web presenta los resultados de la siguiente forma:

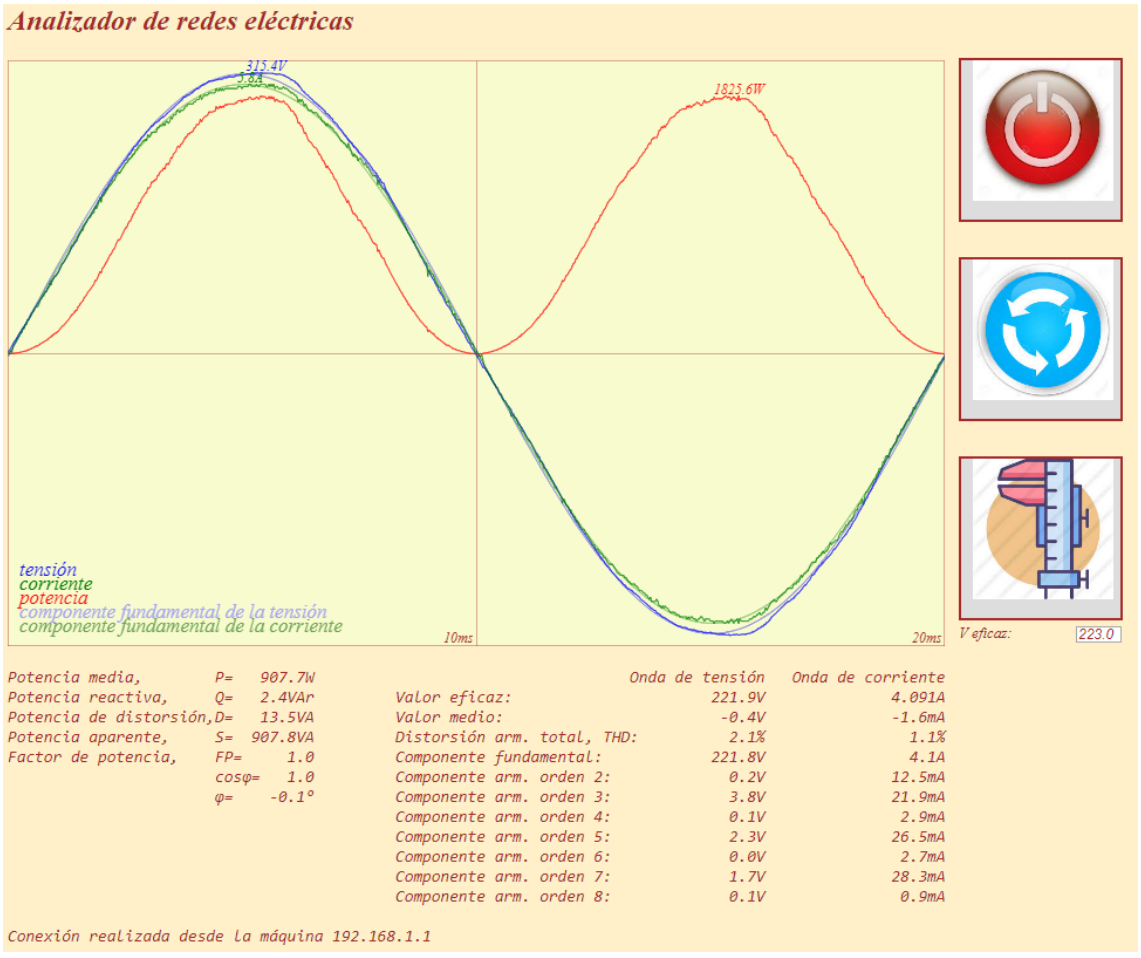


Figura 92: Presentación de resultados de la aplicación web

Se estructura en tres partes bien diferenciadas:

1. *Gráficas*
2. *Botonera*
3. *Parámetros eléctricos*

En la parte gráfica se presentan las ondas de tensión, corriente, potencia, y componentes fundamentales de la tensión y corriente, durante un periodo (20ms):

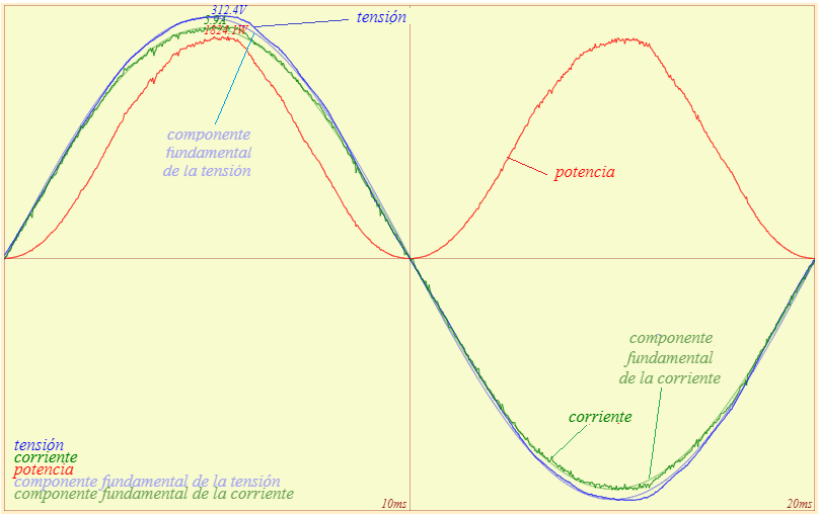


Figura 93: Gráficas de la aplicación web

En la parte inferior aparecen los parámetros eléctricos más relevantes de la carga, tal como potencias, factor de potencia, $\cos\varphi$, así como las componentes armónicas de las ondas de tensión y corriente, y las distorsiones de las mismas:

			Onda de tensión	Onda de corriente
Potencia media,	P= 907.7W	Valor eficaz:	221.9V	4.091A
Potencia reactiva,	Q= 2.4VAr	Valor medio:	-0.4V	-1.6mA
Potencia de distorsión,	D= 13.5VA	Distorsión arm. total, THD:	2.1%	1.1%
Potencia aparente,	S= 907.8VA	Componente fundamental:	221.8V	4.1A
Factor de potencia,	FP= 1.0	Componente arm. orden 2:	0.2V	12.5mA
	cosφ= 1.0	Componente arm. orden 3:	3.8V	21.9mA
	φ= -0.1°	Componente arm. orden 4:	0.1V	2.9mA
		Componente arm. orden 5:	2.3V	26.5mA
		Componente arm. orden 6:	0.0V	2.7mA
		Componente arm. orden 7:	1.7V	28.3mA
		Componente arm. orden 8:	0.1V	0.9mA

Figura 94: Parámetros eléctricos de la aplicación web

En la parte derecha aparecen tres botones:

Figura 95: Botones de la aplicación web

Con el botón superior podemos conectar y desconectar la carga de la red eléctrica.
El botón de en medio se utiliza para recarga de la página web.
El botón inferior se usa para calibrar el dispositivo (en el apartado de calibración se detalla su función).

La pulsación de cualquiera de estos botones provoca una nueva toma de datos.

La corriente máxima que podemos medir con este dispositivo depende únicamente de las capacidades del sensor de corriente y del relé que instalemos. El sensor LEM LTS 6-NP puede medir corrientes de hasta 19.2A, que corresponde a una potencia de 19.2x230=4416W. El relé de estado sólido que hemos instalado tiene una capacidad de 5.83A, que corresponde a una potencia de 5.83x230=1340W. De estas dos limitaciones en potencia, la más restrictiva es la del relé, concluyendo que el dispositivo construido permite medir cargas de hasta 1340W. En el apartado de "Conclusiones" de este trabajo se describe la manera de incrementar esta potencia.

La tensión máxima de red que soporta el dispositivo es de 281V eficaces.

El consumo del dispositivo de unos 10W, aunque depende de la carga que se conecte.

Es importante para que nuestra aplicación funcione correctamente no configurar en nuestro terminal móvil el "modo básico". Dicho modo hace que los navegadores no realicen una descarga completa de la información en el caso de que se prevea que ésta vaya a tardar más de 5 segundos.

Calibración y cálculo de tensiones y corrientes

La calibración es el proceso de comparación entre la medida que da el dispositivo y la medida correspondiente a un patrón de referencia, es decir, contrasta el valor de salida del instrumento frente a un estándar, y en caso de desviación se procede al ajuste del dispositivo.

Para proceder a la calibración de nuestro dispositivo seguiremos estos pasos:

1. *Conectamos el dispositivo a una onda de tensión sinusoidal de valor eficaz conocida, V_{ef} .*
2. *Desconectamos la carga.*
3. *Introducimos el valor eficaz V_{ef} en la caja situada justo debajo del botón calibrar.*
4. *Pulsamos en el botón de calibrar.*

Una vez calibrado podemos dar por válidas las medidas.

La calibración del dispositivo desencadena el siguiente proceso:

1. *La Raspberry solicita al Arduino la captura de las muestras de tensión y corriente, y con ellas obtiene el valor máximo y mínimo de la muestras de tensión ("muestraVmax", y "muestraVmin"), y el valor medio de las muestra de corriente ("muestraImedia").*

2. *Se obtienen los valores de las tensiones (voltios) de cada muestra (enteros de 0 a 1023) de la siguiente forma. Se construye la función de transferencia muestra-tensión como una recta que pasa por los puntos (muestraVmax, $\sqrt{2}V_{ef}$) y (muestraVmin, $-\sqrt{2}V_{ef}$) tal como se aprecia en la figura.*

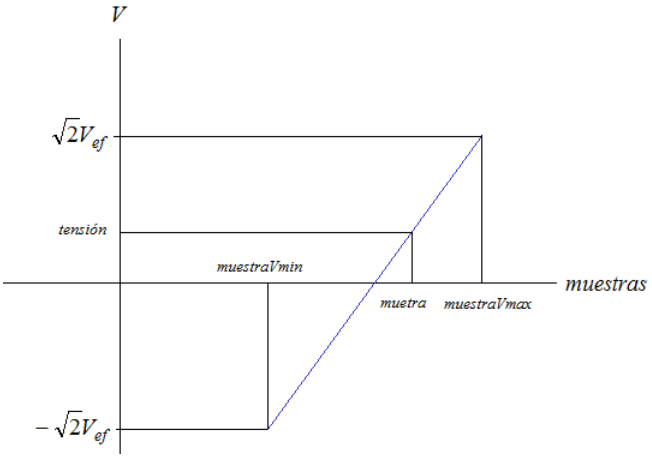


Figura 96: Función de transferencia tensión entrada sensor tensión –lectura arduino

Aplicando semejanza de triángulos:

$$\frac{\sqrt{2}V_{ef} - tensión}{muestraV\max - muestra} = \frac{2\sqrt{2}V_{ef}}{muestraV\max - muestraV\min}\text{, por tanto,}$$

$$\boxed{tensión = \sqrt{2}V_{ef}\left(1 - 2\frac{muestraV\max - muestra}{muestraV\max - muestraV\min}\right)}$$

Con esta ecuación podemos determinar el valor en voltios de cada muestra de tensión.

3. *Se obtienen las intensidades de corriente (amperios) de cada muestra (enteros de 0 a 1023) de la siguiente forma. El ADC del Arduino digitaliza la tensión de salida del sensor de corriente (variable continua, en voltios) obteniendo un valor entero (variable "muestra") que va de 0 a 1023 según esta función:*

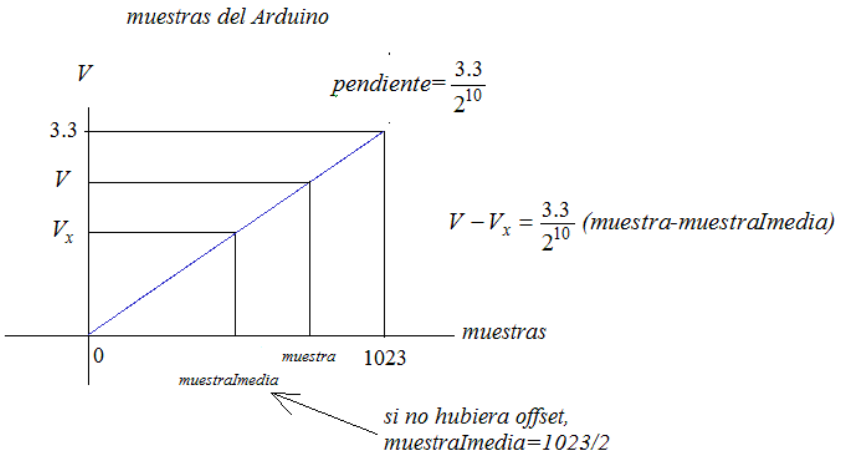


Figura 97: Función transferencia tensión salida sensor corriente – lectura del Arduino

La relación entre la tensión a la salida del sensor y la corriente que circula está determinada por la sensibilidad de dicho sensor según una recta, que debería pasar por el punto (I=0,Vx=1.835V) si el sensor no tuviera "offset" :

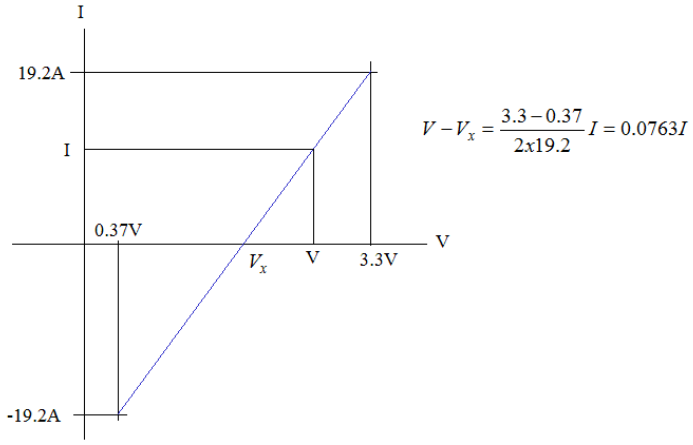


Figura 98: Función transferencia sensor corriente: I-V

Para eliminar el efecto del "offset" (desconocemos el valor real de Vx) igualamos dos ecuaciones de las rectas anteriores:

$$V - V_x = \frac{3.3 - 0.37}{2x19.2} I = 0.0763\, I$$

$$V - V_x = \frac{3.3}{2^{10}} \textit{(muestra-muestraImedia)}$$

, quedando:

$$0.0763\, I = \frac{3.3}{2^{10}} \textit{(muestra-muestraImedia)}$$

, y despejando "I" obtenemos finalmente:

$$\boxed{I = \frac{3.3}{0.0763\, x 2^{10}} (\text{muestra - muestraImedia})}$$

, que no depende del valor Vx, es decir, del "offset" del sensor.

4. *Por último, se guardan los valores "muestraVmax", "muestraVmin","muestraImedia", y "Vef" en el fichero "datos.txt" para asignar tensiones y corrientes a las muestras de las sucesivas mediciones tras la calibración.*

   El fichero "datos.txt" lo podemos visualizar introduciendo en la Raspberry el comando Linux:

   root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# cat datos.txt

   944
   86
   508.3
   223.5

   , siendo los valores respectivos, "muestraVmax","muestraVmin", "muestraImedia", y "Vef".

Capítulo 4: Medidas

En los siguientes apartados someteremos al dispositivo a distintos tipos de cargas, lineales y no lineales, y determinaremos tanto la exactitud como la precisión para los valores aportados de tensión, corriente y potencia. Los datos correspondientes a los armónicos los contrastaremos con la función FFT de Matlab para validarlos.

Medidas sin carga

Los resultados que hemos obtenido cuando la carga está desconectada son los siguientes:

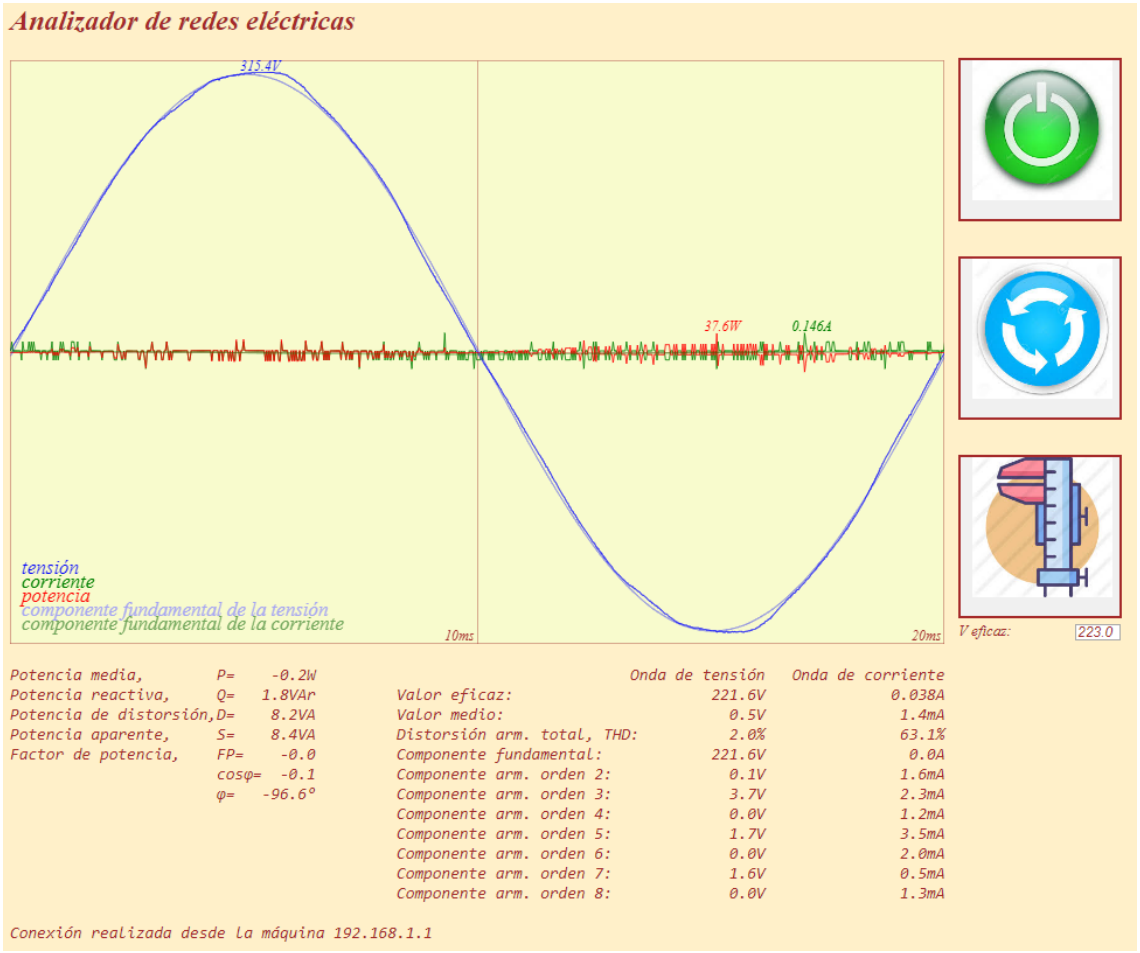


Figura 99: Medidas sin carga

Se aprecia el ruido que incorpora la señal de salida del sensor de corriente, con un valor de pico de 136mA y valor eficaz de 38mA, sin embargo la señal del sensor de tensión está limpia de ruidos.

También se aprecia que la onda de tensión es una función prácticamente impar lo que provoca que sus componentes armónicas pares sean casi nulas.

Podemos comparar la onda de tensión del dispositivo con la salida de un osciloscopio conectado a la red eléctrica para confirmar que estos defectos no son generados por nuestro prototipo. El resultado es el siguiente:

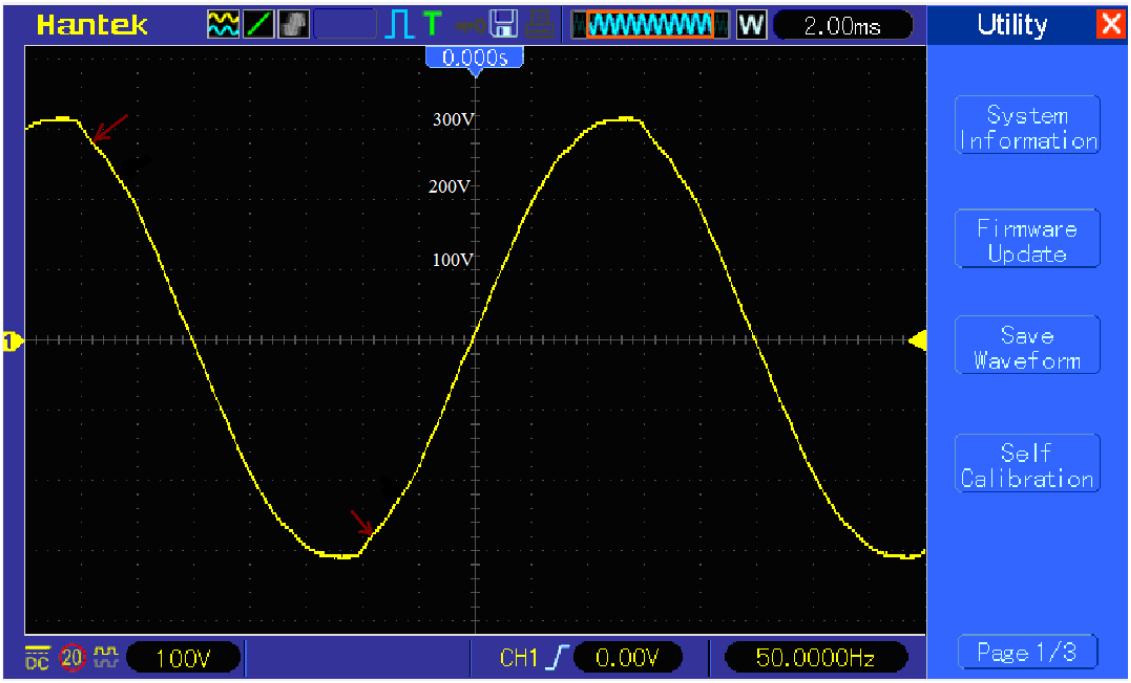


Figura 100: Onda de tensión medida con osciloscopio

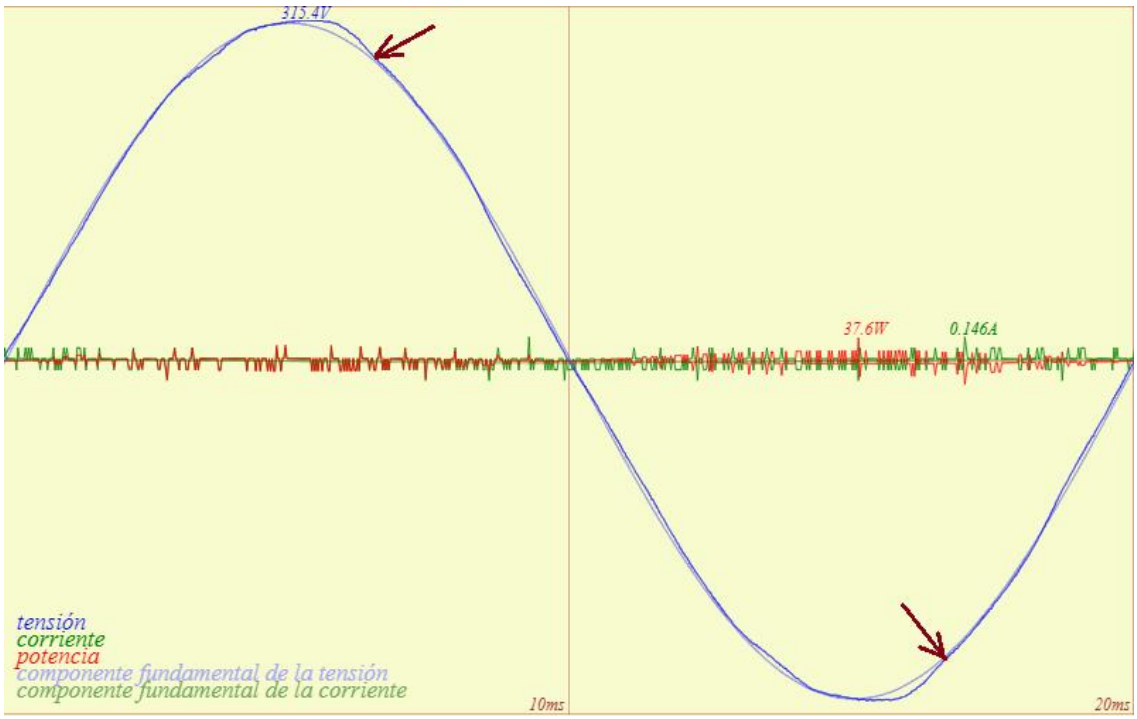


Figura 101: Onda de tensión medida con el prototipo

Las salidas son idénticas lo que valida la calidad gráfica que ofrece el dispositivo.

Medidas para una carga nominal de 60W

Vamos a someter al dispositivo a una carga nominal de 60W, concretamente una bombilla incandescente.

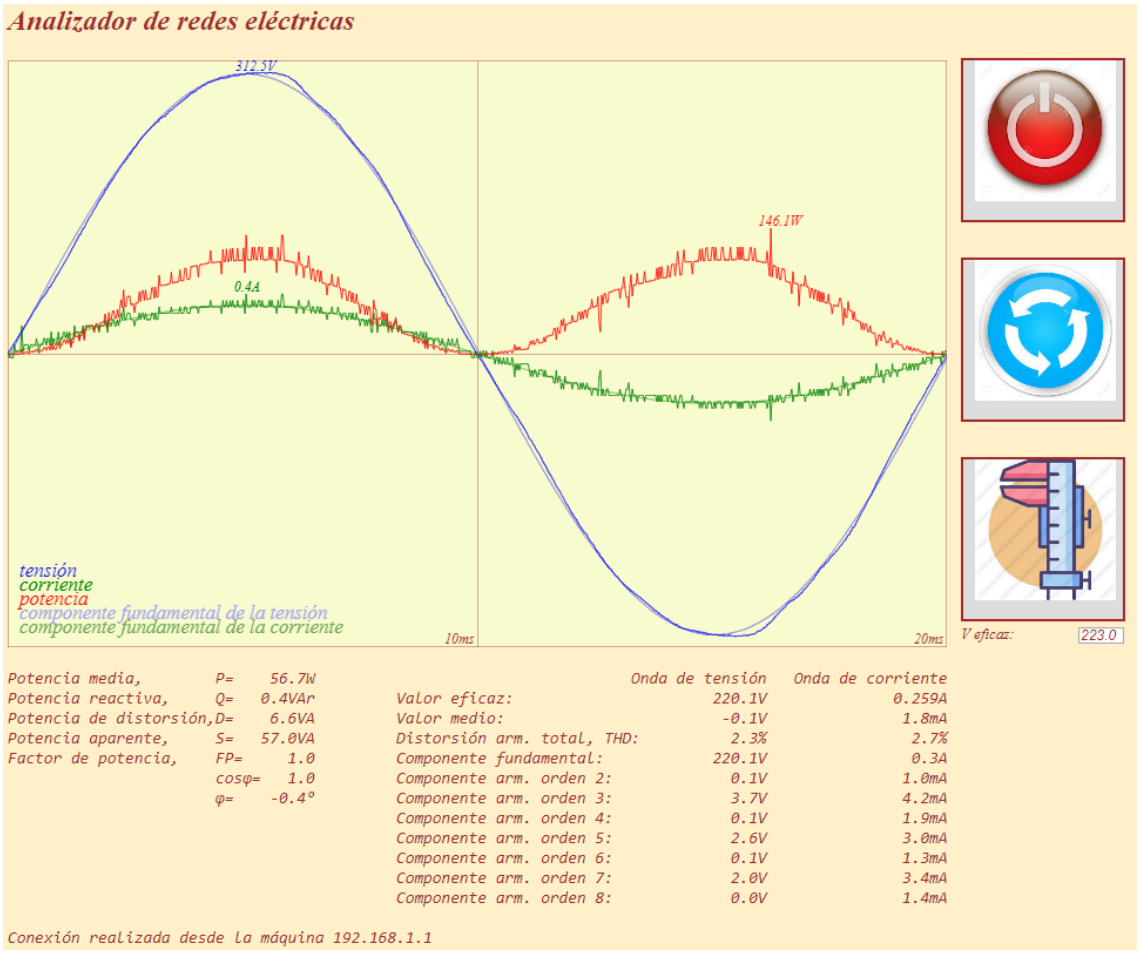


Figura 102: Medidas para una carga de 60W

Como referencia patrón para la comparación tomaremos los siguientes valores obtenidos con un multímetro Fluke 77:

Tensión eficaz: 221V
Corriente eficaz: 259.0mA
Potencia media: 221x0.2590=57.24W

Para poder determinar tanto la exactitud como la precisión de aparato tomaremos 50 medidas simultáneas de la misma carga. Las medias obtenidas nos permitirán conocer cuál es la exactitud del aparato, y las desviaciones típicas (o el coeficiente de variación) nos indicará la precisión del mismo.

Los 50 valores de las tensiones obtenidas han sido:

221.7,221.2,221.6,221.4,221.4,221.7,221.5,221.3,221.5,221.7,221.5,221.4,221.4,221.3,221.5,220.4,220.9, 221.2,221.5,221.5,221.3,221.4,221.6,221.6,221.9,221.7,221.8,221.5,220.7,221.2,221.4,221.4,221.1,221.7, 221.7,221.7,221.5,221.7,221.6,221.6,221.3,219.9,220.8,221.2,221.2,221.2,221.5,221.6,221.4,221.3

, que arrojan una media muestral de $\bar{v} = \frac{\sum_{i=1}^{n} v_i}{n} = 221.38V$ *y cuyo estimador de la desviación típica del aparato es* $s^2 = \frac{\sum_{i=1}^{n} \left(v_i - \bar{v} \right)^2}{n-1} = 0.39V$

Los 50 valores de las corrientes medidas han sido:

0.258,0.257,0.258,0.258,0.258,0.259,0.258,0.257,0.259,0.259,0.258,0.258,0.259,0.259,0.259,0.257,0.258, 0.256,0.258,0.257,0.257,0.259,0.259,0.259,0.258,0.259,0.258,0.258,0.256,0.257,0.257,0.258,0.258,0.259, 0.257,0.258,0.259,0.258,0.259,0.259,0.259,0.258,0.259,0.256,0.258,0.259,0.259,0.257,0.259,0.259

, que corresponden a una media muestra de 0.2581A y un estimador de la desviación típica de 0.00091A

Los 50 valores de las potencias medidas han sido:

56.8,56.6,56.7,56.7,56.7,57.0,56.8,56.6,57.1,57.0,56.8,56.7,56.9,56.9,57.0,56.2,56.5,56.3,56.7,56.4,56.5,5 7.0,57.1,57.0,56.8,56.9,56.8,56.7,56.2,56.5,56.6,56.6,56.7,57.0,56.5,56.8,56.9,56.9,56.9,56.9,56.8,56.4,56 .7,56.1,56.7,56.9,56.9,56.6,56.8,56.8

, que corresponden a una media muestral de 56.73W un estimador de la desviación típica de 0.23W

La precisión de un dispositivo es la aptitud para dar medidas muy próximas durante la aplicación repetida del mesurando. Nosotros la determinaremos a través del coeficiente de variación, definido como la relación entre la desviación típica y la media:

$$CV_V = 100 \frac{0.39}{221.38} = 0.16\%$$
$$CV_I = 100 \frac{0.00091}{0.2581} = 0.35\%$$
$$CV_P = 100 \frac{0.23}{56.73} = 0.41\%$$

La exactitud de un dispositivo es la aptitud para dar medidas próximas al valor verdadero. Nosotros lo determinamos mediante los errores medios relativos:

$$e_V = 100 \frac{|221.38 - 221|}{221} = 0.17\%$$
$$e_I = 100 \frac{|0.2581 - 0.2590|}{0.2590} = 0.35\%$$
$$e_P = 100 \frac{|56.73 - 57.24|}{57.24} = 0.89\%$$

Podemos concluir que nuestro dispositivo, a este nivel de carga, es muy preciso y exacto, cumpliendo con los objetivos que nos marcamos, inferiores al 1%.

Medidas para una carga nominal de 700W

La carga que vamos a utilizar es un radiador de aceite de unos 700W.

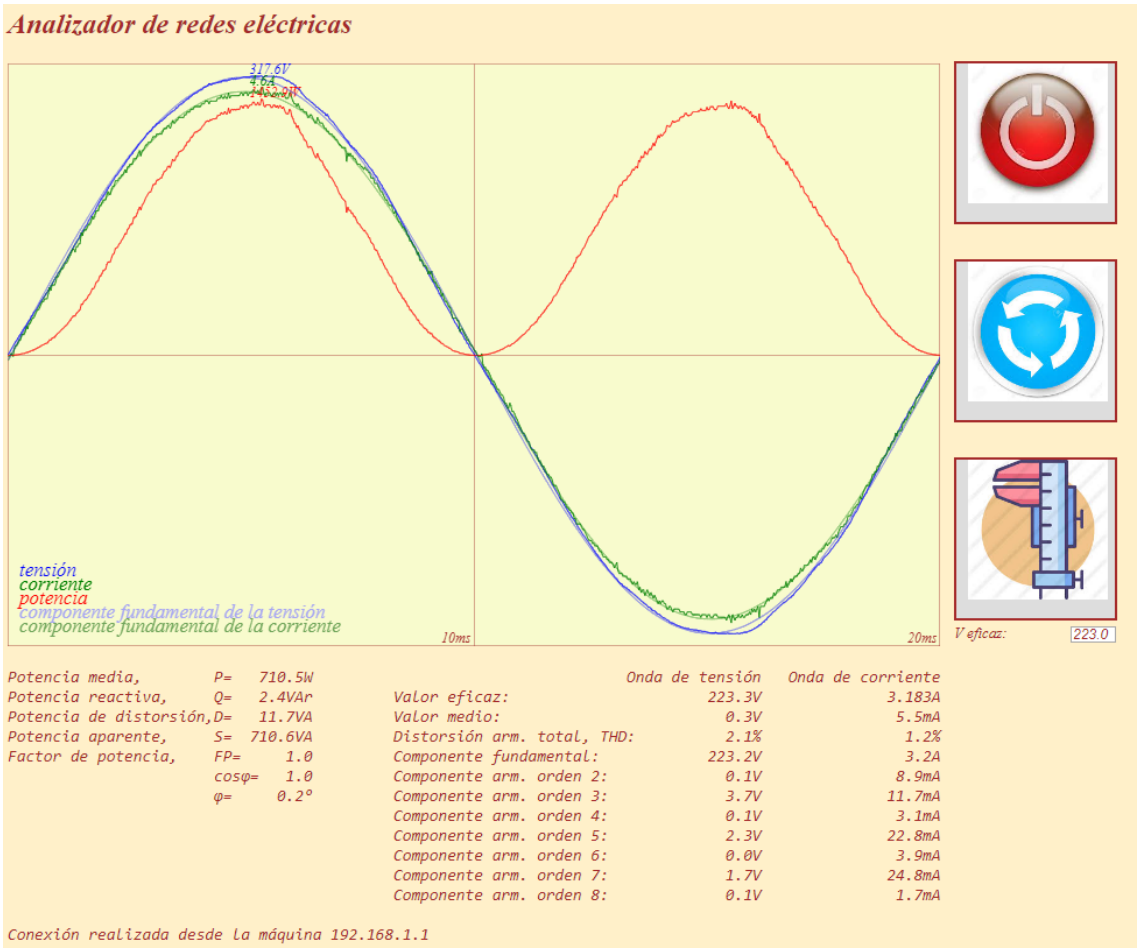


Figura 103: Medidas para una carga de 700W

El ruido ya lo podemos considerar despreciable si lo comparamos con los niveles de corriente que circulan por la carga.

La medición tomada en la carga con un multímetro arroja estos valores:

Tensión eficaz: 224V
Corriente eficaz: 3.19A
Potencia media: 224x3.19=714.56W

Los 50 valores de las tensiones medidas han sido:

222.1,222.0,222.3,222.3,222.4,222.6,223.0,222.8,222.8,223.0,222.6,222.8,222.6,222.6,222.8,223.0,223.1, 223.1,223.0,222.9,223.0,222.8,222.6,222.9,222.8,222.2,222.9,222.6,222.5,222.5,222.7,222.5,222.5,222.7, 222.5,222.4,222.7,222.8,222.7,222.9,222.9,223.0,223.0,223.1,223.1,223.1,223.3,223.0,222.8,223.0

, que corresponden a una media de 222.746V y una desviación típica de 0.29V

Los 50 valores de las corrientes medidas han sido:

3.204,3.203,3.204,3.206,3.209,3.209,3.209,3.212,3.211,3.208,3.204,3.202,3.202,3.198,3.204,3.205,3.207, 3.204,3.208,3.208,3.204,3.201,3.202,3.204,3.205,3.196,3.205,3.198,3.193,3.201,3.199,3.196,3.192,3.201, 3.202,3.199,3.196,3.199,3.202,3.205,3.203,3.204,3.209,3.207,3.200,3.207,3.208,3.199,3.201,3.199

, que corresponden a una media de 3.2031A y una desviación típica de 0.00451A

Los 50 valores de las potencias medidas han sido:

711.5,710.8,712.1,712.5,713.5,714.2,715.5,715.6,715.3,715.1,712.9,713.3,712.6,711.8,713.7,714.9,715.3, 714.7,715.3,715.1,714.4,713.0,712.7,714.0,713.9,710.2,714.3,711.8,710.4,711.9,712.4,710.9,710.1,712.8, 712.2,711.3,711.5,712.5,713.1,714.2,714.0,714.2,715.5,715.5,713.9,715.2,716.2,713.4,713.0,713.3

, que corresponden a una media de 713.35W una desviación típica de 1.60W

Coeficientes de variación:

$$CV_V = 100\frac{0.29}{221.746} = 0.13\%$$
$$CV_I = 100\frac{0.00451}{3.2031} = 0.14\%$$
$$CV_P = 100\frac{1.60}{713.35} = 0.22\%$$

Errores medios relativos:

$$e_V = 100\frac{|222.746 - 224|}{224} = 0.56\%$$
$$e_I = 100\frac{|3.2031 - 3.19|}{3.19} = 0.41\%$$
$$e_P = 100\frac{|713.35 - 714.56|}{714.56} = 0.17\%$$

Seguimos obteniendo coeficientes de variación y errores por debajo del 1%, confirmando tanto la precisión como la exactitud del aparato.

Medidas para una carga nominal de 1000W

La carga que vamos a utilizar es una tostadora con una potencia nominal 1000W.

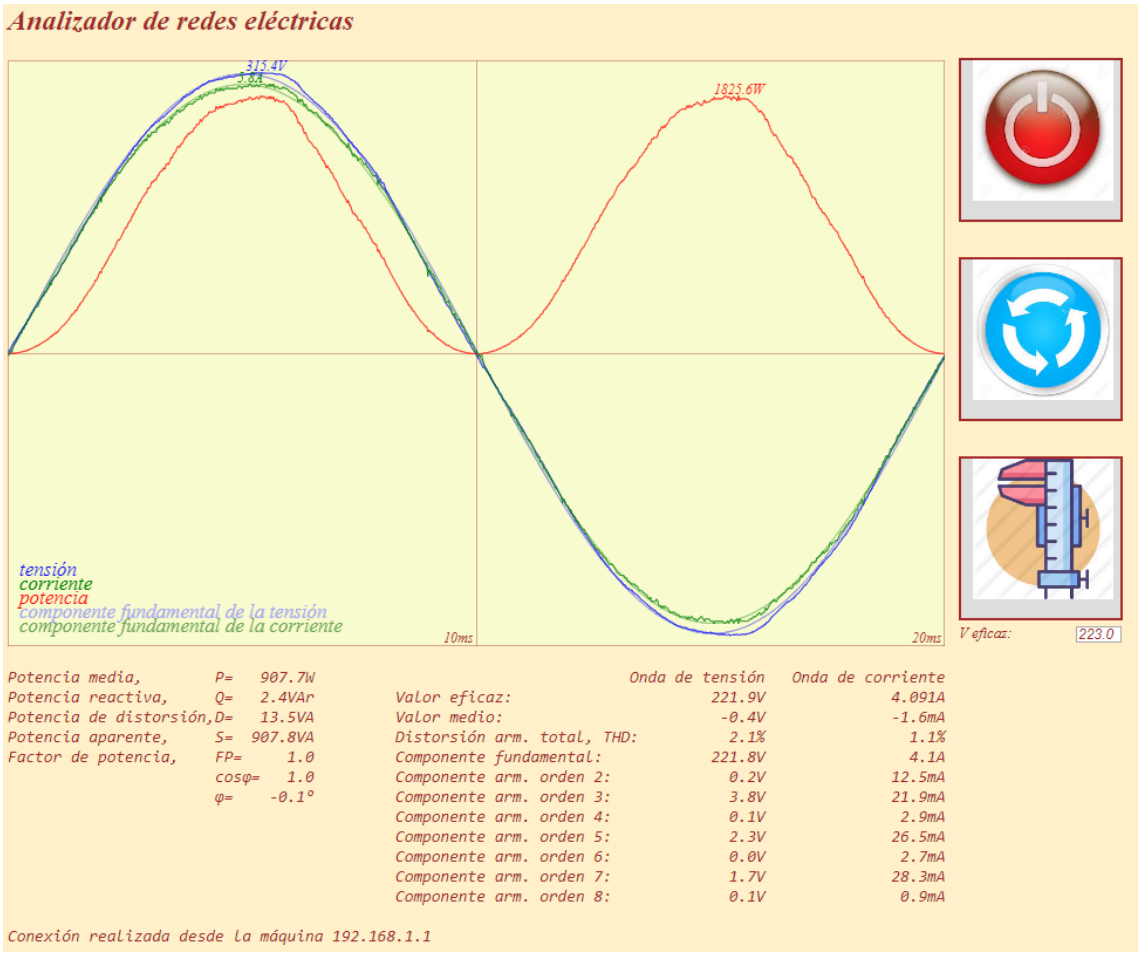


Figura 104: Medidas para una carga de 1000W

La medición tomada en la carga con un multímetro arroja estos valores:

Tensión eficaz: 222.0V
Corriente eficaz: 4.06A
Potencia media: 222.0x4.06=901.3W

Los 50 valores de las tensiones medidas han sido:

221.8,222.2,222.4,222.5,221.9,222.2,222.2,222.3,222.6,222.7,222.8,222.2,222.7,222.2,221.7,221.6,221.6, 221.5,221.2,221.2,221.2,221.6,221.8,221.7,221.5,221.8,221.7,221.0,221.4,221.3,221.4,221.1,221.3,221.1, 221.4,221.8,221.3,221.5,221.3,221.8,221.1,221.2,220.6,221.0,220.8,220.8,220.8,220.8,220.9,221.2

, que corresponden a una media de 221.594V y una desviación típica de 0.57V

Los 50 valores de las corrientes medidas han sido:

4.118,4.128,4.127,4.124,4.116,4.118,4.116,4.113,4.124,4.119,4.127,4.114,4.116,4.115,4.102,4.103,4.102, 4.106,4.100,4.094,4.096,4.095,4.097,4.102,4.093,4.094,4.093,4.088,4.087,4.093,4.095,4.088,4.079,4.083, 4.090,4.085,4.094,4.094,4.078,4.091,4.079,4.086,4.074,4.071,4.068,4.070,4.067,4.074,4.071,4.065

, que corresponden a una media de 4.0964A y una desviación típica de 0.0178A

Los 50 valores de las potencias medidas han sido:

913.4,916.9,917.6,917.4,913.4,914.7,914.5,914.1,918.0,917.3,919.3,914.2,916.5,914.4,909.2,909.4,909.1, 909.4,906.7,905.5,906.1,907.2,908.4,909.3,906.4,907.9,907.2,903.3,904.8,905.6,906.5,903.9,902.7,902.8, 905.4,905.8,906.1,906.7,902.3,907.2,901.8,903.6,898.6,899.8,897.8,898.7,898.0,899.4,899.1,899.1

, que corresponden a una media de 907.65W una desviación típica de 6.1W

Coeficientes de variación:

$$CV_V = 100\frac{0.57}{221.594} = 0.26\%$$
$$CV_I = 100\frac{0.00178}{4.0964} = 0.43\%$$
$$CV_P = 100\frac{6.1}{907.65} = 0.67\%$$

Errores medios relativos:

$$e_V = 100\frac{|221.594 - 222.0|}{222.0} = 0.18\%$$
$$e_I = 100\frac{|4.0964 - 4.06|}{4.06} = 0.9\%$$
$$e_P = 100\frac{|907.65 - 901.32|}{901.32} = 0.7\%$$

Se siguen manteniendo los coeficientes de variación y errores por debajo del 1% confirmando el cumplimiento de los requisitos que nos propusimos como objetivo de este trabajo.

Medidas para una carga electrónica de 70W

La carga que vamos a utilizar es una batidora.

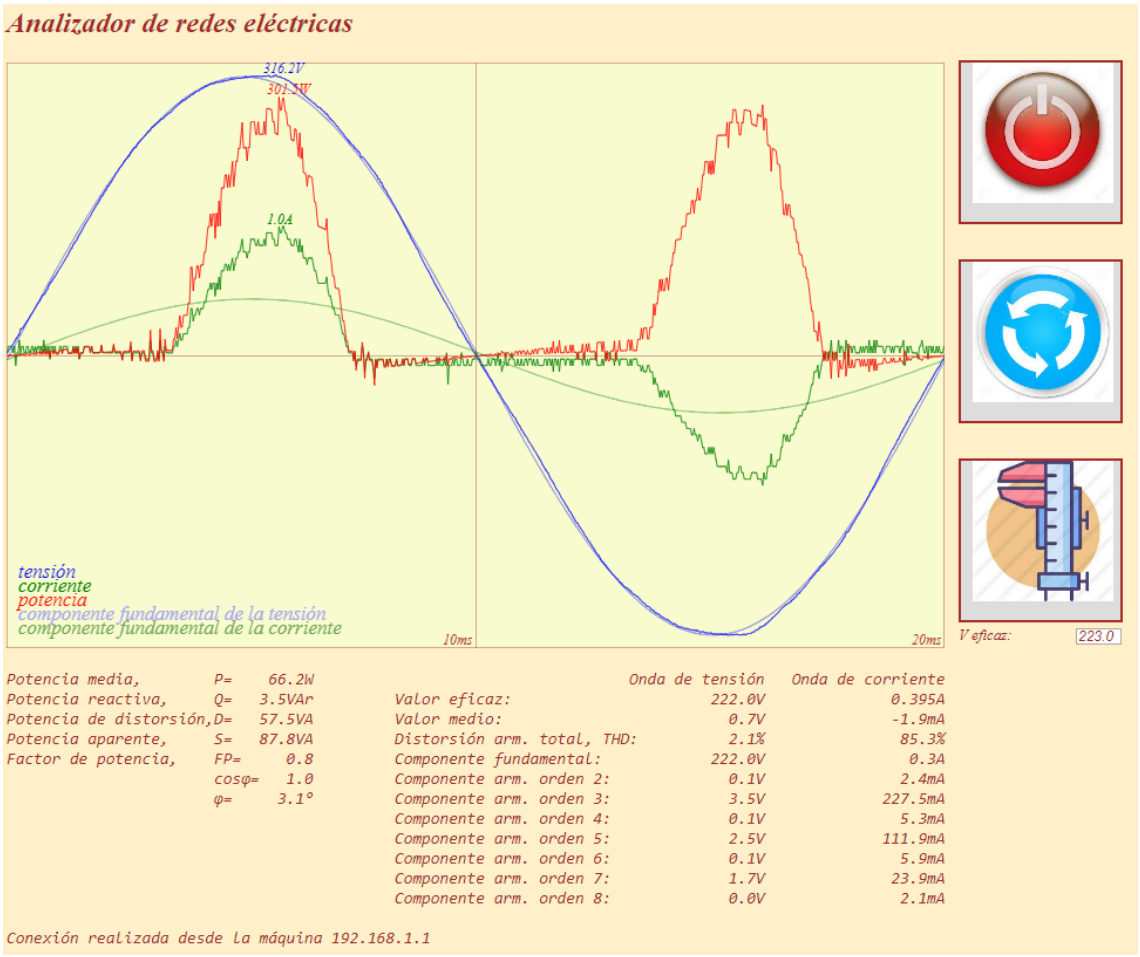


Figura 105: Medidas para una carga electrónica de 70W

Se aprecia que la corriente solo circula es un porcentaje del periodo de la onda, lo que se utiliza para controlar la potencia del motor de la batidora y por tanto su velocidad.

Al tratarse de una carga electrónica, es decir, una carga con elementos no lineales, la distorsión de la onda de corriente es muy alta llegando en este caso a ser del 85.3%.

También podemos observar que al ser una carga inductiva (motor con bobinado), el desfase entre las componentes fundamentales de la tensión y la corriente es superior a las medidas realizadas en los ensayos anteriores. En este caso la corriente se ha retrasado 3.1º respecto a la tensión.

El ruido en la señal de corriente es alto en comparación con los niveles de intensidad que circulan por la carga debido al campo magnético generado por el motor de la batidora.

Medidas para una carga electrónica de 300W

La carga que hemos utilizado es una lavadora.

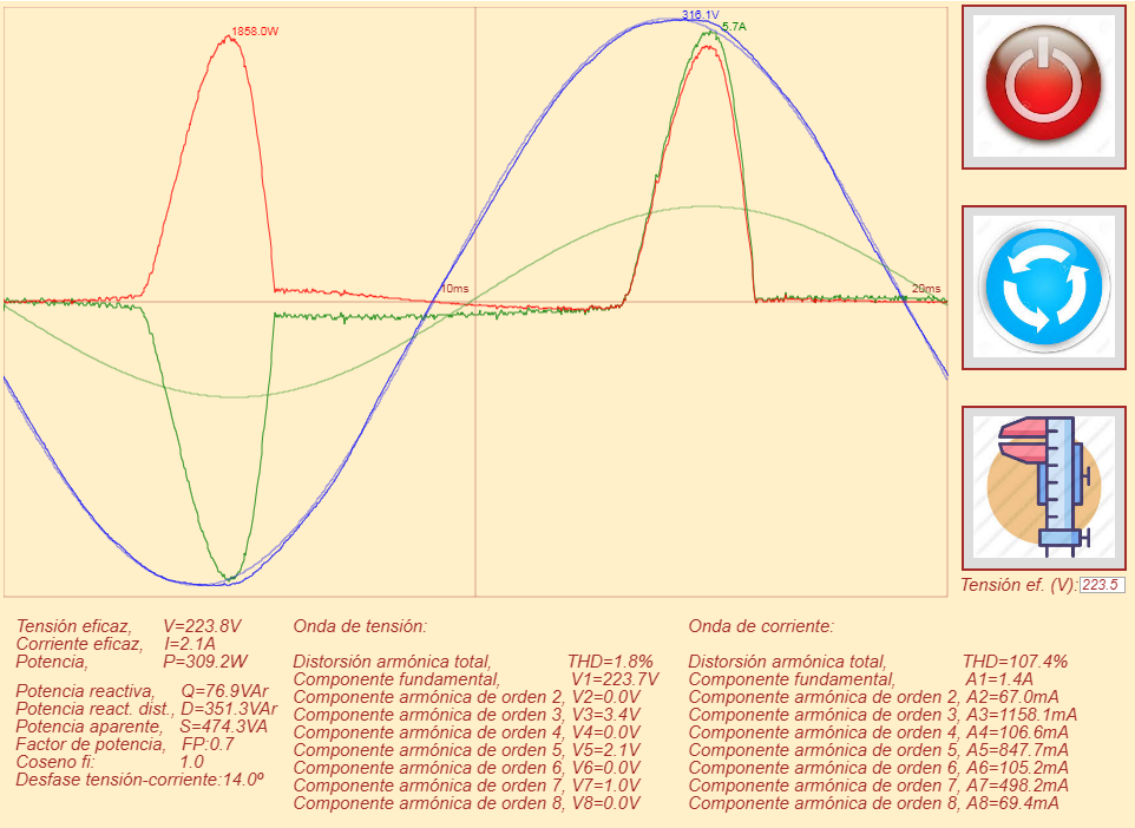


Figura 106: Medidas para una carga electrónica de 300W

Se aprecia, al igual que en el ensayo anterior, el control de velocidad del motor de lavadora, cuyo efecto sobre la corriente es permitir el paso únicamente durante un porcentaje del semiperiodo. Esto provoca la aparición de armónicos principalmente de orden impar en la señal de corriente.

Medidas de armónicos. Comparación con Matlab

Para contrastar los parámetros armónicos aportados por nuestro dispositivo vamos a compararlos con los obtenidos por la función FFT de Matlab. Dicha función calcula la transformada discreta de Fourier (DFT) usando un algoritmo de transformada rápida de Fourier (FFT).

Procederemos de la siguiente forma:

1. *Ejecutamos por consola ("shell") de la Raspberry el programa "tfg", que nos devolverá las tensiones para cada muestra y los valores de los armónicos. Previamente, en el código "tfg.c" tendremos que des-comentar esta instrucción de la función "obtenerTensionesCorrientes()" para que nos presente en pantalla las tensiones de cada muestra:*

```
printf("tension(%3i)=%.1f;\n",i+1,tension[i]);
```

También tendremos que des-comentar esta parte del código de la función "obtenerArmonicosTension()" para que nos presente en pantalla los parámetros armónicos:

```
/*
printf("Valor medio: %7.2f\n",Vm);
printf("Valor eficaz: %7.2f\n",Vef);
printf("Componente fundamental: %7.2f\n",arm1t);
printf("Distorsión armonica total, THD(%): %7.2f\n",100*thdt);
printf("Distorsión armónica de orden 2 (%): %7.2f\n",100*th2t);
printf("Distorsión armónica de orden 3 (%): %7.2f\n",100*th3t);
printf("Distorsión armónica de orden 4 (%): %7.2f\n",100*th4t);
printf("Distorsión armónica de orden 5 (%): %7.2f\n",100*th5t);
printf("Distorsión armónica de orden 6 (%): %7.2f\n",100*th6t);
printf("Distorsión armónica de orden 7 (%): %7.2f\n",100*th7t);
printf("Distorsión armónica de orden 8 (%): %7.2f\n",100*th8t);
*/
```

, y finalmente comentar esta instrucción de la función "main()" para que nos presente en pantalla la salida HTML:

```
crearHTML();
```

Ahora procederemos a compilar el cófigo fuente "tfg.c" introduciendo el comando:

```
g++ -o tfg tfg.c -lwiringPi
```

, y lo ejecutaremos con el commando:

```
./tfg
```

Obtenemos el siguiente resultado (salida en pantalla):

```
root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi# ./tfg
tension(  1)=194.4;
tension(  2)=195.2;
tension(  3)=196.8;
tension(  4)=200.0;
tension(  5)=201.6;
tension(  6)=203.2;
        .
        .
        .
tension(764)=184.7;
tension(765)=186.3;
tension(766)=188.7;
tension(767)=191.1;
Valor eficaz de la tensión: 218.427734
Valor eficaz de la componente fundamental de la tensión: 218.376846
Valor eficaz del armónico 2 de la tensión: 0.022624
Valor eficaz del armónico 3 de la tensión: 3.636654
Valor eficaz del armónico 4 de la tensión: 0.059240
Valor eficaz del armónico 5 de la tensión: 2.229455
Valor eficaz del armónico 6 de la tensión: 0.009907
Valor eficaz del armónico 7 de la tensión: 1.239728
Valor eficaz del armónico 8 de la tensión: 0.010708
Distorsión armónica de la onda de tensión: 0.020344
root@raspberrypi:/var/lib/tomcat8/webapps/TFG/WEB-INF/cgi#
```

Figura 107: Obtención de resultados de la onda de tensión "en local"

2. *Realizaremos un Script en Matlab donde llevaremos estas tensiones obtenidas por el dispositivo. Dicho Script calculará las componentes armónicas así como el coeficiente de distorsión de la señal de tensión muestreada. El detalle de dicho Script se encuentra en "Anexo" de este trabajo.*

3. *Ejecutamos el Scripts de Matlab, y obtenemos el siguiente resultado:*

```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
  Ief: 218.4313V
  Valor eficaz de la componente fundamental: 218.3803A
  Valor eficaz del armónico 2: 0.022075A
  Valor eficaz del armónico 3: 3.6405A
  Valor eficaz del armónico 4: 0.059655A
  Valor eficaz del armónico 5: 2.2252A
  Valor eficaz del armónico 6: 0.010726A
  Valor eficaz del armónico 7: 1.247A
  Valor eficaz del armónico 8: 0.010881A
  Distorsión armónica total: 0.020357
fx >>
```

Figura 108: Ejecución del Script de Matlab para la onda de tensión

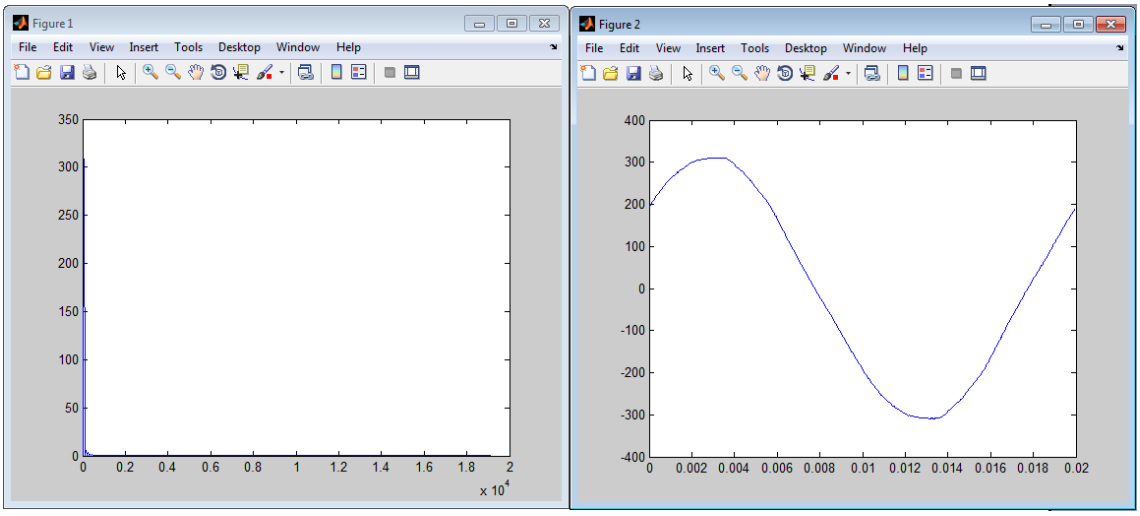


Figura 109: Gráficas obtenidas de la ejecución del Script de Matlab para la onda de tensión

4. *Comparamos los resultados del dispositivo con los de Matlab:*

	Dispositivo	Matlab
Tensión eficaz	218,427734	218,4313
THD	0,020344	0,020357
Tensión eficaz de la componente fundamental	218,376846	218,3803
Tensión eficaz armónico 2	0,022624	0,022075
Tensión eficaz armónico 3	3,636654	3,6405
Tensión eficaz armónico 4	0,05924	0,059655
Tensión eficaz armónico 5	2,229455	2,2252
Tensión eficaz armónico 6	0,009907	0,010726
Tensión eficaz armónico 7	1,239728	1,247
Tensión eficaz armónico 8	0,010708	0,010881

Figura 110: Comparación de resultados dispositivo- Matlab para la onda de tensión

Como vemos, los valores armónicos dados por nuestro dispositivo tienen unos valores prácticamente iguales a los calculados por Matlab, dando por válidos los parámetros armónicos aportados el prototipo.

Capítulo 5: Conclusiones y mejoras

De los ensayos realizados podemos asegurar que el error relativo de nuestro dispositivo es inferior al 1% en cualquiera de las medidas de tensión, corriente o potencia. Igualmente hemos contrastado con Matlab que el cálculo de armónicos realizado por el dispositivo es correcto.

Como ya indicábamos en el apartado de las medidas, el dispositivo construido tiene una limitación en la capacidad de medir corrientes. Esta limitación está condicionada tanto por el sensor de corriente como por el relé utilizado. El sensor de corriente LEM LTS 6-NP puede medir cargas de hasta 19.2A mientras que el relé soporta intensidades de unos 6A. De ambas limitaciones se deduce que este instrumento no es apto para cargas de más de 6A, que a una tensión de 230VAC corresponde a una potencia de 230x6=1380W.

Esta limitación de 6A en el relé se puede incrementar hasta los 19.2A sustituyendo el Triac BTA16 por uno superior, por ejemplo BTA24 o BTA41, y posiblemente también tendremos que sustituir el disipador por uno de menor resistencia térmica. La potencia máxima en este caso sería de 230x19.2=4416W.

Para medir cargas superiores a 4.4kW ya tendríamos que sustituir tanto el sensor de corriente como el relé.

Si deseamos sustituir el sensor de corriente deberemos modificar la siguiente instrucción del fichero /TFG/cgi/tfg.c asignando la sensibilidad correcta del mismo (en voltios/amperio):

```
#define senSensorCorr 0.0763
```

Respecto a la máxima tensión de red que admite el dispositivo, en el apartado donde describimos el sensor de tensión obtuvimos que la señal a su salida esta relacionada con la tensión de la red mediante la siguiente ecuación:

$v_C(t) = 1.65 + \frac{v_i(t)}{241}$ *, y como* $v_C(t) \le 3.3V$ *para no dañar al Arduino,*

$1.65 + \frac{v_i(t)}{241} \le 3.3 \rightarrow v_i(t) \le 398V$

, luego la tensión eficaz de red no debe superar $\frac{397}{\sqrt{2}} = 281V$ *. Las empresas distribuidoras de energía tienen un margen del 10% respecto a una tensión eficaz de 230V, es decir,* $207 \le V \le 253$ *, tensiones inferiores al máximo permitido por nuestro dispositivo.*

Una mejora apreciable habría sido adoptar una distribución mas adecuada de los elementos que componen el dispositivo de forma que se hubieran minimizado la longitud del cableado. Con ello se hubiera incrementado la inmunidad al ruido del dispositivo y por tanto un menor rizado en la señal de corriente.

Otra mejora habría sido el implementar el diseño, tanto del relé como de los sensores de tensión y de corriente, en un mismo circuito impreso (PCB), lo que habría supuesto una mejora en el espacio ocupado así como la eliminación de cableado y reducción de ruido.

Aunque finalmente se ha apantallado el cableado de salida de los sensores de corriente y de tensión, esta se ha realizado de forma "casera" usando papel de aluminio alimentario conectandolo a tierra por uno de los extremos, cuando lo aconsejable habría sido utilizar cableado ya apantallado.

Otra mejora habría sido el diseñar e implementar las fuentes de alimentación tanto de Raspberry como del Arduino, habiéndolas integrado en el mismo circuito impreso (PCB) que el resto de circuitería. De esta forma el dispositivo estaría formado por únicamente tres tarjetas: Raspberty, Arduino y el resto de electrónica.

Finalmente, hubiera sido aconsejable haber protegido la aplicación web con contraseña de forma que el acceso hubiera estado restringido a usuarios autorizados.

Capítulo 6: Presupuesto

En este capítulo se detallará el presupuesto del proyecto desglosado en materiales, mano de obra, gastos generales y beneficio industrial. Los precios de los componentes son aproximados dada la gran variedad de ofertas en el mercado así como una continua bajada de precios al que están sometidos.

Coste de los materiales

Descripción	Cantidad	Coste unitario (€)	Coste total (€)
Arduino DUE	1	11,12	11,12
Cableado de 6mm2 (1 metro)	1	1,00	1,00
Cable micro-USB a USB tipo A de 20cm	1	1,10	1,10
Cableado de colores de 2.54mm	1	1,16	1,16
Caja de proyecto	1	10,00	10,00
Condensador 10uF, 16V	1	0,05	0,05
Conector de 2 terminales para PCB	2	0,07	0,14
Conector de 40 pines, 2.54mm, hembra	1	0,25	0,25
Conector de 40 pines, 2.54mm, macho	1	0,25	0,25
Conector de red hembra	2	1,00	2,00
Fuente de alimentación 230V-12V 2A	1	4,12	4,12
Fuente de alimentación 230V-5V 2A	1	4,90	4,90
LEM LTS 6-NP	1	23,02	23,02
Modem 3G ZTE MF-190	1	12,00	12,00
PCB de 2x8cm de doble cara	3	0,25	0,75
Raspberry Pi 3 modelo B	1	36,00	36,00
Triac BTA16	1	0,15	0,15
MOC3041	1	0,88	0,88
Resistencia de 220, 1/4W	1	0,02	0,02
Resistencia de 360, 1/4W	1	0,02	0,02
Disipador SK12925,4 de Fischer Elektronik	1	0,88	0,88
Resistencia de 100k, 1/4W	1	0,02	0,02
Resistencia variable de 100k, 1/2W	1	0,10	0,10
Resistencia variable de 500k, 1/2W	4	0,10	0,40
Transformador reductor 20.4:1	1	13,00	13,00
Total coste materiales			123,33

Coste de la mano de obra

Descripción	Cantidad	Coste unitario (€)	Coste total (€)
Tiempo de análisis	100	60,00	6000,00
Tiempo de codificación	200	80,00	16000,00
Tiempo de implementación	100	30,00	3000,00
Tiempo de documentación	30	30,00	900,00
Total coste mano de obra			25900,00

Coste total del proyecto

Descripción	Coste total (€)
Total coste de manteriales (MT)	123,33
Total coste mano de obra (MO)	25900,00
Gastos generales: 6%(MT+MO)	1561,40
Beneficio industrial: 13%(MT+MO)	3383,03
Coste total del proyecto	30967,76

Bibliografía

[1] "Medir intensidad y consumo eléctrico con Arduino y ACS712". Luis Llamas
https://www.luisllamas.es/arduino-intensidad-consumo-electrico-acs712/

[2] "Incidencias de Cargas no Lineales en Transformadores de Distribución". Ruggero Ríos B., Sánchez Quintana ME.
https://ucsa.edu.py/yeah/wp-content/uploads/2014/12/8.-AO.-Ruggero-B.-28-44.pdf

[3] "Armónicos en Sistemas Eléctricos". Jose Dariel Arcila
http://ingenieros.es/files/proyectos/Armonicos_en_sistemas_electricos.pdf

[4] "Calculating Power Quantities with Perception Software". Johannes Teigelkotter
https://www.hbm.com/en/3783/calculating-power-quantities-with-perception-software/

[5] "Introducción canvas". Jhonatan Salguero
http://www.novatoz.com/2015/06/introduccion-canvas/

[6] "Tutorial sensor de corriente ACS712". www.naylampmechatronics.com
https://naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html

[7] "Medir tensiones de 220V-230V con arduino y transformador". Luis Llamas
https://www.luisllamas.es/medir-tensiones-de-220v-230v-con-arduino-y-transformador/

[8] "Instalar Raspbian Server en una Raspberry Pi sin monitor"
https://www.flopy.es/instalar-raspbian-server-en-una-raspberry-pi-sin-monitor/

[9] "Cómo instalar Arduino en Windows"
https://arduino.cl/como-instalar-arduino-en-windows/

[10] "MF190 USB Modem Quick Guide"
http://download.ztedevice.com/UploadFiles/product/599/1730/manual/P020110829325168126267.pdf

[11] "Transformada Rápida de Fourier"
https://es.mathworks.com/help/matlab/ref/fft.html

[12] "FFT en Matlab de una onda senoidal"
http://users.df.uba.ar/bragas/Labo3%202016/FFT%20en%20Matlab.pdf

[13] "Evitar autoset Arduino"
http://pikimorgan.blogspot.com/2013/07/evitar-auto-reset-arduino.html

[14] Datasheets del sensor LEM LTS 6-NP
https://www.lem.com/sites/default/files/products_datasheets/lts_6-np.pdf

[15] Datasheets del sensor LEM LV 25-P
https://www.lem.com/sites/default/files/products_datasheets/lv_25-p.pdf

[16] WiringPi Serial Library
http://wiringpi.com/reference/serial-library/

[17] Arduino Uno, partes y componentes
https://descubrearduino.com/arduino-uno/

[18] Raspberry Pi 3 modelo B. Características
https://www.raspberryshop.es/raspberry-pi-3.php

[19] Raspberry Pi 3 modelo B. Guía Raspberry Pi para principiantes
https://www.raspberryshop.es/guia-completa-raspberry-pi.php

[20] Armónicos en redes eléctricas

https://www.academia.edu/10629493/CAPITULO_2_ARMONICOS_EN_LAS_REDES_ELECTRICAS

[21] Raspberry. Usando el puerto GPIO.
http://diymakers.es/usando-el-puerto-gpio/

[22] Librería WiringPi
http://rpiplus.blogspot.com/2013/06/libreria-wiring-pi.html

[23] Raspberry. Gordon Projects
https://projects.drogon.net/raspberry-pi/wiringpi/functions/

[24] Datasheets MOC3041
http://www.farnell.com/datasheets/97984.pdf

[25] Datasheets BTA16
http://www.farnell.com/datasheets/1708247.pdf

[26] Instalación de Eclipse
http://www.edu4java.com/es/servlet/servlet4.html

[27] Dynamic Web Project missing in Eclipse IDE
https://medium.com/kode-blood/dynamic-web-project-missing-in-eclipse-ide-42d72c350f21

Anexos

Programa C++ del Arduino

```
const int N=6; //Número de lecturas que promediarán una muestra de corriente o de tensión
const int numeroMuestras=767;  //Número de muestras de tensiones y corrientes que el Arduino enviará a la Raspberry
                    //para N=6-->numeroMuestras=719
const int numeroLecturas=numeroMuestras*N; //Número de lecturas de tensiones y corrientes que tomará en Arduino

const int pinLecturasTensiones=A10;    //Entrada analógica A10 por donde leeremos la tensión
const int pinLecturasCorrientes=A11;   //Entrada analógica A11 por donde leeremos la corriente
const int pinReleEscritura=53;  //Por el pin 53 actuaremos sobre el relé
const int pinReleLectura=52;    //Por el pin 52 leeremos el estado del relé

int lecturaV[1000*N];   //Lecturas de tensiones
int lecturaI[1000*N];   //Lecturas de corrientes
int muestraV[numeroMuestras]; //Muestras de tensiones
int muestraI[numeroMuestras]; //Muestras de de corriente

void setup() {
  Serial.begin(115200);          // Inicializamos el puerto serie a esa velocidad
  pinMode(pinReleEscritura,OUTPUT); //Establecemos el pin 53 como de escritura (modificará el estado del relé)
  pinMode(pinReleLectura,INPUT);    //Establecemos el pin 52 como de lectura (leerá el estado del relé)
}

void loop() {
  char caracter;
  if (Serial.available()>0){          //Si tenemos caracteres por leer en el puerto serie...
    caracter=Serial.read(); //Leemos un caracter del puerto serie
    if (caracter=='*'){          //Si hemos recibido el caracter '*'

      //Hacemos lecturas de tensiones de preparación durante 20ms
      long t=micros();  //Anotamos el momento en que se comienzan las lecturas de tensiones
      while (micros()-t<20000){
        analogRead(pinLecturasTensiones); //hacemos lecturas (sin guardarlas) durante 20ms
      }

      //Tomamos las lecturas de tensiones
      t=micros();  //Anotamos el momento en que se comienzan las lecturas de tensiones
      for (int i=0;i<1000*N;i++){
        lecturaV[i]=analogRead(pinLecturasTensiones); //Tomamos las lecturas de tensiones
      }

      //Esperamos a que pasen 2 periodos desde la primera lectura de la tensión para que
      //las ondas de tensión y corriente estén en fase en caso de una carga resistiva
      while (micros()-t<40085){  //Realmente es algo mas de 2 periodos (40000us)
        //para hacerlo más inductivo tenemos que disminuir esa cantidad
        analogRead(pinLecturasCorrientes); //Lecturas de corrientes de preparación
      }

      //Tomamos las lecturas de las corrientes
      for (int i=0;i<1000*N;i++){
        lecturaI[i]=analogRead(pinLecturasCorrientes); //Tomamos las lecturas de corrientes
      }

      //Calculamos las muestras como media de cada N lecturas
      for (int i=0;i<numeroMuestras;i++){
        muestraV[i]=0;
        muestraI[i]=0;
        for (int j=0;j<N;j++){
          muestraV[i]=muestraV[i]+lecturaV[N*i+j];
          muestraI[i]=muestraI[i]+lecturaI[N*i+j];
        }
        muestraV[i]=round(muestraV[i]/N);
        muestraI[i]=round(muestraI[i]/N);
      }

      char cadena[5]; //Cada muestra se enviará como 4 caracteres mas 1 de fin de caracter, \0
      //Enviamos a la Raspberry las muestras de las tensiones
      for (int i=0;i<numeroMuestras;i++){
        sprintf(cadena,"%4i",muestraV[i]); //Convertimos la lectura en 4 caracteres
        Serial.print(cadena);
```

```
    }

    //Enviamos a la Raspberry las muestras de las corrientes
    for (int i=0;i<numeroMuestras;i++){
      sprintf(cadena,"%4i",muestraI[i]); //Convertimos la lectura en 4 caracteres
      Serial.print(cadena);
    }
   }
   if (caracter=='0'){ //Si es el caracter recibido es un '0'...
    digitalWrite(pinReleEscritura,LOW); //Ponemos el relé a '0'
   }
   if (caracter=='1'){ //Si es el caracter recibido es un '1'...
    digitalWrite(pinReleEscritura,HIGH);  //Ponemos el relé a '1'
   }
   if (caracter=='2'){ //Si es el caracter recibido por el puerto serie es un '2'...
    int lectura=digitalRead(pinReleLectura);  //Leemos el estado del relé
    if (lectura==1) Serial.print("1");  //Si el relé está en '1' enviamos un '1'
    if (lectura==0) Serial.print("0");  //Si el relé está en '0' enviamos un '0'
   }
  }
}
```

Código C++ TFG/cgi/tfg.c de la Raspberry

```
//Librerías
#include <time.h>            //para poder usar la función time() y ctime()
#include <stdio.h>           //para poder usar la función printf
#include <wiringSerial.h>        //para poder usar las funciones del puerto serie
#include <math.h>                //para poder usar las funciones seno, coseno, potencia
#include <unistd.h>          //para poder usar la función usleep()
#include <string.h>              //para poder usar strcpy()

#define numeroMuestras 767   //número de muestras que se van a tomar en un periodo de 20ms
                                 //180 para el Arduino UNO, y 4420 para el Arduino DUE
#define NBits 10                 //Número de bits del convertidor analógico-digital del Arduino
                                 //10 para el Arduino UNO, y 12 para el Arduino DUE
#define pi 3.1415926
#define bpsUART 115200           //Velocidad del puerto serie UART
#define senSensorCorr 0.0736 //sensibilidad del sensor de corriente (incluido el circuito
                    // de acondicionamiento), medido en voltios/amperios

void capturarLecturasDelArduino();  //La Raspberry solicita las muestras al Arduino
void obtenerTensionesCorrientes();  //Traduce las muestras a voltios y amperios
void ordenarMuestras();
void obtenerArmonicosTension();          //Obtiene armónicos y distorsión de la tensión
void obtenerArmonicosCorriente();    //Obtiene armónicos y distorsión de la corriente
void obtenerFactorPotencia();      //Obtiene potencias y factor de potencia
void crearHTML();                  //Crea el código HTML
void ondasPrueba();
void registro();                 //Registra la dirección IP y fecha de la máquina que ha accedido a la página

int muestraV[numeroMuestras]; //aquí guardamos las lecturas analógicas de tensiones del arduino
int muestraI[numeroMuestras];  //aquí guardamos las lecturas analógicas de corrientes del arduino
int muestraVmax;             //mayor lectura analógica de tensión que lee el arduino
int muestraVmin;             //menor lectura analógica de tensión que lee el arduino
float muestraImedia;         //media de las lectura de corriente que lee el arduino

float tension[numeroMuestras];  //aquí guardamos las tensiones muestreadas (valores en voltios)
float corriente[numeroMuestras];//aquí guardamos las corrientes muestreadas (valores en amperios)
float potencia[ numeroMuestras];//aquí guardamos los valores de las potencias de cada muestra (valores en vatios)

float Vmax;              //mayor tensión muestreada (en voltios)
int nVmax;               //número de lectura en el que se ha tomado la mayor tensión
float Imax;              //mayor corriente muestreada (en amperios)
int nImax;               //número de lectura en el que se ha tomado la mayor corriente
float Pmax;              //mayor potencia de todas las muestras (en vatios)
int nPmax;               //número de lectura en el que se ha dado la mayor potencia

float Vef;           //tensión eficaz (en voltios)
float Ief;           //corriente eficaz (en amperios)
float Vm;            //tensión media (en voltios)
float Im;            //corriente media (en amperios)

float potenciaP;         //Potencia activa (en W)
float potenciaQ;         //Potencia reactiva (en VAR)
float potenciaS;         //Potencia aparente (en VA)
float potenciaD;         //Potencia reactiva de distorsión (var)
float fp;                //Factor de Potencia
float fi_v1;             //fase de la componente fundamental de la tensión
float fi_c1;             //fase de la componente fundamental de la corriente
float cos_fi;            //cos de fi (desfase entre la tensión y la componente fundamental de la corriente)

float VmaxGrafica;       //Tensión máxima que se va a poder dibujar (en voltios)
float ImaxGrafica;       //Corriente máxima que se va a poder dibujar (en amperios)
float PmaxGrafica;       //Potencia máxima que se a va poder dibujar (en vatios)

float arm1t;                         //valor eficaz de la componente fundamental de la tensión (en voltios)
float arm2t;                         //valor eficaz del armónico 2 de la tensión (en voltios)
float arm3t;                         //valor eficaz del armónico 3 de la tensión (en voltios)
float arm4t;                         //valor eficaz del armónico 4 de la tensión (en voltios)
float arm5t;                         //valor eficaz del armónico 5 de la tensión (en voltios)
float arm6t;                         //valor eficaz del armónico 6 de la tensión (en voltios)
float arm7t;                         //valor eficaz del armónico 7 de la tensión (en voltios)
```

```
float arm8t;                    //valor eficaz del armónico 8 de la tensión (en voltios)

float thdt;                     //Coeficiente de distorsión armónica total de la tensión
float th2t;                     //Coeficiente de distorsión del armónic0 2 de la tensión
float th3t;                     //Coeficiente de distorsión del armónic0 3 de la tensión
float th4t;                     //Coeficiente de distorsión del armónic0 4 de la tensión
float th5t;                     //Coeficiente de distorsión del armónic0 5 de la tensión
float th6t;                     //Coeficiente de distorsión del armónic0 6 de la tensión
float th7t;                     //Coeficiente de distorsión del armónic0 7 de la tensión
float th8t;                     //Coeficiente de distorsión del armónic0 8 de la tensión

float arm1c;            //valor eficaz de la componente fundamental de la tensión (en voltios)
float arm2c;            //valor eficaz del armónico 2 de la tensión (en voltios)
float arm3c;            //valor eficaz del armónico 3 de la tensión (en voltios)
float arm4c;            //valor eficaz del armónico 4 de la tensión (en voltios)
float arm5c;            //valor eficaz del armónico 5 de la tensión (en voltios)
float arm6c;            //valor eficaz del armónico 6 de la tensión (en voltios)
float arm7c;            //valor eficaz del armónico 7 de la tensión (en voltios)
float arm8c;            //valor eficaz del armónico 8 de la tensión (en voltios)

float thdc;             //Coeficiente de distorsión armónica total de la tensión
float th2c;             //Coeficiente de distorsión del armónic0 2 de la tensión
float th3c;             //Coeficiente de distorsión del armónic0 3 de la tensión
float th4c;             //Coeficiente de distorsión del armónic0 4 de la tensión
float th5c;             //Coeficiente de distorsión del armónic0 5 de la tensión
float th6c;             //Coeficiente de distorsión del armónic0 6 de la tensión
float th7c;             //Coeficiente de distorsión del armónic0 7 de la tensión
float th8c;             //Coeficiente de distorsión del armónic0 8 de la tensión

char maquina[30];               //Va a contener la dirección IP de la máquina que solicita la página

int main (int argc, char *argv[], char *env[]) {
 strcpy(maquina,env[2]);        //copiamos en la variable "maquina" la dirección IP de host remoto
 capturarLecturasDelArduino();
 obtenerTensionesCorrientes();
 ordenarMuestras();
 //ondasPrueba();
 obtenerArmonicosTension();
 obtenerArmonicosCorriente();
 obtenerFactorPotencia();
 crearHTML();
 registro();
 return 0;
}

void registro(){

   char fecha[25];//ctime devuelve 26 caracteres pero también se podría usar un puntero de char
   time_t current_time;
   current_time=time(NULL);
   ctime(&current_time);
   strcpy(fecha,ctime(&current_time));
   FILE *fp=fopen("../../registro.txt","a"); //abrimos /TFG/registro.txt como escritura al final del mismo
   if (fp==NULL) {
     printf("No se puede abrir el fichero registro.txt desde tfg\n");
   }
   fecha[sizeof(fecha)-1]='\0'; //le quitamos a "fecha" el último carácter, que es un \n
   //fprintf(fp,"IP remoto:%s  Fecha:%s Vef:%.1fV Ief:%.3fA P:%.1fW\n",strchr(maquina,'=')+1,fecha,Vef,Ief,potenciaP);
   fprintf(fp,"IP remoto:%s  Fecha:%s Vef:%.1fV Ief:%.3fA P:%.1fW\n",maquina,fecha,Vef,Ief,potenciaP);
   fclose(fp);
   return;
}

//Genera ondas de prueba de tensión y corriente
//Para llamar a ondasPrueba() no podemos llamar a ninguna de estas funciones:
//            capturarLecturasDelArduino();
//            obtenerTensionesCorrientes();
void ondasPrueba(){
 for (int i=0;i<numeroMuestras;i++){
   tension[i]=230.0*sqrt(2)*sin(100*pi*0.02*i/(numeroMuestras-1));
   corriente[i]=5.0*sqrt(2)*sin(100*pi*0.02*i/(numeroMuestras-1)-pi/8); //carga inductiva
   potencia[i]=tension[i]*corriente[i];
 }
```

```
  ImaxGrafica=8.0;
  VmaxGrafica=230.0*sqrt(2);
  PmaxGrafica=5000.0;
}

//Recoge las lecturas tomadas por el arduino
void capturarLecturasDelArduino(){
  int fd;
  if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
    printf("No se puede abrir el dispositivo serie ttyACM0\n");
    return;
  }
  usleep(150);
  int datoLeido;
  serialPutchar(fd,'*'); //Enviamos este carácter al arduino;
  //Lectura de tensiones
  for (int i=0;i<numeroMuestras;i++){
    int total=0;
    for (int j=0;j<4;j++){
      while (serialDataAvail(fd)<0){};
      char c=serialGetchar(fd);
      if (c==32) datoLeido=0; //Un caracter en blanco lo consideramos como un 0
      else datoLeido=c-48; //Convertimos el caracter a un digito decimal
      if (datoLeido>=0 && datoLeido<=9){
        total=total+datoLeido*pow(10,3-j);
      }
    }
    muestraV[i]=total;
    //printf("%4i %i\n",i,muestraV[i]);
  }
  //Lectura de corrientes
  for (int i=0;i<numeroMuestras;i++){
    int total=0;
    for (int j=0;j<4;j++){
      while (serialDataAvail(fd)<0){};
      char c=serialGetchar(fd);
      if (c==32) datoLeido=0; //Un caracter en blanco lo consideramos como un 0
      else datoLeido=c-48; //Convertimos el caracter a un digito decimal
      if (datoLeido>=0 && datoLeido<=9){
        total=total+datoLeido*pow(10,3-j);
      }
    }
    muestraI[i]=total;
    //printf("%4i %i\n",i,muestraI[i]);
  }
  serialClose(fd);

  //Determinación de los valores máximos y mínimos de las lecturas
  muestraVmax=muestraV[0];
  muestraVmin=muestraV[0];
  muestraImedia=0;
  nImax=0;
  for (int i=0;i<numeroMuestras;i++){
    if (muestraV[i]>muestraVmax){
      muestraVmax=muestraV[i];
      nVmax=i;
    }
    if (muestraV[i]<muestraVmin) muestraVmin=muestraV[i];
    muestraImedia=muestraImedia+muestraI[i];
    if (muestraI[i]>muestraI[nImax]){
      nImax=i;
    }
  }
  muestraImedia=muestraImedia/numeroMuestras;
}

//Obtenemos los valores de las tensiones y corrientes a partir de las muestras tomadas
void obtenerTensionesCorrientes(){
  //leemos del fichero "datos.txt" la tensión eficaz de la red
  FILE *fp=fopen("../../datos.txt","r"); //abrimos /TFG/datos.txt como lectura
  if (fp==NULL) {
    printf("No se puede abrir el fichero datos.txt desde tfg\n");
  }
```

```
  char cadena[5];
  fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de tensión máxima
  int datoVmax=atoi(cadena);
  //printf("%i\n",datoVmax);
  fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de tensión mínima
  int datoVmin=atof(cadena);
  fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la lectura de corriente media
  float datoImedia=atof(cadena);
  fscanf(fp,"%s",cadena); //Leemos del fichero "datos.txt" la tensión eficaz
  float datoVef=atof(cadena);
  fclose(fp);
  Pmax=tension[0]*corriente[0];
  /*
  La tensión en cada instante se calcula en función de:
       muestraV[i]: muestra de tensión tomada por el Arduino en ese instante
              datoVef: valor eficaz de la tensión en la fase de calibración
       datoVmax: muestra máxima de la tensión tomada por el Arduino en la fase de calibración
       datoVmin: muestra mínima de la tensión tomada por el Arduino en la fase de calibración

  La corriente en cada instante se calcula en función de:
       muestraI[i]: muestra de corriente tomada por el Arduino en ese instante
       senSensorCorr: sensibilidad del sensor de corriente (en V/A)
       NBits: número de bits del ADC del Arduino
       datoImedia: muestra media de la corriente tomada por el Arduino en la fase de calibración
  */
  for (int i=0;i<numeroMuestras;i++){
    tension[i]=sqrt(2)*datoVef*(1.0-2.0*(datoVmax-muestraV[i])/(datoVmax-datoVmin));
    corriente[i]=5/(senSensorCorr*pow(2,NBits))*(muestraI[i]-datoImedia);
    potencia[i]=tension[i]*corriente[i];
    if (potencia[i]>Pmax){
      Pmax=potencia[i];
      nPmax=i;
     }
    //printf("tension(%3i)=%.1f;\n",i+1,tension[i]); //para comparar con Matlab
  }

  Vmax=tension[nVmax];
  Imax=corriente[nImax];

  VmaxGrafica=Vmax;
  if (Imax<1) ImaxGrafica=2.0; else ImaxGrafica=Imax;
  if (Pmax<500.0) PmaxGrafica=500.0; else PmaxGrafica=Vmax*Imax;
}

//Ordena las muestras de forma que la fase de la onda de tensión a dibujar sea 0
void ordenarMuestras(){
  int j=0;
  int minimo=abs(tension[0]);
  //buscamos la tensión mínima cuando la función crece
  for (int i=1;i<numeroMuestras;i++){
    if (tension[i-1]<tension[i] && abs(tension[i])<minimo){
      j=i;
      minimo=abs(tension[i]);
     }
  }
  j++;
  if (j==numeroMuestras) j=0;
  //if (j>=numeroMuestras) j=j-numeroMuestras;
  //El valor de j es la muestra con tensión 0 en fase creciente
  float mv[numeroMuestras];
  float mi[numeroMuestras];
  //Copiamos las muestras de tensiones y corrientes
  for (int i=0;i<numeroMuestras;i++){
    mv[i]=tension[i];
    mi[i]=corriente[i];
  }
  //Las muestras desde la j hasta el final las ponemos las primeras
  for (int i=j;i<numeroMuestras;i++){
     tension[i-j]=mv[i];
     corriente[i-j]=mi[i];
     potencia[i-j]=mv[i]*mi[i];
  }
  //Las muestras desde la 0 hasta la j-1 la ponemos las últimas
```

```
  for (int i=0;i<j;i++){
    tension[numeroMuestras-j+i]=mv[i];
    corriente[numeroMuestras-j+i]=mi[i];
    potencia[numeroMuestras-j+i]=mv[i]*mi[i];
  }
  //Actualizamos los instantes en que se producen los máximos
  if (nVmax>=j) nVmax=nVmax-j;
  else nVmax=numeroMuestras-j+nVmax;
  if (nImax>=j) nImax=nImax-j;
  else nImax=numeroMuestras-j+nImax;
  if (nPmax>=j) nPmax=nPmax-j;
  else nPmax=numeroMuestras-j+nPmax;

}

//Obtenemos información de los armónicos
void obtenerArmonicosTension(){
  float a1=0;
  float b1=0;
  float a2=0;
  float b2=0;
  float a3=0;
  float b3=0;
  float a4=0;
  float b4=0;
  float a5=0;
  float b5=0;
  float a6=0;
  float b6=0;
  float a7=0;
  float b7=0;
  float a8=0;
  float b8=0;
  for (int i=0;i<numeroMuestras;i++){
    Vef=Vef+tension[i]*tension[i];
    Vm=Vm+1/0.02*tension[i]*(0.02/numeroMuestras);
    a1=a1+sqrt(2)/numeroMuestras*tension[i]*sin(2*pi*50*i*0.02/numeroMuestras);
    b1=b1+sqrt(2)/numeroMuestras*tension[i]*cos(2*pi*50*i*0.02/numeroMuestras);
    a2=a2+sqrt(2)/numeroMuestras*tension[i]*sin(2*2*pi*50*i*0.02/numeroMuestras);
    b2=b2+sqrt(2)/numeroMuestras*tension[i]*cos(2*2*pi*50*i*0.02/numeroMuestras);
    a3=a3+sqrt(2)/numeroMuestras*tension[i]*sin(3*2*pi*50*i*0.02/numeroMuestras);
    b3=b3+sqrt(2)/numeroMuestras*tension[i]*cos(3*2*pi*50*i*0.02/numeroMuestras);
    a4=a4+sqrt(2)/numeroMuestras*tension[i]*sin(4*2*pi*50*i*0.02/numeroMuestras);
    b4=b4+sqrt(2)/numeroMuestras*tension[i]*cos(4*2*pi*50*i*0.02/numeroMuestras);
    a5=a5+sqrt(2)/numeroMuestras*tension[i]*sin(5*2*pi*50*i*0.02/numeroMuestras);
    b5=b5+sqrt(2)/numeroMuestras*tension[i]*cos(5*2*pi*50*i*0.02/numeroMuestras);
    a6=a6+sqrt(2)/numeroMuestras*tension[i]*sin(6*2*pi*50*i*0.02/numeroMuestras);
    b6=b6+sqrt(2)/numeroMuestras*tension[i]*cos(6*2*pi*50*i*0.02/numeroMuestras);
    a7=a7+sqrt(2)/numeroMuestras*tension[i]*sin(7*2*pi*50*i*0.02/numeroMuestras);
    b7=b7+sqrt(2)/numeroMuestras*tension[i]*cos(7*2*pi*50*i*0.02/numeroMuestras);
    a8=a8+sqrt(2)/numeroMuestras*tension[i]*sin(8*2*pi*50*i*0.02/numeroMuestras);
    b8=b8+sqrt(2)/numeroMuestras*tension[i]*cos(8*2*pi*50*i*0.02/numeroMuestras);
  }
  Vef=sqrt(Vef/numeroMuestras);
  arm1t=sqrt(a1*a1+b1*b1);
  arm2t=sqrt(a2*a2+b2*b2);
  arm3t=sqrt(a3*a3+b3*b3);
  arm4t=sqrt(a4*a4+b4*b4);
  arm5t=sqrt(a5*a5+b5*b5);
  arm6t=sqrt(a6*a6+b6*b6);
  arm7t=sqrt(a7*a7+b7*b7);
  arm8t=sqrt(a8*a8+b8*b8);
  th2t=abs(arm2t/arm1t);
  th3t=abs(arm3t/arm1t);
  th4t=abs(arm4t/arm1t);
  th5t=abs(arm5t/arm1t);
  th6t=abs(arm6t/arm1t);
  th7t=abs(arm7t/arm1t);
  th8t=abs(arm8t/arm1t);
  thdt=sqrt(th2t*th2t+th3t*th3t+th4t*th4t+th5t*th5t+th6t*th6t+th7t*th7t+th8t*th8t);
  if (a1>0) fi_v1=atan(b1/a1);
  else fi_v1=atan(b1/a1)+pi;
  /*
```

```
  printf("Valor eficaz de la tensión: %f\n",Vef);
  printf("Valor eficaz de la componente fundamental de la tensión: %f\n",arm1t);
  printf("Valor eficaz del armónico 2 de la tensión: %f\n",arm2t);
  printf("Valor eficaz del armónico 3 de la tensión: %f\n",arm3t);
  printf("Valor eficaz del armónico 4 de la tensión: %f\n",arm4t);
  printf("Valor eficaz del armónico 5 de la tensión: %f\n",arm5t);
  printf("Valor eficaz del armónico 6 de la tensión: %f\n",arm6t);
  printf("Valor eficaz del armónico 7 de la tensión: %f\n",arm7t);
  printf("Valor eficaz del armónico 8 de la tensión: %f\n",arm8t);
  printf("Distorsión armónica de la onda de tensión: %f\n",thdt);
  */
  /*
  printf("Valor medio: %7.2f\n",Vm);
  printf("Valor eficaz: %7.2f\n",Vef);
  printf("Componente fundamental: %7.2f\n",arm1t);
  printf("Distorsión armónica total, THD(%): %7.2f\n",100*thdt);
  printf("Distorsión armónica de orden 2 (%): %7.2f\n",100*th2t);
  printf("Distorsión armónica de orden 3 (%): %7.2f\n",100*th3t);
  printf("Distorsión armónica de orden 4 (%): %7.2f\n",100*th4t);
  printf("Distorsión armónica de orden 5 (%): %7.2f\n",100*th5t);
  printf("Distorsión armónica de orden 6 (%): %7.2f\n",100*th6t);
  printf("Distorsión armónica de orden 7 (%): %7.2f\n",100*th7t);
  printf("Distorsión armónica de orden 8 (%): %7.2f\n",100*th8t);
  */
}

void obtenerArmonicosCorriente(){
  float a1=0;
  float b1=0;
  float a2=0;
  float b2=0;
  float a3=0;
  float b3=0;
  float a4=0;
  float b4=0;
  float a5=0;
  float b5=0;
  float a6=0;
  float b6=0;
  float a7=0;
  float b7=0;
  float a8=0;
  float b8=0;
  for (int i=0;i<numeroMuestras;i++){
    Ief=Ief+corriente[i]*corriente[i];
    Im=Im+1/0.02*corriente[i]*(0.02/numeroMuestras);
    a1=a1+sqrt(2)/numeroMuestras*corriente[i]*sin(2*pi*50*i*0.02/numeroMuestras);
    b1=b1+sqrt(2)/numeroMuestras*corriente[i]*cos(2*pi*50*i*0.02/numeroMuestras);
    a2=a2+sqrt(2)/numeroMuestras*corriente[i]*sin(2*2*pi*50*i*0.02/numeroMuestras);
    b2=b2+sqrt(2)/numeroMuestras*corriente[i]*cos(2*2*pi*50*i*0.02/numeroMuestras);
    a3=a3+sqrt(2)/numeroMuestras*corriente[i]*sin(3*2*pi*50*i*0.02/numeroMuestras);
    b3=b3+sqrt(2)/numeroMuestras*corriente[i]*cos(3*2*pi*50*i*0.02/numeroMuestras);
    a4=a4+sqrt(2)/numeroMuestras*corriente[i]*sin(4*2*pi*50*i*0.02/numeroMuestras);
    b4=b4+sqrt(2)/numeroMuestras*corriente[i]*cos(4*2*pi*50*i*0.02/numeroMuestras);
    a5=a5+sqrt(2)/numeroMuestras*corriente[i]*sin(5*2*pi*50*i*0.02/numeroMuestras);
    b5=b5+sqrt(2)/numeroMuestras*corriente[i]*cos(5*2*pi*50*i*0.02/numeroMuestras);
    a6=a6+sqrt(2)/numeroMuestras*corriente[i]*sin(6*2*pi*50*i*0.02/numeroMuestras);
    b6=b6+sqrt(2)/numeroMuestras*corriente[i]*cos(6*2*pi*50*i*0.02/numeroMuestras);
    a7=a7+sqrt(2)/numeroMuestras*corriente[i]*sin(7*2*pi*50*i*0.02/numeroMuestras);
    b7=b7+sqrt(2)/numeroMuestras*corriente[i]*cos(7*2*pi*50*i*0.02/numeroMuestras);
    a8=a8+sqrt(2)/numeroMuestras*corriente[i]*sin(8*2*pi*50*i*0.02/numeroMuestras);
    b8=b8+sqrt(2)/numeroMuestras*corriente[i]*cos(8*2*pi*50*i*0.02/numeroMuestras);
  }
  Ief=sqrt(Ief/numeroMuestras);
  arm1c=sqrt(a1*a1+b1*b1);
  arm2c=sqrt(a2*a2+b2*b2);
  arm3c=sqrt(a3*a3+b3*b3);
  arm4c=sqrt(a4*a4+b4*b4);
  arm5c=sqrt(a5*a5+b5*b5);
  arm6c=sqrt(a6*a6+b6*b6);
  arm7c=sqrt(a7*a7+b7*b7);
  arm8c=sqrt(a8*a8+b8*b8);
  th2c=abs(arm2c/arm1c);
```

```
  th3c=abs(arm3c/arm1c);
  th4c=abs(arm4c/arm1c);
  th5c=abs(arm5c/arm1c);
  th6c=abs(arm6c/arm1c);
  th7c=abs(arm7c/arm1c);
  th8c=abs(arm8c/arm1c);
  thdc=sqrt(th2c*th2c+th3c*th3c+th4c*th4c+th5c*th5c+th6c*th6c+th7c*th7c+th8c*th8c);
  if (a1>0) fi_c1=atan(b1/a1);
  else fi_c1=atan(b1/a1)+pi;
  /*
  printf("Valor medio: %7.2f\n",Im);
  printf("Valor eficaz: %7.2f\n",Ief);
  printf("Componente fundamental: %7.2f\n",arm1c);
  printf("Distorsión armónica total, THD(%): %7.2f\n",100*thdc);
  printf("Distorsión armónica de orden 2 (%): %7.2f\n",100*th2c);
  printf("Distorsión armónica de orden 3 (%): %7.2f\n",100*th3c);
  printf("Distorsión armónica de orden 4 (%): %7.2f\n",100*th4c);
  printf("Distorsión armónica de orden 5 (%): %7.2f\n",100*th5c);
  printf("Distorsión armónica de orden 6 (%): %7.2f\n",100*th6c);
  printf("Distorsión armónica de orden 7 (%): %7.2f\n",100*th7c);
  printf("Distorsión armónica de orden 8 (%): %7.2f\n",100*th8c);
  */
}


void obtenerFactorPotencia(){
  cos_fi=cos(fi_v1-fi_c1); //desfase entre la onda de tensión y la fundamental de la corriente
  potenciaS=Vef*Ief;  //Potencia aparente
  float potenciaS1=Vef*arm1c;  //Potencia del primer armónico
  potenciaP=potenciaS1*cos_fi; //Potencia activa
  potenciaQ=sqrt(potenciaS1*potenciaS1-potenciaP*potenciaP); //Potencia reactiva
  potenciaD=sqrt(potenciaS*potenciaS-potenciaP*potenciaP-potenciaQ*potenciaQ);    //Potencia reactiva de distorsión
  fp=potenciaP/potenciaS;
}

//Crea la página HTML con los datos obtenidos
void crearHTML(){
  printf("Content-Type:text/html\n\n");
  printf("<!DOCTYPE html>\n");
  printf("<html>\n");
  printf("<head>\n");
  printf("<meta charset='utf-8'>\n");
  printf("<title>Analizador de red</title>\n");

  printf("<style type='text/css'>\n");
  printf("h2{font-size:24px;color:brown;font-style:italic}\n");
  printf("#botonRele{border:2px solid brown;padding:0;position:absolute;top:65px;left:820px;width:140px;height:140px}\n");
  printf("#botonRecarga{border:2px solid
brown;padding:0;position:absolute;top:235px;left:820px;width:140px;height:140px}\n");
  printf("#botonCalibrar{border:2px solid
brown;padding:0;position:absolute;top:405px;left:820px;width:140px;height:140px}\n");
  printf("p{color:brown;font-size:13px;font-style:italic;position:absolute;top:535px;left:820px;}\n");
  printf("#cajaTexto{color:brown;font-size:12px;font-
style:italic;position:absolute;top:550px;left:920px;width:35px;height:8px}\n");
  printf("pre{color:brown;font-size:14px;font-style:italic}\n");
  printf("div{color:brown;font-size:18px;font-style:italic;position:absolute;top:840px;left:0px}\n");
  printf("img{width:120px;height:120px;absolute;position:absolute;top:0px;left:10px}\n");
  printf("div > pre{position:absolute;top:10px;left:150px;}\n");
  printf("</style>\n");

  printf("</head>\n");
  printf("<body style='background-color:#FFF0C9;'>\n");
  printf("<h2> Analizador de redes eléctricas</h2>\n");

  //Aquí comienza en "canvas" de 800x500
  printf("<canvas id='lienzo' width='800' height='500'>\n");
  printf("<script type='text/javascript'>\n");
  printf("var canvas=document.getElementById('lienzo');\n");
  printf("var ctx=canvas.getContext('2d');\n");

  //Color de fondo del canvas
  printf("ctx.fillStyle='#F8FBCD';\n");
  printf("ctx.fillRect(0, 0, canvas.width, canvas.height);\n");
```

```
//Marcas temporales de 10ms y 20ms
printf("ctx.font = 'oblique 12px italic';\n");
printf("ctx.fillStyle = 'brown';\n");            //Color del texto
printf("ctx.fillText('10ms',372,497);\n");
printf("ctx.fillText('20ms',772,497);\n");
//Indicación de las gráficas
printf("ctx.font = 'oblique 17px italic';\n");
printf("ctx.fillStyle = 'blue';\n");
printf("ctx.fillText('tensión',10,440);\n");
printf("ctx.fillStyle = 'green';\n");
printf("ctx.fillText('corriente',10,452);\n");
printf("ctx.fillStyle = 'red';\n");
printf("ctx.fillText('potencia',10,464);\n");
printf("ctx.fillStyle = '#A6A5F4';\n");
printf("ctx.fillText('componente fundamental de la tensión',10,476);\n");
printf("ctx.fillStyle = '#70A55A';\n");
printf("ctx.fillText('componente fundamental de la corriente',10,488);\n");
//Líneas verticales y horizontales del gráfico (canvas)
printf("ctx.beginPath();\n"); //comienzo del camino
printf("ctx.lineWidth = 0.4;\n"); //línea de espesor 0.4 pixeles
//Eje horizontal superior
printf("ctx.translate(0,0.5);\n"); //elimina el efecto difuminado de la línea
printf("ctx.moveTo(0,0);\n");
printf("ctx.lineTo(800,0);\n");
//Eje horizontal central
printf("ctx.moveTo(0,250);\n");
printf("ctx.lineTo(800,250);\n");
//Eje horizontal inferior
printf("ctx.moveTo(0,499);\n"); //Si lo ponemos en 500 no se ve
printf("ctx.lineTo(800,499);\n");
//Eje vertical izquierdo
printf("ctx.translate(0.5,0);\n"); //elimina el efecto difuminado de la línea
printf("ctx.moveTo(0,0);\n");
printf("ctx.lineTo(0,500);\n");
//Eje vertical central
printf("ctx.moveTo(400,0);\n");
printf("ctx.lineTo(400,500);\n");
//Eje vertical derecho
printf("ctx.moveTo(799,0);\n"); //Si lo ponemos en 800 no se ve
printf("ctx.lineTo(799,500);\n");
printf("ctx.strokeStyle='brown';\n");
printf("ctx.stroke();\n");

//Letra cursiva de tamaño 12px y tipo itálica
printf("ctx.font = 'oblique 12px italic';\n");

//Dibujamos la onda de tensión
float x;
float y;
printf("ctx.lineWidth = 0.8;\n"); //línea de grosor 0.8 pixeles
printf("ctx.beginPath();\n");
x=0;
y=-(250-10)/VmaxGrafica*tension[0]+250;
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;          //0<=x<=800
  y=-(250-10)/VmaxGrafica*tension[i]+250; //10<=y<=490
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
}
printf("ctx.strokeStyle='blue';\n");
printf("ctx.stroke();\n");
printf("ctx.fillStyle = 'blue';\n");
printf("ctx.fillText('%.1fV',%.1f,%.1f);\n",Vmax,nVmax*800.0/numeroMuestras-10,-(250-10)/VmaxGrafica*Vmax+248);

//Dibujamos la gráfica de la corriente:
printf("ctx.beginPath();\n");
x=0;
y=-(250-10)/ImaxGrafica*corriente[0]+250;
printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=-(250-20)/ImaxGrafica*corriente[i]+250; //20<=y<=480
```

```
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
 }
 printf("ctx.strokeStyle='green';\n");
 printf("ctx.stroke();\n");
 printf("ctx.fillStyle = 'green';\n"); //El siguiente texto irá en verde
 printf("ctx.fillText('%.3fA',%.1f,%.1f);\n",Imax,nImax*800.0/numeroMuestras-10,-(250-20)/ImaxGrafica*Imax+248);

 //Dibujamos la gráfica de la potencia:
 printf("ctx.beginPath();\n");
 x=0;
 y=-(250-10)/PmaxGrafica*potencia[0]+250;
 printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
 for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=-(250-30)/PmaxGrafica*potencia[i]+250; //30<=y<=470
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
 }
 printf("ctx.strokeStyle='red';\n");
 printf("ctx.stroke();\n");
 printf("ctx.fillStyle = 'red';\n"); //El siguiente texto irá en rojo
 printf("ctx.fillText('%.1fW',%.1f,%.1f);\n",Pmax,nPmax*800.0/numeroMuestras-10,-(250-30)/PmaxGrafica*Pmax+248);

 //Dibujamos la gráfica de la componente fundamental de la tension:
 printf("ctx.lineWidth = 0.4;\n"); //línea de espesor 0.4 pixeles
 printf("ctx.beginPath();\n");
 x=0;
 y=sqrt(2)*arm1t*sin(fi_v1);
 y=-(250-20)/VmaxGrafica*y+250;
 printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
 for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=sqrt(2)*arm1t*sin(100*pi*0.02*i/(numeroMuestras-1)+fi_v1);
  y=-(250-10)/VmaxGrafica*y+250; //Para PmaxGrafica-->30, para -PmaxGrafica-->470
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
 }
 printf("ctx.strokeStyle='blue';\n");
 printf("ctx.stroke();\n");

 //Dibujamos la gráfica de la componente fundamental de la corriente:
 printf("ctx.beginPath();\n");
 x=0;
 y=sqrt(2)*arm1c*sin(fi_c1);
 y=-(250-20)/ImaxGrafica*y+250;
 printf("ctx.moveTo(%.1f,%.1f);\n",x,y);
 for (int i=1;i<numeroMuestras;i++){
  x=800.0/(numeroMuestras-1)*i;
  y=sqrt(2)*arm1c*sin(100*pi*0.02*i/(numeroMuestras-1)+fi_c1);
  y=-(250-20)/ImaxGrafica*y+250; //Para PmaxGrafica-->30, para -PmaxGrafica-->470
  printf("ctx.lineTo(%.1f,%.1f);\n",x,y);
 }
 printf("ctx.strokeStyle='green';\n");
 printf("ctx.stroke();\n");

 printf("</script>\n");
 printf("</canvas>\n");

 //Mostramos los valores armónicos:
 printf("<pre>\n");
 printf("Potencia media,    P=%8.1fW                    Onda de tensión   Onda de corriente\n",potenciaP);
 printf("Potencia reactiva,  Q=%6.1fVAr     Valor eficaz:          %10.1fV     %10.3fA\n",potenciaQ,Vef,Ief);
 printf("Potencia de distorsión,D=%7.1fVA     Valor medio:          %10.1fV     %10.1fmA\n",potenciaD,Vm,1000*Im);
 printf("Potencia aparente,  S=%7.1fVA     Distorsión arm. total, THD:  %10.1f&#37
%10.1f&#37\n",potenciaS,100*thdt,100*thdc);
 printf("Factor de potencia,  FP=%8.1f     Componente fundamental:    %10.1fV     %10.1fA\n",fp,arm1t,arm1c);
 printf("              cos&#966;=%6.1f     Componente arm. orden 2:   %10.1fV     %10.1fmA\n",cos_fi,arm2t,1000*arm2c);
 printf("              &#966;=%8.1f&deg     Componente arm. orden 3:   %10.1fV     %10.1fmA\n",180/pi*(fi_v1-
fi_c1),arm3t,1000*arm3c);
 printf("                          Componente arm. orden 4:   %10.1fV     %10.1fmA\n",arm4t,1000*arm4c);
 printf("                          Componente arm. orden 5:   %10.1fV     %10.1fmA\n",arm5t,1000*arm5c);
 printf("                          Componente arm. orden 6:   %10.1fV     %10.1fmA\n",arm6t,1000*arm6c);
 printf("                          Componente arm. orden 7:   %10.1fV     %10.1fmA\n",arm7t,1000*arm7c);
 printf("                          Componente arm. orden 8:   %10.1fV     %10.1fmA\n",arm8t,1000*arm8c);
 printf("\nConexión realizada desde la máquina %s\n",strchr(maquina,'=')+1);
```

```
printf("</pre>\n");

printf("<p>V eficaz:</p>\n");

printf("<div>\n");
printf("<img src='/TFG/imagenes/us.png'>\n");
printf("<pre>\n");
printf("Departamento de Electrónica\n");
printf("Ingeniería de las Tecnologías de Telecomunicación\n");
printf("Trabajo Fin de Grado\n");
printf("Araceli García Granados\n");
printf("</pre>\n");
printf("</div>\n");

printf("<script type='text/javascript'>\n");
printf("function encenderRele() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/rele';\n");
printf("var datosPost='data=uno';\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();"); //recargamos la página
printf("}\n");
printf("function apagarRele() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/rele';\n");
printf("var datosPost='data=cero';\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();");
printf("}\n");
printf("function calibrar() {\n");
printf("var xmlhttp=new XMLHttpRequest();\n");
printf("var peticion='/TFG/cgi-bin/calibrar';\n");
printf("var datosPost='%i&%i&%.2f&'+",muestraVmax,muestraVmin,muestraImedia);
printf("document.getElementById('cajaTexto').value;\n");
printf("xmlhttp.open('POST',peticion,false);\n");
printf("xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded');\n");
printf("xmlhttp.send(datosPost);\n");
printf("location.reload();");
printf("}\n");

printf("</script>\n");

int fd;
if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
  printf("No se puede abrir el dispositivo serie ttyACM0\n");
  return;
}
usleep(150);
serialPutchar(fd,'2'); //Enviamos este caracter al arduino;
while (serialDataAvail(fd)<0){};
char c=serialGetchar(fd);
if (c=='1') printf("<button id='botonRele' onclick='apagarRele();'><img src='/TFG/imagenes/off.jpg'></button>\n");
if (c=='0') printf("<button id='botonRele' onclick='encenderRele();'><img src='/TFG/imagenes/on.jpg'></button>\n");
printf("<button id='botonRecarga' onclick='location.reload();'><img src='/TFG/imagenes/recargar.jpg'></button>\n");
printf("<button id='botonCalibrar' onclick='calibrar();'><img src='/TFG/imagenes/calibrar.jpg'></button>\n");
printf("<input id='cajaTexto' type='text' value='223.0'>\n");
printf("</body>\n");
printf("</html>\n");
}
```

Código C++ TFG/cgi/rele.c de la Raspberry

```
//Librerías
#include <stdio.h>              //para poder usar la función printf()
#include <wiringSerial.h>       //para poder usar las funciones del puerto serie
#include <unistd.h>             //para poder usar la función usleep()
#include <stdlib.h>
#include <string.h>

#define bpsUART 115200

int main (void){
  char input[50];
  char *lenstr;
  long len;
  lenstr = getenv("CONTENT_LENGTH");
  if(lenstr != NULL && sscanf(lenstr,"%ld",&len)==1 && len <= 50){
    int fd;
    if((fd=serialOpen("/dev/ttyACM0",bpsUART))<0){
      printf("No se puede abrir el dispositivo serie ttyACM0\n");
      return 0;
    }
    usleep(50);

    fgets(input, len+1, stdin);
    if (strcmp(input,"data=uno")==0){
      serialPutchar(fd,'1');
    }
    else if (strcmp(input,"data=cero")==0){
      serialPutchar(fd,'0');
    }
    serialClose(fd);
  }
  return 0 ;
}
```

Código C++ TFG/cgi/calibrar.c de la Raspberry

```
//Librerías
#include <stdio.h>            //para poder usar la función printf
#include <string.h> //para poder usar strlen()
#include <math.h> //para poder usar la función pow()

float leerNumero(char *cadena,int *n);

int main (void){
 char cadena[50];
 char *lenstr;
 long len;
 FILE *fp=fopen("../../datos.txt","w"); //Ojo, introducir en Linux el comando: chmod 777 datos.txt
 if (fp==NULL) {
  printf("No se puede abrir el fichero datos.txt desde calibrar\n");
 }

 lenstr = getenv("CONTENT_LENGTH");
 if(lenstr != NULL && sscanf(lenstr,"%ld",&len)==1 && len <= 50){
  fgets(cadena, len+1, stdin); //recogemos lo recibido por Post
  int i=0;
  float numero=leerNumero(cadena,&i);
  fprintf(fp,"%.0f\n",numero);
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.0f\n",numero);
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.1f\n",numero);
  numero=leerNumero(cadena,&i);
  fprintf(fp,"%.1f\n",numero);
 }
 fclose(fp);
 return 0 ;
}

//lee el número que hay en "cadena" desde la posición "n"
// Por ejemplo, cadena es "959&82&509.8&226.0"
float leerNumero(char *cadena,int *n){
 int i=*n;
 int j=0;
 float numero=0;
 while (cadena[i]>='0' && cadena[i]<='9'){
  numero=numero+(cadena[i]-'0')/pow(10,i-(*n)+1);
  i++;
  j++;
 }
 if (cadena[i]=='.'){
  i++;
  while (cadena[i]>='0' && cadena[i]<='9'){
   numero=numero+(cadena[i]-'0')/pow(10,i-(*n));
   i++;
  }
 }
 i++;
 *n=i;
 numero=numero*pow(10,j);
 return numero;
}
```

Código C++ TFG/cgi/registro.c de la Raspberry

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(){
 printf("Content-Type:text/html\n\n");
 printf("<!DOCTYPE html>\n");
 printf("<html>\n");
 printf("<head>\n");
 printf("<meta charset='utf-8'>\n");
 printf("<title>Conexiones</title>\n");
 printf("<style type='text/css'>\n");
 printf("p{font-size:16px;color:brown;font-style:italic}\n");
 printf("</style>\n");
 printf("</head>\n");
 printf("<body style='background-color:#FFF0C9;'>\n");
 FILE *fp=fopen("../../registro.txt","r"); //abrimos /TFG/datos.txt como lectura
 if (fp==NULL) {
  printf("No se puede abrir el fichero registro.txt desde tfg\n");
 }
 printf("<p>");
 char caracter=getc(fp);
 while (feof(fp)==0) {
  if (caracter=='\n') printf("</br>");
  else printf("%c",caracter);
  caracter=getc(fp);
 }
 printf("</p>");
 printf("</body>\n");
 printf("</html>\n");
 return 0;
}
```

Código C++ TFG/cgi/resetArduino.c de la Raspberry

```
#include <stdio.h> //para poder usar la función printf()
#include <stdlib.h>  //para poder usar la función system()
#include <wiringPi.h> //para poder usar la función digitalWrite()

int main (void){
  printf("Content-Type:text/html\n\n");
  printf("<!DOCTYPE html>\n");
  printf("<html>\n");
  printf("<head>\n");
  printf("<title>Reset Arduino</title>\n");
  printf("</head>\n");
  printf("<body>\n");
  system("echo out > /sys/class/gpio/gpio17/direction"); //configuramos el puerto GPI0 como de salida
  wiringPiSetupGpio(); //este comando lo puede ejecutar el usuario tomcat8 con nomenclatura BCM para los puertos GPIO
  digitalWrite(17, LOW); //pin GPIO BCM 17 ó pin GPIO wiringPi 0
  delay(1000);   //1s
  digitalWrite(17, HIGH); //pin GPIO BCM 17 ó pin GPIO wiringPi 0
  printf("Resetada la placa Arduino...\n");
  printf("</body>\n");
  printf("</html>\n");
  return 0 ;
}
```

Script de Matlab

```
T=0.02;             %Tiempo de la señal
muestras = 767;     %767 muestras en 20ms
fs=muestras/T;      %frecuencia de muestreo=38350Hz
Ts=1/fs;            %tiempo entre muestras=20ms
t=(0:Ts:T-Ts);

tension = zeros(1,muestras);

%Valores de tensión capturado por el arduino y transformados a voltios
tension( 1)=194.4;
tension( 2)=195.2;
tension( 3)=196.8;
tension( 4)=200.0;
tension( 5)=201.6;
tension( 6)=203.2;
tension( 7)=206.4;
tension( 8)=207.2;
tension( 9)=210.4;
tension( 10)=212.0;
tension( 11)=213.6;
tension( 12)=216.0;
tension( 13)=217.6;
tension( 14)=220.0;
tension( 15)=221.6;
tension( 16)=223.2;
tension( 17)=224.8;
tension( 18)=227.2;
tension( 19)=228.8;
tension( 20)=231.2;
tension( 21)=232.8;
tension( 22)=233.6;
tension( 23)=236.0;
tension( 24)=237.6;
tension( 25)=239.2;
tension( 26)=241.6;
tension( 27)=242.4;
tension( 28)=244.8;
tension( 29)=245.6;
tension( 30)=247.2;
tension( 31)=249.7;
tension( 32)=251.3;
tension( 33)=252.1;
tension( 34)=253.7;
tension( 35)=255.3;
tension( 36)=256.1;
tension( 37)=257.7;
tension( 38)=258.5;
tension( 39)=260.1;
tension( 40)=260.9;
tension( 41)=262.5;
tension( 42)=264.1;
tension( 43)=264.9;
tension( 44)=266.5;
tension( 45)=267.3;
tension( 46)=268.1;
tension( 47)=271.3;
tension( 48)=271.3;
tension( 49)=272.1;
tension( 50)=273.7;
tension( 51)=275.3;
tension( 52)=276.1;
tension( 53)=275.3;
tension( 54)=276.9;
tension( 55)=279.3;
tension( 56)=279.3;
tension( 57)=280.9;
tension( 58)=281.7;
tension( 59)=282.5;
tension( 60)=283.3;
```

```
tension( 61)=284.9;
tension( 62)=284.9;
tension( 63)=286.5;
tension( 64)=287.3;
tension( 65)=288.1;
tension( 66)=288.9;
tension( 67)=289.7;
tension( 68)=291.3;
tension( 69)=291.3;
tension( 70)=292.1;
tension( 71)=293.7;
tension( 72)=294.5;
tension( 73)=295.3;
tension( 74)=296.9;
tension( 75)=298.5;
tension( 76)=297.7;
tension( 77)=299.3;
tension( 78)=300.1;
tension( 79)=300.1;
tension( 80)=300.9;
tension( 81)=301.7;
tension( 82)=300.9;
tension( 83)=301.7;
tension( 84)=302.5;
tension( 85)=302.5;
tension( 86)=303.3;
tension( 87)=304.1;
tension( 88)=304.1;
tension( 89)=304.1;
tension( 90)=305.0;
tension( 91)=305.0;
tension( 92)=305.0;
tension( 93)=305.8;
tension( 94)=305.0;
tension( 95)=305.0;
tension( 96)=306.6;
tension( 97)=306.6;
tension( 98)=307.4;
tension( 99)=308.2;
tension(100)=307.4;
tension(101)=307.4;
tension(102)=307.4;
tension(103)=308.2;
tension(104)=308.2;
tension(105)=308.2;
tension(106)=309.0;
tension(107)=309.0;
tension(108)=309.0;
tension(109)=309.0;
tension(110)=309.0;
tension(111)=309.0;
tension(112)=309.0;
tension(113)=309.8;
tension(114)=309.8;
tension(115)=309.8;
tension(116)=309.8;
tension(117)=309.8;
tension(118)=310.6;
tension(119)=310.6;
tension(120)=309.8;
tension(121)=310.6;
tension(122)=310.6;
tension(123)=310.6;
tension(124)=309.8;
tension(125)=310.6;
tension(126)=310.6;
tension(127)=310.6;
tension(128)=310.6;
tension(129)=310.6;
tension(130)=309.8;
tension(131)=310.6;
tension(132)=309.8;
tension(133)=309.8;
```

```
tension(134)=309.8;
tension(135)=309.8;
tension(136)=309.8;
tension(137)=309.0;
tension(138)=309.0;
tension(139)=309.0;
tension(140)=309.0;
tension(141)=308.2;
tension(142)=307.4;
tension(143)=307.4;
tension(144)=305.8;
tension(145)=305.0;
tension(146)=304.1;
tension(147)=303.3;
tension(148)=302.5;
tension(149)=300.9;
tension(150)=300.1;
tension(151)=300.1;
tension(152)=297.7;
tension(153)=295.3;
tension(154)=292.9;
tension(155)=292.9;
tension(156)=291.3;
tension(157)=289.7;
tension(158)=288.9;
tension(159)=287.3;
tension(160)=286.5;
tension(161)=285.7;
tension(162)=284.1;
tension(163)=284.1;
tension(164)=282.5;
tension(165)=282.5;
tension(166)=280.9;
tension(167)=279.3;
tension(168)=278.5;
tension(169)=276.9;
tension(170)=276.1;
tension(171)=274.5;
tension(172)=272.9;
tension(173)=271.3;
tension(174)=270.5;
tension(175)=268.1;
tension(176)=267.3;
tension(177)=265.7;
tension(178)=264.1;
tension(179)=263.3;
tension(180)=260.9;
tension(181)=260.1;
tension(182)=257.7;
tension(183)=256.9;
tension(184)=255.3;
tension(185)=252.9;
tension(186)=251.3;
tension(187)=249.7;
tension(188)=248.0;
tension(189)=246.4;
tension(190)=244.0;
tension(191)=243.2;
tension(192)=241.6;
tension(193)=239.2;
tension(194)=237.6;
tension(195)=236.0;
tension(196)=234.4;
tension(197)=232.8;
tension(198)=229.6;
tension(199)=229.6;
tension(200)=228.0;
tension(201)=226.4;
tension(202)=224.8;
tension(203)=223.2;
tension(204)=222.4;
tension(205)=220.0;
tension(206)=218.4;
```

```
tension(207)=216.8;
tension(208)=215.2;
tension(209)=213.6;
tension(210)=212.0;
tension(211)=210.4;
tension(212)=208.8;
tension(213)=206.4;
tension(214)=204.8;
tension(215)=202.4;
tension(216)=200.8;
tension(217)=198.4;
tension(218)=196.8;
tension(219)=195.2;
tension(220)=191.9;
tension(221)=189.5;
tension(222)=187.1;
tension(223)=184.7;
tension(224)=182.3;
tension(225)=181.5;
tension(226)=178.3;
tension(227)=175.9;
tension(228)=172.7;
tension(229)=171.1;
tension(230)=167.9;
tension(231)=165.5;
tension(232)=163.1;
tension(233)=159.9;
tension(234)=157.5;
tension(235)=155.1;
tension(236)=152.7;
tension(237)=150.3;
tension(238)=147.1;
tension(239)=145.5;
tension(240)=142.3;
tension(241)=139.9;
tension(242)=137.4;
tension(243)=135.8;
tension(244)=133.4;
tension(245)=130.2;
tension(246)=127.0;
tension(247)=123.8;
tension(248)=121.4;
tension(249)=118.2;
tension(250)=116.6;
tension(251)=114.2;
tension(252)=111.8;
tension(253)=109.4;
tension(254)=107.0;
tension(255)=104.6;
tension(256)=101.4;
tension(257)=99.0;
tension(258)=97.4;
tension(259)=94.2;
tension(260)=91.8;
tension(261)=89.4;
tension(262)=87.0;
tension(263)=85.4;
tension(264)=82.1;
tension(265)=79.7;
tension(266)=77.3;
tension(267)=75.7;
tension(268)=70.9;
tension(269)=69.3;
tension(270)=66.9;
tension(271)=65.3;
tension(272)=62.1;
tension(273)=59.7;
tension(274)=57.3;
tension(275)=55.7;
tension(276)=53.3;
tension(277)=50.9;
tension(278)=48.5;
tension(279)=46.1;
```

```
tension(280)=42.9;
tension(281)=41.3;
tension(282)=38.9;
tension(283)=35.7;
tension(284)=34.1;
tension(285)=31.7;
tension(286)=29.3;
tension(287)=28.5;
tension(288)=25.2;
tension(289)=22.8;
tension(290)=20.4;
tension(291)=18.8;
tension(292)=16.4;
tension(293)=14.0;
tension(294)=10.8;
tension(295)=9.2;
tension(296)=6.0;
tension(297)=3.6;
tension(298)=1.2;
tension(299)=-0.4;
tension(300)=-3.6;
tension(301)=-5.2;
tension(302)=-7.6;
tension(303)=-10.8;
tension(304)=-12.4;
tension(305)=-14.8;
tension(306)=-17.2;
tension(307)=-19.6;
tension(308)=-21.2;
tension(309)=-23.6;
tension(310)=-26.0;
tension(311)=-27.6;
tension(312)=-30.1;
tension(313)=-31.7;
tension(314)=-33.3;
tension(315)=-35.7;
tension(316)=-38.9;
tension(317)=-40.5;
tension(318)=-42.9;
tension(319)=-44.5;
tension(320)=-46.9;
tension(321)=-49.3;
tension(322)=-51.7;
tension(323)=-54.1;
tension(324)=-55.7;
tension(325)=-58.1;
tension(326)=-60.5;
tension(327)=-62.9;
tension(328)=-64.5;
tension(329)=-66.1;
tension(330)=-69.3;
tension(331)=-70.9;
tension(332)=-73.3;
tension(333)=-75.7;
tension(334)=-78.1;
tension(335)=-80.5;
tension(336)=-82.9;
tension(337)=-84.6;
tension(338)=-87.0;
tension(339)=-90.2;
tension(340)=-91.8;
tension(341)=-94.2;
tension(342)=-97.4;
tension(343)=-99.0;
tension(344)=-101.4;
tension(345)=-103.0;
tension(346)=-105.4;
tension(347)=-108.6;
tension(348)=-111.0;
tension(349)=-113.4;
tension(350)=-115.8;
tension(351)=-118.2;
tension(352)=-120.6;
```

```
tension(353)=-123.0;
tension(354)=-124.6;
tension(355)=-127.0;
tension(356)=-129.4;
tension(357)=-131.8;
tension(358)=-134.2;
tension(359)=-135.8;
tension(360)=-139.1;
tension(361)=-140.7;
tension(362)=-143.1;
tension(363)=-145.5;
tension(364)=-148.7;
tension(365)=-150.3;
tension(366)=-151.9;
tension(367)=-153.5;
tension(368)=-155.9;
tension(369)=-158.3;
tension(370)=-160.7;
tension(371)=-163.1;
tension(372)=-165.5;
tension(373)=-167.1;
tension(374)=-170.3;
tension(375)=-171.9;
tension(376)=-174.3;
tension(377)=-175.9;
tension(378)=-178.3;
tension(379)=-179.9;
tension(380)=-182.3;
tension(381)=-184.7;
tension(382)=-187.1;
tension(383)=-188.7;
tension(384)=-191.1;
tension(385)=-192.7;
tension(386)=-195.2;
tension(387)=-196.8;
tension(388)=-199.2;
tension(389)=-201.6;
tension(390)=-202.4;
tension(391)=-205.6;
tension(392)=-207.2;
tension(393)=-209.6;
tension(394)=-212.0;
tension(395)=-213.6;
tension(396)=-215.2;
tension(397)=-216.8;
tension(398)=-219.2;
tension(399)=-220.8;
tension(400)=-222.4;
tension(401)=-224.8;
tension(402)=-226.4;
tension(403)=-228.0;
tension(404)=-230.4;
tension(405)=-232.0;
tension(406)=-233.6;
tension(407)=-235.2;
tension(408)=-237.6;
tension(409)=-238.4;
tension(410)=-240.8;
tension(411)=-243.2;
tension(412)=-243.2;
tension(413)=-244.8;
tension(414)=-247.2;
tension(415)=-248.0;
tension(416)=-250.5;
tension(417)=-252.1;
tension(418)=-254.5;
tension(419)=-254.5;
tension(420)=-256.1;
tension(421)=-256.9;
tension(422)=-257.7;
tension(423)=-258.5;
tension(424)=-260.1;
tension(425)=-261.7;
```

```
tension(426)=-262.5;
tension(427)=-264.1;
tension(428)=-265.7;
tension(429)=-266.5;
tension(430)=-268.1;
tension(431)=-268.9;
tension(432)=-270.5;
tension(433)=-271.3;
tension(434)=-272.9;
tension(435)=-273.7;
tension(436)=-274.5;
tension(437)=-276.1;
tension(438)=-276.9;
tension(439)=-279.3;
tension(440)=-279.3;
tension(441)=-280.1;
tension(442)=-280.1;
tension(443)=-283.3;
tension(444)=-283.3;
tension(445)=-283.3;
tension(446)=-284.1;
tension(447)=-284.9;
tension(448)=-285.7;
tension(449)=-287.3;
tension(450)=-287.3;
tension(451)=-288.9;
tension(452)=-289.7;
tension(453)=-290.5;
tension(454)=-291.3;
tension(455)=-292.1;
tension(456)=-293.7;
tension(457)=-293.7;
tension(458)=-295.3;
tension(459)=-296.1;
tension(460)=-296.1;
tension(461)=-297.7;
tension(462)=-297.7;
tension(463)=-299.3;
tension(464)=-299.3;
tension(465)=-300.9;
tension(466)=-300.1;
tension(467)=-302.5;
tension(468)=-301.7;
tension(469)=-300.9;
tension(470)=-301.7;
tension(471)=-302.5;
tension(472)=-302.5;
tension(473)=-302.5;
tension(474)=-303.3;
tension(475)=-303.3;
tension(476)=-303.3;
tension(477)=-304.1;
tension(478)=-304.1;
tension(479)=-304.1;
tension(480)=-305.0;
tension(481)=-305.0;
tension(482)=-305.0;
tension(483)=-305.0;
tension(484)=-305.8;
tension(485)=-305.8;
tension(486)=-305.8;
tension(487)=-306.6;
tension(488)=-307.4;
tension(489)=-306.6;
tension(490)=-307.4;
tension(491)=-307.4;
tension(492)=-307.4;
tension(493)=-307.4;
tension(494)=-308.2;
tension(495)=-307.4;
tension(496)=-307.4;
tension(497)=-308.2;
tension(498)=-308.2;
```

```
tension(499)=-308.2;
tension(500)=-308.2;
tension(501)=-308.2;
tension(502)=-308.2;
tension(503)=-309.0;
tension(504)=-309.0;
tension(505)=-309.0;
tension(506)=-309.8;
tension(507)=-308.2;
tension(508)=-309.0;
tension(509)=-308.2;
tension(510)=-309.0;
tension(511)=-308.2;
tension(512)=-309.0;
tension(513)=-309.8;
tension(514)=-308.2;
tension(515)=-309.0;
tension(516)=-308.2;
tension(517)=-308.2;
tension(518)=-308.2;
tension(519)=-308.2;
tension(520)=-307.4;
tension(521)=-307.4;
tension(522)=-307.4;
tension(523)=-306.6;
tension(524)=-306.6;
tension(525)=-305.8;
tension(526)=-305.8;
tension(527)=-305.0;
tension(528)=-304.1;
tension(529)=-303.3;
tension(530)=-302.5;
tension(531)=-300.9;
tension(532)=-300.1;
tension(533)=-300.1;
tension(534)=-297.7;
tension(535)=-296.1;
tension(536)=-294.5;
tension(537)=-293.7;
tension(538)=-291.3;
tension(539)=-291.3;
tension(540)=-289.7;
tension(541)=-288.9;
tension(542)=-287.3;
tension(543)=-285.7;
tension(544)=-284.1;
tension(545)=-283.3;
tension(546)=-281.7;
tension(547)=-281.7;
tension(548)=-280.1;
tension(549)=-278.5;
tension(550)=-277.7;
tension(551)=-276.9;
tension(552)=-275.3;
tension(553)=-273.7;
tension(554)=-272.9;
tension(555)=-271.3;
tension(556)=-270.5;
tension(557)=-270.5;
tension(558)=-268.9;
tension(559)=-266.5;
tension(560)=-264.9;
tension(561)=-264.1;
tension(562)=-262.5;
tension(563)=-260.9;
tension(564)=-258.5;
tension(565)=-257.7;
tension(566)=-255.3;
tension(567)=-254.5;
tension(568)=-252.9;
tension(569)=-251.3;
tension(570)=-248.8;
tension(571)=-247.2;
```

```
tension(572)=-245.6;
tension(573)=-244.0;
tension(574)=-242.4;
tension(575)=-240.0;
tension(576)=-238.4;
tension(577)=-236.8;
tension(578)=-235.2;
tension(579)=-233.6;
tension(580)=-232.0;
tension(581)=-231.2;
tension(582)=-228.8;
tension(583)=-228.0;
tension(584)=-226.4;
tension(585)=-224.0;
tension(586)=-222.4;
tension(587)=-221.6;
tension(588)=-219.2;
tension(589)=-217.6;
tension(590)=-216.0;
tension(591)=-214.4;
tension(592)=-212.8;
tension(593)=-211.2;
tension(594)=-209.6;
tension(595)=-208.0;
tension(596)=-205.6;
tension(597)=-204.0;
tension(598)=-202.4;
tension(599)=-200.0;
tension(600)=-198.4;
tension(601)=-196.0;
tension(602)=-193.5;
tension(603)=-191.9;
tension(604)=-189.5;
tension(605)=-187.9;
tension(606)=-185.5;
tension(607)=-183.1;
tension(608)=-179.9;
tension(609)=-177.5;
tension(610)=-175.1;
tension(611)=-173.5;
tension(612)=-170.3;
tension(613)=-167.9;
tension(614)=-164.7;
tension(615)=-162.3;
tension(616)=-159.9;
tension(617)=-157.5;
tension(618)=-155.1;
tension(619)=-152.7;
tension(620)=-149.5;
tension(621)=-147.1;
tension(622)=-145.5;
tension(623)=-142.3;
tension(624)=-139.9;
tension(625)=-136.6;
tension(626)=-134.2;
tension(627)=-131.8;
tension(628)=-129.4;
tension(629)=-127.0;
tension(630)=-124.6;
tension(631)=-122.2;
tension(632)=-119.8;
tension(633)=-116.6;
tension(634)=-115.0;
tension(635)=-111.0;
tension(636)=-109.4;
tension(637)=-106.2;
tension(638)=-103.8;
tension(639)=-101.4;
tension(640)=-99.8;
tension(641)=-96.6;
tension(642)=-94.2;
tension(643)=-91.0;
tension(644)=-89.4;
```

```
tension(645)=-86.2;
tension(646)=-83.8;
tension(647)=-81.3;
tension(648)=-78.9;
tension(649)=-76.5;
tension(650)=-74.1;
tension(651)=-71.7;
tension(652)=-69.3;
tension(653)=-68.5;
tension(654)=-63.7;
tension(655)=-62.1;
tension(656)=-58.9;
tension(657)=-57.3;
tension(658)=-54.9;
tension(659)=-52.5;
tension(660)=-50.1;
tension(661)=-48.5;
tension(662)=-46.1;
tension(663)=-43.7;
tension(664)=-40.5;
tension(665)=-38.1;
tension(666)=-36.5;
tension(667)=-34.1;
tension(668)=-30.9;
tension(669)=-29.3;
tension(670)=-26.8;
tension(671)=-24.4;
tension(672)=-22.0;
tension(673)=-20.4;
tension(674)=-17.2;
tension(675)=-14.8;
tension(676)=-12.4;
tension(677)=-9.2;
tension(678)=-8.4;
tension(679)=-5.2;
tension(680)=-3.6;
tension(681)=-0.4;
tension(682)=2.0;
tension(683)=4.4;
tension(684)=6.8;
tension(685)=8.4;
tension(686)=10.8;
tension(687)=13.2;
tension(688)=14.8;
tension(689)=17.2;
tension(690)=19.6;
tension(691)=21.2;
tension(692)=23.6;
tension(693)=26.0;
tension(694)=28.5;
tension(695)=30.1;
tension(696)=32.5;
tension(697)=34.9;
tension(698)=37.3;
tension(699)=38.9;
tension(700)=41.3;
tension(701)=42.9;
tension(702)=46.1;
tension(703)=48.5;
tension(704)=49.3;
tension(705)=52.5;
tension(706)=54.1;
tension(707)=55.7;
tension(708)=57.3;
tension(709)=61.3;
tension(710)=62.9;
tension(711)=65.3;
tension(712)=67.7;
tension(713)=69.3;
tension(714)=71.7;
tension(715)=74.1;
tension(716)=76.5;
tension(717)=78.9;
```

```
tension(718)=81.3;
tension(719)=83.8;
tension(720)=86.2;
tension(721)=87.8;
tension(722)=90.2;
tension(723)=92.6;
tension(724)=95.0;
tension(725)=97.4;
tension(726)=99.8;
tension(727)=101.4;
tension(728)=105.4;
tension(729)=106.2;
tension(730)=108.6;
tension(731)=111.8;
tension(732)=115.0;
tension(733)=115.8;
tension(734)=119.0;
tension(735)=121.4;
tension(736)=123.8;
tension(737)=125.4;
tension(738)=127.8;
tension(739)=129.4;
tension(740)=131.8;
tension(741)=134.2;
tension(742)=136.6;
tension(743)=139.1;
tension(744)=141.5;
tension(745)=143.9;
tension(746)=146.3;
tension(747)=147.9;
tension(748)=150.3;
tension(749)=153.5;
tension(750)=155.1;
tension(751)=157.5;
tension(752)=159.1;
tension(753)=161.5;
tension(754)=163.9;
tension(755)=165.5;
tension(756)=166.3;
tension(757)=170.3;
tension(758)=172.7;
tension(759)=174.3;
tension(760)=175.9;
tension(761)=178.3;
tension(762)=180.7;
tension(763)=183.1;
tension(764)=184.7;
tension(765)=186.3;
tension(766)=188.7;
tension(767)=191.1;

Y=abs(fft(tension));
Y = Y(1:muestras/2);
Y = 2*Y/muestras;
f = (0:(muestras/2)-1)*fs/muestras;

Vef = rms(tension);
disp(['Ief: ',num2str(Vef),'V']);
disp(['Valor eficaz de la componente fundamental: ',num2str(Y(2)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 2: ',num2str(Y(3)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 3: ',num2str(Y(4)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 4: ',num2str(Y(5)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 5: ',num2str(Y(6)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 6: ',num2str(Y(7)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 7: ',num2str(Y(8)/sqrt(2)),'A']);
disp(['Valor eficaz del armónico 8: ',num2str(Y(9)/sqrt(2)),'A']);
THD=sqrt((Y(3)*Y(3)+Y(4)*Y(4)+Y(5)*Y(5)+Y(6)*Y(6)+Y(7)*Y(7)+Y(8)*Y(8))/(Y(2)*Y(2)));
disp(['Distorsión armónica total: ',num2str(THD)]);
title('Armónicos');
xlabel('Frecuencia (Hz)');
ylabel('Tension');
figure (1);
plot(f,Y);
```

```
figure (2);
plot(t,tension);
```