ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"UTILIZACIÓN DE LA MINICOMPUTADORA RASPBERRY PI PARA LA ADQUISICIÓN Y EVALUACIÓN DE DATOS DE CONSUMO DE ENERGÍA ELÉCTRICA DE EQUIPOS A 220 VOLTIOS"

TESINA DE SEMINARIO

Previa a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Jairo Andrés Rayo Cedillo

José Alfonso Chimbo Cusme

GUAYAQUIL – ECUADOR

AÑO 2013

AGRADECIMIENTO

A Dios por sobre todas las cosas, por haberme dado la vida y la oportunidad de presentarme en ella, a todas las personas que conozco, desde mis padres, familiares, amigos e instructores que con los consejos y guía de todos ellos pude llevar a cabo con éxito y culminar este trabajo de investigación. Un especial agradecimiento a nuestro tutor de tesis el Ing. Carlos Valdivieso por ser nuestra guía y por los conocimientos adquiridos durante este tiempo.

Jairo Rayo

Las palabras no pueden expresar mi agradecimiento a Dios, por darme la oportunidad de seguir en este mundo tras haber superado diversas dificultades, a mis padres por ser soporte moral en el trascurso de mi vida, a la ESPOL por acogerme dentro de sus paredes del conocimiento y finalmente al Ing. Carlos Valdivieso por estar velando por el desarrollo de nuestro proyecto durante la duración de este seminario.

José Chimbo

DEDICATORIA

A mi hijo, por ser la fortaleza que necesitaba para no desmayar y seguir adelante hasta culminar mi carrera universitaria. A mis padres, por haber sido mi sostenimiento y mis consejeros, pilares fundamentales de mi personalidad y mis valores. A la ESPOL por haberme albergado cálidamente durante mi carrera universitaria. A Dios, patria y a la vida.

Jairo Rayo

A Dios Todopoderoso, por haberme otorgado la inteligencia necesaria para cumplir mis objetivos propuestos. A mis padres, hermana y amigos, por estar a mi lado de una u otra forma apoyándome en el inicio y final de esta etapa de mi vida.

José Chimbo

TRIBUNAL DE SUSTENTACIÓN

M. Sc. Carlos Valdivieso

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Ing. Hugo Villavicencio

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Jairo Andrés Rayo Cedillo

_____

José Alfonso Chimbo Cusme

RESUMEN

El siguiente trabajo de estudio académico detalla varios puntos importantes de nuestro proyecto como son investigación, análisis, implementación, adquisición y procesamiento de datos, además de todos los conocimientos adquiridos en nuestra carrera como estudiantes politécnicos, enfocándonos en el campo de los microcontroladores. Puntualizaremos el estudio y correcto funcionamiento de la tarjeta Raspberry Pi, denominada como la computadora de bolsillo debido a su tamaño, semejante a una tarjeta de identificación personal, además del uso del microcontrolador Arduino 1.0 para la adquisición de datos por medio de las entradas analógicas que posee el mismo y a su vez la trasmisión serial vía USB con la Raspberry Pi y la presentación de los datos en un web server casero mediante gráficas en tiempo real.

El proyecto consiste en la utilización de sensores de corriente AC no invasivos, de máximo 30 A del modelo SCT-013-030 como módulos que nos permitan la recolección de datos de consumo eléctrico de cualquier equipo, para de esta manera poderlos procesar por medio del microcontrolador Arduino 1.0 y la minicomputadora Raspberry Pi, así poder realizar una comparación entre varios equipos eléctricos que funcionen a 220 voltios.

Es así que utilizaremos Linux como sistema operativo aprovechando los diferentes lenguajes de programación que este ofrece, para poder interpretar los valores recolectados por los sensores de corriente.

ÍNDICE GENERAL

ÍNDICE DE ABREVIATURAS

DSI (DISPLAY SERIAL INTERFACE)

DIY (DO IT YOURSELF)

IDE (INTEGRADED DEVELOPMENT ENVIRONMENT)

GPIO (GENERAL PURPOSE INPUT OUTPUT)

GPS (GLOBAL POSITIONING SYSTEM)

GPU (GRAPHICS PROCESSING UNIT)

HDMI (HIGH DEFINITION MULTIMEDIA INTERFACE)

HUB

I²C (INTER-INTEGRATED CIRCUIT)

LCD (LIQUID-CRYSTAL DISPLAY)

LED (LIGHT-EMITTING DIODE)

PWM (PULSE WIDTH MODULATION)

RAM (RANDOM ACCESS MEMORY)

RCA (RADIO CORPORATION OF AMERICA)

RX (RECEIVER)

SD (SECURE DIGITAL)

SPI (SERIAL PERIPHERAL INTERFACE)

TX (TRANSMITTER)

URL (UNIFORM RESOURCE LOCATOR)

USB (UNIVERSAL SERIAL BUS)

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Este proyecto se basa en el procesamiento del consumo de energía que se produce en un dispositivo eléctrico, durante un período de tiempo a través de los sensores de corriente AC SCT-013-030 y de la elaboración de un código de programación en Phyton para de esta forma poder interpretar por medio de la tarjeta Raspberry Pi dichos datos y poderlos visualizar en una interfaz gráfica amigable.

En el primer capítulo, se presenta una descripción general del proyecto, sus alcances, limitaciones y objetivos que se quieren lograr al finalizar el mismo. Se hace énfasis en la evolución que han tenido las computadoras, así como los sistemas operativos y los diferentes lenguajes de programación que permiten a cualquier persona trabajar y realizar cualquier modificación en ellos de una forma amigable y sencilla, solamente con un poco de instrucción y compromiso.

En el segundo capítulo, se específica detalladamente las herramientas de hardware y software, además se profundizan las características, funcionamiento y uso de los componentes principales de este proyecto como son la tarjeta Raspberry Pi, los sensores de corriente AC SCT-013-030, el

microcontrolador Arduino 1.0 y demás conceptos claves en base a los cuales se ha trabajado esta tesina.

El tercer capítulo, abarca el diseño y la implementación del proyecto, priorizando cada uno de los bloques de hardware y software que posee el mismo, para esto haremos uso de un diagrama de bloques general, un diagrama de flujo, para paso a paso ir explicando el funcionamiento detallado de cada uno de los componentes, además del código de programación que se trabajó en Python y en el Arduino IDE, que permite poder configurar la tarjeta Raspberry Pi en el sistema operativo Linux y también el microcontrolador Arduino 1.0.

En el cuarto y último capítulo se realizan las pruebas y simulaciones para el análisis de resultados en base a los parámetros establecidos, se realizan comparaciones para pruebas de laboratorio con una fase y con dos fases, pudiendo de esta manera dar conclusiones claras y precisas sobre los resultados obtenidos.

Finalmente redactaremos las conclusiones acerca de nuestro proyecto, además de las recomendaciones que se deben llevar a cabo para efecto del mismo, así como para cualquier posible falla presentada en implementación

o configuración de los dispositivos. También añadiremos información adicional sobre el desarrollo del proyecto.

CAPÍTULO 1

DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 Antecedentes

El gran avance tecnológico en estas últimas décadas ha repercutido directa o indirectamente en la vida de las personas, tanto así que actualmente nos quedamos sorprendidos con las cosas que pueden realizar pequeñas tarjetas o integrados. Es así como el mundo de la electrónica, la programación y las telecomunicaciones se actualizan a diario dejando obsoletos a otros dispositivos, con el fin de hacer más sencilla la vida de las personas.

El enfoque de este proyecto nos ayuda a comprender y discernir muchos aspectos tal vez desconocidos para muchas personas en cuanto al consumo de energía eléctrica de ciertos dispositivos electrónicos, y esto

debido al poco conocimiento que tienen las personas respecto al dispendio de la energía en artefactos electrónicos a 110 V o a 220 V.

Antiguamente la mayoría de equipos médicos, aires acondicionados, lavadoras, secadoras de ropa, etc. funcionaban a 110 V, lo cual representaba una mayor demanda de corriente y por lo mismo una mayor concentración de calor en el cable de alimentación lo que provocaba mayores pérdidas debido a la resistividad total del cobre. Esto se logró disminuir de cierta forma optimizando la cantidad de corriente que demandaba un equipo al colocar una fase más para poder de esta forma tener un voltaje de 220 V, que lo que hacía era disminuir la corriente que pasaba por el cobre en la mitad y por lo tanto se reducirían las pérdidas también y ayudaría en caso de que el calibre del cable de cobre no fuese el correcto, porque de esta manera se podrían evitar incendios.

Actualmente muchas empresas han incursionado en la optimización de la energía consiguiendo así ahorrar hasta en un 40% el consumo de la misma. Este avance no solamente ha ayudado a disminuir el pago de la factura eléctrica en los hogares, sino que también ayuda al planeta, logrando evitar el desperdicio de recursos ambientales y ecológicos en estos tiempos cruciales.

El mundo de los computadores también ha evolucionado enormemente en estos últimos diez años, mencionamos esto porque la tarjeta que nos va a facilitar la adquisición y procesamiento de los datos que obtengamos de los sensores de corriente SCT-013-030 es una poderosa minicomputadora denominada así por muchos programadores, gracias a que les permite realizar un sinnúmero de proyectos por su fácil uso.

La Raspberry Pi también ha sufrido cambios desde su aparición en el año 2006, existen dos modelos A y B. El modelo A solo cuenta con un conector USB y no posee puerto Ethernet, mientras que el modelo B posee dos puertos USB y un puerto Ethernet. Anteriormente la Raspberry Pi era de 256 Mb, actualmente hay hasta de 512 Mb, que pueden ser divididos en memoria RAM y memoria para video por medio del GPU.

1.2 Motivaciones para el proyecto

La principal motivación para llevar a cabo este proyecto de graduación fue adquirir conocimientos nuevos que enriquezcan los ya aprendidos durante nuestra vida académica dentro de la Universidad y a la vez fortalecer y utilizar los conocimientos obtenidos previamente en cada una de las materias teóricas y prácticas de la carrera de Ingeniería en Electrónica y Telecomunicaciones, haciendo un especial enfoque en el estudio de los

microcontroladores y en cada uno de los recursos que estos poseen y escoger el más apropiado para llevar a cabo nuestro proyecto.

Para nosotros como estudiantes nos da una gran satisfacción, la evolución que tiene la tecnología cada día que pasa, más aún haber conocido una tarjeta tan pequeña la cual puede ser utilizada como una computadora sencilla, en la cual se pueden instalar programas, reproducir música, acceder a internet y hasta visualizar videos en alta definición.

Hacer uso de la ética del DIY (Do it yourself), para de esta forma poder realizar nuestras propias convicciones, ahorrando tiempo y dinero y a la vez demostrándonos de la capacidad que poseemos, siempre y cuando se ponga entrega, compromiso y responsabilidad.

Otro de los grandes retos que nos propusimos fue el de aprender a programar para el sistema operativo Linux, utilizando un lenguaje de programación como Python, que nunca lo aprendimos durante la carrera pero que si lo ven otras carreras como Ingeniería en Computación, llama la atención por ser un software libre en el cual cualquier programador puede realizar cambios a las necesidades que requiera.

Nuestro primordial objetivo es poder culminar satisfactoriamente este proyecto de tesina, para de esta forma poder terminar con éxito cinco años de carrera universitaria, formándonos como personas de bien y preparadas para cumplir cualquier reto una vez que nos incorporemos al ámbito profesional, y de esta forma poder aportar así con un granito de arena a nuestras familias y al país entero.

1.3 Identificación del problema

La falta de tecnología en nuestro país motiva cada vez más a jóvenes y profesionales a la realización de proyectos utilizando la práctica del DIY, para de esta forma demostrar a la sociedad que con compromiso y responsabilidad se pueden lograr grandes cosas y a la vez ser competitivos como en otros países del mundo, es así que deseamos llevar a cabo en este proyecto de graduación el procesamiento del consumo de energía en equipos eléctricos y electrónicos que trabajen a 220 V, mediante la previa adquisición de niveles de corriente capturados por los sensores de corriente AC del modelo SCT-013-030, que poseen unos ganchos para poderlos asegurar al cable de cualquier equipo y empezar a tomar mediciones de potencia consumida, los cuales nos entregan valores de voltaje entre 0 y 1 V, luego digitalizarlos y acoplarlos al microcontrolador Arduino 1.0 para de esta manera poder ser referenciados al GPIO (general purpose input/ouput) de la tarjeta

Raspberry Pi. Una vez adquiridos estos datos, procesarlos mediante una aplicación para posteriormente mostrarlos mediante una interfaz gráfica para poder ser analizados. Al finalizar el análisis, realizar una tabla de consumo eléctrico de diferentes dispositivos electrónicos, compararlos y poder dar algún criterio respecto a los mismos.

1.4 Objetivos

El objetivo principal para la implementación de este proyecto de tesina es lograr el perfecto acoplamiento entre los valores de corriente obtenidos a través de los sensores de corriente AC del modelo SCT-013-030, poderlos digitalizar y referenciarlos para poder ser ingresados en el GPIO de la tarjeta Raspberry Pi.

Demostrar que el DIY es posible en cualquier parte del mundo, simplemente siendo exigente, responsable y comprometido con las metas que nos propongamos.

Aprender a utilizar herramientas de software para el manejo de la tarjeta Raspberry Pi, así como aprender nuevos lenguajes de programación de software libre, específicamente para Linux, ya que son las principales herramientas de este proyecto de grado.

Como meta tenemos entregar un análisis exhaustivo con tablas, simulaciones y gráficos que nos permitan poder resumir nuestro proyecto, a la vez dar acertadas conclusiones del mismo y en caso de cualquier inconveniente durante su desarrollo poder entregar las debidas recomendaciones del caso.

Realizar un aporte a la sociedad, a la ESPOL y a los demás estudiantes de la carrera de Ingeniería en Electrónica y Telecomunicaciones para que de esta forma puedan interesarse en otros proyectos en los que puedan demostrar sus conocimientos adquiridos en el área de los microcontroladores y a la vez enriquecerse de nuevos conocimientos que les contribuyan para realizarse profesionalmente.

1.5 Limitaciones

Para efecto del desarrollo de este proyecto la principal limitación que tendremos es trabajar la mayor parte del tiempo con equipos a 110 V, simulando la otra fase para efectos del análisis y esto debido a que en el laboratorio toda la infraestructura eléctrica que funciona a 220 V se encuentra adecuada internamente.

Otro punto esquivo en cuanto a nuestro trabajo de investigación es trabajar por primera vez con la tarjeta Raspberry Pi, aprender su correcto funcionamiento, características y demás componentes, así como prepararla para su configuración inicial y poderla utilizar por primera vez. Con la tarjeta tuvimos un poco de retraso debido al lenguaje de programación que íbamos a utilizar, ya que el sistema operativo es Linux y necesitábamos un lenguaje de programación el cual se adaptara a nuestras necesidades y que cumpliera con estos requisitos, en el cual solo poseíamos conocimientos básicos.

En cuanto a las limitaciones de la tarjeta encontramos su débil procesador de tan solo un núcleo y con apenas una tasa de procesamiento de 700 MHz, la memoria RAM que posee es de baja capacidad, de tan solo 256 Mb y no posee memoria interna de almacenamiento. Tuvimos suerte en que otras personas en el país ya se encuentran trabajando con la Raspberry Pi por lo que pudimos conseguirla a través de un ex estudiante de la ESPOL a un buen precio.

Respecto al lenguaje de programación Python, utilizado para nuestro proyecto de tesina, puede surgir algún inconveniente en el momento de utilizarlo para el procesamiento en tiempo real debido a que existe esta limitación al trabajar con la librería Rpi.GPIO.

Conseguir los sensores de corriente AC SCT-013-030 también retraso en parte nuestro trabajo, debido a que estos no se encontraban disponibles en nuestro mercado electrónico y tuvimos que utilizar los servicios de una persona profesional en importaciones de este tipo para que nos los consiguiera.

CAPÍTULO 2

FUNDAMENTO TEÓRICO

2.1 Introducción

En este capítulo realizamos un análisis sobre el funcionamiento y características de la microcomputadora Raspberry Pi, así como el hardware y software necesario para la realización de este proyecto. La microcomputadora, Raspberry Pi, es una computadora del tamaño de una tarjeta de crédito, la cual con las actualizaciones necesarias puede convertirse en un excelente ordenador portátil para el usuario. Debido al hardware de la tarjeta, es posible conectarla a televisores que existen en el mercado desde hace 20 años, gracias a su salida de video RCA, o a televisores actuales mediante su salida HDMI. El sistema operativo del Raspberry Pi es Linux, propiamente el Raspbian Wheezy el cual se instala en una SD cualquiera, de preferencia y por experiencia de los autores se recomienda utilizar una SD de 8GB de capacidad mínima,

debido a que la instalación del sistema operativo necesita de 2GB de espacio libre. Con respecto al hardware se requiere el microcontrolador Arduino 1.0 y el sensor de corriente SCT-013-030, capaz de medir el paso de corriente a través de equipos eléctricos, los cuales serán ingresados y procesados por la Raspberry Pi para su estudio y próximo uso. Finalmente en la parte de software (programación) es posible desarrollar el proyecto mediante el uso de 2 programas diferentes cuyas librerías son las más utilizadas por el Raspberry Pi, estas son la WiringPi y la Rpi.GPIO, librerías creadas para los programas Code Blocks y Python respectivamente.

2.2 Raspberry Pi

2.2.1 ¿Qué es?

El Raspberry Pi es una microcomputadora, como se muestra en la figura 2.1, que a diferencia de la mayoría de computadoras existentes trabaja con un sistema operativo propio de ella (Raspbian Wheezy). Posee entradas USB que generalmente son utilizadas para conectar el mouse y el teclado a la misma. Con respecto a la salida de video, el Raspberry Pi posee 2 alternativas para el usuario, la salida HDMI para televisores modernos o en su defecto la salida de video RCA para televisores analógicos. **[1]**

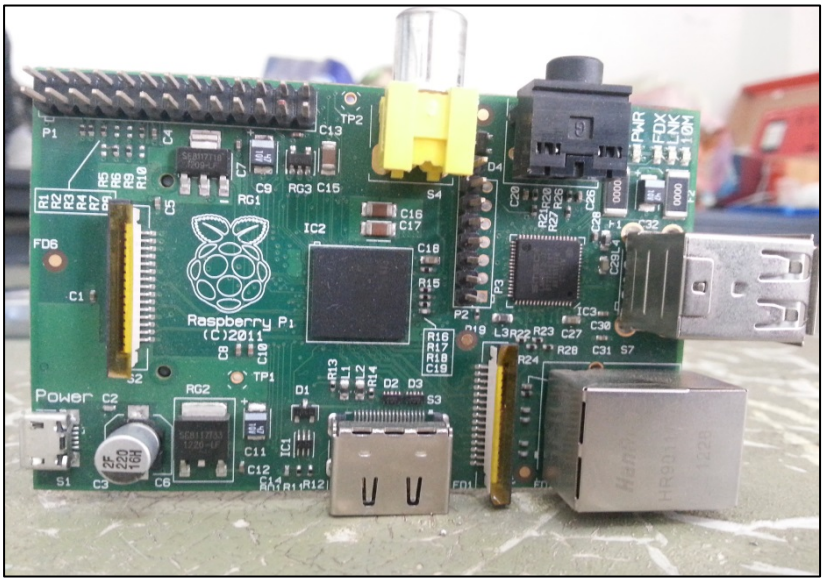

Figura 2.1. Raspberry Pi

2.2.2 Características físicas básicas del Raspberry Pi

El procesador BMC2835

Este procesador es considerado el corazón del Raspberry Pi debido que se encarga de los procesos del audio, video, procesamiento gráfico y comunicación de las interfaces de hardware (teclado y mouse). Todas estas tareas están repartidas en dos memorias especiales: la RAM y el GPU. La primera fue diseñada para el procesamiento de los datos, mientras que la segunda su misión es la del acelerador de video propio de la tarjeta. Un total de 256Mb* está repartida entre las dos memorias, permitiendo al usuario seleccionar cuanta memoria desea distribuir

entre ambos de acuerdo a sus necesidades, por ejemplo para la reproducción de videos en el Raspberry Pi el estándar utilizado es 192/64, es decir distribuir 192 Mb en el procesamiento de los datos y 64 Mb para el acelerador de videos. El procesador del Raspberry Pi se muestra en la figura 2.2. **[2]**

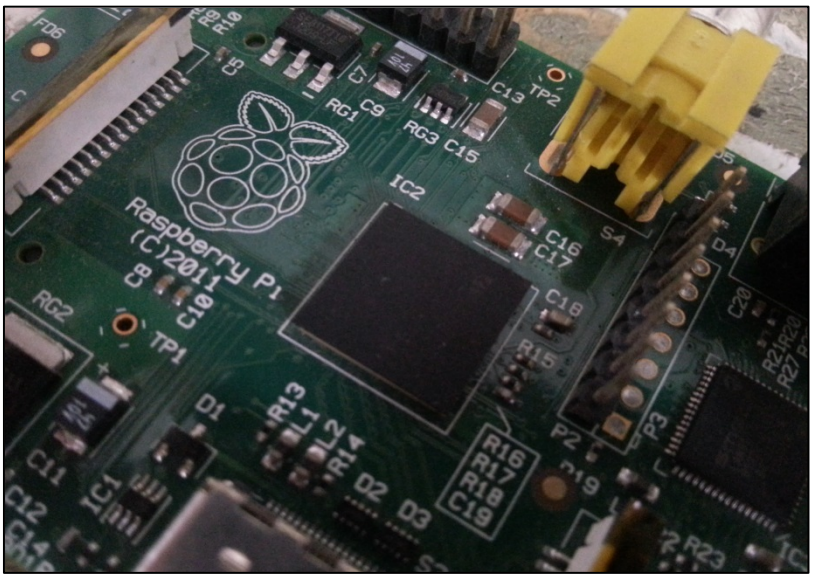

Figura 2.2. Procesador BMC2835 en la Raspberry Pi

Salidas de Video

El Raspberry Pi está diseñado para soportar tres estándares diferentes de salida de video: RCA, HDMI, DSI. Los dos primeros son de fácil acceso al usuario ya que se encuentran en los televisores analógicos y digitales de la era actual, mientras que

para el último estándar, es necesario un hardware adicional para su óptimo funcionamiento.

Para un televisor analógico es necesario utilizar la salida RCA, para un televisor digital o bien puede seguir usando esta salida o en su defecto usar la salida HDMI que se muestra en la figura 2.4. **[3]**

Figura 2.3. Salida de video RCA en la Raspberry Pi

Figura 2.4. Salida de video HDMI en la Raspberry Pi

Conexión de Mouse, Teclado e Internet

EL Raspberry Pi (modelo B) posee 2 puertos USB, como se muestra en la figura 2.5, para la conexión de dispositivos como el teclado y el mouse, cabe recalcar que es posible adicionar un hub para alimentar a más dispositivos con salida USB para así conectarlos a la tarjeta, pero no utilice estos puertos como recarga de equipos electrónicos (como celulares) ya que reducen el tiempo de vida del Raspberry Pi. **[4]**

Además el modelo actual del Raspberry Pi, posee un puerto Ethernet para conectarse al internet, como se aprecia en la figura

2.5, pero si es la primera vez que enciende este microcomputador necesitará configurar la tarjeta para poder acceder a la red. Esto se explica en los anexos de este texto.

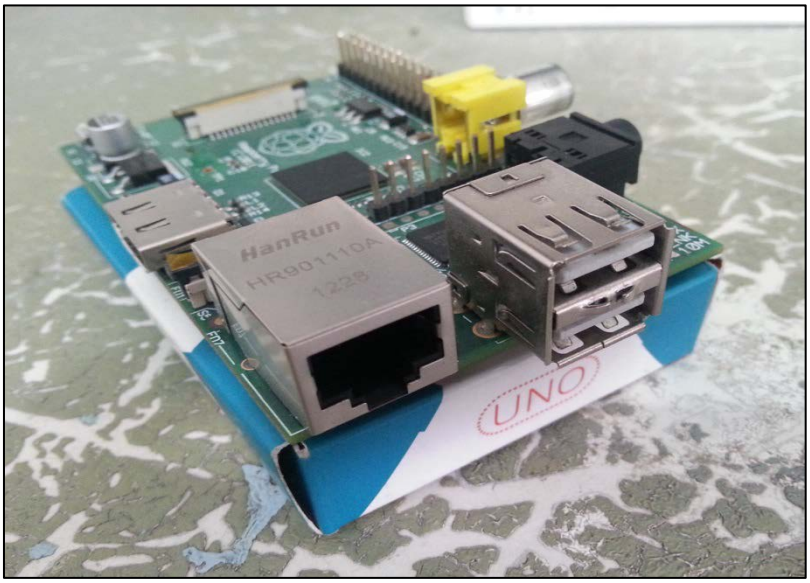

Figura 2.5. Conectores USB para el teclado, el mouse y puerto Ethernet para acceder a la red

Fuente de Alimentación

Esta es una de las principales ventajas de este microcomputador, su procesador requiere un bajo consumo de energía, por lo tanto basta con un cargador que tenga para conectar un cable USB y tenga por salida un micro USB, como se muestra en la figura 2.6. Es importante tener en mente que el voltaje y amperaje ideal de

alimentación sea de 5V y 1A, este consumo se justifica cuando se conectan varios elementos externos en la tarjeta a parte del teclado y mouse. **[5]**

Figura 2.6. Fuente de Alimentación de la Raspberry Pi

2.2.3 Sistema Operativo Raspbian Wheezy

El Raspbian Wheezy es considerado el sistema operativo estándar del Raspberry Pi, tenga en cuenta que no solo es el único disponible en la red, pero sí el más usado por los usuarios que no poseen mucha experiencia utilizando Linux.

Este sistema operativo ocupa poco espacio en la memoria de la SD (2GB) debido a que apenas vienen instalados un mínimo de aplicaciones para el usuario. En otras palabras si el usuario necesita de algún programa extra u otra aplicación deberá instalarla manualmente usando líneas de comando en el LX Terminal del escritorio del Raspberry Pi.

Para aprender más acerca de la correcta instalación del sistema operativo, recomiendo visitar la siguiente página web http://www.raspberrypi.org/downloads la cual le ofrece de manera gratuita todas las herramientas necesarias para una instalación exitosa del Raspbian Wheezy.

2.2.4 GPIO en la Raspberry Pi

El puerto de entrada/salida para propósitos generales (GPIO) son los pines que se aprecian en la figura 2.7, estos pines le permiten al Raspberry Pi comunicarse con otros componentes y circuitos electrónicos, además con una programación adecuada se dispone de la opción de censar temperatura, controlar servo motores y utilizando los protocolos adecuados como SPI e I2C comunicarse con otra computadora e incluso con otra Raspberry Pi. **[6]**

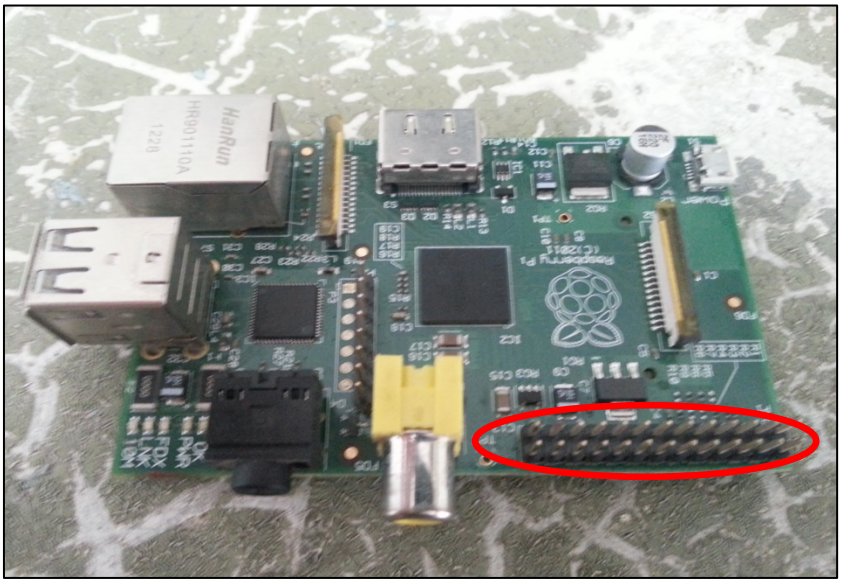

Figura 2.7. Localización del puerto GPIO en la Raspberry Pi

Definición de los pines en la Raspberry Pi

El propósito de cada uno de los pines de este puerto se muestra en la figura 2.8, no todos los pines se encuentran habilitados para uso del usuario, hay pines reservados, es decir que no deben ir conectados a ningún equipo externo o a la propia Raspberry Pi debido a que estos pines se encuentran conectados directamente con el procesador BMC2835 y cualquier conexión podría terminar dañando permanentemente la tarjeta. Otros de los pines tienen como función permitir el uso de los protocolos I2C, SPI, otros están habilitados para el uso del PWM, TX, RX, y un par de pines son de alimentación de 5 V y 3.3 V. **[7]**

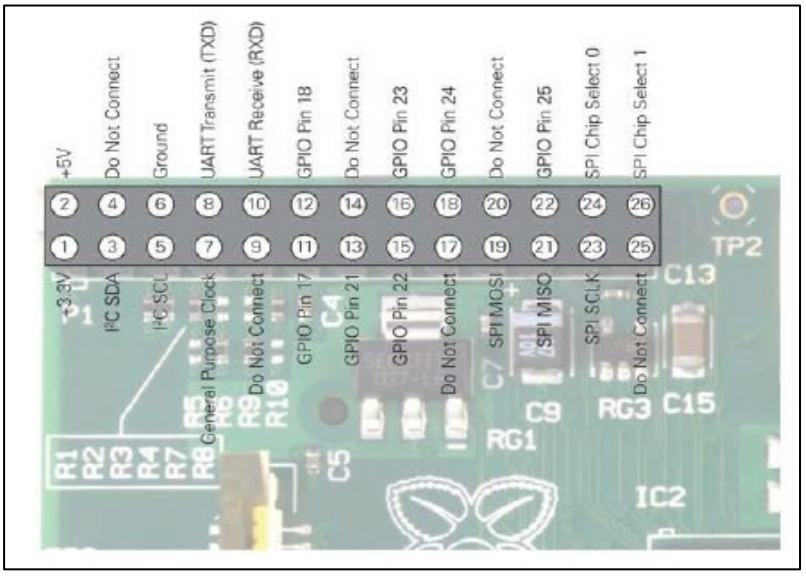

Figura 2.8. Designación de pines del puerto GPIO en la Raspberry Pi **[8]**

2.2.5 Software utilizado

Synaptic

Una de las desventajas a la hora de instalar un programa/aplicación en la Raspberry Pi, resulta en que el usuario se ve en la necesidad de utilizar líneas de comando en el LX Terminal para descargar los distintas aplicaciones en la red, pero existe un programa que se encarga de esta tarea con un entorno gráfico agradable al usuario y este es Synaptic. Esta aplicación es útil al usuario porque le presenta a este los distintos programas existentes para la Raspberry Pi clasificados por categorías como

juegos, reproductor de video, electrónica, servidores web, programación etc. El entorno gráfico de Synaptic se muestra en la figura 2.9.

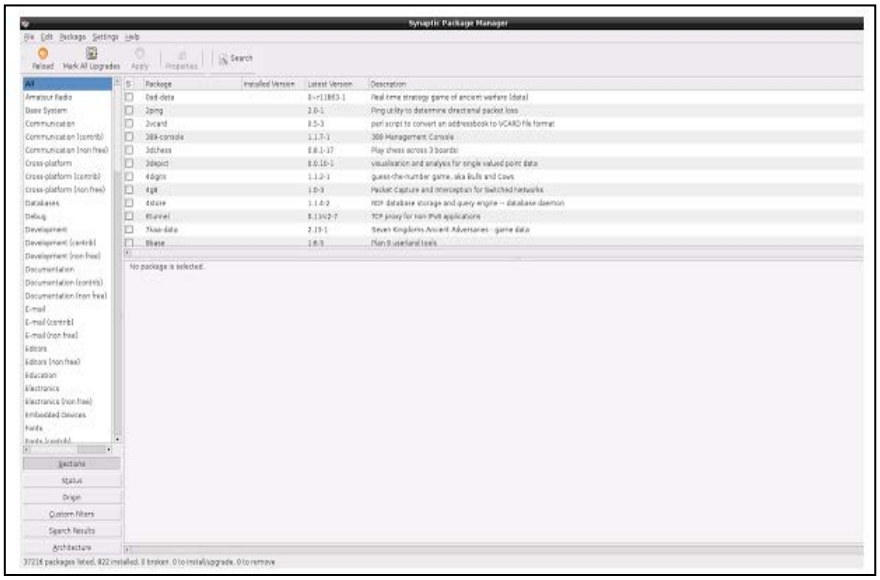

Figura 2.9. Pantalla de descarga del programa Synaptic **[9]**

Con la ayuda de esta herramienta, se instalaron las dos posibles opciones para realizar la interfaz entre el GPIO y los sensores externos necesarios en el proyecto, Python y Code Blocks.

Code Blocks

Code Blocks es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C y

C++. Está basado en la plataforma de interfaces gráficas WxWidgets, lo cual quiere decir que puede usarse libremente en diversos sistemas operativos. Con la ayuda de Synaptic se puede realizar la descarga de este programa sin ninguna dificultad, teniendo en cuenta que las librerías que necesita tener el programa no vienen incluidas en este y es necesario descargarlas por separado. **[10]**

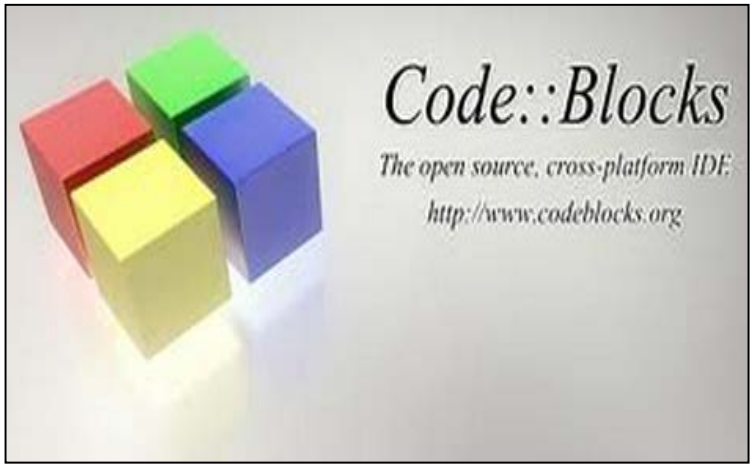

Figura 2.10. Logotipo de Code Blocks instalado en la Raspberry Pi **[11]**

Este programa se caracteriza por su entorno grafico el cual es muy semejante a otros compiladores de Microsoft como Visual Studio 2008. Una posible desventaja que tiene este programa es que no

posee una librería adecuada para el GPIO, debido a esto el usuario necesita descargar dicha librería la cual es conocida como WiringPi.

WiringPi

Esta librería gratuita se encuentra disponible en el siguiente enlace http://www.electroensaimada.com/parpadeo-led.html, el proceso de instalación se asemeja al de cualquier otro programa que se haya instalado con anterioridad, claro está que es necesario agregar esta librería al programa Code Blocks para poder utilizarla.

Instalación y uso de la librería WiringPi con Code Blocks

Para hacer de esta librería, primero es necesario descargarla e instalarla por lo tanto ejecute los siguientes pasos:

- Asegúrese de que su Raspberry Pi esté conectada al internet.
- Diríjase al LX terminal y digite lo siguiente
- Cuando le pregunten acerca de instalar ciertos componentes, declare (Y).
- Después de pocos minutos la instalación se completará exitosamente.

Una vez instalada ahora es necesario adjuntarla junto con las otras librerías disponibles en Code Blocks, para ello siga el siguiente proceso:

Project->Build options

Seleccionar la pestaña linker settings y hacemos clic en Add Solicitara el path de la librería, en cuya dirección debemos colocar: /usr/local/lib y luego seleccionar libwiringPi.

Una vez realizado esto puede llamar a la librería en la cabecera de cualquier programa creado en Code Blocs si el usuario así lo crea conveniente junto con las librerías que vienen por default como por ejemplo:

```
#include<stdio.h>

#include<WiringPi.h>
```

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma , ya que soporta orientación a objetos,

programación imperativa y, en menor medida, programación funcional.

De igual manera que Code Blocks, Python no incluye una librería que permita trabajar con el GPIO del Raspberry Pi, esto puede ser un problema para el usuario, sin embargo una de las librerías más usadas para lidiar con este problema es Rpi.GPIO. **[12]**

Rpi.GPIO

Esta librería gratuita se encuentra disponible en el siguiente enlace http://muyraspi.blogspot.com.es/2013/02/programar-gpio.html, como muchos programas pueden ser instalados con Synaptic o en su defecto con el LX terminal del escritorio.

Instalación y uso de Rpi.GPIO en Python

El proceso de instalación de esta librería se describe a continuación:

- Descargue la librería de la siguiente dirección: *wget http://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.3.1a.tar.gz*

- Extraer el archivo con la siguiente línea de código *tar zxf RPi.GPIO-0.3.1a.tar.gz*
- Crear el siguiente directorio *cd RPi.GPIO-0.3.1a*
- Instalar la librería *sudo python setup.py install*
- Remover el directorio y escribir

  cd ..

  sudo *rm -rf RPi.GPIO-0.3.1a/*
- Finalmente borrar el archivo con el siguiente escrito *rm RPi.GPIO-0.3.1a.tar.gz*

2.3 Herramientas de Hardware

2.3.1 Sensor de corriente Non-invasive AC current sensor (30 A máx) SCT-013-030

Es un sensor de energía eléctrica de gancho el cual puede soportar hasta 30 A de corriente como se muestra en la figura 2.11.

Figura 2.11. Non-invasive AC current sensor

Entre sus características están:

- Resistencia interna de 64 ohmios.
- Corriente máxima de 30 A.
- Temperatura de operación entre -25 C – 70 C.
- Tensión de voltaje de salida entre 0V y 1V.

Aplicaciones: Adecuado para el monitoreo de mediciones de corriente en equipamientos de iluminación, compresores de aire, etc.

2.3.2 Arduino 1.0

El Arduino es una placa basada en un microcontrolador Atmel AVR, que dispone de puertos de entrada y salida, además de un oscilador para trabajar a una frecuencia de reloj predeterminada de hasta 16 MHz. Para programar el Arduino se requiere utilizar el Arduino IDE (Integrated Development Environment) basado en lenguaje C. Tanto el hardware y software del Arduino son libres, es decir que códigos, esquemático y diseño pueden ser utilizados libremente por cualquier persona. Al Arduino se pueden acoplar bases para ciertos tipos de sensores como son receptores de GPS, módulos Ethernet, LCD displays, etc., con la finalidad de añadir otras funcionalidades al Arduino. Existen diferentes modelos de Arduino, entre ellos el Arduino 1.0 que se puede observar en la figura 2.12, utilizado en este proyecto de graduación. Tal vez la variación más poderosa es el Arduino Mega 2560, el cual ofrece incremento de memoria y de puertos de entrada y salida. **[13]**

Figura 2.12. Apariencia física del Arduino 1.0

El Arduino dispone de la facilidad de poder reemplazar el chip o microcontrolador AVR que contiene por uno nuevo en caso de que sea necesario.

2.4 Herramientas de Software

2.4.1 Gnuplot

Es un programa de licencia gratuita, el cual permite realizar gráficas de funciones y datos, con diferentes tipos de formatos como el png, jpeg, eps entre otros. Este software se caracteriza por su compatibilidad con los diversos sistemas operativos que operan en el mercado (Windows y Linux). Debido a que es de

licencia gratuita y código abierto, es posible modificar su código fuente aunque dichos cambios solo se podrían distribuir en forma de parches para el usuario.

Para poder realizar las gráficas de los datos, es necesario ingresar los parámetros de configuración mediante líneas de comando en el Lx Terminal, es decir declarar el tamaño de la gráfica, sus leyendas el título principal, el formato de salida, la ubicación de los datos a graficar entre otras cosas. Finalmente después de declarar todo lo anterior, y si esto se realizó con éxito, la imagen deberá aparecer en la dirección /home/pi en el formato y con el nombre de creación definido en las líneas de comando. **[14]**

2.4.2 Web server casero phpMyAdmin

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos.

Su objetivo en el desarrollo del proyecto, es de almacenar los resultados del consumo eléctrico en una base de datos, para su uso en la implementación de proyectos futuros. **[15]**

2.4.3 Cosm.com

Es una plataforma web que permite conectarse con diversos dispositivos y productos en tiempo real, es decir enviar cierto tipo de información a la página y visualizarlo cada cierto tiempo a escoger por el usuario. Claro que para realizar dicha tarea es necesario poseer ciertas claves de conexión, las cuales se entregan al momento de crear una cuenta en Cosm (Mayor información véase Anexo 9). Una vez creada la cuenta y el feed, es necesario configurar los parámetros de presentación de la gráfica, unidades de las variables, cuantas feeds se van a crear, esto con el objetivo de tener la misma cantidad de variables que se enviarán a Cosm como graficas que se realizaran para evitar errores de transmisión y recepción de los datos.

Cada día Cosm administra miles de datos de información alrededor del mundo proveniente de empresas, organizaciones y miles de personas alrededor del mundo.

El servicio que ofrece Cosm permite extraer y analizar los datos, enviar alertas en tiempo real desde cualquier flujo de datos para el control de secuencias de comandos y dispositivos. **[16]**

Figura 2.13. Logotipo de Cosm **[17]**

2.4.4 LX Terminal

Es un emulador de terminal independiente de escritorio basado en VTE sin ninguna dependencia innecesaria. LX Terminal es considerado el emulador de terminal estándar de LXDE.

En esta ventana parecida al cmd de Windows, se realizan las instalaciones de los programas por línea de comando. **[18]**

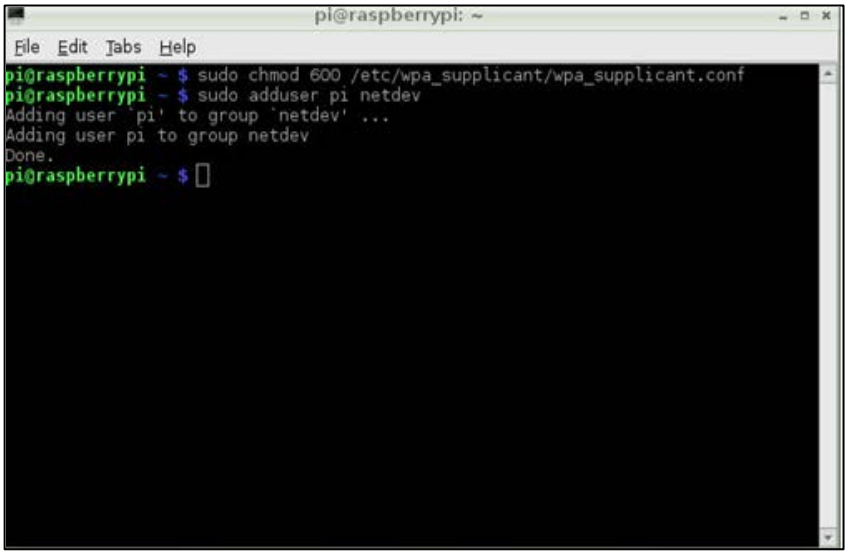

Figura 2.14. Lx Terminal del Raspberry Pi **[19]**

2.4.5 LXDE

LXDE es el acrónimo inglés de Lightweight X11 Desktop Environment (Entorno de Escritorio X11 Liviano), un entorno de escritorio extremadamente rápido, ágil y eficiente en consumo de energía. LXDE utiliza menos CPU y RAM que otros entornos de escritorio. Está diseñado especialmente para ordenadores en la nube con especificaciones de hardware de escasa potencia tales como netbooks, dispositivos móviles (por ejemplo, los dispositivos móviles para conectividad a Internet), u ordenadores antiguos. Proporciona una experiencia de escritorio rápida que conecta fácilmente con aplicaciones en la nube. LXDE soporta una gran

cantidad de programas que pueden instalarse localmente en sistemas Linux. El código fuente de LXDE tiene una licencia en parte bajo los términos de la General Public License (GPL) y en parte bajo la Lesser General Public License (LGPL). **[20]**

CAPÍTULO 3

EJERCICIOS PREVIOS Y REALIZACIÓN DEL PROYECTO

3.1 Introducción

Este capítulo describe en forma detallada cada uno de los ejercicios previos a la realización completa de nuestro proyecto de grado. En el mismo se presenta una breve descripción de cada ejercicio realizado, presentando cuatro partes fundamentales en cada uno que son: diagrama de bloques, diagrama de flujo, descripción del algoritmo y el código fuente que se utilizó.

Todos los ejercicios realizados fueron elaborados secuencialmente con la finalidad de ir probando etapa por etapa cada uno de los componentes que utilizamos en nuestro proyecto, además del software respectivo para cada uno, así de esta forma logrando completar un perfecto acoplamiento entre los sensores de corriente, el Arduino 1.0 y la

minicomputadora Raspberry Pi, lo que nos llevó a la finalización de nuestro proyecto en una interfaz no muy amigable que es el LX Terminal del Raspberry Pi, por lo que tuvimos que tomar la decisión de como presentaríamos los datos gráficamente, para lo cual llevamos a cabo los últimos tres ejercicios de este capítulo, los cuales tienen ciertas ventajas y desventajas que detallaremos dentro de cada uno de estos ejercicios.

Los primeros cinco ejercicios fueron realizados con el propósito de poder lograr el acoplamiento entre los dispositivos que corresponden al hardware de nuestro proyecto para la correcta adquisición de los datos de consumo eléctrico, para esto tuvimos que instalar ciertos programas y librerías que nos facilitarían el desarrollo normal del mismo.

El procesamiento de los datos lo hemos realizado con ciertos programas instalados en la memoria SD de la minicomputadora Raspberry Pi como son el IDE del Arduino 1.0 que es un compilador en lenguaje C donde se encuentra el programa de la captura de los datos y su debido procesamiento adaptado a las necesidades de nuestro proyecto de grado, también hemos utilizado el programa Phyton para la conexión serial vía USB entre el Arduino 1.0 y el Raspberry Pi, así como para la generación de un archivo .csv para poder realizar la gráfica en el web

server casero phpMyAdmin y también la generación de un .txt para tener como otra opción de poder realizar la gráfica en el programa Gnuplot.

El punto más interesante de este capítulo es el último ejercicio en donde realizamos la gráfica del consumo de energía eléctrica en tiempo real vía streaming en un servidor libre conocido como pachube o también como el internet de las cosas cuya url es www.cosm.com.

3.2 Ejercicios desarrollados

3.2.1 EJERCICIO 1.- Familiarización del GPIO del Raspberry Pi mediante el encendido de un LED.

Descripción

Este ejercicio consiste en el encendido y apagado de un LED a una frecuencia de 1 Hz, logrando esto por medio de un retardo de 1 segundo. Para esto hemos realizado un pequeño programa en Phyton el cual nos permitirá aprender a utilizar las entradas y salidas de propósito general o GPIO que posee la minicomputadora Raspberry Pi. Vale aclarar que el GPIO del Raspberry Pi solo se lo puede configurar como entradas o salidas digitales. Para poder hacer uso del GPIO tuvimos que importar la librería Rpi.GPIO dentro del programa Phyton, así como la librería

de tiempo para poder hacer uso de los retardos dentro del programa. En este código hemos implementado una función que nos permite realizar el encendido y apagado de un LED, por medio de uno de los pines del GPIO configurado como salida digital, el cual hacemos que permanezca en HIGH durante un segundo y en LOW durante un segundo más, utilizando un lazo for y llevando a cabo una iteración para que se realice durante 10 veces este proceso. La finalidad de este ejercicio es aprender el uso adecuado de los pines de entrada/salida de propósitos generales (GPIO) y recordarles a los lectores y futuros diseñadores que el Raspberry Pi solo posee salidas o entradas digitales.

Herramientas utilizadas

Herramientas de hardware:

- 1 Raspberry Pi con sistema operativo instalado.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 hub-USB con alimentación propia.
- 1 diodo LED (cualquier color).
- 1 resistencia de 1kΩ.

- Cables de conexión.

Herramientas utilizadas de software:

- Python versión 2.7 o superior.
- Instalar las librerías del GPIO para Python (RGPIO.GPIO).

Diagrama de Bloques

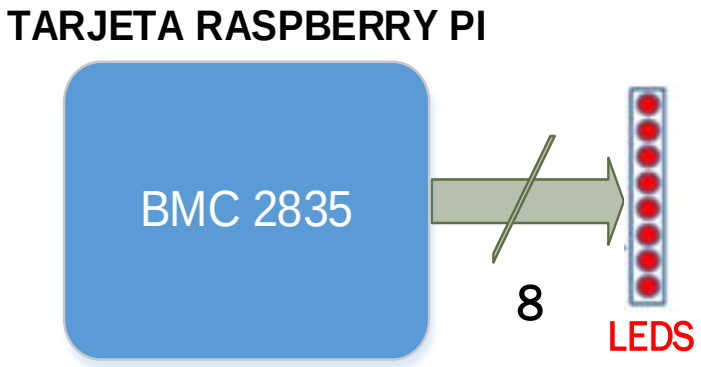

Figura 3.1. Diagrama de Bloques del Ejercicio 1

Diagrama de Flujo

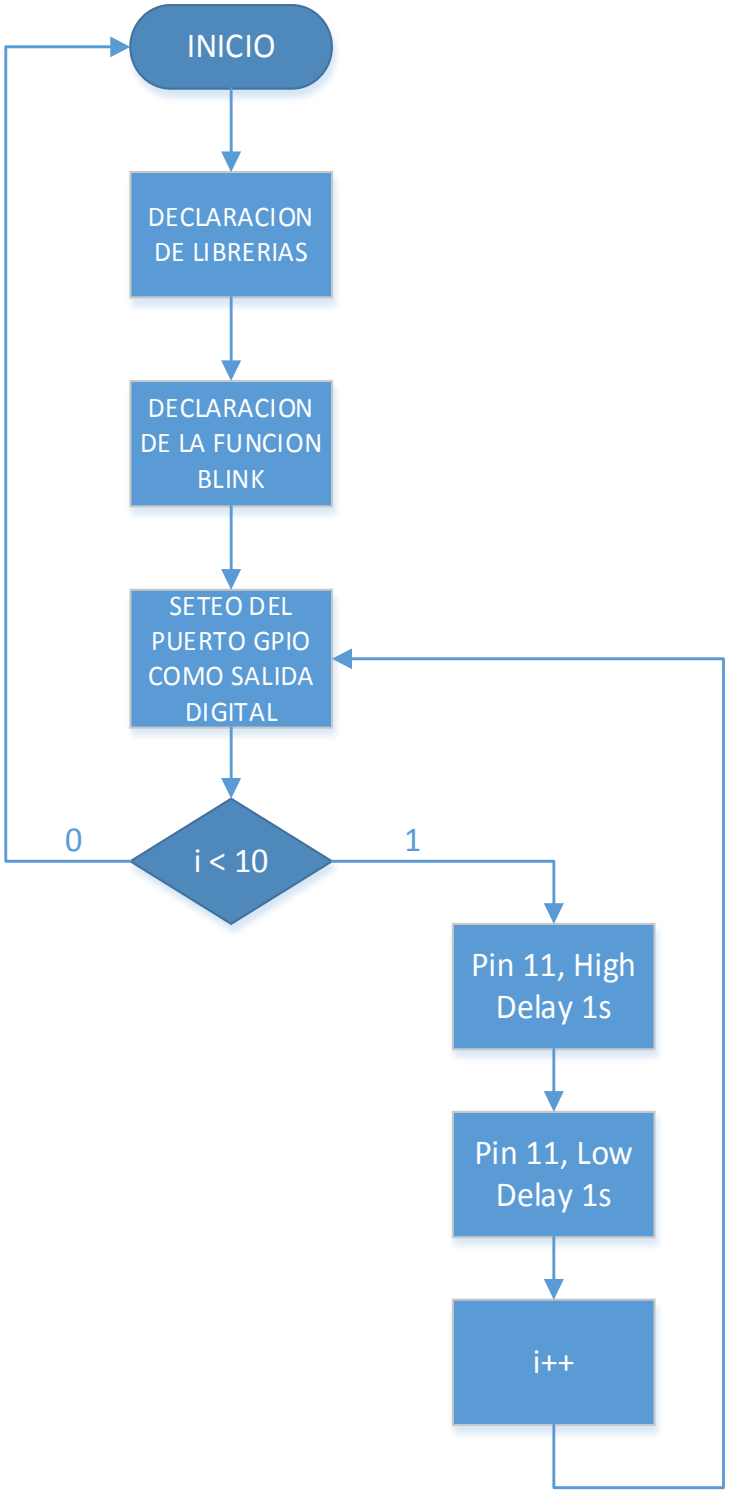

Figura 3.2. Diagrama de Flujo del Ejercicio 1

Descripción del Algoritmo

1) Se realiza la declaración de las librerías Rpi.Gpio y Time.
2) Se define la función blink.
3) Se realiza el seteo del pin 11 del puerto GPIO del Raspberry Pi como salida digital.
4) Se inicia un lazo for con 10 iteraciones.
5) Se pregunta si la variable i es menor que 10, de no ser así regresa al inicio del programa.
6) El pin 11 se pone en high y se prende el LED durante un segundo.
7) El pin 12 se pone en low y se apaga el LED durante un segundo.
8) Si el contador i es mayor o igual que 10 termina el programa.

Código Fuente

```
#declaración de librerías
import RPi.GPIO as GPIO
import time

# funcion de tiempo
def blink(pin):
        GPIO.output(pin,GPIO.HIGH)
```

```
    time.sleep(1)
        GPIO.output(pin,GPIO.LOW)
time.sleep(1)
return

#numeros de pines
GPIO.setmode(GPIO.BOARD)

#SETEAR COMO SALIDA PIN 11
GPIO.setup(12,GPIO.OUT)

#REPETIR EL ON/OFF 10 VECES
for i in range(0,10):
        blink(12)
GPIO.cleanup()
```

Imágenes

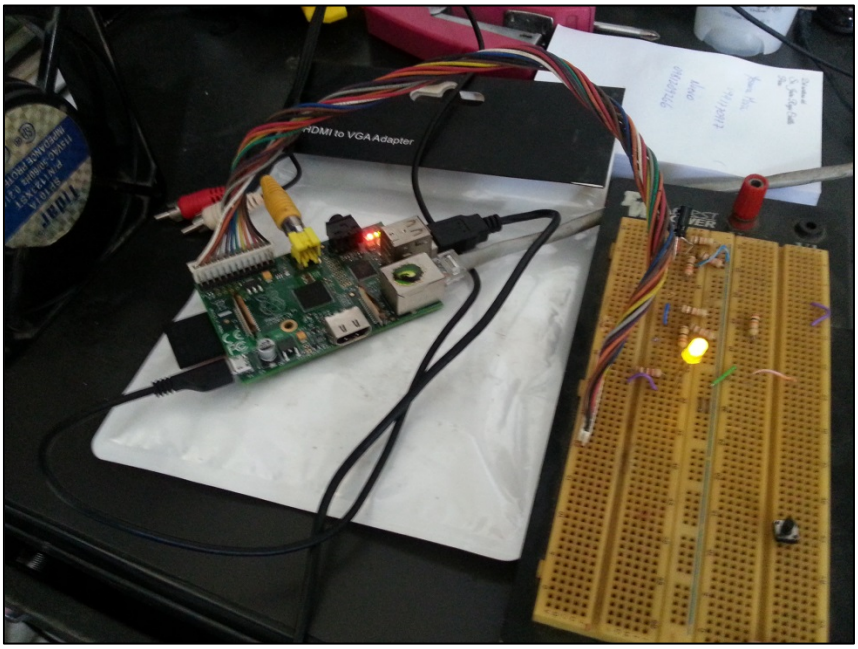

Figura 3.3. Encendido/Apagado del LED usando el Raspberry Pi

Conclusiones

El desarrollo de este ejercicio se realizó con la finalidad, de que sea como base para el posterior uso del GPIO en proyectos más avanzados. De igual manera se refuerza la correcta conexión de cables del o hacia el GPIO, tomando con consideración de que el alto lógico es de apenas 3.3V eso significa que si nuestro pin del GPIO va a ser programado como entrada, el voltaje que reciba no puede sobrepasar este valor, y a su vez recordar que no todos los pines se pueden usar, y el mal uso de esto puede llevar desde la avería de un pin hasta el daño completo del Raspberry Pi.

3.2.2 EJERCICIO 2.- Familiarización de los puertos digitales del Arduino 1.0 mediante el encendido de LEDS.

Descripción

Este ejercicio consiste en probar el funcionamiento de los puertos digitales que posee el microcontrolador Arduino 1.0 mediante el encendido de un LED. Aunque para efecto de nuestro proyecto utilizaremos los puertos analógicos del mismo y esto ya que los niveles de referencia de voltaje que captura el sensor SCT-013-030 van entre 0 y 1V, pero este ejercicio se lo realizó de tal forma que se pudiera tener una primera experiencia con el Arduino 1.0 mediante el encendido y apagado de un LED, un ejemplo básico y visible.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 hub-USB con alimentación propia.

- 1 diodo LED (cualquier color).
- 1 resistencia de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino IDE, en donde se realiza la programación para el Arduino 1.0.

Diagrama de Bloques

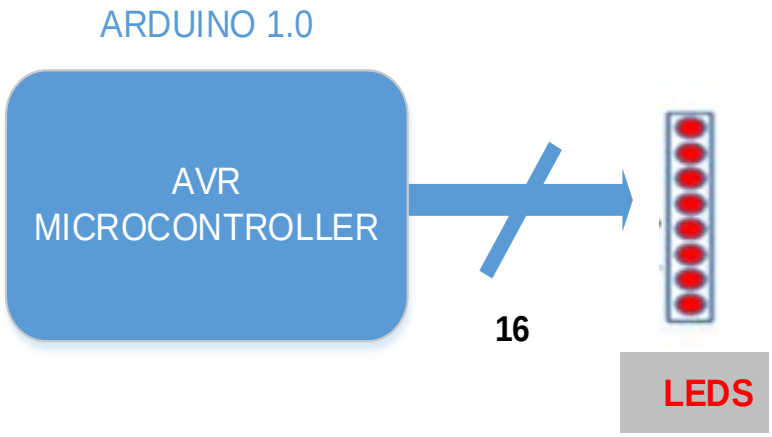

Figura 3.4. Diagrama de Bloques del Ejercicio 2

Diagrama de Flujo

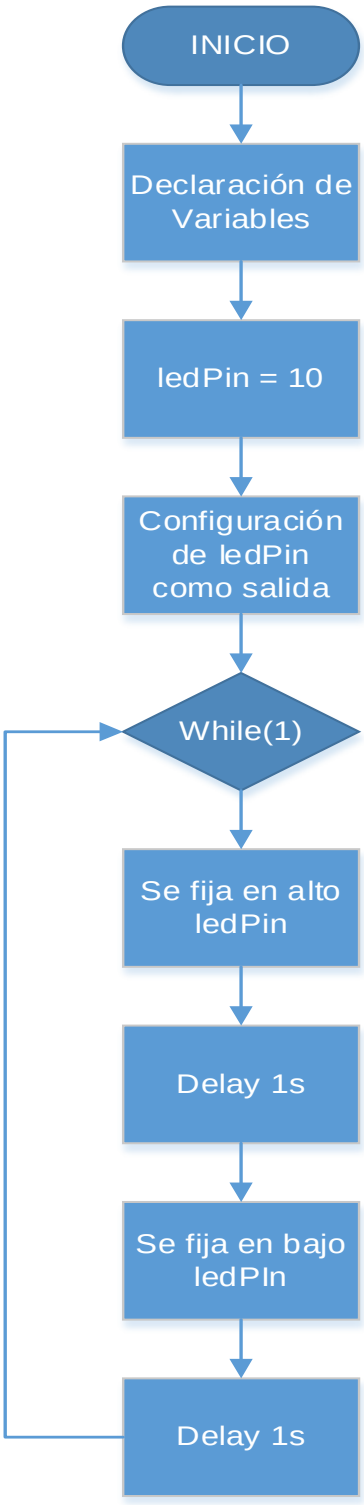

Figura 3.5. Diagrama de Flujo del Ejercicio 2

Descripción del Algoritmo

1) Se asigna uno de los pines digitales del Arduino 1.0 a una variable, en este caso el pin 10 lo estamos asignando a la variable ledPin.
2) Se define la función void setup() en la cual se llama a la función pinmode(ledPIn, OUTPUT), donde se fija el pin del Arduino 1.0 como salida digital.
3) En el main() se llama a la función digitalWrite(ledPin, HIGH), que fija en alto el pin del Arduino 1.0 que se está utilizando.
4) Se establece un delay de 1s.
5) Se llama a la función digitalWrite(ledPin, LOW), que fija en bajo el pin del Arduino 1.0 que estamos utilizando.
6) Se establece otro delay de 1s con el propósito de que sea visible el encendido y apagado de un LED.
7) Este proceso se realiza indefinidamente hasta que manualmente se detenga la compilación del programa.

Código Fuente

```
//Project – Led Flasher
int ledPin = 10;
```

```
void setup() {
      pinMode(ledPin, OUTPUT);
}

Void loop(){
      digitalWrite(ledPin, HIGH);
      delay(1000);
      digitalWrite(ledPin, LOW);
      delay(1000);
}
```

Imágenes

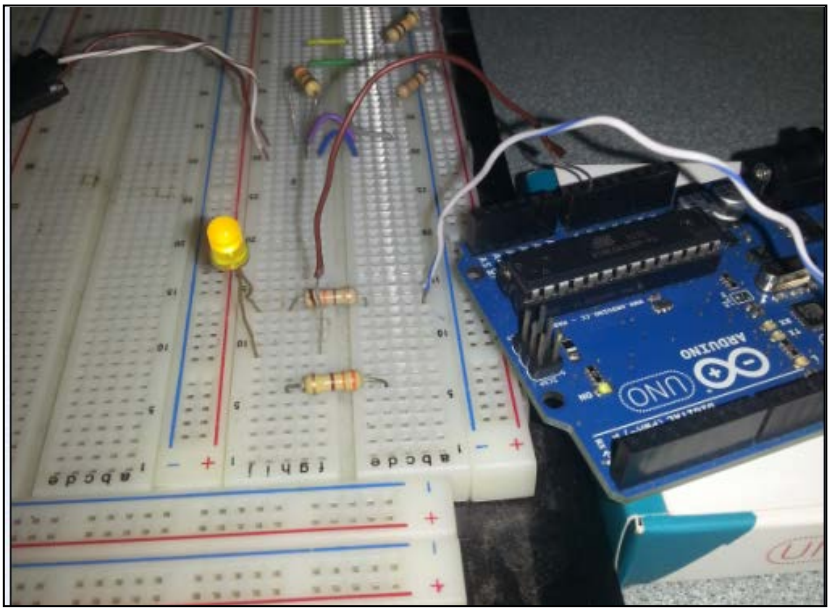

Figura 3.6. Encendido/Apagado del LED usando el Arduino 1.0

Conclusiones

A diferencia del Raspberry Pi, el Arduino 1.0 es capaz de recibir/entregar voltajes de 5 voltios, por lo cual el diseñador se siente un poco más seguro a la hora de ensamblar proyectos en esta plataforma, ya que además posee un fusible contra sobre corrientes lo cual no posee el Raspberry Pi. En cuanto al lenguaje de programación del Arduino 1.0, es muy parecido, en sintaxis, al lenguaje C con algunas pequeñas diferencias en cuanto a las funciones y la forma de programar, pero no perdamos de vista otro propósito importante detrás de este ejercicio, la cual es que el Arduino 1.0 no sólo posee salidas digitales sino también

analógicas, las cuales precisamente serán usadas en el proyecto final, mientras que el Raspberry Pi no posee pines analógicos.

3.2.3 EJERCICIO 3.- Conexión serial vía USB entre el Raspberry Pi y el Arduino 1.0 con LEDS de comprobación.

Descripción

Este ejercicio fue realizado con el propósito de establecer una comunicación serial vía USB half dúplex entre la minicomputadora Raspberry Pi y el microcontrolador Arduino 1.0 mediante el encendido de LEDS que indicaran si existe o no comunicación serial entre ellos. Primeramente se incluyen las librerías de comunicación serial en el compilador de Phyton y se configura la comunicación serial, para luego leer por teclado un valor entero el cual lo enviaremos vía serial al Arduino 1.0 el cual lo capturará y dependiendo de si exista o no comunicación entre ambos dispositivos hará que se enciendan y se apaguen los LEDS con un retardo la cantidad de veces definidas por el valor entero que se envió desde el Raspberry Pi.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 Raspberry Pi.
- 1 cable USB-impresora.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 hub-USB con alimentación propia.
- 2 diodos LED (cualquier color).
- 2 resistencias de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino Idle que es donde se escribe la programación para el Arduino.
- Python versión 2.7 o superior.
- Instalar las librerías de py-serial y python-serial en el Raspberry Pi.

Diagrama de Bloques

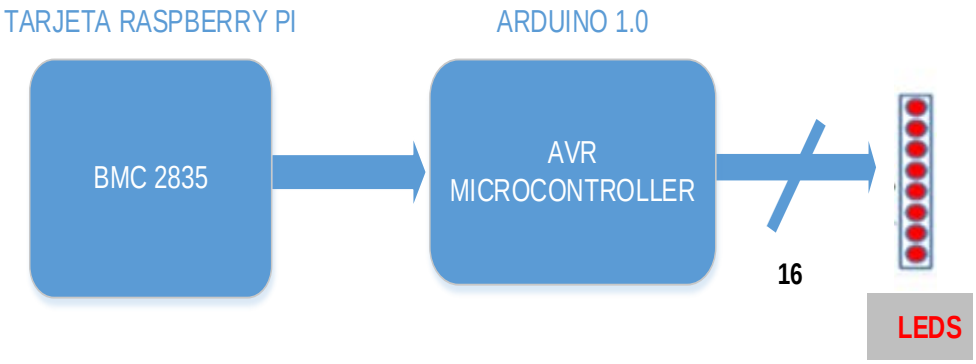

Figura 3.7. Diagrama de bloques del Ejercicio 3

Diagrama de Flujo

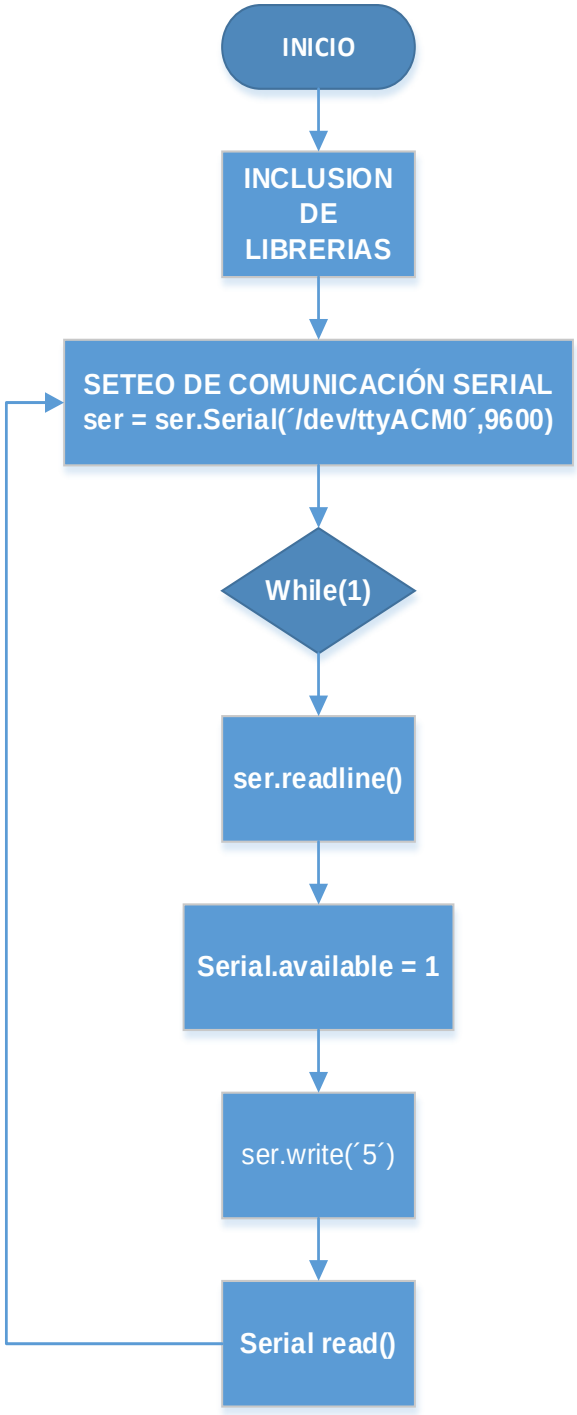

Figura 3.8. Diagrama de Flujo del Ejercicio 3 en Phyton

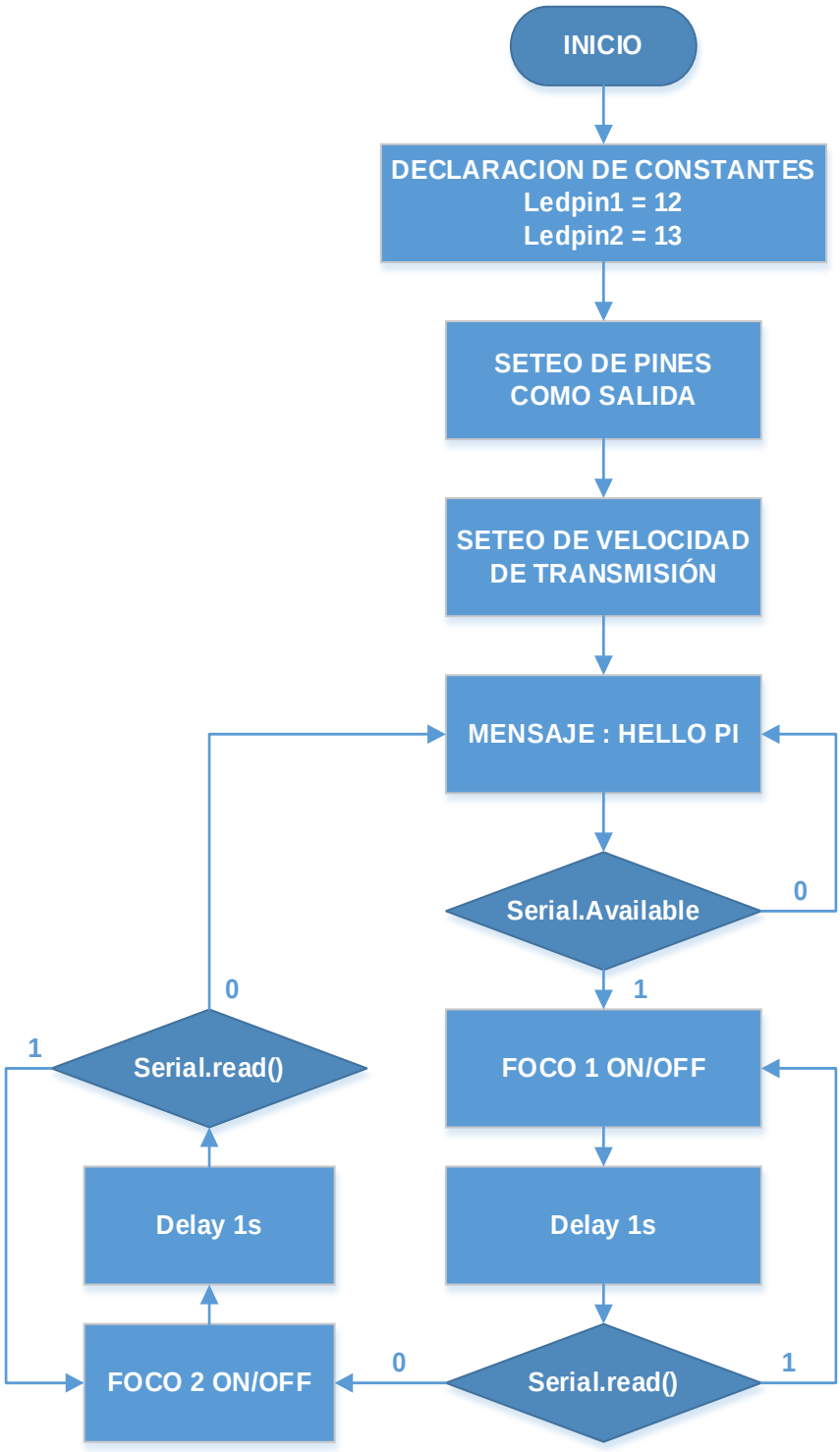

Figura 3.9. Diagrama de Flujo del Ejercicio 3 en Arduino IDE

Descripción del Algoritmo

Código implementado en Phyton

1) Incluimos las librerías que habilitan la comunicación serial.
2) Se configura la tasa de transmisión de la comunicación serial en 9600, y se almacena en la variable ser.
3) Se establece un lazo infinito, mediante la condición while(1).
4) Se activa el comando ser.realine() que lee n cantidad de bytes hasta encontrar un \n.
5) Se activa el comando ser. write(5) que permite enviar el número 5 vía serial.

Código implementado en el Arduino IDE

1) Se declara la constante Ledpin1 y se la asigna al pin 12 del Arduino 1.0.
2) Se declara la constante Ledpin2 y se la asigna al pin 13 del Arduino 1.0.
3) Se configuran los pines 12 y 13 del Arduino 1.0 como salidas digitales.
4) Se configura la velocidad de transmisión de la comunicación serial en 9600.
5) Se imprime en pantalla un mensaje de bienvenida ¨Hello Pi¨.

6) Se consulta por el comando serial.available(), el cual indica si existe o no comunicación serial entre los dos dispositivos.
7) Si el resultado de serial.available() es igual a 1 significa que si hay comunicación y se imprime en pantalla ¨Conexión exitosa #1¨, ¨Foco Amarillo ON/OFF¨, además realiza el llamado de la función flash1(int n).
8) Se envía Serial.read() - '0' como parámetro de la función flash1(int n), el cual devuelve la lectura del valor entero enviado durante la comunicación.
9) Se ejecuta la función flash1(Serial.read() - '0'), la cual realiza un lazo for con el número de iteraciones obtenidas del comando Serial.read().
10) Dentro del lazo for se manda a high y a low el pin 12 del Arduino 1.0 cada 300 ms durante el número de iteraciones establecidas.
11) Nuevamente se vuelve a preguntar por el resultado del comando serial.available(), en caso de no existir comunicación termina el lazo.
12) Si el resultado de serial.available() es igual a 1 significa que sigue existiendo comunicación y se imprime en pantalla ¨Conexión exitosa #2¨, ¨Foco Rojo ON/OFF¨, además realiza el llamado de la función flash2(int n).

13) Se envía Serial.read() - '0' como parámetro de la función flash2(int n), el cual devuelve la lectura del valor entero enviado durante la comunicación.
14) Se ejecuta la función flash2(Serial.read() - '0'), la cual realiza un lazo for con el número de iteraciones obtenidas del comando Serial.read().
15) Dentro del lazo for se manda a high y a low el pin 13 del Arduino 1.0 cada 300 ms durante el número de iteraciones establecidas.
16) Este proceso se realiza indefinidamente hasta que se corte la comunicación serial entre ambos dispositivos.

Código fuente

Código implementado en Phyton

```
import serial

ser = serial.Serial('/dev/ttyACM0', 9600)

while 1 :
        ser.readline()
   ser.write('5')
```

Código implementado en el Arduino IDE

```
const int ledPin1 = 12;
const int ledPin2 = 13;

void setup()
{
    pinMode(ledPin1, OUTPUT);
    pinMode(ledPin2, OUTPUT);
    Serial.begin(9600);
}

void loop()
{
    Serial.println("Hello Pi");
     if (Serial.available())
     {
            Serial.println("Conexion exitosa #1");
            Serial.println("Foco Amarillo ON/OFF ");
            flash1(Serial.read() - '0');
    }
    delay(1000);
    if (Serial.available())
    {
```

```
            Serial.println("Conexion exitosa");
            Serial.println("Foco Rojo ON/OFF ");
            flash2(Serial.read() - '0');
    }
    delay(1000);
}

void flash1(int n)
{
    for (int i = 0; i < n; i++)
    {
            digitalWrite(ledPin1, HIGH);
            delay(300);
            digitalWrite(ledPin1, LOW);
            delay(300);
    }
}

void flash2(int n)
{
    for (int i = 0; i < n; i++)
    {
            digitalWrite(ledPin2, HIGH);
            delay(300);
```

```
            digitalWrite(ledPin2, LOW);
            delay(300);
    }
}
```

Imágenes

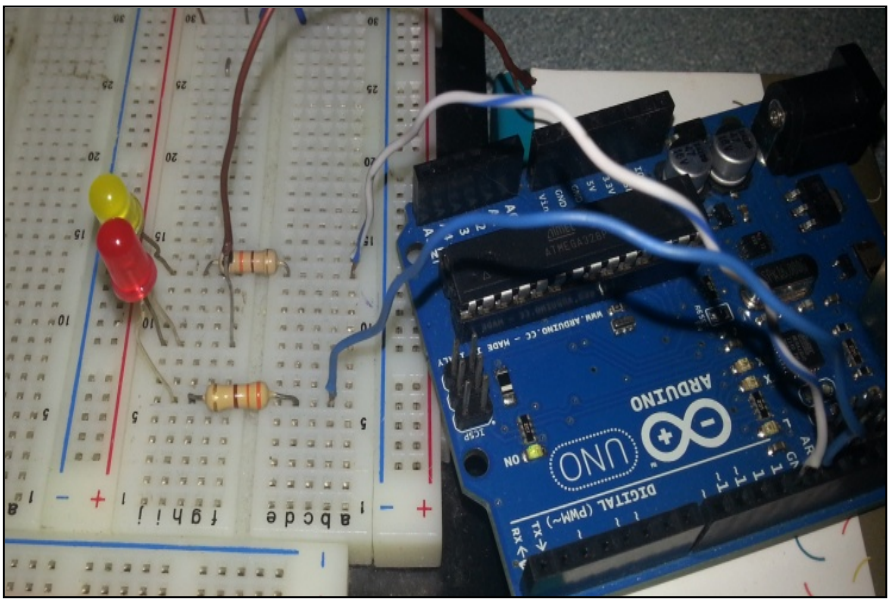

Figura 3.10. Funcionamiento de la comunicación entre el Raspberry Pi y el Arduino 1.0

Conclusiones

Los resultados de este proyecto muestran que para realizar la conexión exitosa entre estos dos dispositivos es necesario incluir retardos en la programación, de lo contrario cuando se intente

enviar información durante los primeros segundos, puede que se reciba datos extraños o en su defecto no haya suficiente tiempo de espera y finalice la ejecución del programa antes siquiera de iniciar.

3.2.4 EJERCICIO 4.- Adquisición de Datos de Consumo Eléctrico utilizando los puertos análogos del Arduino 1.0 y presentación de los mismos en el Serial Monitor del Arduino IDE.

Descripción

Este ejercicio fue realizado con el objetivo de comprobar la adquisición de los datos de consumo de energía eléctrica utilizando únicamente el Arduino 1.0 y el sensor de corriente no invasiva SCT-013-030, aprovechándonos de las entradas analógicas que posee el mismo y esto debido a que el sensor de corriente SCT-013-030 entrega niveles de voltaje entre 0 y 1V los cuales hay que interpretarlos en el código implementado en el Arduino IDE, para posteriormente poderlos presentar en el serial monitor. Para poder hacer uso de las funciones para el cálculo de la energía consumida tenemos que incluir dentro del código la librería Emonlib.h.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 Raspberry Pi.
- 1 sensor de corriente SCT-013-030.
- 1 cable USB-impresora.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 hub-USB con alimentación propia.
- 1 resistencia de 33 Ω.
- 2 resistencias de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino IDE que es donde se realiza la programación para el Arduino 1.0.
- Instalar la librería Emonlib.h para el Arduino 1.0.

Diagrama de Bloques

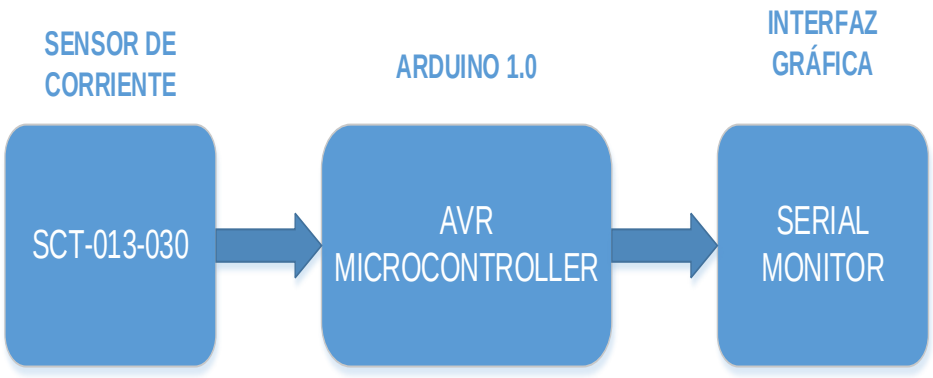

Figura 3.11. Diagrama de Bloques del Ejercicio 4

Diagrama de Flujo

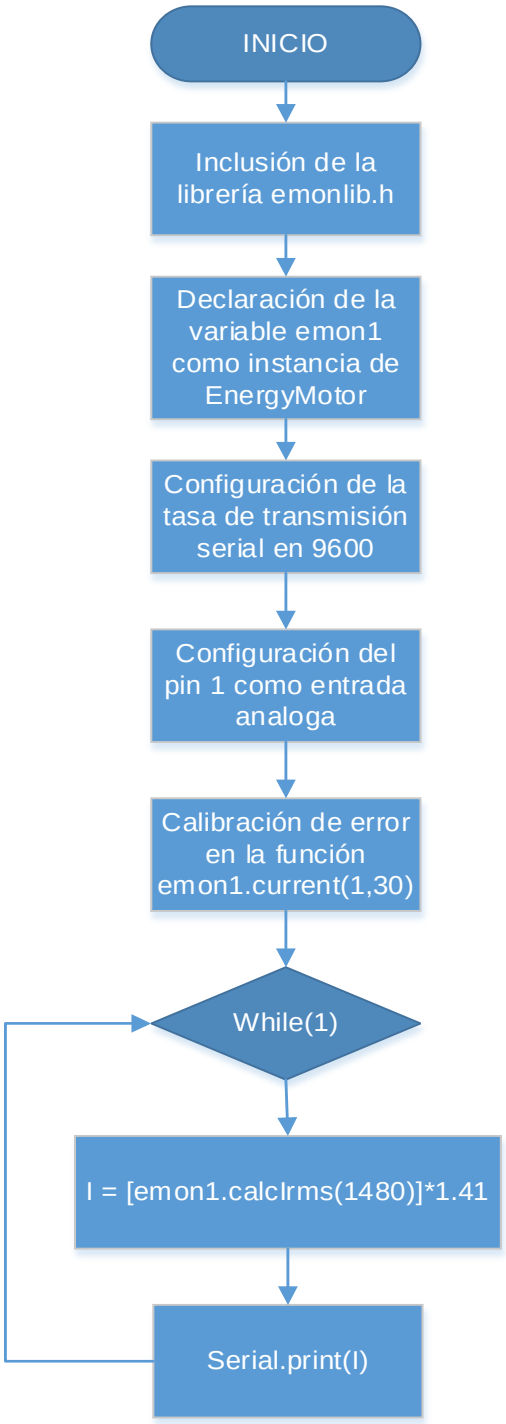

Figura 3.12. Diagrama de Flujo del Ejercicio 4

Descripción del Algoritmo

1) Se incluye la librería Emonlib.h que posee ciertas funciones ya implementadas con el manejo de sensores de corriente.
2) Se crea una instancia de la clase EnergyMotor y se lo almacena en la variable emon1.
3) Se realiza la configuración de la tasa de transmisión serial para la comunicación en 9600.
4) Se configura el pin 1 del Arduino 1.0 como entrada analógica para ingresar por ese pin los valores de voltaje tomados por el sensor SCT-013-030.
5) Se realiza un llamado a la función emon1.current(1,30) y se fija el porcentaje de calibración de error en 30.
6) Se crea un lazo infinito en el cual se realizan tomas secuenciales de corriente mediante la función [emon1.calcIrms(1480)] * 1.41.
7) Finalmente se imprime en pantalla los valores de corriente calculados.

Código fuente

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance
```

```
void setup()
{
        Serial.begin(9600);
        emon1.current(1, 30);     // Current: input pin, calibration.
}

void loop()
{
      double Irms = emon1.calcIrms(1480);  // Calculate Irms only
      Serial.print(Irms);
}
```

Imágenes

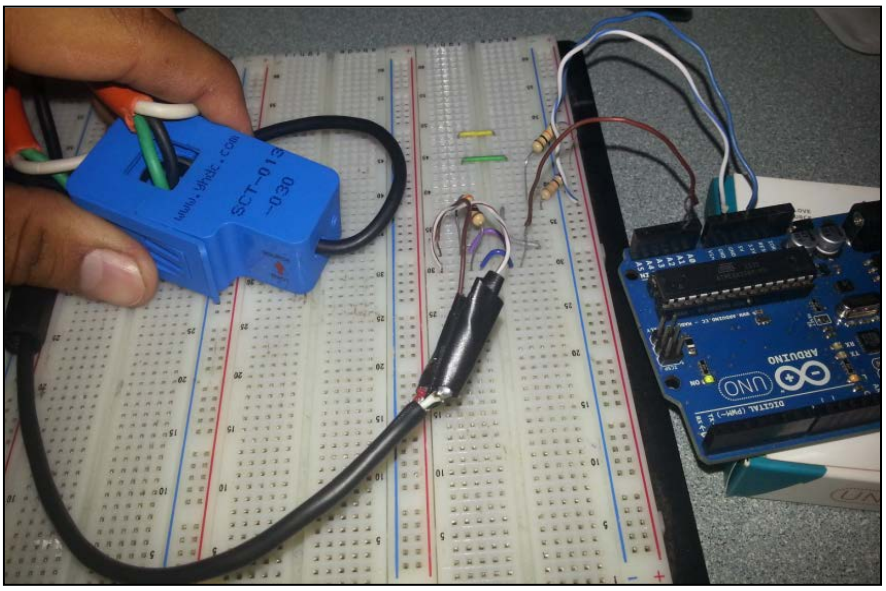

Figura 3.13. Censado de Corriente mediante el sensor SCT- 013-030 y el Arduino 1.0

Figura 3.14. Visualización de la corriente usando el serial monitor del Arduino IDE

Conclusiones

A pesar de tener un porcentaje de error bajo, el código del Arduino y su librería incorporada ofrecen una manera muy eficiente de realizar este ejercicio, sin embargo hay ciertas fallas que no pueden pasar por alto, como que al inicio del programa se aprecia en el serial monitor del Arduino datos iniciales extraños. Esto es debido a fallas del muestreo inicial del sensor y a retardos en abrir la conexión serial. Mediante la finalización de este ejercicio se puede tomar medidas para que este tipo de errores no se manifieste en el proyecto final, y de hecho hay un procedimiento

sencillo para solucionar este problema el cual fue aplicado en el ejercicio siguiente.

3.2.5 EJERCICIO 5.- Adquisición de datos de consumo eléctrico utilizando el Arduino 1.0, el Raspberry Pi y como interfaz gráfica el Lx Terminal del Raspberry Pi.

Descripción

Este ejercicio es una variación del ejercicio anterior en donde interactúan todos los dispositivos de hardware que hemos utilizado en este proyecto de tal manera que es un gran avance para la finalización del mismo, formado básicamente por tres etapas, primero la recolección de los datos de consumo de energía eléctrica cada cierto tiempo por medio del sensor de corriente SCT-013-030, este sensor entrega niveles de voltaje entre 0 y 1V, los cuales son ingresados en uno de los pines analógicos que tiene el microcontrolador Arduino 1.0, en esta etapa se procesan los datos para posteriormente enviarlos vía serial USB a la minicomputadora Raspberry Pi y presentarlos en el Lx Terminal en el formato de fecha, hora, valor numérico de la corriente y la potencia consumida en tiempo real.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 Raspberry Pi.
- 1 sensor de corriente SCT-013-030.
- 1 cable USB-impresora.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 hub-USB con alimentación propia.
- 1 resistencia de 33 Ω.
- 2 resistencias de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino IDE que es donde se realiza la programación para el Arduino 1.0.
- Instalar la librería Emonlib.h para el Arduino 1.0.
- Python versión 2.7 o superior.

- Instalar las librerías de py-serial y python-serial en el Raspberry Pi.

Diagrama de Bloques

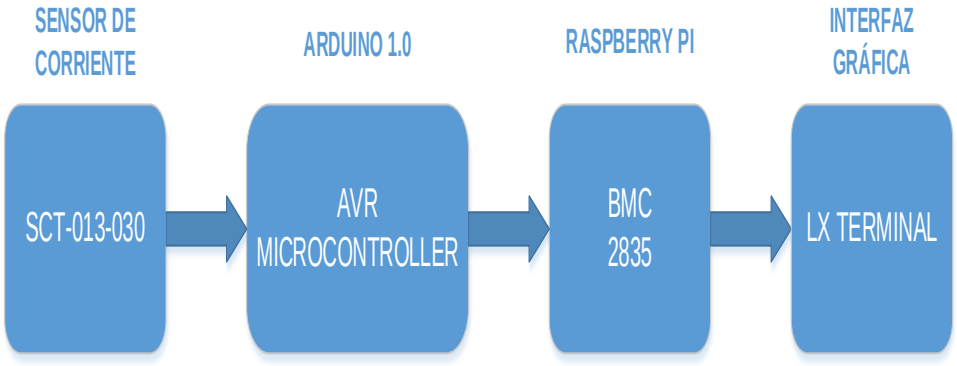

Figura 3.15. Diagrama de Bloques del Ejercicio 5

Diagrama de Flujo

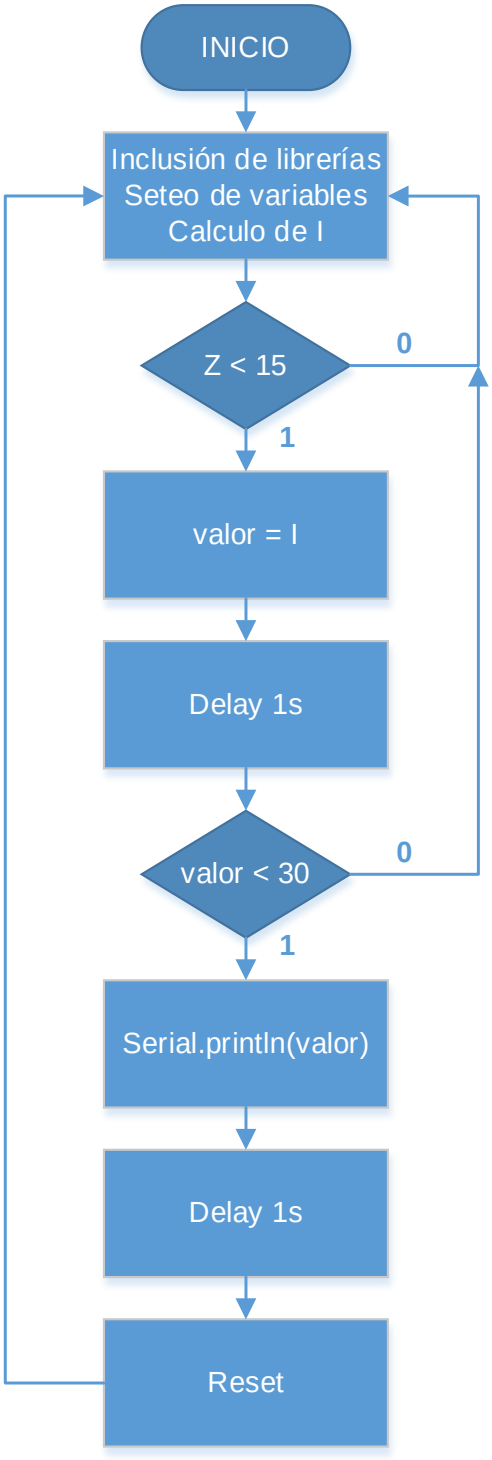

Figura 3.16. Diagrama de Flujo del Ejercicio 5 en Arduino IDE

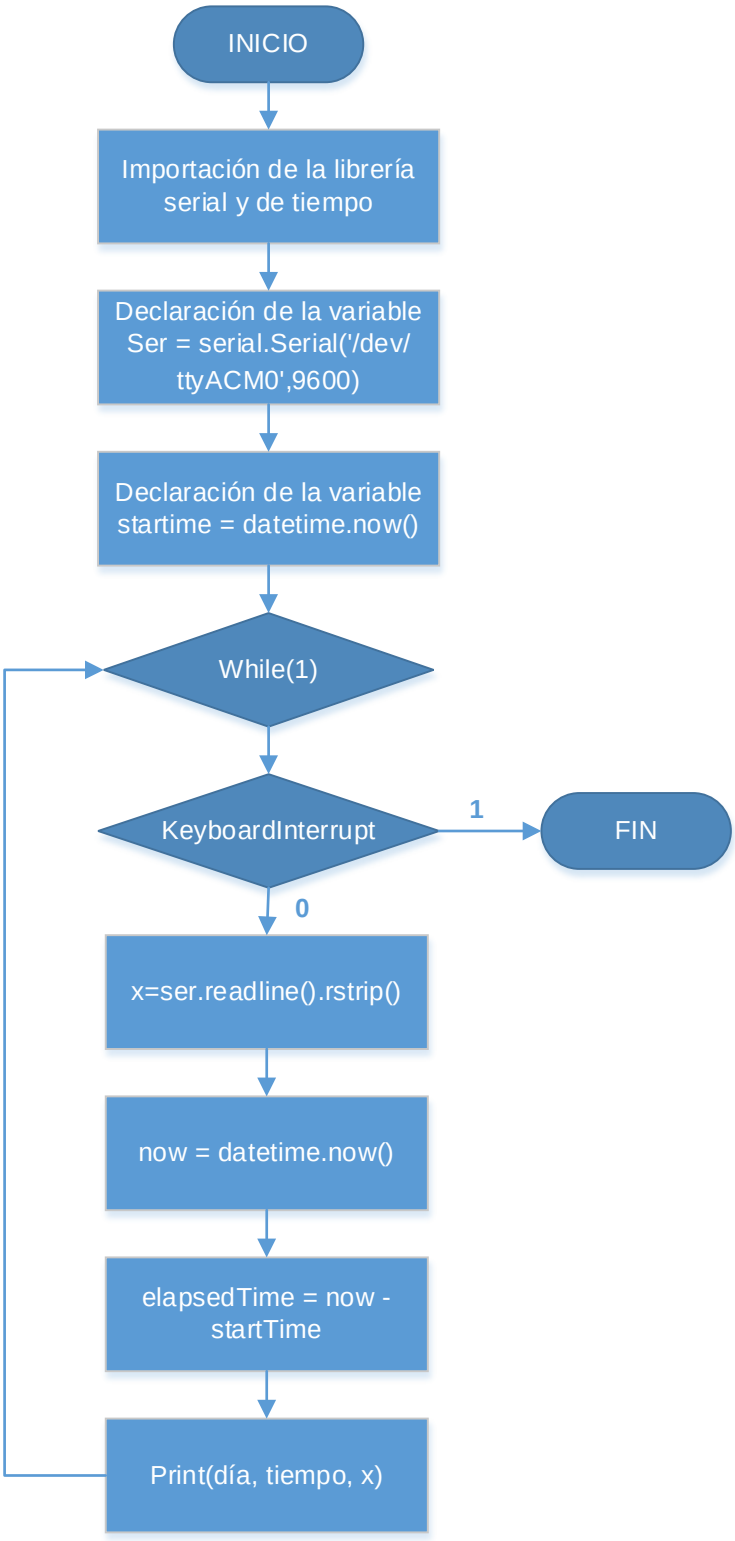

Figura 3.17. Diagrama de Flujo del Ejercicio 5 en Phyton

Descripción del Algoritmo

Código implementado en el Arduino IDE

1) Se incluye la librería EmonLib.h que contiene internamente ya ciertas funciones implementadas para el cálculo de corriente a través de ciertos sensores, además de la calibración por si existe porcentajes de error en los mismos.
2) Se declaran las variables que van a ser utilizadas durante el programa.
3) Se declara la variable entera z y se la encera a la vez.
4) Se declara el procedimiento void setup() donde se fijan las configuraciones de la tasa de transmisión serial en 9600 y se calibra el porcentaje de error de la medición de corriente.
5) Se realiza un lazo infinito y dentro de él un lazo for en donde se incrementa la variable z.
6) Se consulta por la variable z, si es 1 calcula el valor de la corriente en ese instante y se lo asigna a la variable valor.
7) Se establece un retardo de 1s.
8) Se pregunta por la variable valor, si es mayor a 30 sale del lazo infinito.
9) Si la variable valor es menor a 30 envía serialmente el valor calculado de la corriente al Raspberry Pi mediante el comando serial.println(valor).

10) Se establece un retardo de 1s.
11) Finalmente se llama a la función reset(), donde se enceran todas las variables.

<u>Código implementado en Phyton</u>

1) Se importan la librería serial y la de tiempo.
2) Se realiza la declaración de la variable ser = serial.Serial('/dev/ttyACM0',9600) y se establece la velocidad de transmisión en 9600.
3) Se realiza la declaración de la variable startTime = datetime.now().
4) Se entra a un lazo while infinito el cual solamente puede ser interrumpido si se presiona una tecla.
5) Se le asigna a la variable x = ser.readline().rstrip().
6) Se le asigna a la variable now = datetime.now().
7) Se le asigna a la variable elapsedTime la diferencia entre la variable now y startTime, es decir elapsedtime = now – startTime.
8) Se lee la información enviada serialmente por el Arduino 1.0.
9) Se imprime en la pantalla del Lx Terminal del Raspberry Pi el día, el tiempo y el valor de la corriente.

Código Fuente

Código implementado en el Arduino IDE

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

 float Irms;
 float Irms2;
 double valor;
 double valor2;
 int z=0;

void setup()
{
      Serial.begin(9600);
      emon1.current(1,30);
}

void loop()
{
      reset();
      for (z=0;z<5;z++)
      {
            valor=calcular();
```

```
            valor2=110*valor;//watts
        }
        z=0;
        if (valor<30)
        {
            Serial.print(valor);
            Serial.print(" ");
            Serial.print(valor2);
            delay(1);
        }
}

double calcular () ;
{
    float Irms = emon1.calcIrms(1480);  // Calculate Irms only
    double Irms2 = (Irms + (Irms*0.40)) * sqrt(2);
    return Irms2;
}

void reset()
{
    float Irms = 0;
    float Irms2 = 0;
    double valor = 0;
```

```
        double valor2 = 0;
        Serial.flush();
}
```

Código implementado en Phyton

```
import serial
from time import strftime
from datetime import datetime, time

ser = serial.Serial('/dev/ttyACM0',9600)
startTime = datetime.now()

try:
        while 1:
        x=ser.readline().rstrip();
        now = datetime.now();
        elapsedTime = now-startTime;
        elapsedSeconds =
        ((elapsedTime.days*24*3600+elapsedTime.seconds)*10**6)
        /10**6;
        print("%s;%s"%(now.strftime("%Y-%m-%d;%H:%M:%S"),x)
        except KeyboardInterrupt:
        print "\nGracias"
```

Imágenes

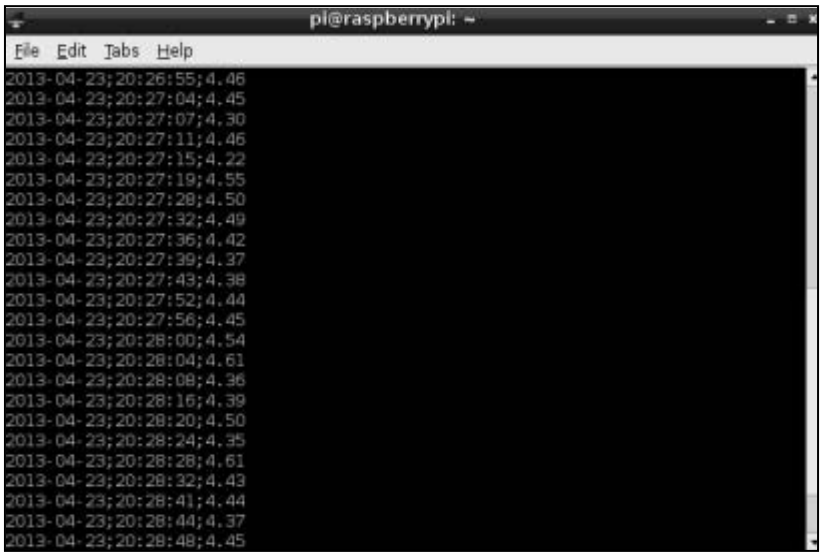

Figura 3.18. Envió de los datos entre el Raspberry Pi y el Arduino 1.0

Conclusiones

Presentar el dato en pantalla no es necesariamente el primer dato que se obtuvo, fue la técnica usada para disminuir drásticamente la aparición de errores o datos basura que podrían aparecer y dañar el muestreo de los datos, debido a que si el usuario quisiera utilizar estos datos para crear una gráfica existirían picos muy elevados de corriente lo cual no es verdad. Desde luego que ahora con el proyecto funcionado pero mostrando los datos de una manera no muy estética, faltaría agregar una manera de

presentación más agradable para el usuario, y es precisamente de lo que se trata los siguientes ejercicios a continuación.

3.2.6 EJERCICIO 6.- Uso del programa Gnuplot para la presentación de los datos de consumo eléctrico.

Descripción

Este ejercicio se realizó con la finalidad de agregar otra vía más para presentar los datos de consumo eléctrico de forma gráfica. Para esto hemos utilizado el programa Gnuplot que dado un archivo .txt en un formato de columnas puede realizar una gráfica en dos dimensiones de cualquier variable que tengamos almacenada.

Herramientas utilizadas

Herramientas de hardware:

- Ver página 69.

Herramientas de software:

- Ver página 69
- Instalar el programa Gnuplot.

Diagrama de Bloques

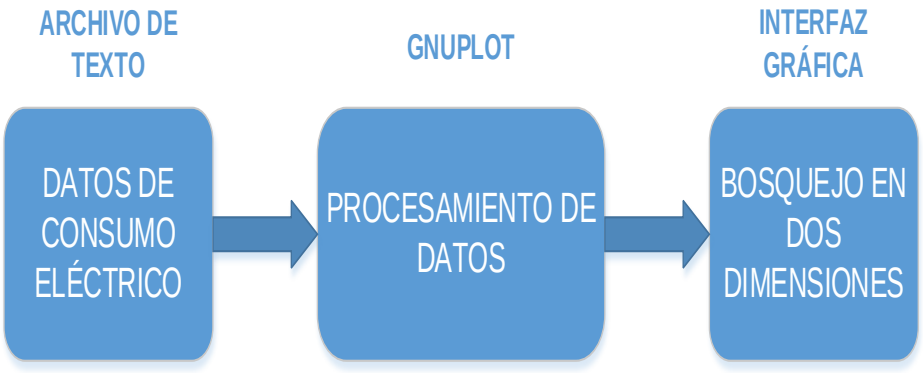

Figura 3.19. Diagrama de Bloques del Ejercicio 6

Diagrama de Flujo

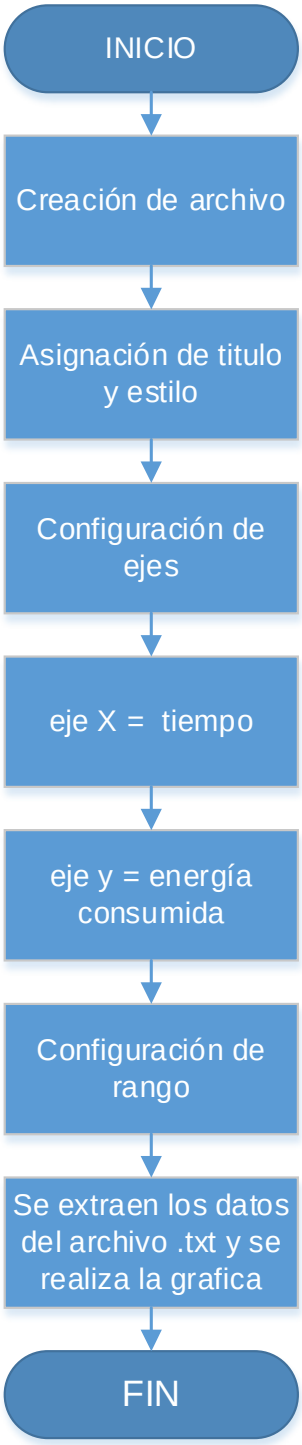

Figura 3.20. Diagrama de flujo del Ejercicio 6

Descripción del algoritmo

1) Se crea un archivo con extensión .gnuplot donde se configuran todos los parámetros necesarios para realizar la gráfica.
2) Se fija la extensión de la gráfica en formato .png.
3) Se asigna un título a la gráfica.
4) Se asigna un estilo determinado a la gráfica mediante el comando set style data fsteps.
5) Se configuran los ejes X y Y por cuadrículas.
6) Se asigna al eje x la variable de tiempo con el formato de horas, minutos y segundos.
7) Se asignan títulos a cada uno de los ejes.
8) Se establece el rango en el cual se quiere graficar.
9) Se indica el nombre del archivo .txt del cual se extraen los datos para realizar la gráfica.

Código fuente

Dentro del archivo .gnuplot configurar los siguientes parámetros.

```
reset
set term png
set title "Proyecto de Graduación"
set style data fsteps
```

```
set grid
set grid xtics
set grid ytics
set ytics 0.5
set xdata time
set timefmt "%H:%M:%S"
set format x "%H:%M:%S"
set xlabel "Tiempo"
set ylabel "Corriente Irms"
set yrange [ 0 : 10 ]
set xrange [ "02:22:03" : "02:23:10" ]
set output "x10.png"
plot "grafica4.txt" u 1:2 with linespoints
```

Se lo ejecuta usando el gnuplot de la siguiente manera

```
gnuplot> load "nombredelarchivo.gnuplot"
```

Imágenes

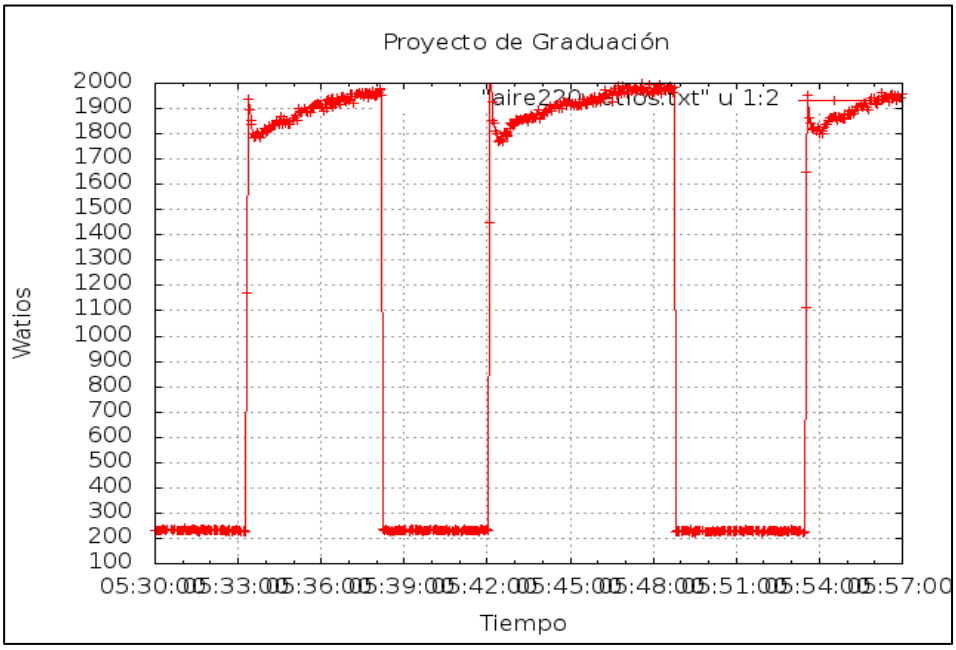

Figura 3.21. Grafica realizada con los datos generados durante un minuto

Conclusiones

Una ventaja que ofrece usar este programa, es que el usuario decide el rango del tiempo del consumo eléctrico, aunque la desventaja es que esto no puede ser ejecutado en tiempo real, es decir la gráfica no se genera automáticamente aun cuando reciba los datos obtenidos por el sensor de corriente. Debido a esto es necesario encontrar una manera de graficar estos valores en tiempo real y esto es la finalidad del ejercicio final.

3.2.7 PROYECTO COMPLETO.- Envío de datos vía streaming en tiempo real a la url del web server Cosm para la presentación de datos de consumo de energía eléctrica y graficación de los mismos en Gnuplot.

Descripción

El siguiente ejercicio corresponde a la finalización total del proyecto, en la cual interactúan todos los dispositivos de hardware y software empleados en el mismo, así como a su vez se utiliza un servidor online para realizar las gráficas de consumo eléctrico en tiempo real utilizando la red de redes ¨el Internet¨.

Primeramente se ha instalado un pequeño circuito compuesto por un divisor de voltaje, que permite centrar en 2.5 V el voltaje de referencia que entrega el sensor de corriente SCT-013-030, para luego ingresar ese valor por uno de los pines analógicos del Arduino 1.0, internamente se procesan los datos calculando el valor de corriente y de consumo de energía eléctrica, para luego enviarlos serialmente vía USB a la minicomputadora Raspberry Pi, y a su vez aprovechar la conectividad a internet que posee la misma para enviar los datos en tiempo real a la url de Cosm, que es un servidor online el cual nos permite graficar la potencia consumida durante un intervalo de tiempo, representándonos así

el consumo eléctrico, a su vez también presentamos la gráfica de corriente vs tiempo.

Herramientas utilizadas

Herramientas de hardware:

- Ver página 69.

Herramientas de software:

- Ver página 69.
- Crear una cuenta en Cosm.com y modificar el programa escrito en Python para que realice la conexión con este servicio web.

Diagrama de Bloques

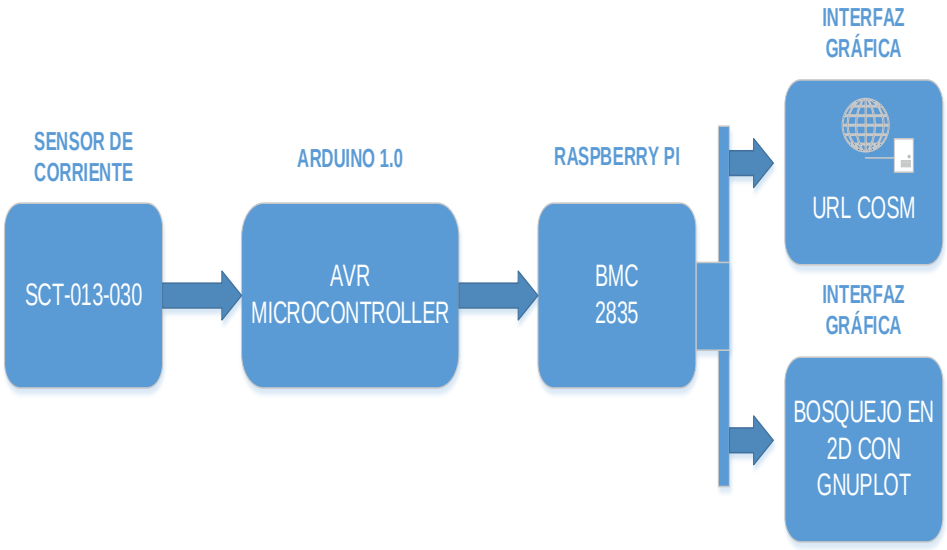

Figura 3.22. Diagrama de bloques del Ejercicio 6

Diagrama de Flujo

El diagrama de flujo realizado para el Arduino IDE en el ejercicio anterior es exactamente el mismo por lo cual solamente hacemos referencia a la figura 3.11.

Descripción del Algoritmo

Código implementado en el Arduino IDE

Ver página 73.

Código implementado en Phyton

1) Se importan las librerías utilizadas en el proyecto que son las siguientes:
    - eeml: permite la conexión con la dirección url cosm.com.
    - serial: activa la comunicación vía usb con el Raspberry Pi.
    - strftime y datetime: librerías de tiempo.
2) Se incluyen las llaves que permiten acceso con la cuenta creada en cosm.com(ver anexos).
3) Se define la variable y que permite el control de la toma de datos cada 5 segundos.
4) Se define la variable ser que permite configurar la tasa de transmisión serial de la comunicación USB en 9600.
5) Se define la variable startTime y se le asigna el tiempo que registra el ordenador.
6) Se crea un lazo infinito mediante el comando while 1.
7) Se asigna a la variable x los datos de consumo eléctrico que están llegando vía USB en la comunicación serial.
8) Se asigna el valor de tiempo que registra el ordenador en ese momento a la variable now.
9) Se asigna la diferencia de tiempos entre las variables now y startTime a la variable elapsedTime.

10) Se imprimen los valores de tiempo, corriente y potencia consumida en el Lx Terminal de la minicomputadora Raspberry Pi.
11) Se crea un archivo con extensión .csv en modo de escritura y se almacenan los datos de tiempo, corriente y potencia consumida.
12) Se crea un archivo con extensión .txt en modo de escritura y se almacenan la hora, minutos y segundos como parte de tiempo, y como datos de consumo eléctrico almacenamos la corriente y la potencia consumida.
13) Se cierran los archivos con extensión .csv y .txt.
14) Se asignan los permisos para la conexión con la url cosm.com a la variable pac.
15) Se envían los datos en tiempo real a la url cosm.com, en el formato indicado de acuerdo a los parámetros establecidos.
16) Se realiza la gráfica en tiempo real mediante el comando pac.put().
17) Se encera la variable x.
18) Se consulta por la variable y, si es igual a 5 se desabilita la comunicación serial, se establece un retardo de 1 segundo y se encera la variable y.

19) El lazo while terminará en el momento que exista una interrupción por teclado.

Código Fuente

Código implementado en el Arduino IDE

Ver página 75.

Código implementado en Phyton

```
import eeml
import serial
from time import strftime
from datetime import datetime, time

API_KEY =
'SctJFL61WhQHt9L8nXmhW1XfjGWSAKxOa0JGSmMwaExqbz0g'
FEED = 124917
API_URL = '/v2/feeds/124917.xml'
y=0
startTime = datetime.now()
try:

    while 1:
```

```
import time
time.sleep(0.1)
y=y+1
ser = serial.Serial('/dev/ttyACM0', 9600)
readings = ser.readline().strip().split(' ')
now = datetime.now()
elapsedTime = now-startTime
elapsedSeconds = ((elapsedTime.days*24*3600+
                   elapsedTime.seconds)*10**6)/10**6
print("%s,%s,%s"%(now.strftime("%Y-%m-%d;%H:%M:
%S"),readings[0],readings[1]))
#f = open('base7.csv','a')
#print>>f,("%s,%s,%s"%(now.strftime("%Y/%m-%d;%H:
%M:%S"),readings[0],readings[1]))
#g = open('grafica7.txt','a')
#print >>g,("%s %s %s"%(now.strftime("%H:%M:%S"),
readings[1]))
#f.close()
#g.close()
pac = eeml.Pachube(API_URL, API_KEY)
pac.update([eeml.Data(0,readings[0], unit=eeml.Celsius
())])

pac.update([eeml.Data(1,readings[1],unit=eeml.Celsius())
])
pac.put()
```

```
            if y==5 :
                ser.close()
                time.sleep(0.1)
                y=0

except KeyboardInterrupt:
    print "\nGracias"
```

Imágenes

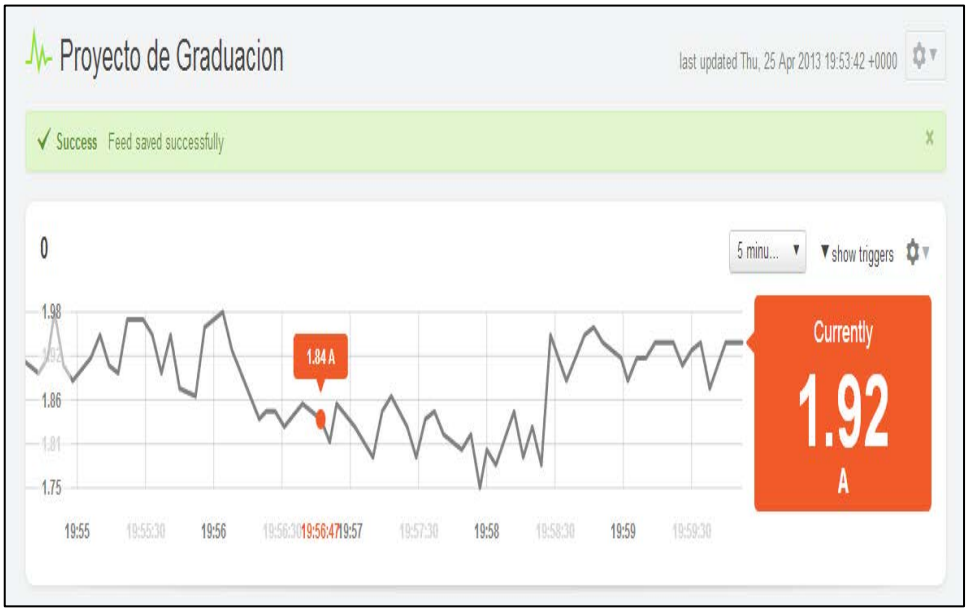

Figura 3.23. Envió de los datos vía streaming al servidor Cosm.com

Conclusiones

Con la incorporación de este servicio web, es posible apreciar de manera real toda la información del consumo eléctrico al equipo al cual se le esté realizando pruebas de medición. Una de las principales ventajas que ofrece haber logrado este ejercicio, es que se puede investigar estos datos ya sea en diferentes puntos del tiempo y en diferentes lugares, ya que la página web es de libre acceso y no hay la necesidad de abrir la página web desde el RaspberryPi.

CAPÍTULO 4

SIMULACION Y PRUEBAS EXPERIMENTALES

4.1 Introducción

Este capítulo se basa en las diferentes pruebas que se realizaron respecto a este proyecto de graduación. Para esto hemos llevado a cabo el estudio y el comportamiento de los datos de consumo de energía de dos dispositivos eléctricos, ambos funcionan con corriente alterna a 220V, el primero es un aire acondicionado de hogar monofásico y el segundo una secadora de ropa casera. Paro ambos dispositivos eléctricos hemos realizado el respectivo análisis de consumo de corriente y de potencia, de tal forma que pueda ser monitoreado remotamente desde cualquier lugar con acceso a internet gracias a la ventaja del envío de los datos vía streaming, en tiempo real al servidor casero Cosm, en el cual se puede visualizar el consumo de energía en un intervalo de tiempo específico.

4.2 Análisis del consumo de energía eléctrica de un aire acondicionado monofásico a 220V

Descripción

Hemos llevado a cabo la adquisición y evaluación de los datos de consumo de energía eléctrica en un aire acondicionado de hogar de 24000 BTU, con un consumo de corriente teórico de máximo 14.5 A. Al no conocer el factor de potencia real del aire acondicionado hemos estimado un valor aproximado de 0.85, este valor es el que utilizan ciertos fabricantes para esta clase de motores. Con ayuda de nuestro proyecto nos permitirá analizar el período de sleep que se produce en ciertos aires acondicionados, que se realiza cuando se ha llegado a la temperatura deseada a través del valor que censa el termostato, en este momento se apaga el compresor y disminuye el consumo de corriente, por lo cual también disminuirá el consumo de energía eléctrica. Esta prueba la realizamos conectando un cable Ethernet directamente al puerto LAN de la tarjeta, con lo cual no hubo ningún problema durante el tiempo que se realizó la captura y presentación de los datos. También realizamos varias gráficas del consumo de energía eléctrica haciendo uso del programa Gnuplot.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 Raspberry Pi.
- 1 aire Acondicionado monofásico a 220V, 14.5A máx.
- 1 sensor de corriente SCT-013-030.
- 1 cable USB-impresora.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 convertidor de HDMI a VGA.
- 1 hub-USB con alimentación propia.
- 1 resistencia de 33 Ω.
- 2 resistencias de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino IDE que es donde se realiza la programación para el Arduino 1.0.
- Instalar la librería Emonlib.h para el Arduino 1.0.
- Python versión 2.7 o superior.
- Instalar las librerías de py-serial y Python-serial en el Raspberry Pi.

- Crear una cuenta en Cosm.com y modificar el programa escrito en Python para que realice la conexión con este servicio web.

Desarrollo

Realizamos la conexión y el cableado de todos los elementos de hardware por los cuales está conformado el proyecto como se puede apreciar en la figura 4.1. Se observa parte de la conexión destacando el circuito eléctrico implementado en el protoboard de color rojo, además del Raspberry Pi (tarjeta de color verde) y el Arduino 1.0 (tarjeta de color azul), también se aprecia el convertidor de HDMI a VGA (color blanco).

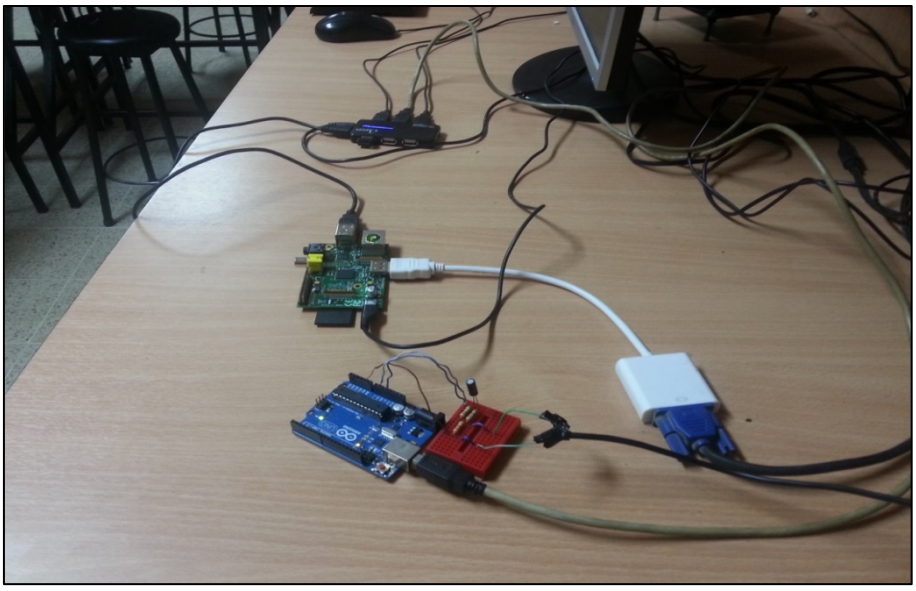

Figura 4.1. Conexión y cableado de los elementos de hardware del proyecto – aire acondicionado 220 V

Tuvimos que realizar un pequeño corte en el cable de alimentación del aire acondicionado para poder separar los cables de línea viva y el neutro que se encontraban dentro del recubrimiento del mismo como se observa en la figura 4.2 y esto debido a que el sensor de corriente SCT-013-030 que utilizamos en nuestro proyecto solamente censa los niveles de corriente que circulan por uno de los cables de línea viva. Por el otro cable de línea viva circula la misma corriente siempre y cuando la carga se encuentre balanceada, mientras que por el neutro no circula corriente, aunque dependiendo de la instalación eléctrica pueden presentarse corrientes parásitas de muy baja magnitud.

Figura 4.2. Sensor SCT-013-030 de color azul y amperímetro de gancho

Una vez realizada todo el cableado y de haber colocado el sensor de corriente en uno de los cables de línea viva se procede a poner en marcha el proyecto para de esta forma poder monitorear el consumo de energía del aire acondicionado remotamente desde cualquier computadora o dispositivo con acceso a internet.

Podemos observar en la figura 4.3 algunas de las características del aire acondicionado especificadas por el fabricante como por ejemplo la fase, el voltaje de entrada, la frecuencia de operación, la corriente máxima, la potencia que consume, el tipo de refrigerante, etc. Todos estos valores son teóricos y esto lo mencionamos porque debido a los años que tiene el aire acondicionado hace que el aire acondicionado ya no entregue la misma capacidad que cuando era nuevo.

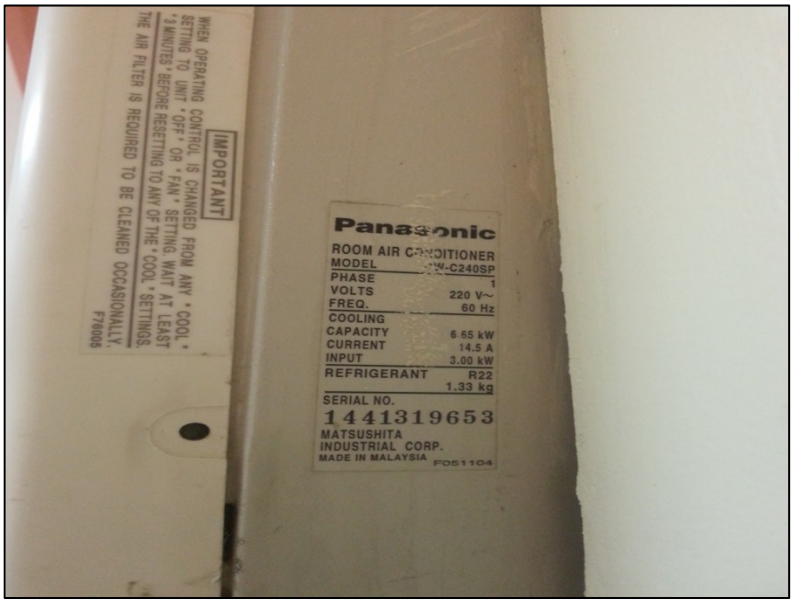

Figura 4.3. Características del aire acondicionado especificadas por el fabricante

Podemos observar en la figura 4.4 la adquisición de los datos de consumo de energía eléctrica del aire acondicionado que se actualizan cada tres segundos aproximadamente en el LX Terminal del Raspberry Pi, en el formato de fecha, hora, minutos, segundos, corriente y potencia consumida.

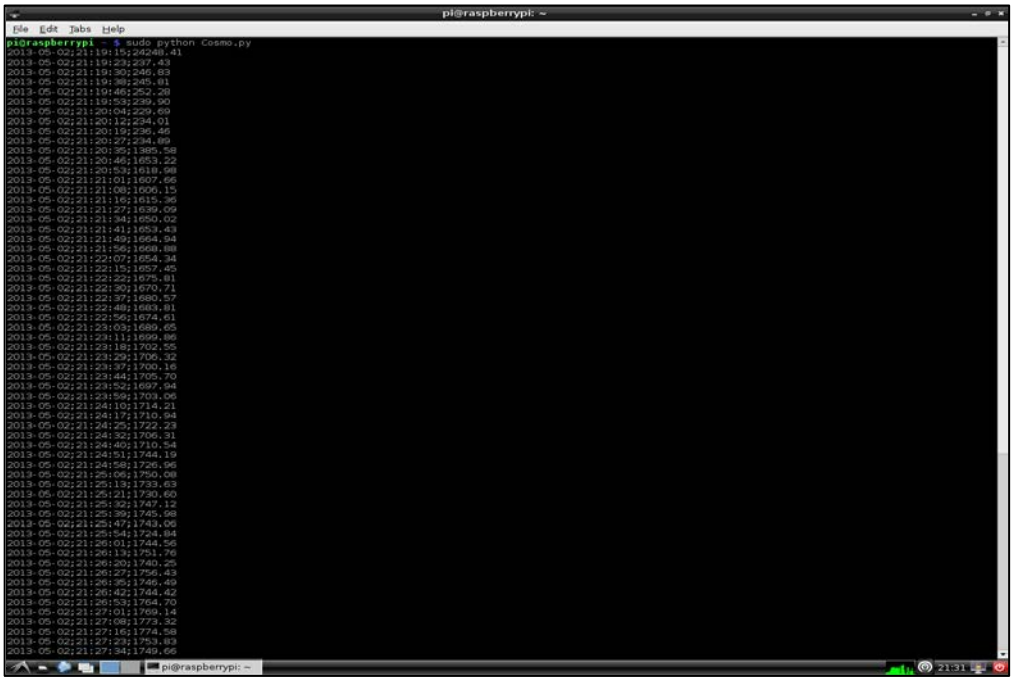

Figura 4.4. Presentación de los datos de consumo de energía eléctrica en el LX Terminal del Raspberry Pi

Todos estos datos se están enviando vía streaming a una computadora personal, desde la cual estamos monitoreando remotamente el consumo de energía eléctrica del aire acondicionado a través del web server casero Cosm, accediendo a nuestra cuenta a través de un usuario y contraseña.

En las figuras 4.5, 4.6 y 4.7 podemos ver las gráficas de corriente y potencia consumida durante los últimos cinco minutos, media hora y una hora, donde se puede apreciar que la corriente máxima durante ese

intervalo de tiempo es de 10.95 A y la mínima es de 1.19 A, así mismo la amplitud máxima de potencia consumida es de 2047 Watts, mientras que la mínima es de 222 Watts. Nos preguntaremos a que se debe que exista un máximo y un mínimo de corriente y potencia consumida, y no sea constante como esperamos, la respuesta es sencilla debido a que ciertos aires acondicionados tienen un período de sleep en el cual su compresor se apaga trabajando al mínimo posible cuando el termostato censa que ha llegado a la temperatura indicada.

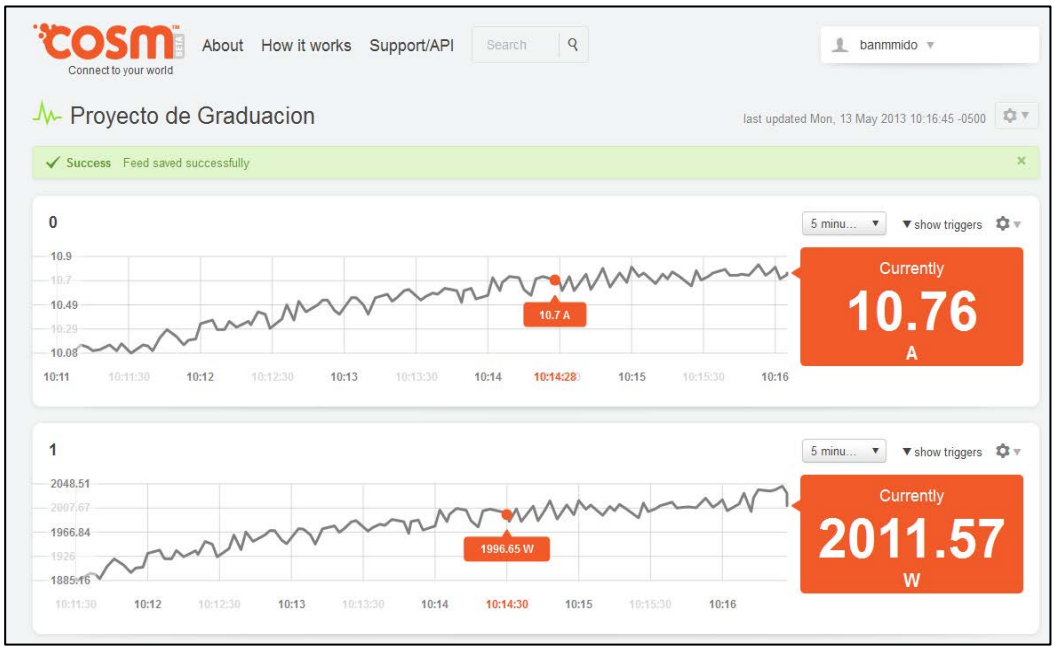

Figura 4.5. Gráfica del consumo de energía eléctrica del aire acondicionado en Cosm durante los últimos 5 minutos

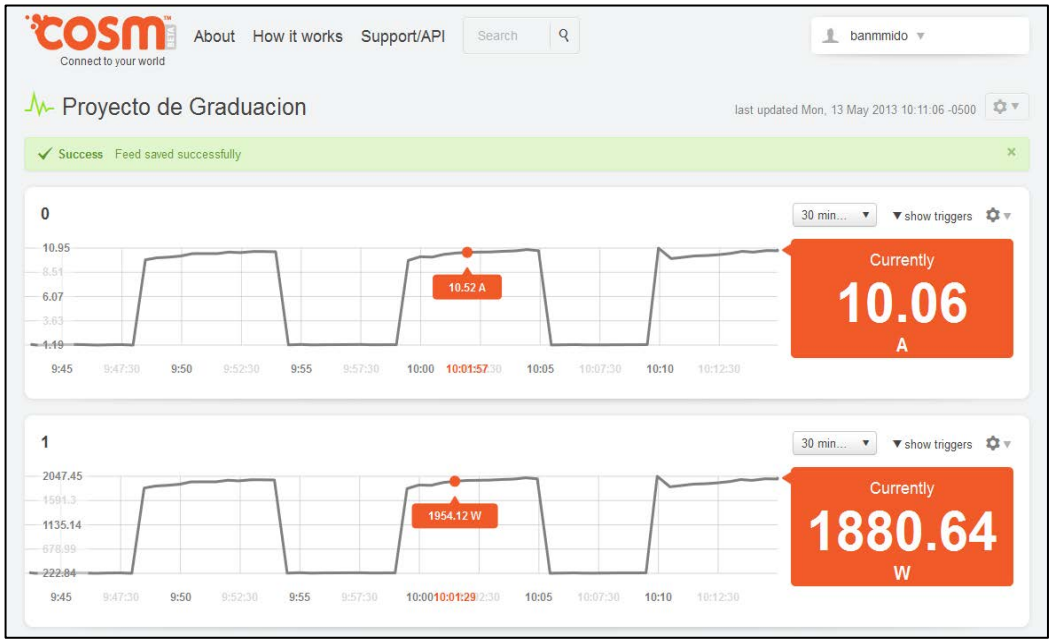

Figura 4.6. Gráfica del consumo de energía eléctrica del aire acondicionado en Cosm durante los últimos 30 minutos

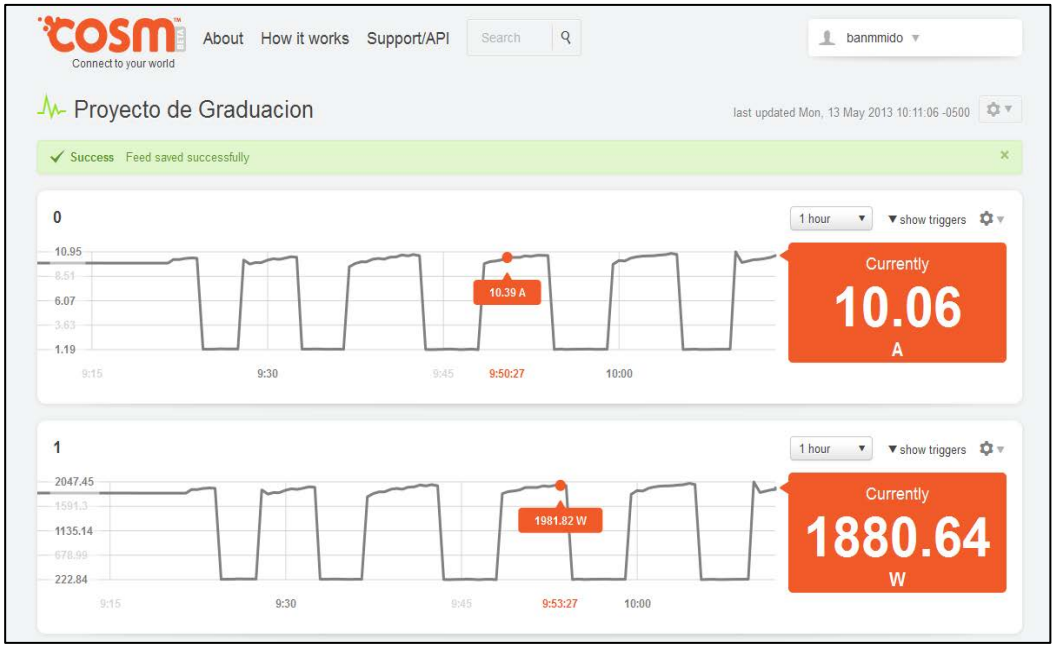

Figura 4.7. Gráfica del consumo de energía eléctrica del aire acondicionado en Cosm durante la última hora

Tanto la gráfica de corriente y potencia consumida tienen la misma tendencia debido a que son directamente proporcionales al permanecer el voltaje y el factor de potencia constantes.

Si queremos saber el costo del consumo de energía del aire acondicionado simplemente procedemos a sacar el área bajo la curva durante un intervalo de tiempo específico y lo multiplicamos el valor comercial al que se encuentre regulado el kilowatt/hora en el país.

Estos datos de consumo de energía a la vez que se envían vía streaming para la presentación de los mismos en Cosm, también se están guardando en un archivo con extensión .csv y en un archivo con extensión .txt en el formato indicado.

Con los datos que se almacenaron en el archivo .csv podemos guardarlos en el servidor phpMyAdmin con el propósito de no perder esta información para poder ser utilizada en cualquier momento.

Con los datos que se almacenaron en el archivo .txt y haciendo uso del programa Gnuplot podemos realizar gráficas del consumo de energía

eléctrica del aire acondicionado en un intervalo de tiempo específico el cual lo configuramos en los parámetros del programa Gnuplot.

En la figura 4.8 se puede apreciar que el consumo de corriente bordeo los 10.5 A durante el último minuto. El eje vertical representa a la variable de corriente, mientras que el eje horizontal representa a la variable de tiempo.

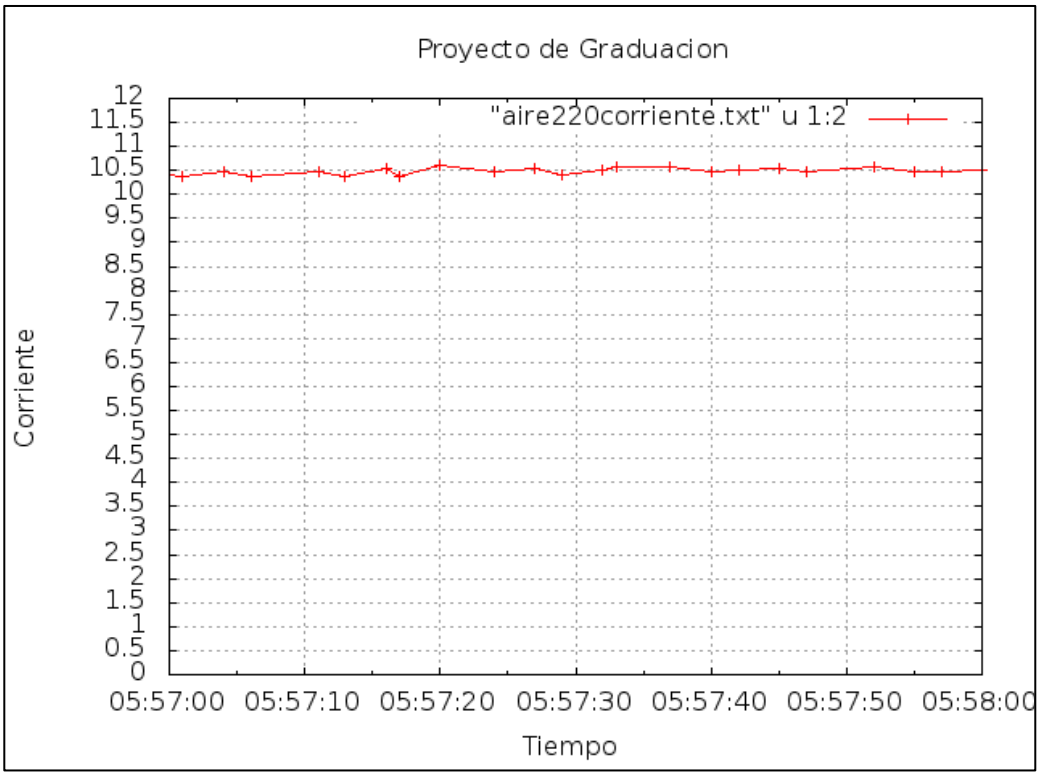

Figura 4.8. Gráfica de corriente del aire acondicionado a 220 V realizada en Gnuplot durante 1 minuto

En la figura 4.9 podemos observar el período de sleep que se produce en el aire acondicionado al alcanzar la temperatura indicada, registrandose un valor de corriente de 10.5 A aproximadamente cuando esta trabajadando el compresor, mientras que en período de sleep registra un valor de corriente de 1 A aproximadamente.

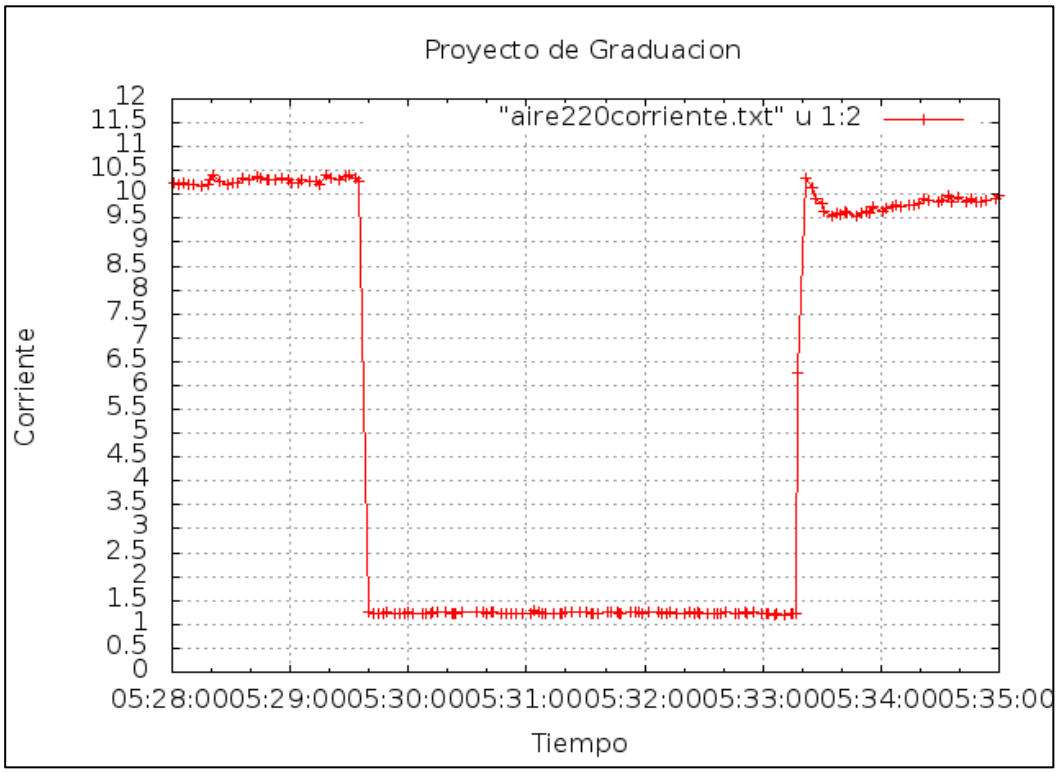

Figura 4.9. Gráfica de corriente del aire acondicionado a 220 V realizada en Gnuplot durante 7 minutos

Ya en la figura 4.10 podemos apreciar una gráfica un poco más completa de cómo se comporta el consumo de corriente del aire acondicionado en

un intervalo de tiempo más largo de unos treinta minutos aproximadamente. Se puede diferenciar que el tiempo que permanece trabajando el compresor es de unos seis minutos aproximadamente y el tiempo en que permanece apagado es de unos cuatro minutos aproximadamente.

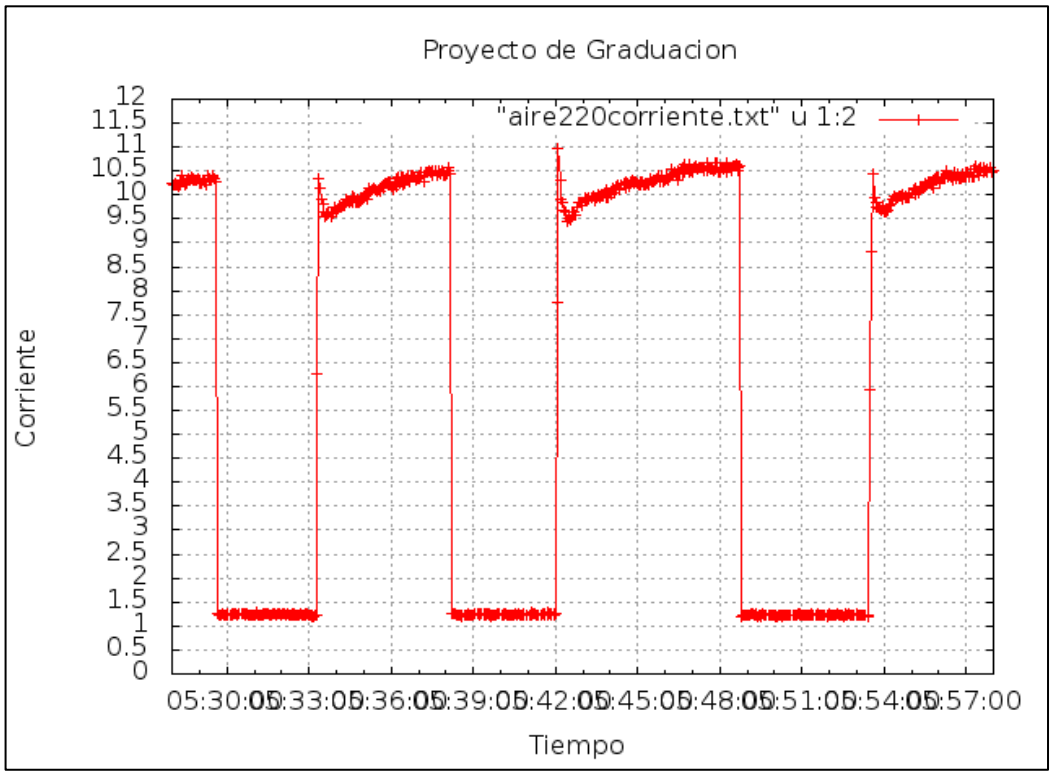

Figura 4.10. Gráfica de corriente del aire acondicionado a 220 V realizada en Gnuplot durante los últimos 30 minutos

En las figuras 4.11, 4.12 y 4.13 observamos las gráficas de potencia consumida en los mismos intervalos de tiempo que para las gráficas de

corriente, como dijimos anteriormente la tendencia de las gráficas son similares a las de corriente y el análisis que se realizó previamente es el mismo para estas gráficas.

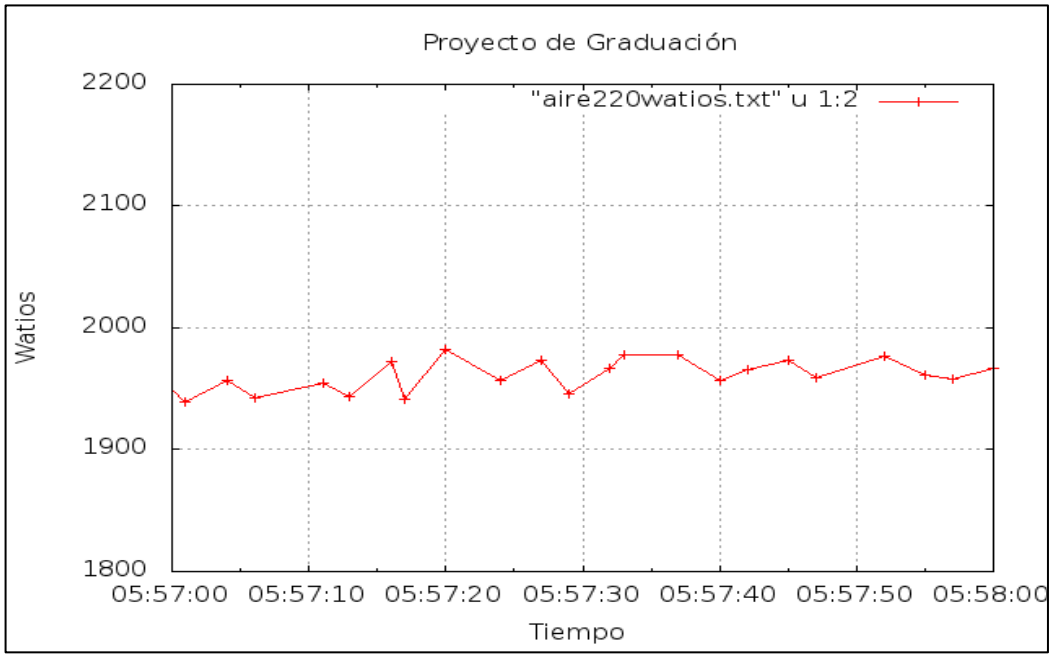

Figura 4.11. Gráfica de potencia consumida del aire acondicionado a 220 V realizada en Gnuplot durante 1 minuto

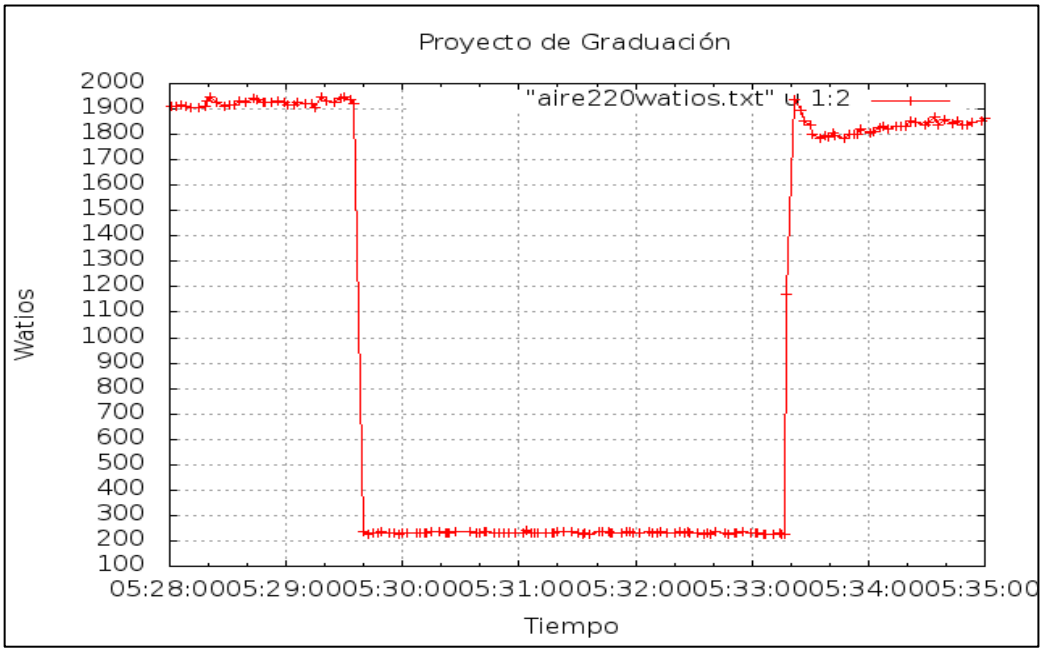

Figura 4.12. Gráfica de potencia consumida del aire acondicionado a 220 V realizada en Gnuplot durante 7 minutos

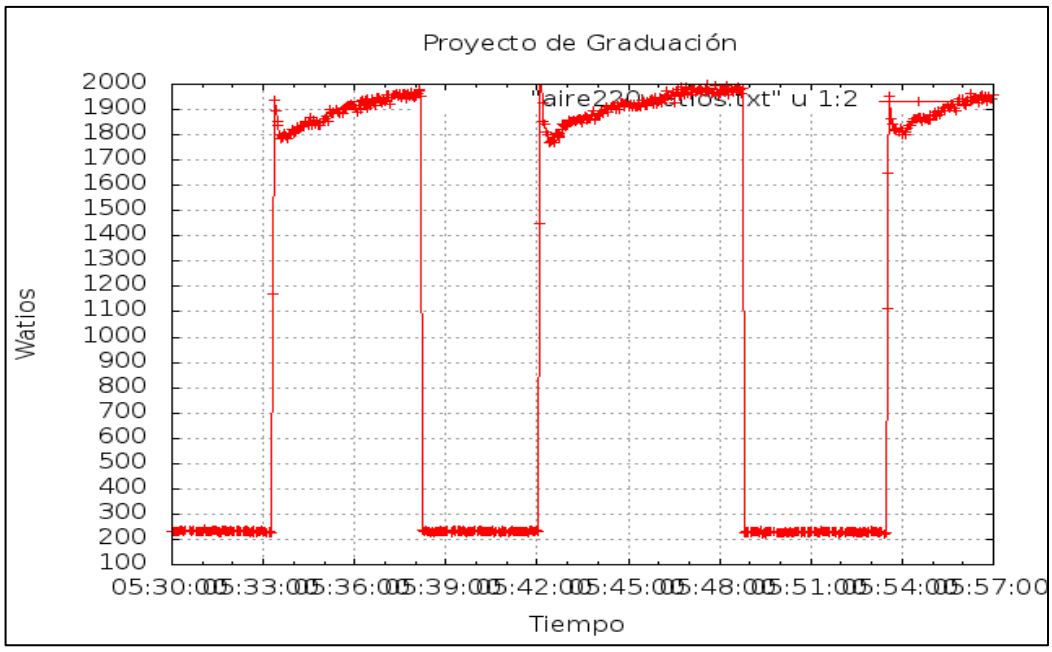

Figura 4.13. Gráfica de potencia consumida del aire acondicionado a 220 V realizada en Gnuplot durante 30 minutos

Vale recalcar que este proyecto puede funcionar de manera continua con la adquisición y presentación de los datos en tiempo real siempre y cuando exista una perfecta conexión a internet sin ningún corte, al igual que la tarjeta Raspberry Pi no se sobrecaliente o exista alguna falla en el sistema operativo, cualquiera de estas dos causas harán que se paralice la evaluación de datos y se requiera de supervisión.

4.3 Análisis del consumo de energía eléctrica de una secadora de ropa a 220V

Descripción

Se llevó a cabo también la captura y procesamiento de los datos de consumo de energía eléctrica para una secadora de ropa a 220V durante un ciclo de secado de 60 minutos. Para esta prueba también se consideró un factor de potencia de 0.85, la corriente teórica máxima que podía entregar la secadora era de 20A. Se monitoreo remotamente los datos de corriente y potencia, visualizándolos gráficamente en el servidor Cosm. Para este caso conectamos inalámbricamente la tarjeta Raspberry Pi, al contrario del primer ejercicio, realizando esto para experimentar el correcto funcionamiento de la tarjeta Raspberry Pi debido a las limitaciones q esta posee, sin embargo nos dimos cuenta que el conectar muchos dispositivos al hub-USB que está directamente conectado a uno de los puertos USB de la tarjeta hace que dejen de funcionar otros

dispositivos por la sobrecarga de la misma. Igual que con la prueba del aire acondicionado también se realizaron varias gráficas en diferentes intervalos de tiempo haciendo uso del programa Gnuplot.

Herramientas utilizadas

Herramientas de hardware:

- 1 Arduino 1.0.
- 1 Raspberry Pi.
- 1 Secadora de ropa a 220V, 20A máx.
- 1 Sensor de corriente SCT-013-030.
- 1 Cable USB-impresora.
- 1 cargador micro-USB con 0.7 A de salida como mínimo.
- 1 teclado USB.
- 1 mouse USB.
- 1 convertidor de HDMI a VGA.
- 1 hub-USB con alimentación propia.
- 1 resistencia de 33 Ω.
- 2 resistencias de 1kΩ.
- Cables de conexión.

Herramientas de software:

- Instalar el Arduino IDE que es donde se realiza la programación para el Arduino 1.0.
- Instalar la librería Emonlib.h para el Arduino 1.0.
- Python versión 2.7 o superior.
- Instalar las librerías de py-serial y python-serial en el Raspberry Pi.
- Crear una cuenta en Cosm.com y modificar el programa escrito en Python para que realice la conexión con este servicio web.

Desarrollo

Trabajando ahora con una secadora de ropa casera con un consumo mayor al del aire acondicionado realizamos la conexión y cableado de todos los elementos de hardware del proyecto como se ilustra en la figura 4.14.

Figura 4.14. Conexión y cableado de los elementos de hardware del proyecto – secadora 220 V

En la figura 4.15 podemos apreciar el conector tipo hembra para dispositivos a 220 V como en este caso es la secadora de ropa.

Figura 4.15. Conector hembra para dispositivos a 220 V

Para esta prueba tuvimos que realizar una separación en el cable de alimentación de la secadora de ropa como se observa en la figura 4.16, para así poder adaptar el sensor de corriente SCT-013-030.

Figura 4.16. Adaptación del sensor de corriente SCT-013-030 al cable de alimentación de la secadora de ropa a 220 V

Dejamos funcionando el proyecto con la secadora configurada para un ciclo de trabajo de 60 minutos y a la vez monitoreando remotamente el consumo de energía eléctrica desde nuestra computadora personal como se observa en la figura 4.17.

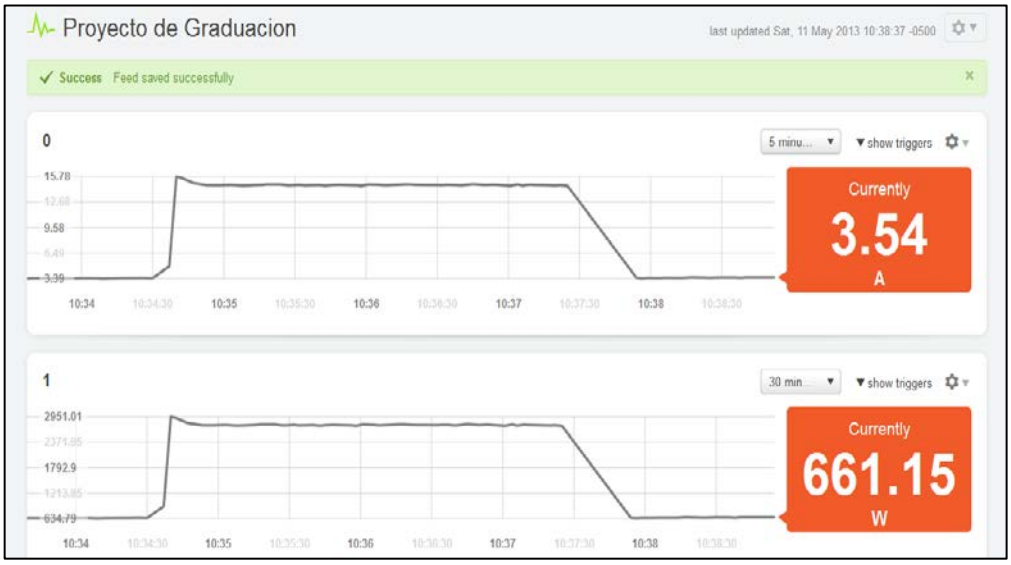

Figura 4.17. Gráfica del consumo de energía eléctrica de la secadora de ropa en Cosm durante 5 minutos

Como podemos apreciar en la figura 4.17 durante los últimos cinco minutos la corriente máxima fue de 15.78 A y la mínima de 3.39 A. Al igual que el aire acondicionado la secadora de ropa tiene un período de tiempo en donde su consumo de energía disminuye. Así mismo sucede con la potencia consumida, la máxima bordea los 2950 Watts y la mínima los 630 Watts.

En las figuras 4.18, 4.17 y 4.19 podemos observar el consumo de corriente de la secadora de ropa en diferentes intervalos de tiempo utilizando el programa Gnuplot para la construcción de gráficas en 2D. Se puede apreciar que los valores son muy similares a los obtenidos en

Cosm y esto es porque los valores que se envían a Cosm y los que se guardan en el archivo con extensión .txt son los mismos, lo único diferente es el método de graficación.

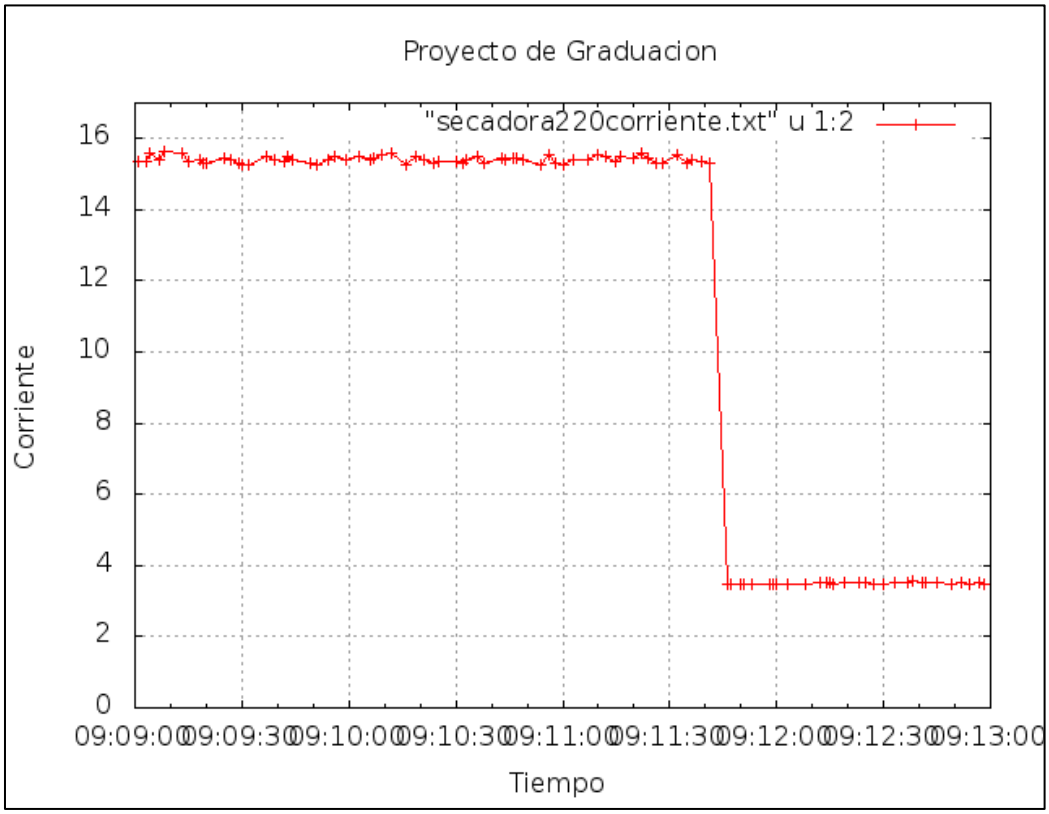

Figura 4.18. Gráfica de corriente de la secadora de ropa a 220 V realizada en Gnuplot durante 4 minutos

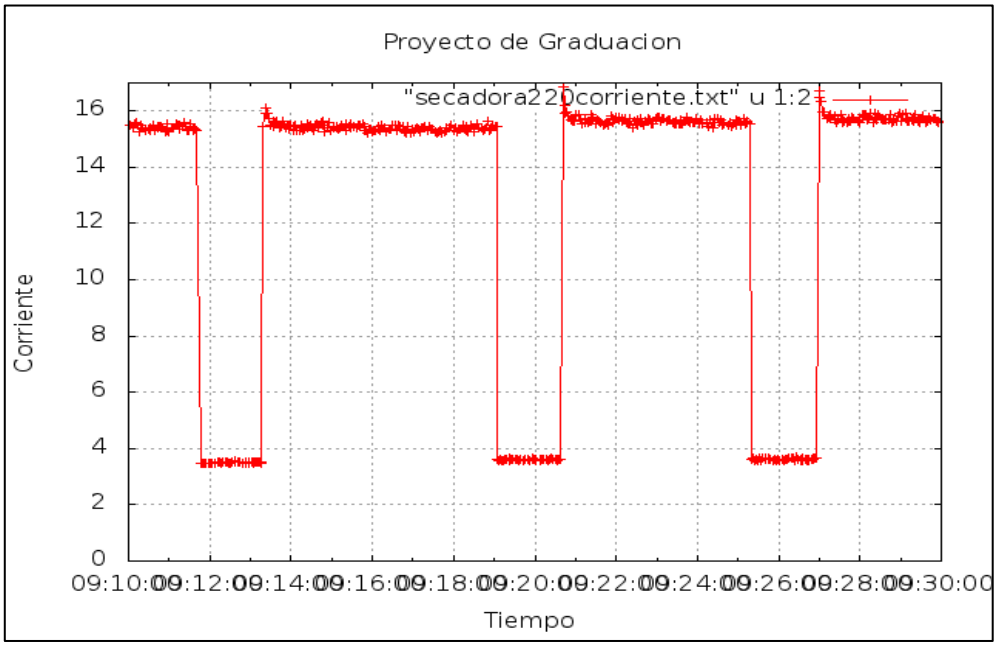

Figura 4.19. Gráfica de corriente de la secadora de ropa a 220 V realizada en Gnuplot durante 20 minutos

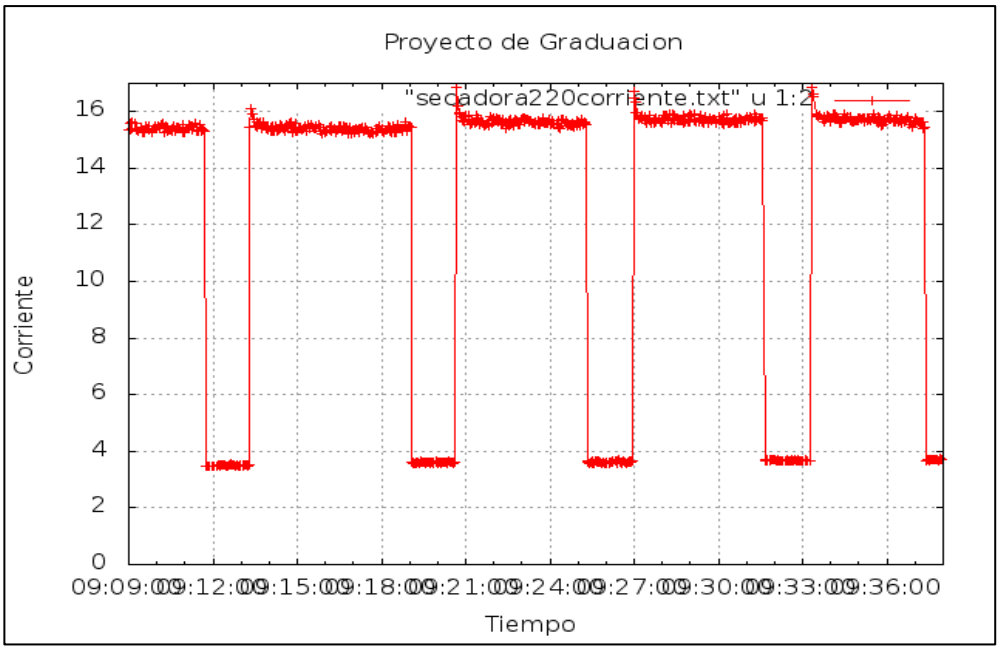

Figura 4.20. Gráfica de corriente de la secadora de ropa a 220 V realizada en Gnuplot durante 30 minutos

En la figura 4.20 se puede observar que la corriente máxima bordea los 15.5 A, mientras que cuando baja su consumo de energía registra un valor cercano a los 3.8 A.

Debido a que la escala de tiempo se encuentra en el formato de horas, minutos y segundos hace que se traslapen los valores, pero lo importante es que la gráfica se encuentra correctamente bosquejada.

Finalmente presentamos las gráficas de potencia consumida para la secadora de ropa a 220 V como se puede apreciar en las figuras 4.21, 4.22 y 4.23 realizadas en el programa Gnuplot.

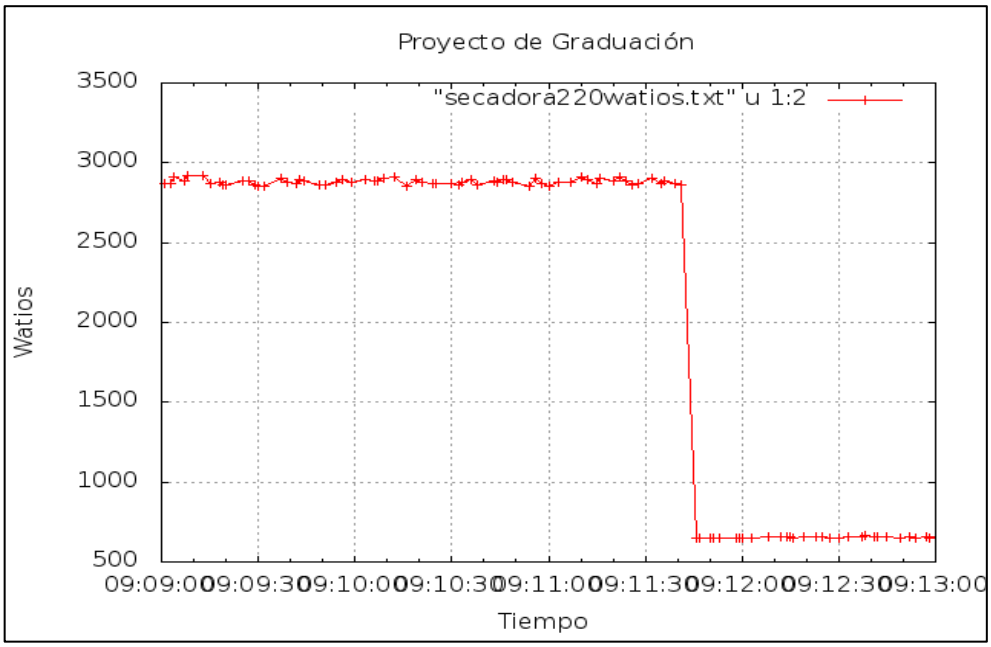

Figura 4.21. Gráfica de potencia consumida de la secadora de ropa a 220 V realizada en Gnuplot durante 4 minutos

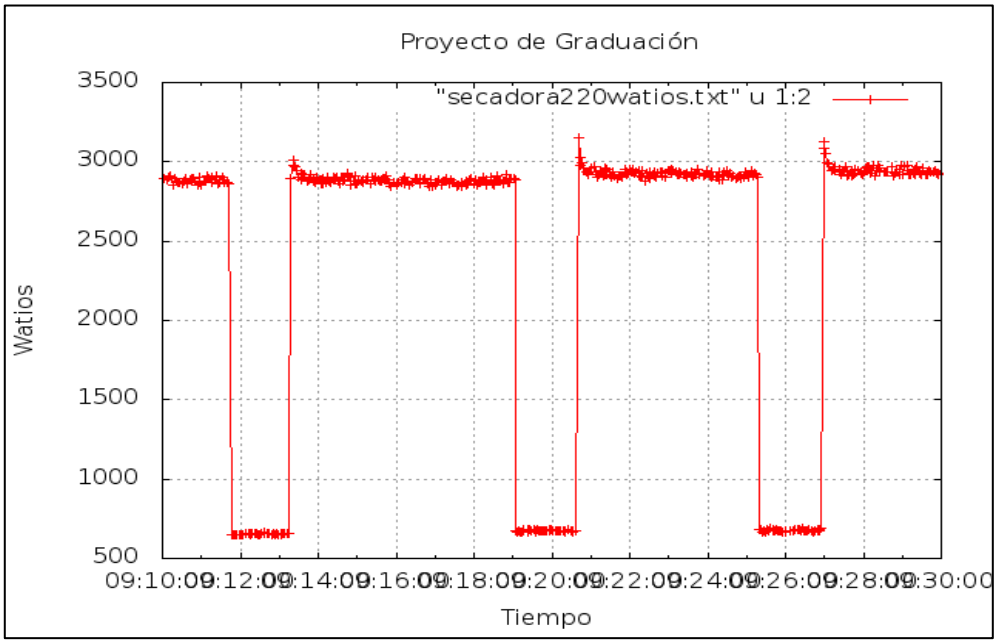

Figura 4.22. Gráfica de potencia consumida de la secadora de ropa a 220 V realizada en Gnuplot durante 20 minutos

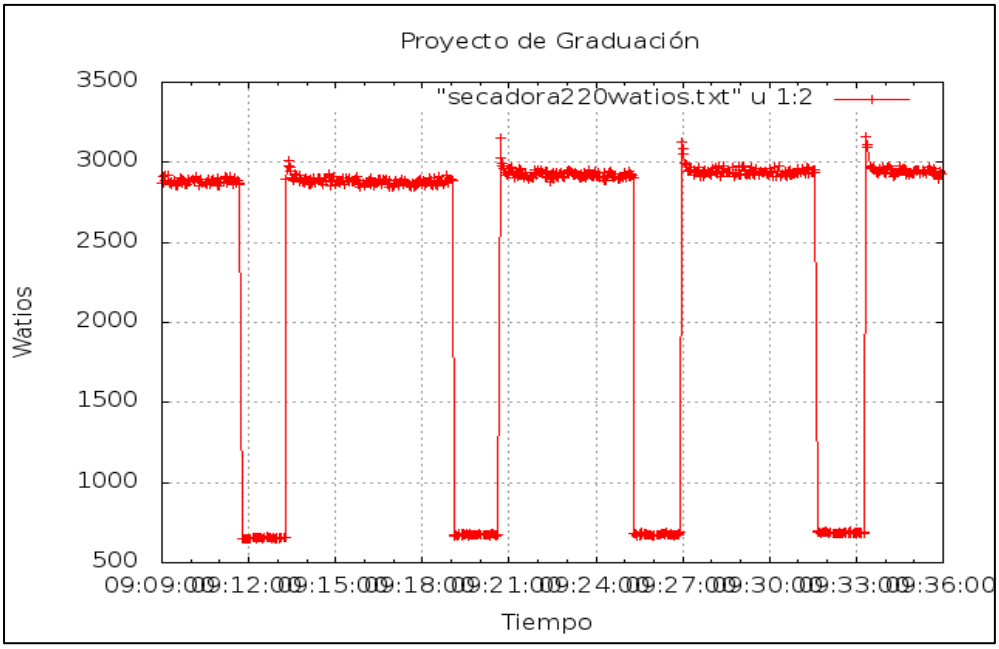

Figura 4.23. Gráfica de potencia consumida de la secadora de ropa a 220 V realizada en Gnuplot durante 30 minutos

CONCLUSIONES

De acuerdo a la investigación, desarrollo e implementación de este proyecto, como a su vez de los distintos problemas que hubo que superar para la finalización del mismo, los autores de esta obra podemos concluir lo siguiente:

1) El Raspberry Pi es una microcomputadora la cual a pesar de sus limitaciones es perfectamente capaz de realizar la mayoría de las tareas de cualquier ordenador actual, siempre y cuando el usuario que la posea le dé el mantenimiento y desarrollo necesario. Esto se debe a que la mayoría de actualizaciones, programas y software necesarios son open source, y existe una vasta cantidad de información acerca de ellos en foros por toda la red, lo cual le permitiría al usuario desarrollar más su Raspberry Pi con un poco de esfuerzo y dedicación.

2) Sin embargo si hablamos de open source, necesariamente debemos mencionar al Arduino 1.0, el cual fue clave en la implementación de este proyecto, ya que este posee más tiempo en el mercado y debido precisamente a eso se han desarrollado códigos de programación que permiten comunicarse ya sea mediante el uso del puerto serial (Rx, Tx) o por el puerto USB con otro dispositivo externo, y es ahí donde está la participación del Raspberry Pi. Cabe recalcar que este tipo de comunicación no es posible lograr sino se posee los drivers adecuados para dicho fin, de hecho depende mucho de que programa se esté utilizando en el Raspberry Pi para lograr la comunicación con el Arduino IDE.

3) Una de las principales razones por la cual se escogió como plataforma de programación al Python 2.7, es que ya existen librerías adecuadas para la comunicación USB entre el Raspberry Pi y el Arduino 1.0. Es importante destacar que dicha comunicación es factible en una versión inferior a la 3.0 del programa Python, la razón es que las librerías Python-serial y py-serial, que fueron utilizadas en el desarrollo del proyecto, fueron creadas en base a una versión del Python anterior a la 3.0. Además al intercomunicar ambos hardware (Raspberry Pi y Arduino 1.0) mediante un cable USB la alimentación necesaria que

requiere el Arduino 1.0 para su funcionamiento se provee a su vez por este medio físico.

4) Durante la toma de datos de consumo eléctrico, es necesario refrescar la comunicación USB entre ambos equipos, sin mencionar que es de vital importancia tomar en cuenta los retardos de transmisión y recepción de datos. Esto con la finalidad de evitar presentar datos extraños o erróneos los cuales darán una gráfica errónea con picos de corriente inexistentes que serán trasmitidos vía streaming a la página Cosm.com que podrían dar mal interpretaciones del consumo de cierto aparato eléctrico que se esté analizando .

5) El web server Cosm.com posee una deficiencia a la hora de presentar los datos, esta es que el rango de almacenamiento de los datos no es muy flexible en cuanto al rango de tiempo. Por lo tanto, el web server casero phpMyAdmin se instaló en el Raspberry Pi con el objetivo de preservar toda la información del proyecto y que esta fuera utilizada en un uso posterior para otro análisis de estudio más detallado.

RECOMENDACIONES

1) Debido a los limitados puertos USB que posee el Raspberry Pi, el usuario solo podría conectar el teclado y mouse necesarios, limitando los otros dispositivos USB que existen en el mercado actualmente como por ejemplo un pendrive. Para solucionar este inconveniente los autores recomiendan adquirir un hub con varias entradas USB pero que a su vez posea una fuente de alimentación externa, esto con la finalidad de evitar que el Raspberry Pi no necesite *alimentar* a estos dispositivos sino que solo reciba la comunicación de los mismos.

2) Si el usuario desea utilizar la salida HDMI del Raspberry Pi por mucho tiempo, y además de eso la utiliza como reproductor de media center este se sobrecalentará y podría afectar la vida útil del Raspberry Pi. Por lo tanto lo ideal sería mantenerlo en constante ventilación para evitar sobrecalentamientos innecesarios.

3) Debido a la diferencia de voltajes que pueden soportar el Raspberry Pi y el Arduino, para lograr la comunicación serial entre ellos se recomendaría la opción por USB. Debido a que esta ofrece mayor seguridad en cuanto a que no sobrepasa valores de voltaje que podrían dejar inservible al Raspberry Pi, pero acarreando ciertos inconvenientes a la hora de transmitir y recibir datos por posible congestionamiento del buffer. Por lo cual los autores aconsejan cada cierto tiempo limpiar el buffer para evitar errores en la transmisión y recepción de datos.

ANEXO 1

Instalación del Sistema Operativo Raspbian en el Raspberry Pi.

Para llevar a cabo la instalación de este sistema operativo se necesita del programa **Win32DiskImager** así como de la imagen del sistema operativo que será copiada dentro de la SD que será utilizada como el disco duro del Raspberry Pi.

Una vez instalado el Win32DiskImager se escoge la unidad donde se encuentra alojada la SD para quemar el sistema operativo en ella, como se muestran en la figura A-1 y A-2.

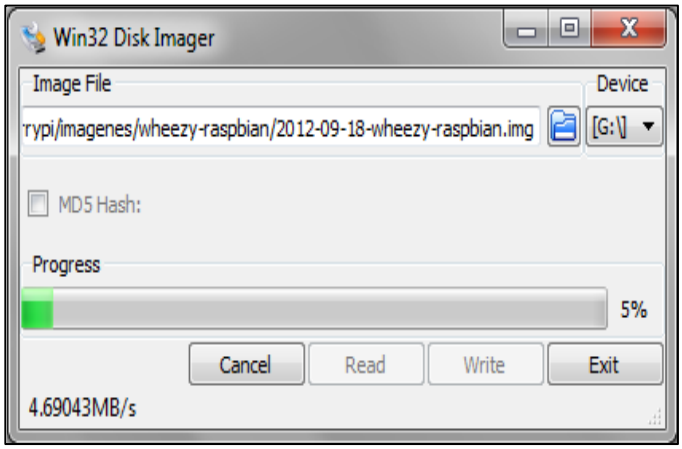

Figura A-1. Se aprecia el proceso de copiado de la SD

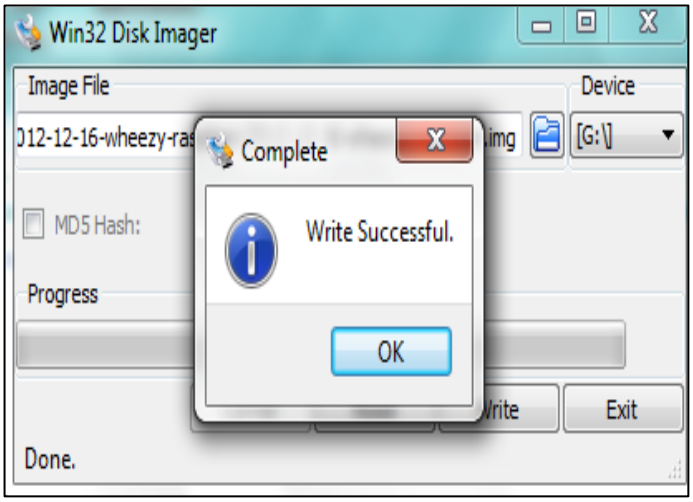

Figura A-2. Se aprecia la escritura exitosa de la SD

Luego de esto, se procede a insertar la SD y junto con el teclado y mouse estamos listos para encender el Raspberry Pi por primera vez.

Al encender el Raspberry Pi se puede apreciar una pantalla como la que se muestras en la figura A-3. Es necesario realizar una configuración del sistema operativo una vez instalado este, con el objetivo de definir parámetros de configuración como el uso de la memoria, la clase de teclado, la distribución adecuada entre la memoria RAM y GPU entre otras cosas.

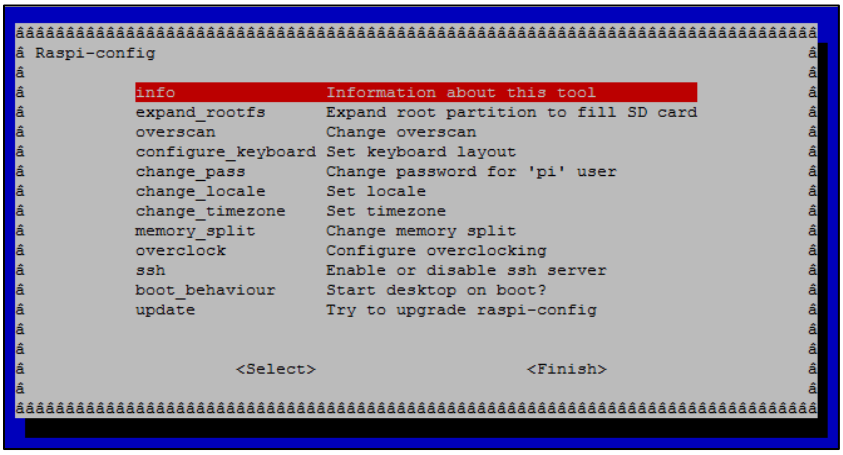

Figura A-3. Configuración de parámetros del sistema operativo pos instalación

Entre las diferentes opciones a configurar se destacan las más importantes a continuación:

expand_rootfs - para que el sistema de ficheros de nuestra distribución de Linux utilice todo el espacio que permita nuestra tarjeta SD.

configure_keyboard - para elegir la configuración de teclado (en mi caso, uno "Genérico de 105 teclas", que aparece como "Generic 105 key") y su distribución (se propone "English (UK)", pero deberíamos escoger "Other" y luego "Spanish"; se nos preguntará ciertas variantes, como el teclado de Apple Macintosh, o el catalán, pero generalmente bastará con dejar simplemente "Spanish"). Finalmente se nos preguntará también cómo queremos que se comporte la tecla AltGr (se puede dejar con su valor por

defecto) y si queremos que la combinación Ctl+Alt+Borrar pueda cerrar el entorno gráfico, por si el entorno gráfico llegase a bloquearse.

memory_split - para elegir el reparto entre memoria principal y memoria de video.

boot_behaviour - para indicar el comportamiento de arranque: si queremos que comience con entorno gráfico o con una consola en modo texto.

Luego de configurar a gusto del usuario estos parámetros, el sistema procederá a solicitar un reinicio del Raspberry Pi como se muestra en la figura A-4.

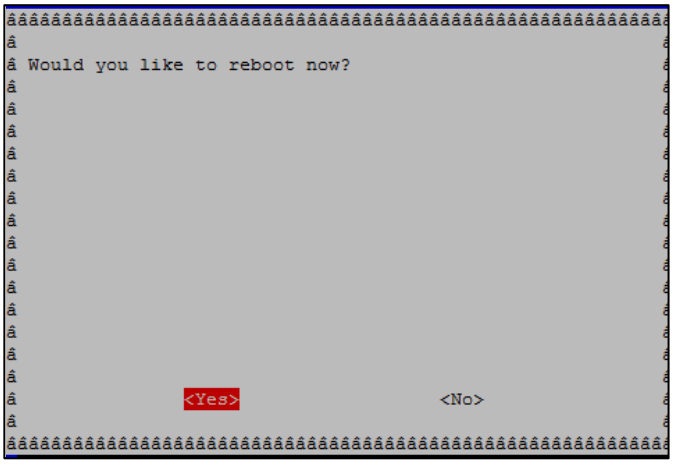

Figura A-4. Mensaje de solicitud de reinicio para guardar los cambios realizados

Finalmente al encender por segunda vez el Raspberry Pi deberá aparecer en pantalla un fondo de pantalla como el que se muestras en la figura A-5, claro está que eso sucederá siempre y cuando el lector haya seleccionado el entorno modo gráfico en la configuración anterior.

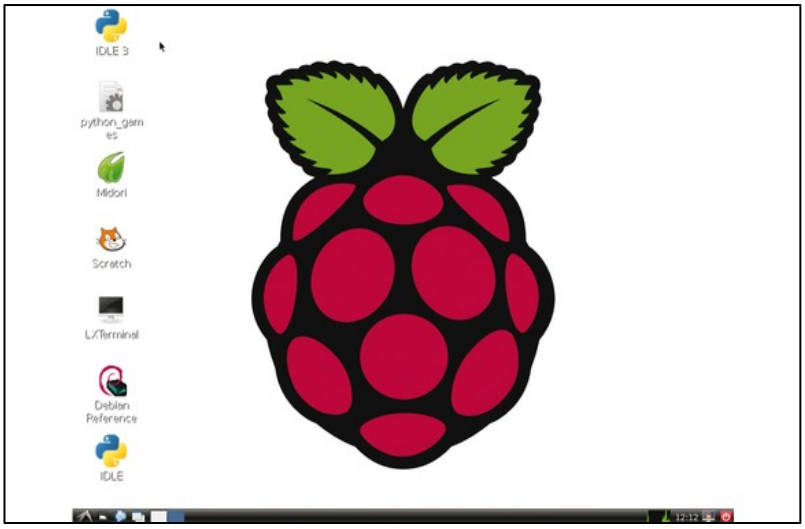

Figura A-5. Entorno gráfico inicial del Raspberry Pi

ANEXO 2

Conectando el Raspberry Pi al Internet

Para poder acceder al internet desde el Raspberry Pi, a esta se le asigna una dirección IP estática utilizando las siguientes líneas de comando en el Lx Terminal:

- sudo cp /etc/network/interfaces /etc/network/interfaces.old
- sudo chmod 666 /etc/network/interfaces

Luego de esto se procede a cambiar el archivo interfaces alojado en la dirección */etc/network*, de tal manera que aquí escribiremos la dirección ip, la máscara de red y el Gateway como se muestra en la figura A-6.

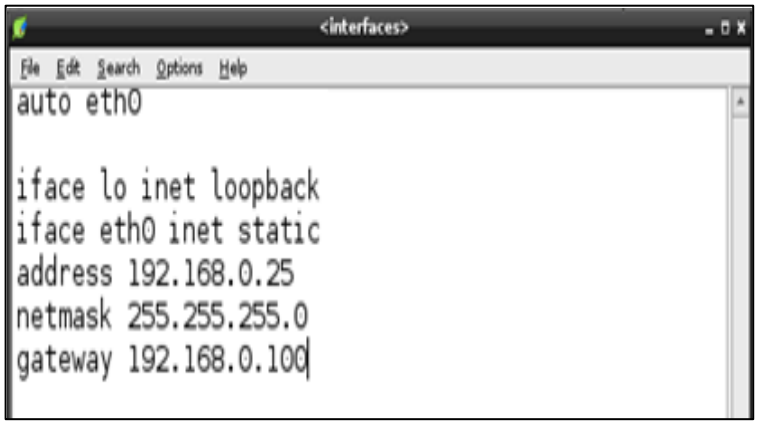

Figura A-6. Configuración de los parámetros de Internet en el Raspberry Pi

El siguiente paso es guardar los cambios en el archivo, y continuar ingresando las siguientes líneas de comando en el Lx Terminal.

- sudo chmod 644 /etc/network/interfaces
- sudo chmod 666 /etc/resolv.conf
- sudo echo "nameserver 8.8.8.8" > /etc/resolv.conf
- sudo chmod 644 /etc/resolv.conf
- sudo /etc/init.d/networking restart

Finalmente se recomienda reiniciar el Raspberry Pi, y al encenderlo de nuevo debería estar habilitada la conexión a Internet.

ANEXO 3

Instalación del programa Synaptic

Para llevar a cabo la instalación del programa Synaptic, esta puede hacerse desde el Lx Terminal ejecutando la siguiente línea de comando:

- Sudo apt-get install Synaptic

Cuando el Lx Terminal le realice una pregunta, el usuario debe contestar Y, y seguir con la instalación hasta su culminación exitosa.

Cada vez que ingrese al programa, este a su vez le pedirá la clave de acceso la cual es Raspberry con el usuario pi por supuesto como se muestra en la figura A-7.

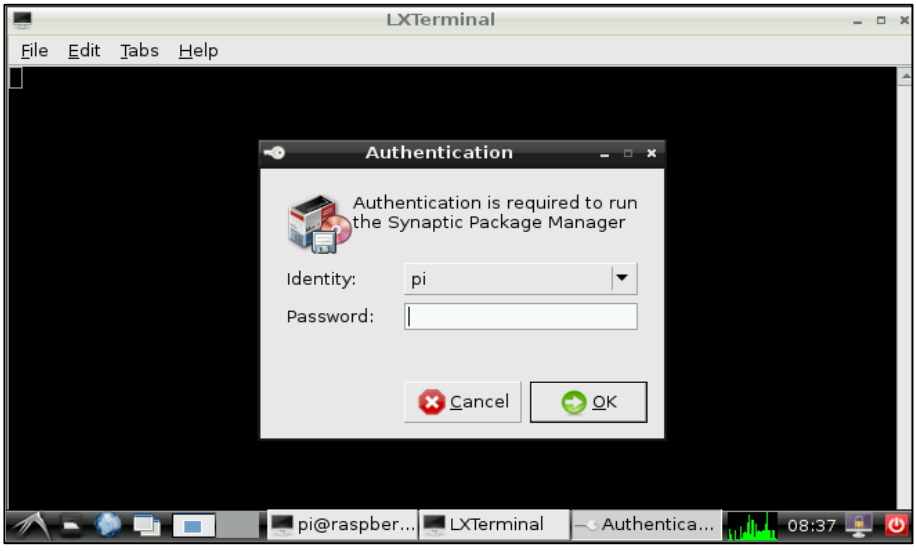

Figura A-7. Synaptic solicitando clave de acceso para iniciar

Como se mencionó en el capítulo 2 este programa facilita la descarga e instalación de ciertos drivers, librerías y programas necesarios no solo para el desarrollo del proyecto sino que también para cual propósito general.

ANEXO 4

Instalación del programa Arduino IDE

Para poder utilizar el Arduino uno es necesario tener instalado su plataforma de programación, para lo cual nos apoyaremos en el programa Synaptic para obtener dicha plataforma.

En la opción de búsqueda de Synaptic se escribe la palabra Arduino, como se muestra en la figura A-8, a continuación aparecen los drivers existentes de acuerdo a la búsqueda realizada anteriormente. Los autores recomiendan seleccionar todas las opciones que tengan relación con la búsqueda, ya que por lo general todos los drivers necesarios no se encuentran únicamente en un link de enlace, en este caso se seleccionaron tres.

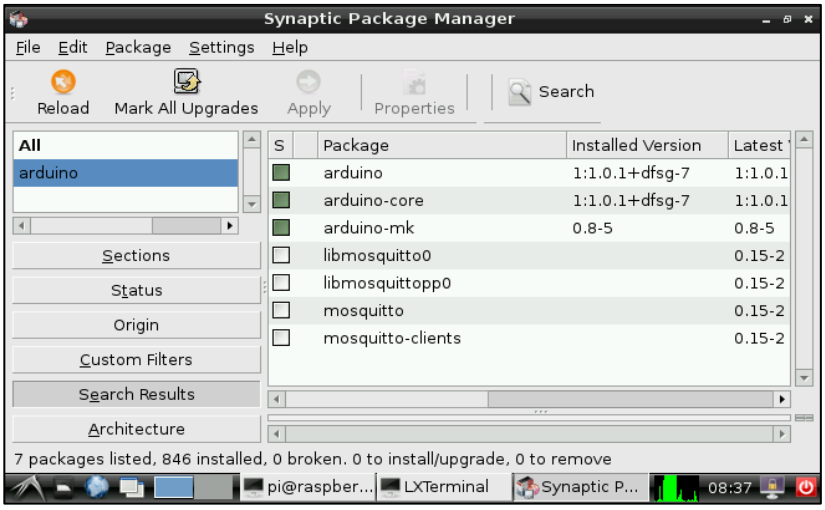

Figura A-8. Drivers necesarios para la instalación de la interfaz Arduino IDE

Una vez seleccionados se procede a descargarlos y Synaptic se encarga de instalarlos automáticamente. Para este caso el enlace de acceso directo del Arduino se ubicara en la pestaña Programming del menú inicial del Raspberry Pi.

ANEXO 5

Instalación de la librería python-serial y py-serial para lograr la comunicación serial entre el Raspberry Pi y el Arduino.

Estas librerías son cruciales para que ambos equipos puedan comunicarse mediante conexión USB. Para ello nuevamente usamos el programa Synaptic y simplemente repetimos el procedimiento anterior (instalación Arduino IDE) para instalar ambas librerías sin inconveniente alguno.

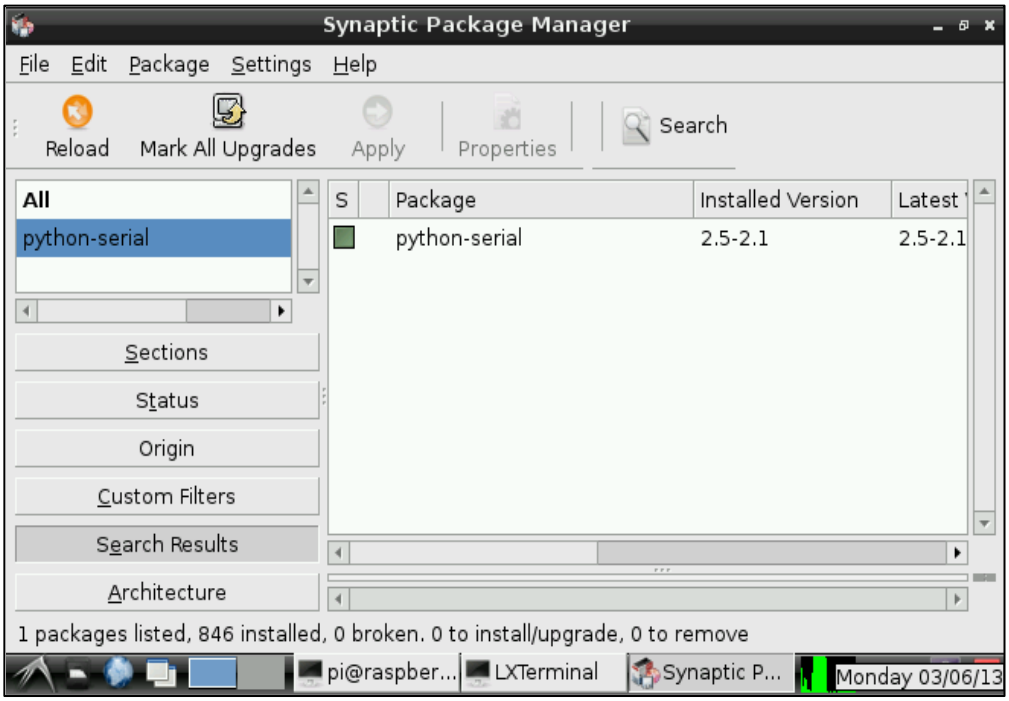

Figura A-9. Instalación de la librería python-serial

ANEXO 6

Instalación del programa Gnuplot

Para dicha instalación se utilizó el Lx terminal para digitar la siguiente línea de comando:

- sudo apt-get install gnuplot

Este programa se encarga de realizar gráficas cuyos parámetros son ingresados a manera de líneas de comando, aunque como se menciona en el capítulo 3 se puede almacenar todos los parámetros de configuración en un archivo de texto pero con extensión *nombre.gnuplot* y mandarlo a llamar desde gnuplot para así no tener que escribir tantas líneas de comando por gráfica, sino que la llamamos a crear con solo una que es la siguiente:

Gnuplot> load "*nombre.gnuplot*"

ANEXO 7

Instalación del web server casero phpMyAdmin

La instalación de este web casero, se realiza mediante el ingreso de líneas de comando en el Lx Terminal, las cuales son las siguientes:

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

Esto es para actualizar el sistema para evitar errores en la instalación, la siguiente línea de comando sería:

- sudo apt-get install apache2 php5 php5-mysql mysql-server

La descarga llevara un cierto tiempo debido a que los paquetes de instalación pesan alrededor de 116 Mb y durante este proceso se le pregunta dos veces por la clave ingresada de Mysql.

Después de finalizada la instalación se procede a verificar desde una página en blanco de un browser cualquiera en el cual se ingresa como dirección electrónica la ip de asignada al Raspberry pi. Si todo se ha instalado correctamente procedemos con la siguiente línea de comando:

- sudo apt-get install lynx

Para verificar que el servidor web funciona en nuestra red local abrimos una ventana de un browser externo y vamos a la dirección de la Raspberry Pi, veremos un mensaje como el de la figura A-10

Figura A-10. La página web por default funciona correctamente

Ahora, si probamos con lynx, debemos usar el comando:

- lynx http://localhost

En la vista de lynx tendremos una imagen parecida a la figura A-11.

Figura A-11. Mensaje de correcto funcionamiento usando el lynx

Para probar PHP, hacer lo siguiente:

- sudo nano /var/www/info.php

Escriba lo siguiente

- <?php phpinfo(); ?>

y guardar los cambios. Ahora a probar en la dirección local con el comando

- lynx http://localhost/info.php

Figura A-12. Prueba de php en el navegador Lynx

Ahora que hemos instalado los paquetes necesarios, procedemos a instalar el phpMyAdmin. Para hacer esto se digita en el Lx Terminal la siguiente línea de comando:

- sudo apt-get install phpmyadmin

Después de esto aparecerán las siguientes pantallas para completar la instalación, como la que se muestra en la figura A-13.

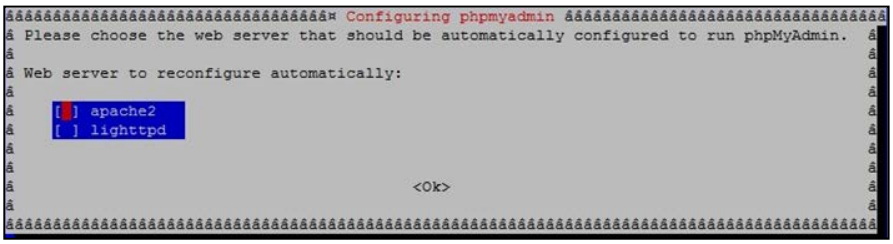

Figura A-13. Selección del web server a elegir

Luego escoja la opción Yes, como aparece en la figura A-14.

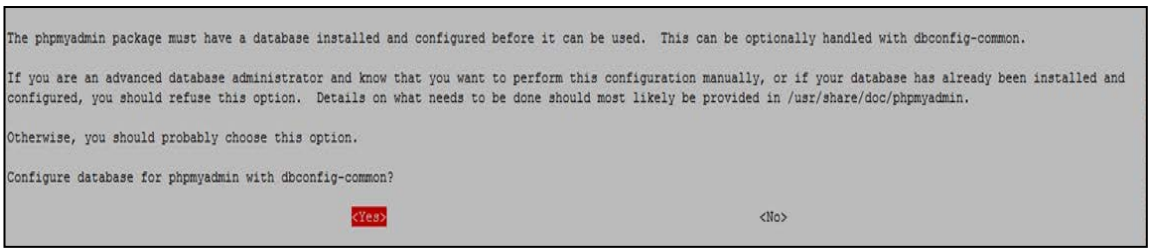

Figura A-14. Selección de respuesta al escoger Apache2

Luego, le pedirá ingresar una contraseña dos veces como se muestra en la figura A-15 y A-16.

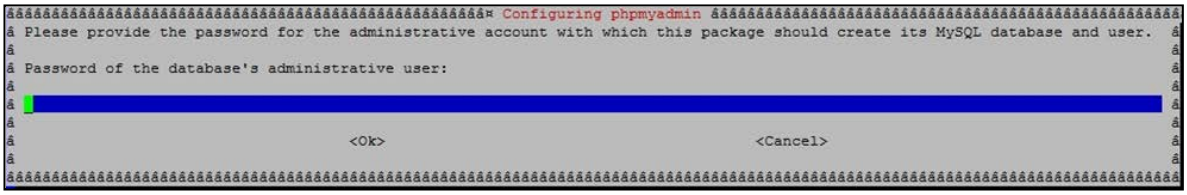

Figura A-15. Ingreso de la contraseña

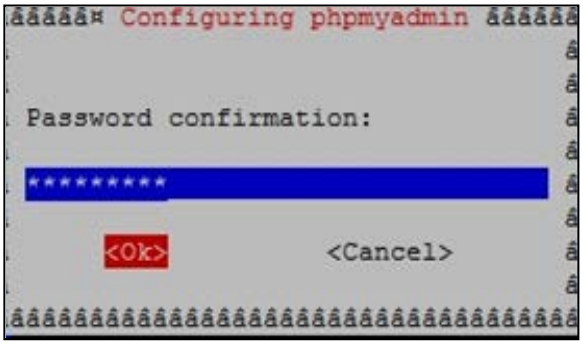

Figura A-16. Confirmación de la contraseña

Después se escribe el comando:

- sudo nano /etc/php5/apache2/php.ini

Buscamos dentro de este archivo, una parte del código que diga Dynamics Extensions y debajo de ella se escribe lo siguiente:

- extension=mysql.so

Se guardan los cambios y se ejecutan los siguientes comandos:

- sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf

- sudo /etc/init.d/apache2 reload

Finalmente se puede acceder al web server utilizando la dirección, y la página inicial debería lucir similar a lo que muestra la figura A-17.

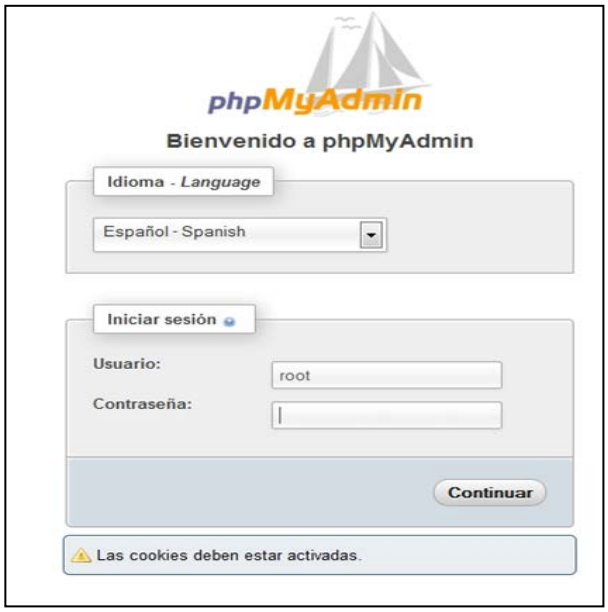

Figura A-17. Página Inicial de phpMyAdmin

ANEXO 8

Instalación de la librería eeml

Primero es necesario poseer todos los drivers y librerías para poder establecer la conexión con la página web Cosm.com, por lo tanto ingrese en el LX Terminal las siguientes líneas de comando:

- sudo apt-get install python-dev
- sudo easy_install -U distribute
- sudo apt-get install python-pip
- wget-Ogeekman-python-eeml.tar.gz https://github.com/geekman/python-eeml/tarball/master
- tar zxvf geekman-python-eeml.tar.gz
- cd geekman-python-eeml-*
- sudo python setup.py install

Al terminar, se habrá instalando la librería eeml, la cual hace referencia a la página Cosm.com.

ANEXO 9

Creación de una cuenta y de un feed en Cosm.com

Para la creación de una cuenta es necesario los siguientes pasos:

- Ingresar a la página Cosm.com (Figura A-18).
- Presionar el botón “Get Started”, digite su correo electrónico y contraseña. La página enviará a este correo un mensaje de verificación.

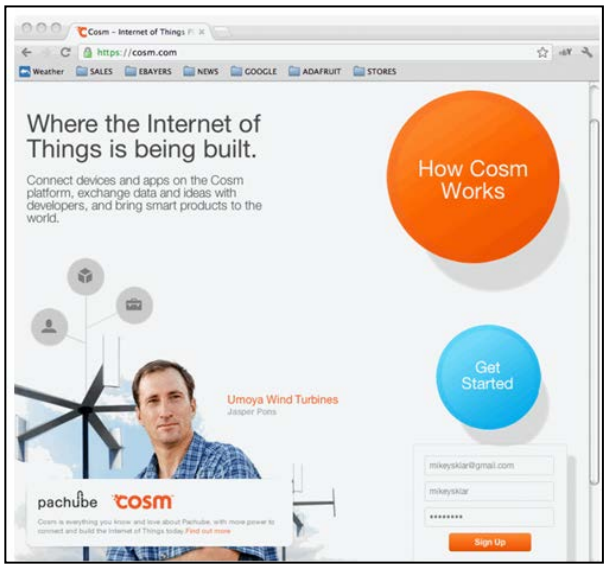

Figura A-18. Página de Inicio de Cosm.com

- Agregar un feed, presionando el botón azul como aparece en la figura A-19.

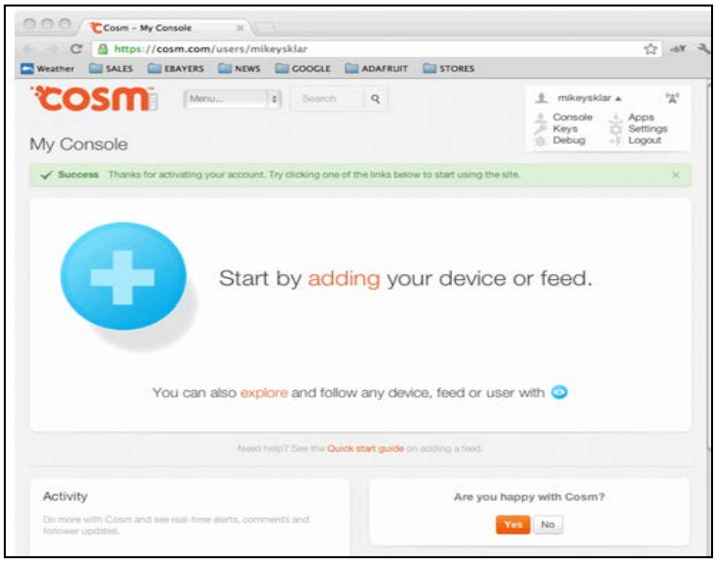

Figura A-19. Opción para agregar un nuevo feed

Seleccione como dispositivo el Arduino (Figura A-20).

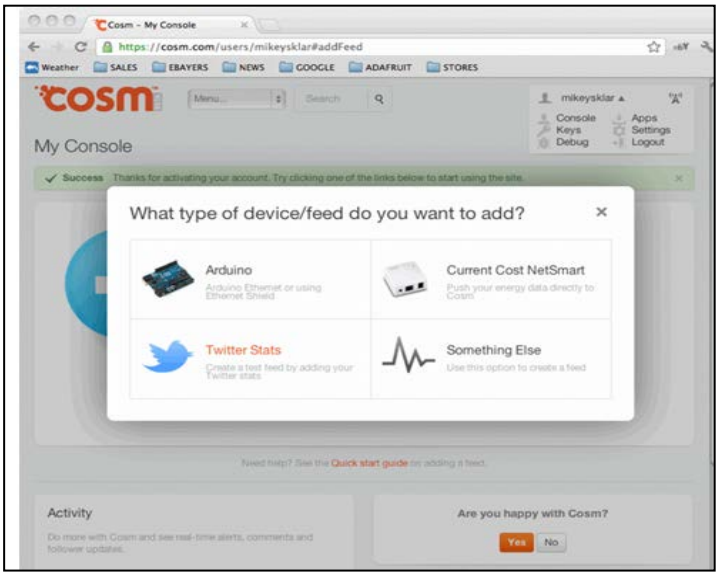

Figura A-20. Selección del dispositivo externo a agregar

Cree parámetros de configuración como el título de proyecto, las leyendas entre otras cosas.

Finalmente si no hay errores el sistema le otorgara varias claves para lograr la conexión con Cosm.com, como se muestra en la figura A-21, las cuales son:

API_KEY

FEEDID

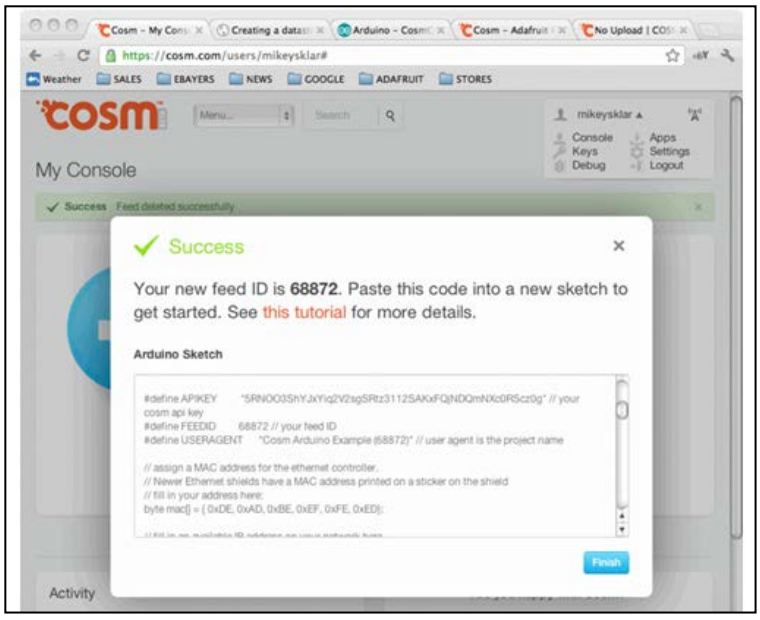

Figura A-21. Adquisición de las claves API_KEY y FEEDID al finalizar el proceso

BIBLIOGRAFÍA

[1] Simon Monk, Programing the Raspberry Pi Getting started with Python, McGraw Hill, 2013, p 1

[2] Eben Upton and Gareth Halfacree, Raspberry Pi User Guide, John Wiley and Sons Ltd., 2012, pp 12-13

[3] Eben Upton and Gareth Halfacree, Raspberry Pi User Guide, John Wiley and Sons Ltd., 2012, p 14

[4] Eben Upton and Gareth Halfacree, Raspberry Pi User Guide, John Wiley and Sons Ltd., 2012, p 15

[5] Simon Monk, Programing the Raspberry Pi Getting started with Python, McGraw Hill, 2013, p 7

[6] Simon Monk, Programing the Raspberry Pi Getting started with Python, McGraw Hill, 2013, p 125

[7] Eben Upton and Gareth Halfacree, Raspberry Pi User Guide, John Wiley and Sons Ltd., 2012, p 188

[8] Eben Upton and Gareth Halfacree, Raspberry Pi User Guide, John Wiley and Sons Ltd., 2012, p 18

[9] Everyday Linux User, Everyday Linux User guide to installing applications on the Raspberry Pi, http://www.everydaylinuxuser.com/2013/01/everyday-linux-user-guide-to-installing.html, fecha de consulta marzo 2013.

[10] Wikipedia, Code::Blocks, https://es.wikipedia.org/wiki/Code::Blocks, fecha de consulta abril 2013

[11] Code::Blocks, www.codeblocks.org, fecha de consulta abril 2013

[12] Wikipedia, Python, http://es.wikipedia.org/wiki/Python, fecha de consulta marzo 2013

[13] Michael McRoberts, Beginning Arduino, Springer Science+Business Media, 2010

[14] http://el-directorio.org/Gnuplot, fecha de consulta abril 2013

[15] Wikipedia, phpMyAdmin, http://es.wikipedia.org/wiki/PhpMyAdmin, fecha de consulta marzo 2013

[16] Appropedia, Pachube, http://www.appropedia.org/Pachube, fecha de consulta abril 2013

[17] Cosm, http://www.cosm.com, fecha de consulta abril 2013

[18] LXDE Wiki, LXTerminal, http://wiki.lxde.org/en/LXTerminal, fecha de consulta abril 2013

[19] LXDE, LXTerminal, http://wiki.lxde.org/en/LXTerminal, fecha de consulta abril 2013

[20] LXDE.org, LXDE, http://lxde.org/, fecha de consulta abril 2013