

**UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES
DEPARTAMENTO DE COMPUTACIÓN Y SIMULACIÓN DE SISTEMAS**

***CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA
DOMÓTICO PARA MONITOREO DEL CONSUMO
ELÉCTRICO VÍA WEB MEDIANTE UNA APLICACIÓN
ANDROID Y LA TECNOLOGÍA ARDUINO / RASPBERRY PI***

Asesor:

ING. EUCLIDES SAMANIEGO GONZÁLEZ, PhD

Presentado por:

**BORIS ANTONIO GONZÁLEZ ZAMBRANO
JOSÉ FRANCISCO HERAZO BRAVO**

**TRABAJO DE GRADUACIÓN PARA OPTAR AL TÍTULO DE LICENCIADO EN
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**

PANAMÁ, 2016

DEDICATORIA

Le dedicamos esta tesis a nuestros compañeros, futuros colegas, y demás personas dentro del mundo tecnológico, como mensaje de que la tecnología también debe ser utilizada e implementada en beneficio del medio ambiente, de nuestro único hogar.

Y a la sociedad panameña, como un mensaje de concientización, vivimos en un país privilegiado por la cantidad de recursos naturales que tenemos en nuestro alrededor, pero si no nos preocupamos por cuidarlos y preservarlos, no los tendremos por mucho tiempo.

Boris G. y José H.

AGRADECIMIENTOS

*Primero, gracias a Dios por permitirme llegar hasta donde
he llegado.*

*A mi familia, por darme la oportunidad de realizar y culminar
mis estudios, por ampararme y apoyarme en todo momento
y por haber estado ahí para mí cuando más lo necesité.*

*Agradezco a la Universidad Tecnológica de Panamá, y a
sus profesores, por habernos inculcado los conocimientos
que hoy nos han convertido en los profesionales que somos.*

*Y a nuestro profesor asesor, Euclides Samaniego, por
habernos guiado en todo el camino hasta llegar aquí.*

Boris G.

*Me gustaría agradecer a cada una de las personas que
formaron parte de los escenarios más trascendentales de mi
vida universitaria y que de manera especial me han inspirado
a cambiar y convertirme en la persona que soy actualmente.*

*Entre aquellas personas destaco a mi padre y madre por su
arduo apoyo y su orientación; a mi valioso hermano;
aquellos que más que amigos son mi familia; a mis
apreciados compañeros y respetados profesores.*

José H.

ÍNDICE

INTRODUCCIÓN	2
Capítulo 1. ASPECTOS GENERALES	4
1.1 Antecedentes	4
1.2 Caracterización de la Problemática	5
1.3 Justificación	7
1.4 Restricciones y Limitaciones	8
1.5 Objetivos	8
1.5.1 Objetivo General	8
1.5.2 Objetivos Específicos	8
Capítulo 2. MARCO TEÓRICO	9
2.1 Domótica	9
2.1.1 Aplicaciones de la Domótica	10
2.1.1.1 Gestión energética en el hogar	11
2.1.1.2 Seguridad y protección de bienes en el hogar	11
2.1.1.3 Confort y Ocio	11
2.1.1.4 Telecomunicaciones	12

2.2	El Internet de las Cosas	13
2.2.1	Aplicaciones del Internet de las Cosas	14
2.3	Medidores Convencionales de Consumo Eléctrico	16
2.3.1	Medidor Digital	16
2.3.2	Medidor Analógico	18
2.4	Sensores CT para la medición de consumo eléctrico	19
2.4.1	¿Qué es el sensor CT?	19
2.4.2	Funcionamiento del sensor CT	20
2.4.3	Ventajas de los sensores CT	20
2.4.4	Desventajas de los sensores CT	20
2.5	Arduino	21
2.5.1	¿Qué es Arduino?	21
2.6	Raspberry PI	22
2.6.1	Lectura de Datos desde la GPIO	23
2.7	ARM VERSUS x86	24
2.8	Entornos de Programación	25
2.8.1	Python	25
2.8.2	Java	26
2.8.3	JSON	27

2.9 Comparación entre Arduino y Raspberry PI	29
2.10 Plataforma Xively	33
2.10.1 Envío de Datos a la Plataforma	33
2.11 Android	34
2.11.1 Conceptos básicos del sistema operativo	34
2.11.2 Versiones de Android	36
Capítulo 3. DISEÑO E IMPLEMENTACIÓN DEL HARDWARE	41
3.1 Descripción del sistema	41
3.2 Conexión del sensor CT	42
3.3 Interfaz inalámbrica WiFi	43
Capítulo 4. DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE	45
4.1 Instalación del Sistema Operativo	45
4.2 Registro del Dispositivo en la Plataforma Xively	47
4.3 Codificación de Programa Python para el Procesamiento de Datos en el Sistema Operativo	48
4.3.1 Instalación de la Librería Python de Xively	48
4.3.2 Conexión con Plataforma Xively	49
4.3.3 Lectura de Datos de los Sensores CT	50
4.3.3.1 Formato de los Datos	50
4.3.3.2 Decodificación de los datos	52

4.3.4	Lectura de Datos del Termómetro Interno de la Raspberry Pi	52
4.3.5	Envío de Datos a la Plataforma Xively	53
4.3.5.1	Creación u Obtención de los Canales de Transmisión	54
4.3.6	Ejecución Automática del Programa Python en el Sistema Operativo	
	55	
4.4	Desarrollo de la aplicación Android	56
4.4.1	Conexión a la plataforma Xively	56
4.4.1.1	Importación de la Librería org.apache.http.legacy	56
4.4.1.2	Creación de clase para Manipular Peticiones	57
4.4.2	Lectura de Datos desde Xively	58
4.4.3	Creación de registros de lectura del medidor	62
Capítulo 5.	PRUEBAS Y RESULTADOS	63
5.1	Verificación de las lecturas de mediciones	63
5.1.1	Recolección de Datos	64
5.1.2	Comparación del Error Relativo versus el Consumo	66
5.2	Tiempo de Conectividad	67
CONSIDERACIONES FINALES		68
RECOMENDACIONES		70
REFERENCIAS BIBLIOGRÁFICAS		72
ANEXOS		78

PAGE
1
MERGE
OR
MA

ÍNDICE DE FIGURAS

Ilustración 1. Consumo Eléctrico per cápita de Panamá, Colombia y Costa Rica. Fuente: Google Charts, Banco Mundial.	6
Ilustración 2. Evolución de los dispositivos en el hogar.	10
Ilustración 3. Las áreas en donde se verá aplicado el IdC.	16
Ilustración 4. Medidor Eléctrico Digital.	17
Ilustración 5. Medidor Eléctrico Analógico.	19
Ilustración 6. Sensor CT.	19
Ilustración 7. Placa Arduino UNO	21
Ilustración 8. Raspberry Pi Zero. Con su pequeño tamaño, es un computador de cinco dólares	23
Ilustración 9. Diagrama de los GPIO de la Raspberry Pi.	24
Ilustración 10. Placa Arduino Duemilanove	30
Ilustración 11. Placa Raspberry Pi 2 Modelo B.	31
Ilustración 12. Las capas de la arquitectura de Android.	35
Ilustración 13. Gráfica mostrando la distribución de ventas de dispositivos móviles según su sistema operativo. Fuente: Gartner Inc.	36
Ilustración 14. Diagrama de Componentes del Dispositivo y su funcionamiento. 42	
Ilustración 15. Adaptador RPICT3 manufacturado por LeChacal.com	43
Ilustración 16. Adaptador WiFi USB optimizado para Raspberry Pi.	44

- Ilustración 17. Primera ventana que aparece al conectar la Raspberry Pi. Se elige el sistema operativo que se encuentre en la tarjeta SD. 46
- Ilustración 18. El escritorio de Raspbian Jessie, con sus ventanas y programas pre instalados. 47
- Ilustración 19. Diseño del prototipo de la aplicación Android. 62
- Ilustración 20. Lectura de Amperaje con un Amperímetro Convencional. 63
- Ilustración 21. Gráfico de Correlación de Pearson entre Error Relativo Porcentual y Medición del Dispositivo. 67

ÍNDICE DE TABLAS

Tabla 1. Tabla comparativa con las características de Arduino y Raspberry Pi.	32
Tabla 2. Tabla con las versiones y características de Android. Fuente: Xataka.com (Espeso, 2015)	37
Tabla 3. Tabla de Datos JSON de una petición HTTP	59
Tabla 4. Tabla de Valores de Pruebas Realizadas	64
Tabla 5. Errores Relativos Porcentuales en Cada Medición	65

RESUMEN

El objetivo de este proyecto fue medir el consumo eléctrico en un hogar a través de internet en tiempo real, y con la posibilidad de ver los resultados en el dispositivo móvil del usuario.

El hardware del prototipo está compuesto por sensores de corriente CT que mide el flujo de corriente que pasa por el cable al que se conecta, luego envía los datos medidos a un adaptador que está conectado a la entrada GPIO de una placa o sistema embebido Raspberry Pi 2 Modelo B. El adaptador transforma los datos para ser manipulados correctamente por la Raspberry Pi. El sistema procesa la información y la envía hacia un servidor en la plataforma web Xively que permite enviar la información hacia una aplicación en el dispositivo móvil del usuario.

El software consiste en un programa escrito en Python que se encarga de leer los datos medidos por los sensores CT. Este programa descansa sobre un sistema operativo Raspbian, basado en Linux. Los datos leídos por el programa en Python son enviados a la plataforma Xively, una plataforma para almacenamiento de datos creados para el Internet de las Cosas.

La comunicación entre el programa en Python y la plataforma Xively se realiza mediante el protocolo HTTP. Además el acceso remoto a los datos de la plataforma se realiza desde una aplicación Android que de igual modo utiliza el protocolo HTTP para realizar la comunicación con la plataforma Xively.

PAGE
1
MERGE
OR
MA

INTRODUCCIÓN

El mundo está cambiando, y con él nuestras necesidades. En las últimas décadas hemos sido parte de un incremento en el consumo de energía eléctrica a nivel mundial, el cual ha tenido sus consecuencias y parece no estabilizarse ni mucho menos detenerse.

Según los indicadores de desarrollo del Banco Mundial, en el 2011 se registró un consumo global de casi 18 Tera Watts hora/año, en el cual el 83% se veía representado por combustibles fósiles no renovables. Esto trae como consecuencia la excesiva generación de gases de invernadero, siendo esta una de las causas principales del cambio climático.

Nuestro país no se queda atrás. Hemos formado parte del alza de consumo eléctrico más elevada en los últimos años, razón por la cual se han creado programas para la toma de conciencia entre los ciudadanos, al igual que una organización gubernamental como lo es la Secretaría Nacional de Energía, la cual se encarga de promover el uso racional y eficiente de la energía en Panamá.

La mejor manera de solucionar un problema es prevenirlo, es por eso que este proyecto de tesis plantea el desarrollo de un dispositivo que nos permita conocer el consumo eléctrico de nuestro hogar en tiempo real.

El mismo se conectará a nuestros dispositivos móviles (teléfonos y/o tabletas) a través del internet, para así llevar un mejor monitoreo de la energía que se consume en nuestras casas, y así poder implementar un plan de ahorro energético y una mejor racionalización de la energía en el hogar de los panameños.

En el capítulo I, **Aspectos Generales**, se presentan los antecedentes del proyecto, en este caso proyectos parecidos que fueron desarrollados anteriormente, con algunas características parecidas al proyecto presente,

además, se caracteriza la problemática, se explica la justificación del proyecto y se definen los objetivos del mismo.

El capítulo II, **Marco Teórico**, se desarrolla la teoría sobre la cual se fundamenta el proyecto con base al planteamiento del problema que se ha realizado. Destacan temas como el Internet de las Cosas, los sistemas embebidos, los medidores eléctricos, diferentes lenguajes de programación, entre otras cosas.

En el III capítulo, **Diseño e Implementación del Hardware**, se describe detalladamente el sistema a desarrollar, y los componentes de hardware que se ven implicados en el funcionamiento del mismo. Se describe el funcionamiento de la interfaz de internet WiFi y el sensor de corriente CT.

El capítulo IV, **Diseño e Implementación del Software**, se expone el funcionamiento de los programas utilizados para el desarrollo del proyecto, entre ellos, el sistema operativo a utilizar, la lectura de valores por medio del sensor CT, el desarrollo de la plataforma web que se implementa en el proyecto y la conexión de la misma, y por último se explica el desarrollo de la aplicación móvil, en este caso para Android.

Finalmente, en el capítulo V, **Pruebas y Resultados**, se analizan los resultados obtenidos con el uso e implementación del sistema. En este capítulo se estudian los resultados para evaluar la confiabilidad de la data que se expone a través de nuestro sistema, y se verifica que los datos concuerden y sean confiables.

PAGE
1
MERGE
OR
MA

Capítulo 1. ASPECTOS GENERALES

En este capítulo se presentan los aspectos generales del presente trabajo.

1.1 Antecedentes

El ser humano normalmente busca ahorrar o disminuir sus gastos, y un gasto del que casi nadie se escapa es el consumo de energía eléctrica en los hogares. Y para regular o disminuir este gasto se han ideado gran cantidad de programas para generar conciencia en la sociedad, ya que el consumo excesivo de energía es un problema que nos afecta a todos. También se han desarrollado algunos artefactos que ayudan al usuario a saber su consumo eléctrico, como es el caso del proyecto de tesis de Jonathan Alberto Zaldaña, titulado "*MEDIDOR INALÁMBRICO DE CONSUMO DE ENERGÍA ELÉCTRICA DE BAJO COSTO.*" en la Universidad de El Salvador, El Salvador. Este proyecto, parecido al nuestro, consiste en el diseño y construcción de un medidor inalámbrico de consumo de energía eléctrica, utilizando micro controladores y transductores de corriente.

También podemos encontrar en el mercado productos que tienen como objetivo la medición del consumo eléctrico en el hogar, ya sea en puntos específicos del hogar o para toda la casa, con la problemática de que estos productos no son realmente comercializados en nuestro país, sólo en Europa y algunas ciudades de Norte América, y además, los productos son muy costosos. Como es el caso de *Eferry Elite*, *Owl Intuition-e*, *OD Energy*, entre otros.

El IdC o Internet de las Cosas, concepto que radica en que los objetos que nos rodean y que usamos diariamente estén conectados a internet en cualquier momento y lugar. (Mansilla & López, 2015). Conecta personas y cosas, está en continuo crecimiento. Según Cisco, el número de aparatos conectados a la red sobrepasa a la población humana en 1.5 puntos sobre 1. La conectividad a la

Red es la base del Internet de las Cosas. Sin embargo, los objetos conectados deben desempeñar un rol para sacar crédito a su ubicuidad y poder ser considerados como objetos inteligentes. (Mansilla & López, 2015)

Desde que se comenzó a desarrollar el concepto de IdC se han creado una gran cantidad de proyectos alrededor del mundo, todos con el mismo objetivo, conectar los objetos de uso diario a la Internet. Uno de los proyectos que valen la pena recalcar es la tesis desarrollada por Edison Chuquimarpa Sarango de la Universidad de Loja, Ecuador, con el nombre "*Diseño e implementación del prototipo de un sistema domótico para la medición del consumo de agua potable a través de internet y correo electrónico.*"

La misma cuenta con objetivos relacionados al presente proyecto, diseñar e implementar un sistema domótico, o sistema capaz de automatizar una vivienda, que permita monitorear el consumo de agua potable a través de correo electrónico y acceso Web en una vivienda de la ciudad de Loja. Para lograr esto, se estudió y analizó las diferentes tecnologías en sistemas domóticos, los sensores, dispositivos de transmisión de datos, micro-controladores y sistemas embebidos. Luego se implementó el hardware y software para la lectura digital del consumo de agua potable. Finalmente se desarrolla una interfaz gráfica para que el usuario logre monitorear su consumo.

1.2 Caracterización de la Problemática

Consideramos que somos conscientes de las consecuencias del cambio climático que se ha venido experimentando en los últimos 20 años y lo que esto está provocando al ambiente. Éste ha sido uno de los temas más controversiales alrededor del mundo, ya que la principal causa de este cambio climático es el consumo desmesurado de energía eléctrica, un problema en el cual nosotros somos los artífices.

La globalización y el desarrollo de las nuevas tecnologías nos han llevado a un excesivo aumento en el consumo eléctrico. En Panamá, en 1980 se promediaban 750 kWh per cápita al año, mientras que en el 2011 este valor se multiplicó casi 2.5 veces llegando a unos 1830 kWh per cápita al año. (Ilustración 1)

Esto ha marcado una tendencia en el consumo, estos valores nos indican que si no nos detenemos el consumo eléctrico de la nación llegará a ser uno de los mayores en la región.

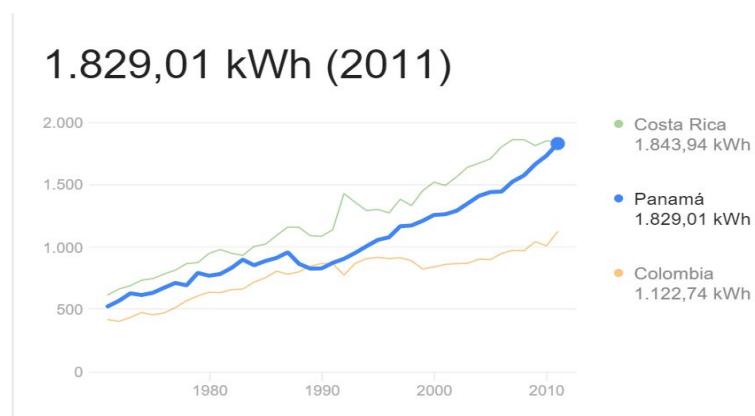


Ilustración 1. Consumo Eléctrico per cápita de Panamá, Colombia y Costa Rica. Fuente: Google Charts, Banco Mundial.

El récord de consumo eléctrico en el país se registró en junio del 2015, con 30 mil 249 megavatios/hora al día. El año anterior el consumo fue de 27 mil 500 megavatios, o un crecimiento del 10%, de acuerdo con las cifras del Centro Nacional de Despacho de la Empresa de Transmisión Eléctrica (ETESA).

Según el Plan de Expansión del Gobierno, para el periodo 2014-2017 se calcula un crecimiento en la demanda de energía entre 6.8 y 7.5%. A este ritmo se requiere que cada año se incorporen aproximadamente 100 megavatios al sistema, o una inversión de 300 millones del Estado. (Jordan, 2015).

Debido a que en Panamá el 52% de la energía es generada por hidroeléctricas, se están sobreexplotando nuestras fuentes de recursos naturales, como los ríos y fuentes de agua dulce, para poder satisfacer el consumo actual, lo cual está teniendo consecuencias negativas en la flora y fauna de nuestro país.

Los autores consideran que una forma para contribuir al ahorro de consumo eléctrico es saber el consumo eléctrico en tiempo real en nuestros teléfonos inteligentes.

1.3 Justificación

Gracias a la proliferación de electrodomésticos, se hace evidente que en gran parte del tiempo en el que nos encontramos en nuestros hogares estamos consumiendo energía eléctrica de una manera u otra, lo cual hace que la cuenta de consumo eléctrico del hogar sea cada vez más elevada.

Y en muchos de los casos no sabemos que somos totalmente capaces de reducir este consumo radicalmente sin cambiar nuestros hábitos ni disminuir nuestra calidad de vida.

La tecnología avanza a medida que el hombre se lo propone, y todos los días podemos ver que el potencial de estas tecnologías es cada vez mayor, permitiéndonos hoy en día hacer cosas que hace un par de años se quedaban en ideas. Uno de los últimos avances es el llamado Internet de las Cosas (IdC).

Esta tecnología ha demostrado tener un potencial de desarrollo a gran escala, principalmente dentro del hogar, conectando los dispositivos de uso diario a internet, para así dar soluciones más sencillas y eficientes.

Por lo tanto, se dispone tomar estos dos puntos de inflexión, el Internet de las Cosas como solución a situaciones dentro del hogar que necesiten de tecnología de avanzada y que sea de fácil implementación, y el incremento del consumo

eléctrico dentro de los hogares como problemática que debe ser tratada lo más pronto posible.

1.4 Restricciones y Limitaciones

- El prototipo será capaz de medir solamente el consumo eléctrico total de un hogar o un grupo de dispositivos eléctricos.
- El prototipo funcionará correctamente con una conexión a internet.

1.5 Objetivos

1.5.1 Objetivo General

- Construir e implementar un sistema domótico que permita monitorear el consumo eléctrico de un hogar a través de internet mediante una aplicación Android y el empleo de la tecnología Arduino y Raspberry Pi.

1.5.2 Objetivos Específicos

- Estudiar y analizar las distintas tecnologías en sistemas domóticos para micro controladores, sistemas embebidos, sensores, dispositivos de recepción y transmisión de datos vía internet.
- Diseñar e implementar el hardware de un prototipo de sistema domótico para la lectura digital del consumo eléctrico.

- Desarrollar el software de un prototipo de sistema domótico para la comunicación de datos a través de internet.
- Desarrollar una aplicación Android para el monitoreo de los datos del sistema domótico.

PAGE
1
MERGE
OR
MA

Capítulo 2. MARCO TEÓRICO

Iniciamos con conceptos referentes al entorno de trabajo y la introducción teórica a las herramientas utilizadas.

2.1 Domótica

*“La palabra domótica deriva de la unión de *domus* (casa) y de *informática*, y hace referencia a la incorporación a la vivienda de un conjunto de tecnologías informáticas y de comunicaciones que permiten gestionar y automatizar desde un mismo sistema las diferentes instalaciones de uso cotidiano en una vivienda, proporcionando una mejor calidad de vida de los usuarios de la misma y una mejor conservación y cuidado del edificio.”* (Sáez, 2006)

“Se entiende por domótica el conjunto de sistemas capaces de automatizar una vivienda, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar.” (Orghidan, 2012)

En los últimos quince años hemos sido testigos de un cambio radical en el estilo de vida de las personas, no solo para los que viven a la vanguardia de las tecnologías, sino para todos aquellos que cuentan con la posibilidad de adquirir nuevos productos para el hogar, por ejemplo, televisores, teléfonos, refrigeradoras, lavadoras, entre otros.

Una de las principales modificaciones ha sido la unión de estos productos con el internet. Cada vez es más común ver en venta productos para el hogar que llaman “inteligentes”, los cuales cuentan con la posibilidad de conectarse a internet, para así facilitar la vida de los usuarios. (Ilustración 2) Esto es a lo que se le conoce como “casa digital” o “casa inteligente” una nueva manera de vivir, que poco a poco, está transformando el hogar tradicional en el que muchos nos

criamos hace más de quince o veinte años, pero la mayoría de nosotros consideramos que este cambio está mejorando nuestra calidad de vida y de la familia.



Ilustración 2. Evolución de los dispositivos en el hogar.

2.1.1 Aplicaciones de la Domótica

La domótica se puede dividir en cuatro grandes aspectos, ya que la mayoría de los dispositivos desarrollados y comercializados están destinados a cumplir con alguna de estas cuatro áreas:

2.1.1.1 Gestión energética en el hogar

Cada vez son más los aparatos eléctricos que se incorporan a la vivienda, de forma que el consumo de energía puede llegar a ser importante. Mediante un sistema domótico es posible implementar mecanismos que regulen y optimicen dicho consumo, como el control de la climatización y regulación de la temperatura por zonas; la utilización de electrodomésticos en tarifa nocturna; la iluminación por detección de presencia; el riego controlado por sensor meteorológico; la desconexión automática de dispositivos, etc.

El dispositivo que se va a desarrollar y construir en este proyecto entra en esta categoría, ya que el mismo busca asistir al usuario para que se le facilite el ahorro energético dentro del hogar.

2.1.1.2 Seguridad y protección de bienes en el hogar

Actualmente, la seguridad es la función más demandada de un sistema domótico y la más implantada. Puede incorporar múltiples aplicaciones, y el objetivo fundamental es evitar riesgos y accidentes domésticos así como asegurar y proteger a los usuarios así como a sus bienes. Se puede dividir en seguridad de personas y seguridad de bienes. (Rocamora, 2008)

Para asegurar la integridad de las personas y de los edificios, una instalación domótica puede proporcionar mecanismos como detección de intrusos; simulación de presencia; conexión con centrales de alarma; alarmas de salud o alertas médicas (tele asistencia); alarmas técnicas: Incendios, fugas de agua o gas; control de accesos, etc.

2.1.1.3 Confort y Ocio

Es un hecho comprobado que la calidad de vida actual es muy superior a la de un par de generaciones anteriores. El simple hecho de automatizar los elementos del hogar y poder gestionarlos de forma remota proporciona unos niveles de comodidad antes inimaginables. Con una instalación domótica el

usuario se libera de invertir tiempo y energía en realizar acciones mecánicas y cotidianas y de preocuparse por aspectos que el sistema podría resolver automáticamente.

Probablemente el confort sea el aspecto más valorado de un sistema domótico para usuarios residenciales, pues es el que se percibe directamente dentro del hogar. El control inalámbrico de todo sistema domótico a través de mandos a distancia, la automatización del riego del jardín, la apertura automática de puertas por detección de presencia, el control de persianas y cortinas, la integración de audiovisuales en el sistema, de modo que la televisión (*SmartTV*), los sistemas de audio, el computador y demás puedan ser empleados por el sistema domótico como un componente más. Estos son sólo algunos ejemplos de los dispositivos destinados al ocio o confort dentro del hogar que ofrece la domótica.

2.1.1.4 Telecomunicaciones

La aparición de nuevas tecnologías en el campo de las comunicaciones y redes de transmisión de datos, y el hecho de que los sistemas domóticos avanzados se basen en el empleo de estos tipos de redes, hacen de éste un campo rentable para la investigación y el desarrollo de nuevas arquitecturas y sistemas de integración.

El objetivo fundamental de una vivienda domótica en materia de telecomunicaciones es el de asegurar y establecer comunicaciones dentro del propio hogar y de forma remota. Estas comunicaciones deben ser bidireccionales, es decir: el usuario podrá establecer una comunicación remota con su vivienda y el sistema domótico podrá comunicarse con el usuario.

Por ejemplo, se podría distribuir imágenes y sonido por el interior del edificio; manejar internamente el sistema a través de mando a distancia; enviar alarmas y señales hacia el exterior; realizar un control remoto del sistema a través de teléfono fijo, móvil, Internet, etc.

2.2 El Internet de las Cosas

El IERC (*European Research Cluster on the Internet of Things*) define el internet de las cosas como “una infraestructura de red global dinámica con capacidad de autoconfiguración basada en los estándares y protocolos de comunicación interoperables donde las “cosas” físicas y virtuales tienen identidad, atributos físicos, y personalidad virtual, usan interfaces inteligentes, y están continuamente integradas en la información de la red.”

La ITU (*International Telecommunications Union*) define el internet de las cosas de manera similar como “una infraestructura global para la información de la sociedad que habilita servicios avanzados a través de las cosas interconectadas (físicas o virtuales) basadas en la información interoperable existente y desarrollada y en las tecnologías de la comunicación.”

De manera más simple, se podría definir el internet de las cosas como la red que interconecta objetos físicos valiéndose del internet. Estos objetos se valen de sistemas embebidos, o mejor dicho, hardware especializado que le permite no solo la conectividad al internet, sino que además programa eventos específicos en función de las tareas que se le sean dictadas remotamente. Cada uno de los objetos conectados al internet tiene una IP específica y mediante esta IP puede ser accedido para recibir instrucciones. Así mismo, puede contactar con un servidor externo y enviar los datos que recoja.

La conectividad a la Red es la base del Internet de las Cosas. Sin embargo, los objetos conectados deben desempeñar un rol para sacar crédito a su ubicuidad y poder ser considerados como objetos inteligentes. Para ello, según Diego Casado Mansilla y Juan López de Armentia, investigadores de DeustoTech de la Universidad de Deusto, España, son necesarios tres pilares:

- 1) componentes computacionales que permitan procesar información.
Ejemplo: micro-controladores;

- 2) sensores que permitan obtener información física del entorno y convertirla en información procesable digitalmente. Ejemplo: luminosidad, movimiento, temperatura;
- 3) actuadores, que son dispositivos electrónicos que permiten modificar o generar un efecto sobre la física del entorno. Ejemplo: motores, altavoces.

Las aplicaciones del Internet de las Cosas cubren un amplio espectro de nuestra vida cotidiana. Uno de los campos en el que está empezando a tener y se prevé tendrá gran relevancia, es la sostenibilidad medioambiental, impulsado principalmente por el ahorro energético.

2.2.1 Aplicaciones del Internet de las Cosas

El principal objetivo de IdC es la creación de entornos inteligentes y cosas conscientes para aplicaciones relacionadas con el clima, la alimentación, la energía, la movilidad, la sociedad digital y la salud (por ejemplo: transporte, productos, ciudades, edificios, zonas rurales, energía, salud, inteligente, etc.).
(Ilustración 3)

Los desarrollos de las “entidades inteligentes” también fomentará el desarrollo de las nuevas tecnologías necesarias para hacer frente a los nuevos retos de la salud pública, el envejecimiento de la población, la protección del medio ambiente y el cambio climático, la conservación de la energía y la escasez de materias primas, mejoras en la seguridad y la continuación y crecimiento de la prosperidad económica. Estos problemas serán abordados de las siguientes maneras: (Madrid Network, 2013)

- Gestión, detección y tecnología de red fiable, inteligente, auto-administrada, sensible al contexto y adaptable.
- Perfeccionamiento de la interacción entre el hardware, software, algoritmos, así como el desarrollo de interfaces inteligentes entre las cosas (máquina a máquina inteligente, interfaces cosas a cosas) y los

interfaces inteligentes entre humanos-máquinas y cosas, permitiendo así el software inteligente y móvil.

- Incorporación de la funcionalidad inteligente a través de desarrollos adicionales en el área de la nano-electrónica, sensores, actuadores, antenas, almacenamiento, fuentes de energía, sistemas integrados y redes de sensores.
- Desarrollos en todas las disciplinas para hacer frente a las comunicaciones multifuncionales y multi-dominio, las tecnologías de la información y procesamiento de señal, la tecnología de identificación y mejoras en la tecnología de los motores de búsqueda.
- El desarrollo de nuevas técnicas y conceptos para mejorar la seguridad y la privacidad de las tecnologías existentes con el fin de adaptarse a los nuevos retos tecnológicos y sociales.
- Mejorar la estandarización, la interoperabilidad, la validación y la modularización de las tecnologías y soluciones de IoT.
- Definición de nuevos principios de gobierno que se ocupan de la evolución de la tecnología y permitir el desarrollo de negocios y el acceso libre al conocimiento en línea con las necesidades globales, manteniendo el respeto por la privacidad y la seguridad.

Creemos que en el futuro la mayoría de los objetos tendrán conectividad inalámbrica y el Internet de los objetos impulsará las aplicaciones de eficiencia energética tales como la red eléctrica, o redes inteligentes, vehículos eléctricos, edificios energéticamente eficientes.

El Internet de las Cosas proporcionará la tecnología y soluciones que permitirán hacer pleno uso de las tecnologías integradas en las redes de comunicaciones y tecnologías de Internet para construir ciudades verdes inteligentes, ofreciendo una amplia variedad de métodos interactivos y de control para el sistema de

información urbana y más apoyo para la construcción de sistemas integrales para el desarrollo de la ecología urbana. (Madrid Network, 2013)

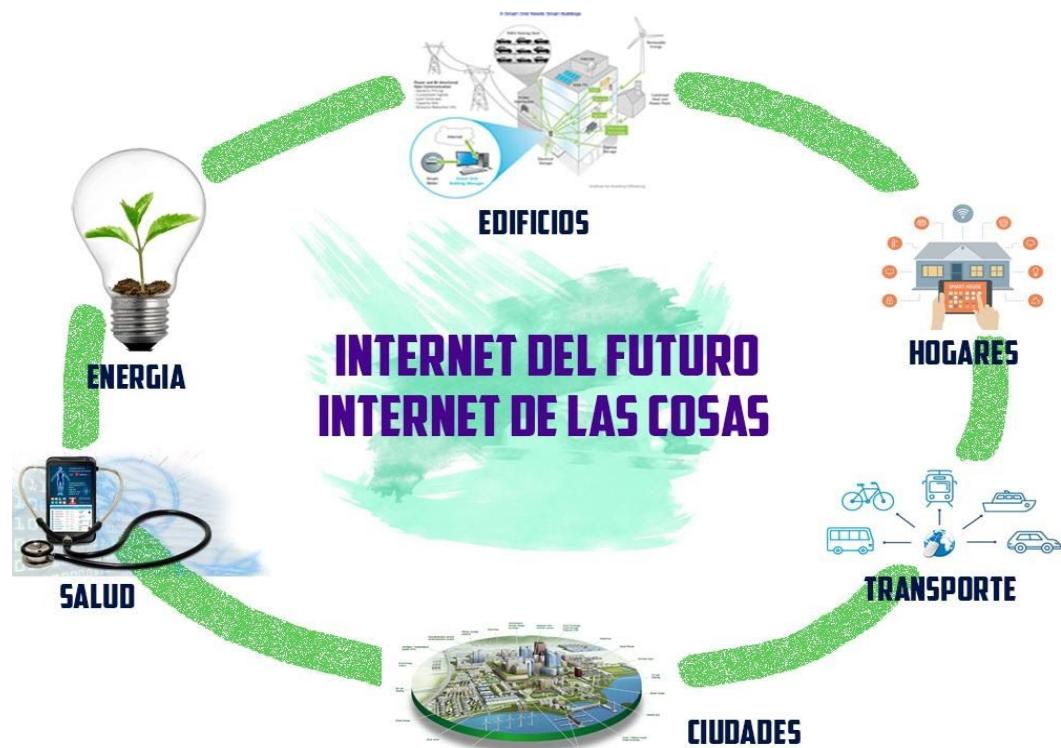


Ilustración 3. Las áreas en donde se verá aplicado el IdC.

2.3 Medidores Convencionales de Consumo Eléctrico

2.3.1 Medidor Digital

Un medidor eléctrico digital involucra, a partir de alguna etapa, un proceso digital, es decir, ante una señal de entrada cuya variación sea continua, proporciona una salida un número finito o discreto de valores.

La medición aparece en forma numérica.

Ventajas

- Tienen alta resolución alcanzando en algunos casos más de 9 cifras en lecturas de frecuencia.
- Mucha exactitud.
- No están sujetos al error de paralelaje.
- Pueden eliminar la posibilidad de errores por confusión de escalas.
- Tienen una rapidez de lectura que puede superar las 1000 lecturas por segundo.

Desventajas

- El costo es elevado.
- Son complejos en su construcción.
- Las escalas no lineales son difíciles de introducir.
- En todos los casos requieren de fuente de alimentación.



Ilustración 4. Medidor Eléctrico Digital.

2.3.2 Medidor Analógico

Un medidor eléctrico analógico involucra un proceso analógico, es decir ante una señal de entrada cuya variación es continua, proporciona una salida también continua, la cual puede tomar cualquiera de los valores entre los límites especificados. Es aquel en el cual la indicación se obtiene a partir de una posición de un índice material o no, sobre una referencia adecuada.

Ventajas

- Bajo Costo.
- En algunos casos no requieren de energía de alimentación.
- No tienen gran sofisticación.
- Presentan con facilidad las variaciones de los parámetros para visualizar si el valor aumenta o disminuye.
- Es sencillo adaptarlos a diferentes tipos de escalas no lineales.

Desventajas

- Tienen poca resolución, típicamente no proporcionan más de 3 cifras.
- El error de paralaje limita la exactitud a $\pm 0.5\%$ a plena escala en el mejor de los casos.
- Las lecturas se presentan a errores graves cuando el instrumento tiene varias escalas.
- La rapidez de lectura es baja.



Ilustración 5. Medidor Eléctrico Analógico.

2.4 Sensores CT para la medición de consumo eléctrico

2.4.1 ¿Qué es el sensor CT?

Los transformadores de corriente (CT por sus siglas en inglés) son sensores que miden la corriente alterna. Son particularmente útiles para la medición del consumo o flujo de corriente eléctrica.



Ilustración 6. Sensor CT.

2.4.2 Funcionamiento del sensor CT

Como los demás transformadores, el de corriente tiene un cable devanado primario, un núcleo magnético y un cable devanado secundario. La corriente alterna fluyendo por el cable primario produce un campo magnético en el núcleo, el cual induce una corriente en el circuito del cable secundario. La corriente en el cable secundario es proporcional a la corriente que fluye por el cable primario.

2.4.3 Ventajas de los sensores CT

- Controla la corriente y el arranque y la parada de motores.
- No posee partes móviles que puedan fallar.
- Facilita el cableado.
- Facilitan la instalación.
- Puede ubicarse en lugares pequeños.
- Reduce los tiempos y costos asociados a la instalación.
- Brinda una amplia selección de rangos de amperaje para diferentes aplicaciones.

2.4.4 Desventajas de los sensores CT

- Sólo tiene conexión Jack de 3.5mm, lo cual reduce la cantidad de dispositivos que pueden utilizarlo.
- De manera que sólo utiliza conexión Jack de 3.5mm, en algunos dispositivos es necesario el uso de adaptadores.
- Por lo general, la distancia del cable es bastante corta.
- Sólo funciona si el cable del que se quiere medir su potencial eléctrico está dividido, ya que el sensor sólo puede leer la corriente de ida o de vuelta, no ambas.

2.5 Arduino

2.5.1 ¿Qué es Arduino?

Arduino es una plataforma electrónica de código abierto (open source) basada en una sencilla placa con entradas y salidas analógicas y digitales, de fácil uso.

En un entorno de desarrollo que implementa el lenguaje Processing / Wiring. Es decir, Arduino es una placa electrónica que cualquiera puede acceder a su esquema, montarla y utilizarla sin adquirir ninguna licencia, aunque también se puede adquirir ya montada.

La tecnología Arduino puede “sentir” el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar su entorno mediante el control de luces, motores y otros artefactos.

El corazón de la placa Arduino es el chip AtMega8, un chip sencillo y de bajo precio que permite el desarrollo de múltiples diseños. (Samaniego, 2011)



Ilustración 7. Placa Arduino UNO

2.6 Raspberry Pi

Raspberry Pi es un ordenador de placa reducida de bajo coste. Se desarrolla con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. De software *open source*, utiliza como sistema operativo una versión adaptada de Debian, denominada Raspbian.

Al igual que la placa de un computador común, la Raspberry Pi contiene un procesador central (CPU), un procesador gráfico (GPU), y 512 MB de memoria RAM. El diseño no incluye disco duro, ya que utiliza una tarjeta SD para el almacenamiento permanente, tampoco incluye fuente de alimentación ni carcasa. (Schmidt, 2012)

La placa, la cual en sus modelos básicos es del tamaño de una tarjeta de crédito (Ilustración 8), tiene varios puertos y entradas, dos USB, uno de Ethernet y salida HDMI. Estos puertos permiten conectar el miniordenador a otros dispositivos, teclados, ratones y pantallas.

Al momento de desarrollar este proyecto, existían cinco tipos de Raspberry Pi, desde la más pequeña, la Raspberry Pi Zero, que cuesta cinco dólares, hasta la más compleja, la Raspberry Pi 2 modelo B, con un costo de 35 dólares.

Sin dudas éste ha sido uno de los inventos más revolucionarios en el área de la computación y los gadgets en los últimos años, ya que permite a los aficionados de ésta área personalizar, modificar y hasta crear sus propios dispositivos y soluciones con el uso de esta pequeña y accesible placa.

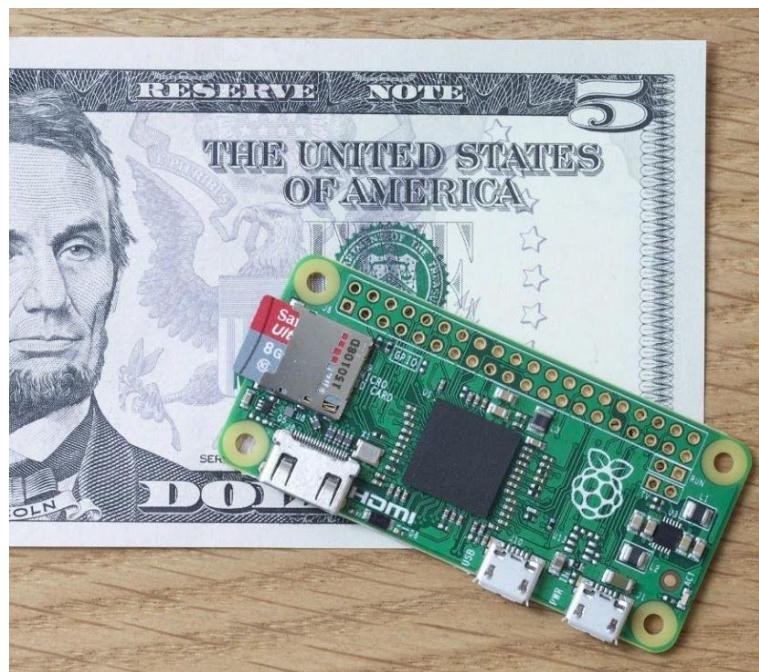


Ilustración 8. Raspberry Pi Zero. Con su pequeño tamaño, es un computador de cinco dólares

2.6.1 Lectura de Datos desde la GPIO

La GPIO, por sus siglas en inglés de Entrada/Salida de Propósito General, son los pines genéricos en un chip, cuyo comportamiento puede ser controlado o programado por el usuario en tiempo de ejecución.

Esta es una gran fuente de posibilidades que podemos encontrar en la Raspberry Pi, ya que nos permite utilizar estos pines para controlar sensores, servomotores y dispositivos externos.

En este caso se controla desde Python, lo que lo hace relativamente sencillo. Para la Raspberry Pi se debe conocer el funcionamiento de cada pin antes de utilizarlos, ya que hay algunos destinados a la comunicación, etc. La única deficiencia que presenta esta placa es que no tiene ninguna entrada de lectura analógica. Cabe mencionar que a diferencia de otras placas, los pines de entrada y salida de la Raspberry Pi son de 3.3 voltios, y no toleran los 5V. (Monnk, 2013)

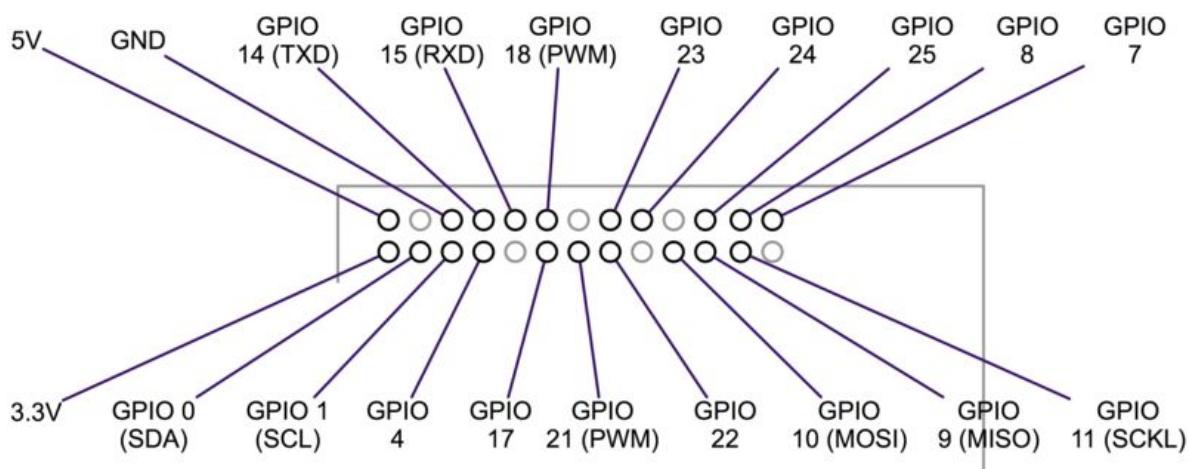


Ilustración 9. Diagrama de los GPIO de la Raspberry Pi.

2.7 ARM VERSUS x86

Es un procesador basado en la arquitectura CISC (*Complex instruction set computing*) con soporte para instrucciones complejas, simultáneas y de ejecución más lenta, que a pesar de simplificar la estructura de programación lo que se acaba obteniendo es un alto desempeño, que desafortunadamente viene

acompañado de un mayor consumo de energía y también de la necesidad de más espacio físico.

Por otro lado, los procesadores ARM son de tipo RISC (*Reduced Instruction Set Computer*); cuyas propiedades son que poseen instrucciones de tamaño fijo con pocos formatos y que sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos. El objetivo de diseñar máquinas con esta arquitectura es facilitar el paralelismo en la ejecución de instrucciones y permitir realizar tareas menores con procesos más cortos lo que al final conlleva una disminución de la energía empleada.

Intel y AMD son los dos grandes fabricantes que han hecho evolucionar los procesadores x86 en entornos básicamente de escritorio. Mientras que el desarrollo de los chips ARM han evolucionado gracias a distintos fabricantes como Qualcomm, Texas Instruments, Apple, Samsung, entre otros. (Pérez, 2012)

2.8 Entornos de Programación

2.8.1 Python

Python fue creado a principio de la década de los 90 por Guido van Rossum en Holanda, como sucesor del lenguaje ABC. Desde ahí Python se ha vuelto muy popular entre los desarrolladores, que se han sentido atraídos por su sintaxis limpia y por la reputación de alta productividad que tiene el lenguaje. Además que se caracteriza por poseer una licencia de código abierto, denominada Python Software Foundation License.

El creador del lenguaje lo define como un lenguaje de muy alto nivel, intérprete y orientado a objetos, implementado de una manera que enfatiza la interactividad con el usuario. Python comparte algunas características con los lenguajes que

utilizan scripts, pero también trae características de otros lenguajes de programación más tradicionales. (Rossum, 2003)

Python tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. La elegante sintaxis de Python, su gestión de tipos dinámica y su naturaleza interpretada hace de él el lenguaje ideal para scripts y el desarrollo rápido de aplicaciones, en muchas áreas y en la mayoría de las plataformas. (van Rossum & Drake, 2000)

El intérprete de Python y la extensa librería estándar están disponibles en forma de fuentes o ejecutables, para las plataformas más importantes, y se pueden distribuir libremente.

Se puede decir que Python es multiparadigma, ya que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional.

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++.

```
int factorial(int x)
{
    if (x == 0)
        return 1;
    else
        return x * factorial(x - 1);
}
```

Ejemplo sencillo con los elementos más comunes del lenguaje Python.

2.8.2 Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. (Oracle, 2008)

Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos.

JAVA es el lenguaje de programación con que se desarrollan las aplicaciones en Android. El funcionamiento de las aplicaciones en Android, de cualquier tipo (fuera del propio sistema operativo), hace que estas corran sobre una máquina virtual llamada DALVIK. Esta máquina es la responsable de que no sea necesario diseñar las aplicaciones específicamente para cada teléfono, sino que solamente haya que diseñarlos para Android.

Esta máquina, aunque es compatible con JAVA, no es esa misma plataforma estrictamente hablando, pero a la hora de desarrollar, es idéntica para muchas cosas e incluso existen múltiples herramientas para hacer conversiones de JAVA a Android. (Benito, 2013)

```
// Hello.java
import javax.swing.JApplet;
import java.awt.Graphics;

public class Hello extends JApplet {

    public void paint(Graphics g) {
        g.drawString("Hola, mundo!", 65, 95);
    }
}
```

Ejemplo del clásico Hola Mundo, escrito en Java.

2.8.3 JSON

JSON, por sus siglas en inglés de Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo.

Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros.

Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. (ECMA International, 2013)

```
var http_request = new XMLHttpRequest();
var url = "http://example.net/jsondata.php";

// Descarga los datos JSON del servidor.
http_request.onreadystatechange = handle_json;
http_request.open("GET", url, true);
```

```
http_request.send(null);

function handle_json() {
    if (http_request.readyState == 4) {
        if (http_request.status == 200) {
            var json_data = http_request.responseText;
            var the_object = eval("(" + json_data + ")");
        } else {
            alert("Ocurrio un problema con la URL.");
        }
        http_request = null;
    }
}
```

Un ejemplo de acceso a datos JSON usando XMLHttpRequest. (Yahoo! Developer Network, 2010).

2.9 Comparación entre Arduino y Raspberry Pi

La evolución de las nuevas tecnologías ha llegado a tal punto que podemos obtener nuestros propios sistemas embebidos o placas con micro controladores desde cinco dólares.

Este es el precio base de la última versión de Raspberry Pi Zero desarrollada hasta la fecha. Pero esta no es la única opción que tenemos, Raspberry Pi nos permite escoger entre cinco modelos con características distintas cada uno.

Arduino tiene más de 25 placas en el mercado. Todas estas placas parecen ser similares, pero cada una de ellas está desarrollada para un objetivo distinto, es por esto que el dilema de la mayoría de las personas que van a desarrollar su propio proyecto es elegir entre Arduino o Raspberry Pi.

Estas tecnologías tienen sus diferencias. (Tabla 1) Para empezar, Raspberry Pi es una computadora completamente funcional, mientras que Arduino es un micro controlador, el cual es sólo un componente de una computadora.

Aunque el Arduino puede ser programado con pequeñas aplicaciones como C, este no puede ejecutar todo un sistema operativo y ciertamente no podrá ser el sustituto de un computador en un tiempo cercano.

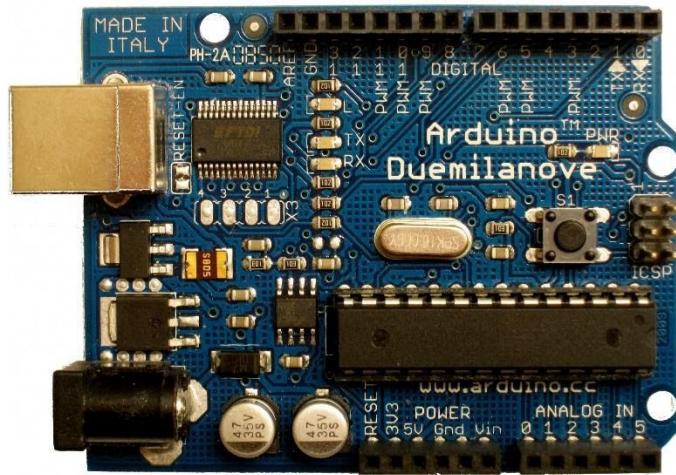


Ilustración 10. Placa Arduino Duemilanove

Tanto Raspberry Pi como Arduino fueron diseñadas originalmente para ser herramientas de enseñanza, es por ello que se han vuelto tan populares, ambos dispositivos son muy fáciles de aprender a usar. Hasta que se examina su hardware y software es cuando se hace evidente que están orientadas a diferentes tipos de proyectos.

La Raspberry Pi es 40 veces más rápido que un Arduino cuando se trata de velocidad de reloj. Además, Pi tiene 128,000 veces más memoria RAM. La Raspberry Pi es una computadora independiente que puede ejecutar un sistema operativo real en Linux. Puede realizar varias tareas, soportar dos puertos USB y conectarse de forma inalámbrica a Internet. En pocas palabras, es lo suficientemente potente como para funcionar como una computadora personal.



Ilustración 11. Placa Raspberry Pi 2 Modelo B.

Puede parecer que Raspberry Pi es superior a Arduino, pero eso es sólo cuando se trata de aplicaciones de software. La simplicidad de Arduino hace que éste sea una apuesta mucho mejor para proyectos de hardware.

El Arduino IDE es mucho más fácil de usar que Linux. Por ejemplo, si se desea escribir un programa para hacer parpadear un LED con Raspberry Pi, se necesita instalar un sistema operativo y algunas librerías de código.

En Arduino, se puede obtener una luz LED parpadeando con tan sólo ocho líneas de código. Dado que Arduino no está diseñado para funcionar con un sistema operativo o una gran cantidad de software, funciona con la tecnología Plug & Play. (Galicia, 2014)

Raspberry Pi puede procesar varias tareas, éste puede ejecutar múltiples programas en segundo plano mientras está activado. Por ejemplo, puede estar funcionando como un servidor de impresión y un servidor VPN al mismo tiempo.

Por otro lado, se puede dejar un Arduino conectado, ya que lleva a cabo un proceso único por un largo periodo de tiempo, y desconectarlo cuando no se esté utilizando.

Tabla 1. Tabla comparativa con las características de Arduino y Raspberry Pi.

Características	Arduino	Raspberry Pi modelo B
Precio en dólares	\$30	\$35
Tamaño	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memoria	0.002MB	512MB
Velocidad de reloj	16 MHz	700 MHz
On Board Network	Ninguna	10/100 wired Ethernet RJ45
Multitarea	No	Sí
Voltaje de entrada	7 a 12 V	5 V
Memoria Flash	32KB	Tarjeta SD (2 a 16G)
Puertos USB	Uno	Dos
Sistema operativo	Ninguno	Distribuciones de Linux
Entorno de desarrollo integrado (IDE)	Arduino	Scratch, IDLE, cualquiera con soporte Linux

2.10 Plataforma Xively

Xively es una compañía que cuenta con una plataforma *cloud* pública en que se pueden conectar sensores y dispositivos a través del Internet de las Cosas. Ofrece una plataforma de servicio para el IdC, servicio de negocios y socios que permiten conectar rápidamente los productos y operaciones a Internet. (Carrasco, 2013)

Las principales características de este servicio son:

- Herramienta para desarrolladores novatos.
- Centro de desarrollo con tutoriales, guías para las APIs, videos y biblioteca para conectar los distintos equipos.
- Centro de aprovisionamiento.
- Servicio comercial orientado a empresas que requieran un soporte dedicado para su propia Internet de las Cosas.

La plataforma permite publicar los datos recogidos por distintos sensores (como pueden ser sensores de humedad, temperatura, gases, luminosidad, radiación, etc.) mediante gráficas en tiempo real y widgets. (Corvis, 2013)

2.10.1 Envío de Datos a la Plataforma

Xively provee librerías de manera open source, gratuitas y abiertas a todos, además de tutoriales y documentación que nos permite conectarnos a Xively utilizando el hardware que queramos, en este caso Raspberry Pi, y los lenguajes de programación que se escojan.

Para facilitar las cosas, Xively se alía cada vez con más plataformas de hardware de una gran variedad de empresas, lo cual nos permite elegir entre una gran cantidad de dispositivos.

Las librerías permiten manipular los estandarizados APIs sobre MQTT, que es un protocolo de comunicación entre dispositivos, Websockets y HTTP, para hacer que la conexión a Internet de las Cosas sea simple, intuitiva y rápida.

Xively también facilita la actualización de los dispositivos, realiza diagnósticos remotos y provee un mejor soporte y mantenimiento. Con la misma herramienta existe la posibilidad de acceder de forma remota a los dispositivos o aplicaciones que necesiten control o reparación, aumentando así de manera exponencial el alcance del producto.

2.11 Android

2.11.1 Conceptos básicos del sistema operativo

Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo tabletas, notebooks, reproductores de música, televisores e incluso PC's.

Android permite programar un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución).

Además, lo que le diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre. (Báez, et al.)

Este sistema operativo fue desarrollado por Android Inc, empresa que en 2005 fue comprada por Google, pero no es hasta octubre del 2008 cuando sale al mercado el primer teléfono que ejecuta Android.

Android está formado por alrededor de 12 millones de líneas de código, de estas, 2.8 son lenguaje C, 2.1 de lenguaje Java, 1.75 de lenguaje C++ y 3 son de XML. La estructura de este sistema operativo se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos. (Mejía, 2012)

La arquitectura del SO está compuesta por cuatro capas, la primera de ellas es un kernel basado en Linux, le siguen las bibliotecas entre las que se encuentran las básicas correspondientes a la máquina virtual, a continuación está el marco de aplicaciones o framework y finalmente las aplicaciones. (Ilustración 12)



Ilustración 12. Las capas de la arquitectura de Android.

A través de los años Android se ha ido popularizando, hasta abarcar el mercado casi en su totalidad, en el tercer trimestre de 2015 se vendieron 353 millones de smartphones en el mundo, 15,5% más que en igual período del año anterior.

Si se los clasifica de acuerdo al sistema operativo, el amplio dominador del mercado es Android, utilizado por el 84,7% de los dispositivos vendidos, según datos de Gartner Inc. (Ilustración 13)

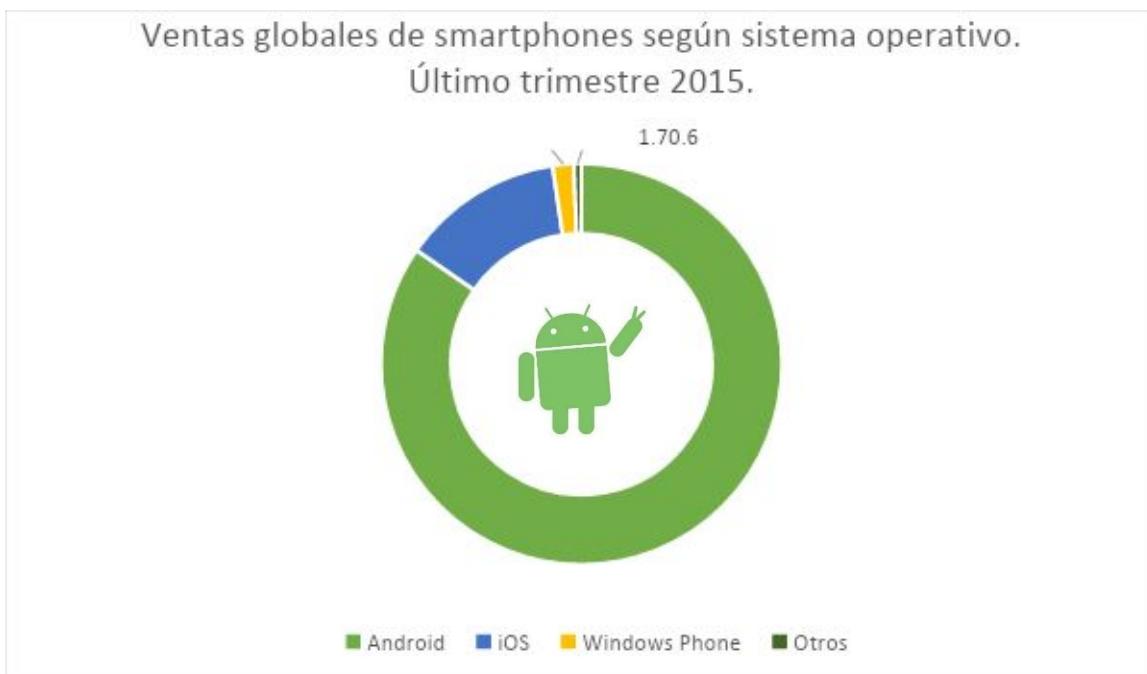


Ilustración 13. Gráfica mostrando la distribución de ventas de dispositivos móviles según su sistema operativo. Fuente: Gartner Inc.

2.11.2 Versiones de Android

Desde sus inicios, Android ha ido mejorando su sistema operativo, haciéndolo más amigable, más eficiente y eficaz en cada versión que lanzan al mercado.

Hasta la fecha, se han lanzado al mercado 11 versiones de Android, sin contar las previas a la primera versión comercializada, de Android 1.5 a Android 6.0 han pasado seis años en los que se han dado a conocer muchos nombres.

Algo que caracteriza a las versiones de Android es que a cada una de ellas le dan un nombre de dulce, que es como habitualmente se le conoce al software de modo general, y en orden alfabético.

Tabla 2. Tabla con las versiones y características de Android. Fuente: Xataka.com (Espeso, 2015)

Versión	Lanzamiento	Características Principales
---------	-------------	-----------------------------

Apple Pie 1.0	Octubre 2008	Incluyó la primera versión de la Android Market, un Navegador Web, soporte para mensajes de texto SMS y MMS, discador para llamadas, y una aplicación para tomar fotos.
Banana Bread 1.1	Febrero 2009	Entre sus novedades se encontraban el soporte para marquesina en diseños de sistemas, la posibilidad de guardar los archivos adjuntos en los mensajes, y las reseñas al buscar negocios en los mapas.
Cupcake 1.5	Abril 2009	Se introdujo el teclado en pantalla y la posibilidad de insertar widgets. Copiar y pegar en el navegador, grabación y reproducción de videos, la capacidad de subir videos a Youtube, auto-rotación y auto-sincronización.
Donut 1.6	Junio 2009	Trajo consigo el cuadro de búsqueda Google Search, la diversidad en el tamaño de pantallas que iban saliendo al mercado, se introdujo el Android Market,
Eclair 2.0	Octubre 2009	Integró las redes sociales y la sincronización con Facebook y Twitter, mejoró el menú de contactos, se mejoró la interfaz de usuario, la cámara, el teclado y se dio soporte Multi-Touch. Además se introdujo la navegación con Google Maps y el speech-to-text.
Froyo 2.2	Mayo 2010	Incorpora el motor de Java V8 y ofrece a los usuarios un aumento de velocidad con el compilador JIT. Se introduce la posibilidad de ser puerto de anclaje, se da soporte a Adobe Flash, pasar aplicaciones a la tarjeta microSD.

Gingerbread 2.3	Diciembre 2010	Se mejoró la estética e incrementó la velocidad y simpleza, se adaptó para teléfonos con doble núcleo y pantallas más grandes. Se mejoró el teclado, y el manejo de funciones por voz, se introdujo el NFC y el manejo de la batería.
Honeycomb 3.0	Febrero 2011	Fue la primera versión en adaptarse a tabletas, se introdujo la barra de sistema, que permite el acceso rápido a notificaciones y configuración básica.
Ice Cream Sandwich 4.0	Octubre 2011	Se introdujo el diseño Holo y mejoró la interfaz de usuario con botones propios del sistema. Además el dock de aplicaciones mostraba también los widgets, y la posibilidad de acceder a ciertas aplicaciones desde la pantalla de bloqueo. Se agregó un control para el consumo de datos.
Jelly Bean 4.1	Junio 2012	Mejoró la estabilidad, funcionalidad y rendimiento de la interfaz de usuario, se mejoró la barra de notificaciones. Se introdujo Google Now y la posibilidad de utilizar varias cuentas de usuario en un solo dispositivo.
KitKat 4.4	Octubre 2013	Se redujo el tamaño del sistema operativo, se agregó el “OK Google”, y las barras desaparecerían en ciertas aplicaciones de pantalla completa. Fue una de las versiones que más cambios ha introducido a nivel estético.
Lollipop 5.0	Noviembre 2014	Se presentó el Material Design, en donde se renovó la estética e interfaz de usuario. Se renovó el teclado. Nuevo control de batería y notificaciones. Soporte para procesadores de 64 bits.
Marshmallow 6.0	Septiembre 2015	Se introduce Google Now on Tap, permiso de aplicaciones, mejoras en la gestión de batería, manejo de huellas dactilares, USB tipo C, y mejoras en funciones internas.

PAGE
1
MERGE
OR
MA

PAGE
1
MERGE
OR
MA

Capítulo 3. DISEÑO E IMPLEMENTACIÓN DEL HARDWARE

Este capítulo corresponde a la descripción de los procesos realizados para construir el dispositivo físicamente.

3.1 Descripción del sistema

El sistema está compuesto por cuatro partes esenciales: los sensores, un adaptador Raspberry PI para sensores CT, una Raspberry PI y la interfaz inalámbrica.

Los sensores utilizados para el funcionamiento del dispositivo son los siguientes:

- Sensores CT: con los cuales se calcula el consumo eléctrico a partir de la medición de corriente eléctrica y previa medida de voltaje.
- Sensor de Temperatura Interno de la Raspberry PI: con la función de monitorear la temperatura interna del dispositivo Raspberry PI.

El componente principal del dispositivo es la Raspberry Pi, en ella está conectado el sensor de temperatura, se conectan los sensores CT a través del adaptador correspondiente, y se logra una conexión a internet inalámbrica mediante un adaptador WiFi.

La siguiente figura muestra el diagrama de los componentes principales del dispositivo y su funcionamiento.

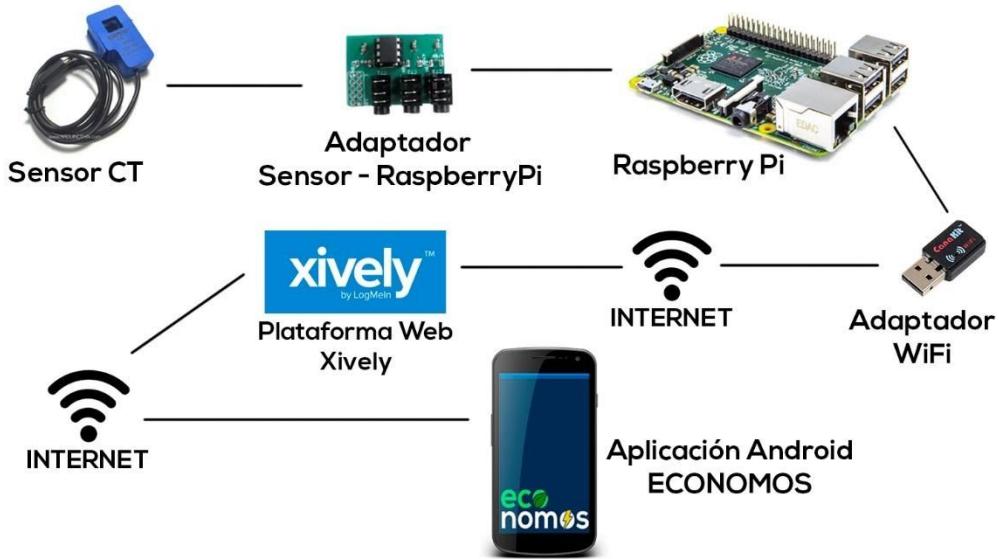


Ilustración 14. Diagrama de Componentes del Dispositivo y su funcionamiento.

3.2 Conexión del sensor CT

Para poder extraer los datos medidos por los sensores CT, estos deben estar conectados a un adaptador debido a que el puerto Jack 3.5 mm de la Raspberry PI es exclusivamente para salida de audio.

El adaptador para sensores CT es un sensor RPICT3 manufacturado por LeChacal.com (Ilustración 15). Este cuenta con tres conexiones Jack 3.5 mm para sensores CT. Estos puertos están conectados a un microcontrolador Arduino programable ATTINY 85.

Este microcontrolador lleva instalado un programa Arduino que calcula la potencia en Watts desde la medición de corriente del sensor CT (código fuente en anexos).

El microcontrolador calcula la potencia utilizando un valor de 240 V como valor fijo de voltaje. Debido a esto, en caso de que se conecte los dispositivos que funcionen con un voltaje diferente a 240 V, se necesitaría hacer cambios en el firmware de la placa ATTINY 85 o recalculando el valor enviado por el adaptador de sensor CT RPICT3 en un software exterior mediante la ecuación $P = V \cdot I$ (*Potencia es igual a voltaje por intensidad de corriente*).

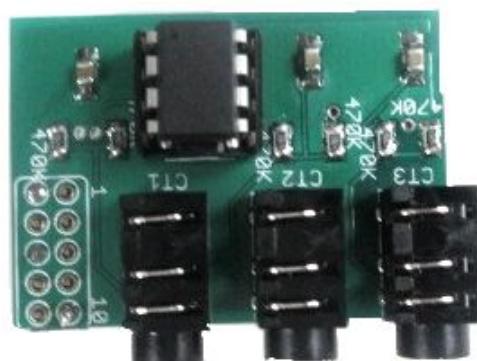


Ilustración 15. Adaptador RPICT3 manufacturado por LeChacal.com

3.3 Interfaz inalámbrica WiFi

Se utilizó un adaptador de red WiFi tipo USB 2.0 (Ilustración 16), el mismo al ser conectado a la placa Raspberry Pi funciona como una tarjeta de red en un computador convencional.

Nuestro adaptador soporta hasta 150 Mbps bajo el estándar 802.11 g/b/n, el cual permite al usuario un alcance más lejano y una cobertura más amplia.

El mismo funciona con la tecnología Plug & Play, al ser introducido en la placa se puede utilizar de inmediato, sin necesidad de ningún tipo de configuración

previa, sólo se introduce el adaptador USB, se conecta a la red más cercana y de inmediato se puede comenzar a navegar desde la Raspberry Pi.

El mismo provee dos tipos de funcionamiento, como infraestructura y como Ad-Hoc.



Ilustración 16. Adaptador WiFi USB optimizado para Raspberry Pi.

Nuestro adaptador WiFi de red es una de las piezas fundamentales, de gran importancia dentro de nuestro proyecto, el mismo es utilizado principalmente para obtener una conexión a internet estable para enviar los datos que son leídos por la placa hacia el servidor de Xively.

Estos datos serán los que reconozca el sensor CT, los mismos serán enviados a la placa a través del adaptador, y la placa, con la ayuda del programa, lo enviará a la plataforma Xively a través de la conexión de internet que nos brinda nuestro adaptador de red WiFi.

PAGE
1
MERGE
OR
MA

Capítulo 4. DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE

Este capítulo describe los pasos realizados para el funcionamiento a nivel de software del dispositivo.

4.1 Instalación del Sistema Operativo

Se decide utilizar Raspbian con nuestra Raspberry Pi, ya que es una de los sistemas operativos más nuevos que funcionan con la placa, y nos brinda la posibilidad de utilizar un entorno gráfico amigable, es una de las versiones más extendidas y con mejor soporte.

Raspbian es una distribución de Linux, basada en Debian, adaptada para trabajar específicamente con la Raspberry Pi. En su página web se definen a sí mismos como “Un sistema operativo gratuito basado en Debian optimizado para el hardware Raspberry Pi.

Nuestro sistema operativo es un conjunto de programas básicos y utilidades que hacen que tu Raspberry Pi funcione. Raspbian es más que un sistema operativo, el mismo viene con más de 35 mil paquetes y softwares pre-compilados organizados para una fácil instalación en la placa.”

Para la instalación de nuestro sistema operativo necesitaremos una tarjeta SD de al menos 8 GB, la cual hará el papel del disco duro en una computadora normal, ahí se almacenará la imagen del sistema operativo, en este caso Raspbian, el cual puede ser descargado de manera gratuita en la web oficial de Raspberry Pi. Se decidió utilizar la edición Raspbian Jessie, ya que es la más recomendada y es la última edición estable.

Al momento de descargar la imagen a la tarjeta SD ya es posible introducirla a la placa. Al conectarla comenzará el proceso de instalación automáticamente, nos saldrá una ventana en donde escogeremos cuál sistema queremos instalar, al escogerlo se comenzará a instalar Raspbian. (Ilustración 17)

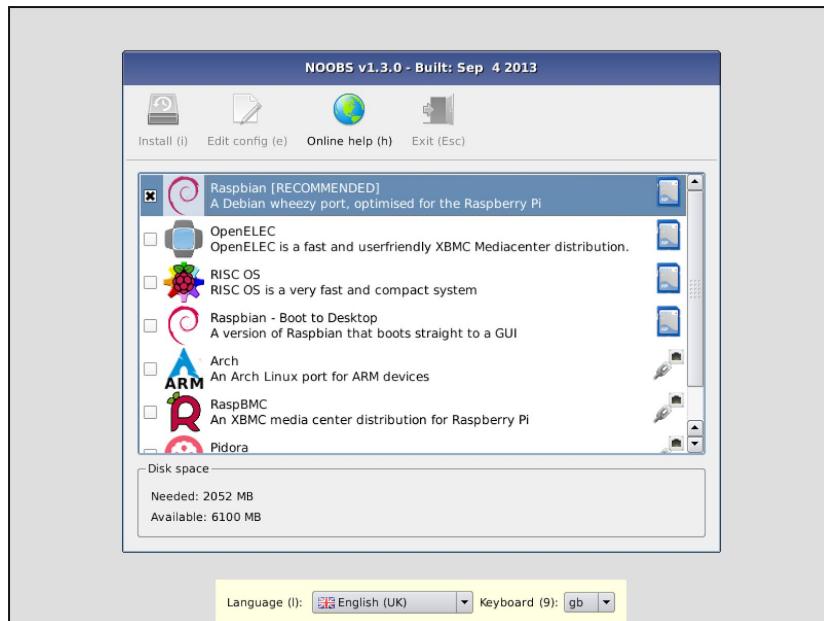


Ilustración 17. Primera ventana que aparece al conectar la Raspberry Pi. Se elige el sistema operativo que se encuentre en la tarjeta SD.

Al momento de la instalación se configuran automáticamente los componentes del hardware y software, y se instalan los paquetes de utilidades. En esta versión de Raspbian viene pre instalados entornos de desarrollo para Phyton, Java, C. Utilidades como LibreOffice, calculadora, Wolfram, entre otros.

Lo primero que se puede observar al iniciar la Raspberry Pi es su interfaz gráfica amigable, con ventanas y menús, muy parecida a lo que estamos acostumbrados a ver en Windows o en sistemas operativos más complejos. (Ilustración 18)

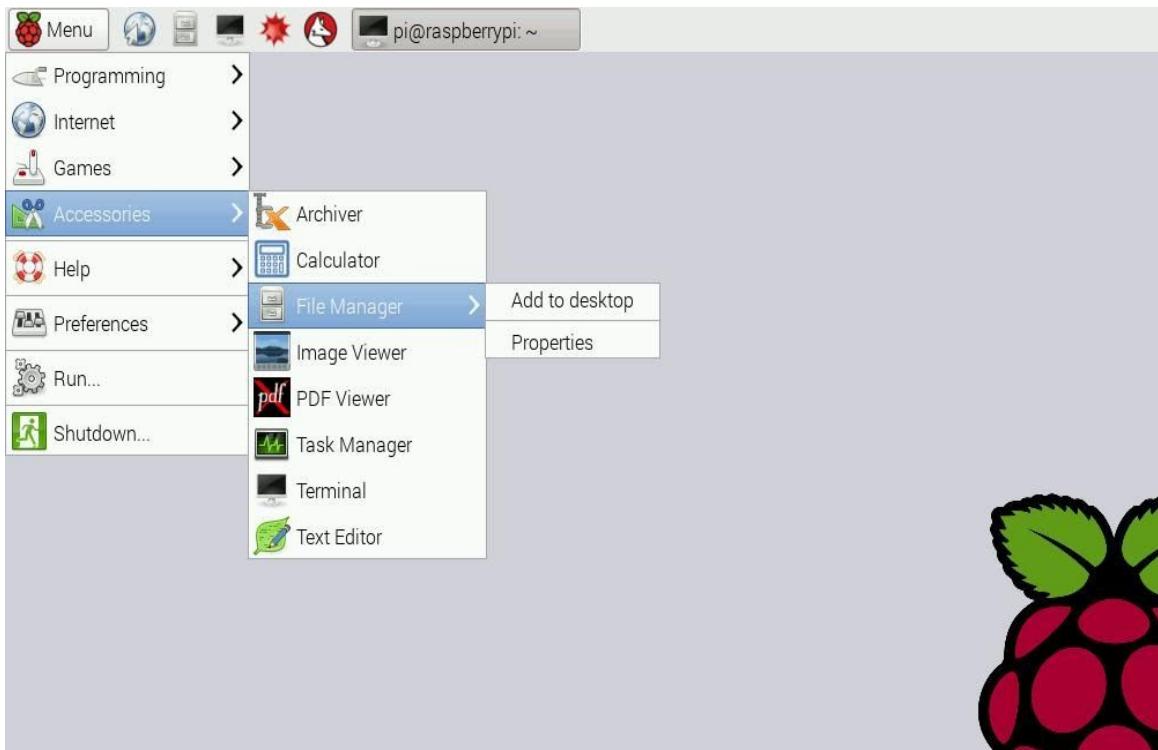


Ilustración 18. El escritorio de Raspbian Jessie, con sus ventanas y programas pre instalados.

4.2 Registro del Dispositivo en la Plataforma Xively

Para el uso de la plataforma Xively como servidor de datos, es necesario el registro previo del dispositivo a conectarse.

En nuestro caso es necesario registrar la Raspberry Pi como dispositivo conectado a Xively. Luego de registrar el dispositivo en la plataforma, esta asigna algunos datos únicos y privados para el dispositivo, adicionalmente se crea por defecto una API Key.

Las API Key, como su nombre lo dice, es una llave o clave formada por un conjunto de caracteres que son usados para controlar el acceso desde cualquier interfaz de programación de aplicaciones. Puede existir distintas API Key para

un mismo dispositivo en la plataforma Xively, esto con el fin de delimitar el acceso a los datos o feeds desde distintas direcciones IP.

Estas API Key son utilizadas para realizar la conexión remota desde otros dispositivos que quieran acceder a los datos en la plataforma. En nuestro caso, el programa codificado en Python en la Raspberry Pi y la aplicación Android necesitarían una API Key para conectarse mediante el protocolo HTTP.

4.3 Codificación de Programa Python para el Procesamiento de Datos en el Sistema Operativo

Para lograr enviar los datos a la plataforma Xively, en la cual se van a almacenar la lectura de datos en tiempo real, se necesita la ayuda de un programa extra que realice la lectura de datos desde el puerto serial (al cual están conectados los sensores CT).

Este programa debe realizar la conexión a la plataforma Xively, y al mismo tiempo debe enviar los datos a la plataforma para su almacenamiento y posterior uso.

En base a que el sistema descansa sobre el sistema operativo Raspbian Jessie, que es un sistema basado en Linux, se decidió codificar el programa para el procesamiento de datos utilizando el lenguaje de programación Python.

4.3.1 Instalación de la Librería Python de Xively

Se necesita de una librería llamada **xively**, que provee Xively para realizar la conexión desde Python a la plataforma.

La librería es de código abierto y se encuentra en la plataforma GitHub, y se instala mediante los siguientes pasos:

1) Instalación del software de control de versiones de GitHub para GNU/Linux

Para la instalación de la librería se necesita tener instalado en el sistema operativo el software *git*. Utilizado por GitHub como software de control de versiones.

La instalación de *git* se realiza mediante el comando `apt-get install git`.

2) Clonar la librería de Xively-Python desde GitHub al Sistema Operativo

Luego de tener instalado el software *git*, se puede proceder a clonar la librería xively-python en el sistema operativo mediante el comando `git clone [dirección de repositorio]` utilizando la dirección del repositorio de la librería

3) Instalar la Librería

Para finalizar la instalación se utiliza el comando `python setup.py install`.

4.3.2 Conexión con Plataforma Xively

Para realizar la conexión con la plataforma Xively desde Python se necesita el API Key que provee la plataforma al momento de registrar el dispositivo, y la conexión se realiza mediante la función

`xively.XivelyAPICliente([APIKey])` de la librería Xively instalada desde GitHub.

A continuación se muestra un segmento del código Python donde se observa la conexión a la plataforma.

4.3.3 Lectura de Datos de los Sensores CT

Para realizar la lectura de los datos debe conocerse el formato con el cual están siendo enviados al puerto serial antes de realizar la lectura en el programa Python.

4.3.3.1 Formato de los Datos

El adaptador de sensores CT envía cada cierto tiempo los datos medidos por los sensores a la interfaz serial de la Raspberry, estos pueden leerse desde el terminal con los siguientes comandos:

```
$ stty -F /dev/ttyAMA0 raw speed 38400  
$ cat /dev/ttyAMA0
```

Con el comando **stty** se indica el nombre del puerto serial de la Raspberry Pi, en nuestro caso **/dev/ttyAMA0** y la velocidad en baudios de la conexión, en esta ocasión siendo de 38400.

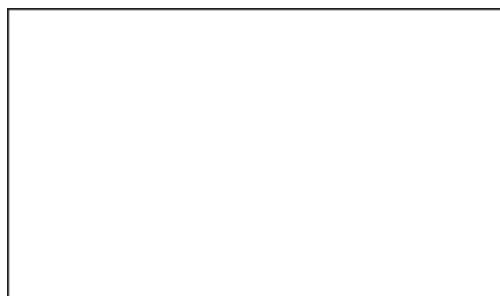
Se utiliza el comando `cat` para desplegar la información actual del puerto serial en pantalla.

El adaptador de sensores CT envía los datos al puerto serial con el siguiente formato:

```
ID 1A 1B 2A 2B 3A 3B
```

- **ID:** representa el nodo asignado en el software del adaptador, es único y no cambia.
- **1A y 1B:** representan el valor codificado de un sensor conectador en el puerto 1 del adaptador.
- **2A y 2B:** representan el valor codificado de un sensor conectador en el puerto 2 del adaptador.
- **3A y 3B:** representan el valor codificado de un sensor conectador en el puerto 3 del adaptador.

Para decodificar los valores medidos por cada sensor se debe seguir la siguiente ecuación para cada uno de ellos, la ecuación representa el cálculo del consumo en Watts para el sensor conectador en el puerto 1 del adaptador:

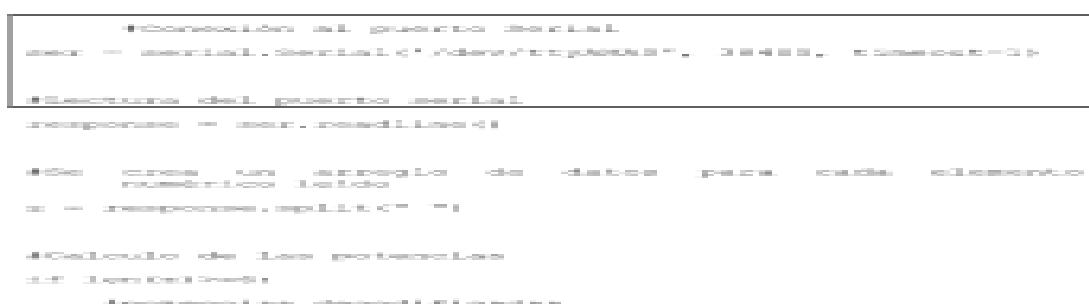


4.3.3.2 Decodificación de los datos

La ecuación mostrada anteriormente debe efectuarse cada vez que se quieran leer los datos del puerto serial de la Raspberry Pi.

A continuación se muestra un segmento de código con el cual se realiza la lectura del puerto serial y la conversión de los datos a su equivalente en consumo eléctrico.

El segmento de código realiza la lectura del puerto serial y calcula el valor de las potencias para los sensores conectados en los puertos 1 y 2 del adaptador de sensores CT.



4.3.4 Lectura de Datos del Termómetro Interno de la Raspberry Pi

Para la lectura de los datos medidos por el termómetro interno de la Raspberry Pi, debe utilizarse una línea de comandos del sistema operativo. Para utilizar estos comandos en el programa Python era necesario importar la librería `os` de la biblioteca estándar de Python.

El módulo `os` nos provee de distintas funciones para interactuar directamente con el sistema operativo, en nuestro caso utilizamos la función `popen` para

ejecutar el comando `vcgencmd measure_temp` que retorna el valor medido de la temperatura.

En las siguientes líneas de códigos se muestra la definición de la función para la lectura del termómetro interno desde un programa Python.

Al final del código se utiliza la función `res.replace` para reemplazar los primeros cinco caracteres de la respuesta del sistema operativo y dejar solamente el valor en números.

```
import os

# Funcion para obtener el valor de la temperatura del SoC

def getTemp():
    res = os.popen('vcgencmd measure_temp').readline()
    return(res.replace("temp=", "") .replace("°C\n", ""))
```

4.3.5 Envío de Datos a la Plataforma Xively

El envío de datos a la plataforma Xively se realiza mediante la creación de distintos canales para el envío de datos a un feed del dispositivo. Los dispositivos conectados a la plataforma únicamente cuentan con un sólo feed, y este es identificado por un Feed ID que se utiliza para crear canales de transmisión desde otras APIs.

Para el envío de datos a la plataforma se necesita de dos procesos, crear los canales de transmisión y luego actualizarlos.

4.3.5.1 Creación u Obtención de los Canales de Transmisión

Para nuestro caso era necesario la creación de tres canales de transmisión, dos para los valores de dos sensores CT conectados y uno para la lectura de la temperatura de la Raspberry Pi.

Los canales de transmisión también pueden ser creados en la plataforma Xively desde la página web correspondiente al perfil de dispositivo vinculado. Por lo tanto en caso de que ya se haya creado un canal de transmisión, sólo basta obtenerlo desde la plataforma para actualizarlo después.

A continuación se muestra un segmento de código que crea un canal de transmisión llamado “Power1”. El código consta de una función para el canal de trasmisión (datastream) en la cual mediante una estructura **try except**, se crea o se obtiene un canal de transmisión (si el canal está creado se obtiene, si el canal no existe se crea).

```
import xively  
  
FEED_ID = "1898014877"  
  
# Funcion para crear canal de datos de Powerl  
  
def create_datastreams_powerl(feed):  
  
    try:  
  
        datastream = feed.datastreams.get("Powerl")  
  
    return datastream
```

Se utilizan las funciones `feed.datastreams.get` o `feed.datastreams.create` para obtener o crear canales de transmisión respectivamente.

Al final del código se utiliza la función `api.feeds.get` para asignarle el valor del feed a una variable para su posterior uso en el llamado a la función.

4.3.6 Ejecución Automática del Programa Python en el Sistema Operativo

Para mantener la plataforma Xively actualizada en tiempo real con los valores leídos y procesados por el programa Python, se debe procurar que éste se ejecute iterativamente en el sistema operativo.

Antes de poder ejecutarlo mediante el sistema operativo de manera automática, el archivo a ejecutarse debe tener permisos de ejecución. Este permiso se le asigna mediante el comando:

```
$ sudo chmod +x [nombre del archivo]
```

Cron es un administrador regular de procesos. Este se encarga de ejecutar líneas de comandos o procesos iterativamente cada cierto intervalo de tiempo. Para esto se modificó el archivo `crontab` en el sistema operativo.

En nuestro caso el programa en Python se ejecutaría cada minuto luego de insertar el siguiente comando al abrir el fichero `crontab` en el terminal mediante la instrucción `crontab -e`.

```
*/1 * * * * python [dirección del archivo python]/[nombre  
del archivo python].py
```

El 1 representa un minuto de intervalo de ejecución para el archivo ubicado mediante su dirección y nombre en el sistema operativo.

Cada minuto se ejecuta el programa Python y actualiza los datos en la plataforma Xively.

4.4 Desarrollo de la aplicación Android

A continuación presentamos los pasos llevados a cabo para la elaboración de la aplicación Android cuya función es acceder a los datos de la plataforma Xively y presentarlos en tiempo real en el dispositivo móvil.

4.4.1 Conexión a la plataforma Xively

La plataforma Xively utiliza el protocolo HTTP para comunicarse con otros dispositivos vía internet. Mediante este protocolo se manipulan los datos del dispositivo de manera remota.

4.4.1.1 Importación de la Librería org.apache.http.legacy

La librería `org.apache.http.legacy` es utilizada para manipular las peticiones HTTP que se realicen desde el dispositivo. Sin esta librería el dispositivo no estaría habilitado para comunicarse con la plataforma Xively vía internet.

La librería se importa escribiendo el siguiente comando directamente en **dependencies{}** en el archivo **build.gradle** del proyecto Android. A continuación se muestra la línea de código que permite importar la librería.

```
dependencies {  
  
    /*Se importa la librería de org.apache.http para  
     * manipular las peticiones HTTP*/  
    android {  
        useLibrary 'org.apache.http.legacy'  
    }  
}
```

Con esta librería se creó una clase que manipulara la conexión con la plataforma.

4.4.1.2 Creación de clase para Manipular Peticiones

Como la única función de la aplicación Android es leer datos de la plataforma Xively, en la clase para manipular datos de la misma se necesita una función capaz de acceder a los datos utilizando como parámetros la **url** permanente del dispositivo suministrada por Xively y su **APIKey**.

Para obtener los datos debe realizarse una petición HTTP a la **url** del dispositivo enviando en el encabezado su **APIKey**. A continuación se presenta la función **getFeed** con el manejo de excepciones.

```

public String getFeed(String url, String apikey) {

    /*Se trata de realizar la conexión*/
    try {
        /*Datos del Cliente HTTP*/
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpEntity httpEntity = null;
        HttpResponse httpResponse = null;

        /*Se especifica la función Get de la Petición*/
        HttpGet httpGet = new HttpGet(url);

        /*Se almacena la APIKey en el encabezado de la petición*/
        httpGet.addHeader("X-ApiKey", apikey);

        /*Se ejecuta la petición*/
        httpResponse = httpClient.execute(httpGet);
        httpEntity = httpResponse.getEntity();

        /*Se almacena la respuesta en la variable*/
        response = EntityUtils.toString(httpEntity);

        /*Manejo de Excepciones*/
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return response;
}

```

4.4.2 Lectura de Datos desde Xively

Para acceder a los datos desde la clase principal de la aplicación Android para posteriormente visualizar en pantalla los datos elegidos a mostrar, se debe efectuar una llamada a la función anterior y procesar la respuesta.

La plataforma Xively utiliza JSON (JavaScript Object Notation) como formato de intercambio de datos. Por lo que es conveniente utilizar la librería org.json para el manejo de los datos.

La respuesta de la plataforma Xively a la petición HTTP con una APIKey sin restricciones presenta los siguientes datos:

Tabla 3. Tabla de Datos JSON de una petición HTTP

Campo (TAGS)	Descripción
Datos del Dispositivo	
“ id ”	Número de identificación del dispositivo
“ title ”	Título del Dispositivo
“ private ”	“ True ” para dispositivos privados y “ False ” para dispositivos públicos
“ feed ”	URL más el número de Feed. Ejemplo: “ https://api.xively.com/v2/feeds/1601.json ”
“ status ”	“ live ” para dispositivo conectado actualmente a Xively y “ frozen ” para dispositivos que no han actualizado los campos de la plataforma después de 15 minutos.
“ updated ”	Fecha y hora de la última actualización
“ created ”	Fecha y hora de creación
“ creator ”	Dirección del perfil creador del dispositivo
“ location ”	Localización del dispositivo
“ version ”	Versión del dispositivo
Campos de un Canal de Transmisión (Datastream)	
“ id ”	Identificación del campo
“ current_value ”	Valor
“ tag ”	Unidades de medición para el valor del datastream
“ at ”	Fecha y hora de la última actualización
“ max_value ”	Valor máximo capturado

“min_value”	Valor mínimo capturado
--------------------	------------------------

De los valores anteriores en un arreglo de datos JSON se seleccionan los valores del título del dispositivo, estado del dispositivo, última actualización como valores generales del dispositivo y el campo de valor actual para las unidades de los datastream.

A continuación se muestra el segmento de código que accede a los datos del objeto JSON obtenido después de la petición, en donde las constantes con sufijo **TAG_** corresponden al valor correspondiente del campo mostrado de la tabla anterior.

```

@Override
protected Void doInBackground(Void... arg0) {
    // Instancia de la Clase
    HTTPHandle sh = new HTTPHandle();

    String jsonStr = sh.getFeed(url, API_KEY);

    // En caso de obtener respuesta
    if (jsonStr != null) {
        try {
            JSONObject jsonObj = new JSONObject(jsonStr);

            /*Generales del dispositivo*/
            title = jsonObj.getString(TAG_TITLE);
            status= jsonObj.getString(TAG_STATUS);
            updated= jsonObj.getString(TAG_UPDATED);

            /*Obteniendo el nodo de datastreams*/
            JSONArray jsontDataStream =
            jsonObj.getJSONArray(TAG_DATASTREAMS);

            /*JsonObject de Power 1*/
            JSONObject c = jsontDataStream.getJSONObject(0);
            power1=c.getString(TAG_CURRENT_VALUE);

            /*JsonObject de Power 2*/
            c = jsontDataStream.getJSONObject(1);
            power2=c.getString(TAG_CURRENT_VALUE);

            /*JsonObject de Temperatura*/
            c = jsontDataStream.getJSONObject(2);
            temperatura=c.getString(TAG_CURRENT_VALUE);

        } catch (JSONException e) {
            e.printStackTrace();
        }
    } else {
        internet=false;
        //Se visualiza mensaje de error en terminal
        Log.e("ServiceHandler", "Couldn't get any data from the
url");
    }
}

```

4.4.3 Creación de registros de lectura del medidor

Se utilizó el siguiente diseño para la visualización de los datos en el prototipo (código XML en anexos).

La siguiente figura ilustra el diseño del prototipo funcionando en el emulador del entorno de Android Studio.



Ilustración 19. Diseño del prototipo de la aplicación Android.

Capítulo 5. PRUEBAS Y RESULTADOS

A continuación se presentan los resultados de las pruebas realizadas.

5.1 Verificación de las lecturas de mediciones

Para la verificación de las mediciones se hicieron pruebas comparando el consumo medido por el dispositivo versus el consumo calculado a partir de una medición utilizando un amperímetro convencional. En la siguiente figura se muestra el amperímetro utilizado para las pruebas.



Ilustración 20. Lectura de Amperaje con un Amperímetro Convencional.

El proceso consistió en medir la corriente de un hogar pequeño utilizando el dispositivo y revisando el consumo con el amperímetro periódicamente. A partir de la medición de corriente del amperímetro calculamos el consumo neto utilizando un voltaje de 120 V. A partir de este cálculo comparamos estos valores con los valores visualizados en la aplicación Android.

5.1.1 Recolección de Datos

Se recopilaron 36 mediciones para realizar el estudio del error absoluto y relativo del dispositivo.

Tabla 4. Tabla de Valores de Pruebas Realizadas

Medición	Amperaje Medido [I]	Consumo Calculado [W]	Consumo Economos [W]
1	4.46	535.20	596.50
2	4.54	544.80	596.50
3	4.41	529.20	596.50
4	3.96	475.20	596.50
5	3.96	475.20	596.50
6	3.07	368.40	428.50
7	3.07	368.40	428.50
8	3.04	364.80	427.50
9	3.04	364.80	427.50
10	3.04	364.80	427.50
11	3.02	362.40	427.50
12	3.00	360.00	427.50
13	3.00	360.00	427.50
14	2.98	357.60	427.50
15	2.99	358.80	427.50
16	3.00	360.00	427.50
17	3.10	372.00	427.50
18	11.00	1320.00	1271.00
19	10.30	1236.00	1203.00
20	11.60	1392.00	1276.00

21	12.58	1509.60	1548.00
22	3.83	459.60	533.00
23	3.83	459.60	506.00
24	3.60	432.00	360.00
25	3.60	432.00	377.00
26	12.30	1476.00	1249.00
27	3.84	460.80	503.00
28	12.69	1522.80	1557.00
29	3.75	450.00	500.00
30	12.80	1536.00	1560.50
31	3.08	369.60	427.00
32	5.34	640.80	707.50
33	5.30	636.00	707.50
34	5.26	631.20	697.00
35	5.28	633.60	697.50
36	5.13	615.60	697.50

Con los datos anteriores podemos calcular el error relativo en las mediciones realizadas por el dispositivo. La siguiente tabla muestra los valores calculados.

Tabla 5. Errores Relativos Porcentuales en Cada Medición

Consumo Economos (W)	Error Relativo Porcentual (%)
596.50	11.45
596.50	9.49
596.50	12.72
596.50	25.53
596.50	25.53
428.50	16.31
428.50	16.31
427.50	17.19
427.50	17.19
427.50	17.19
427.50	17.96
427.50	18.75
427.50	18.75
427.50	19.55
427.50	19.15

427.50	18.75
427.50	14.92
1271.00	3.71
1203.00	2.67
1276.00	8.33
1548.00	2.54
533.00	15.97
506.00	10.1
360.00	16.67
377.00	12.73
1249.00	15.38
503.00	9.16
1557.00	2.25
500.00	11.11
1560.50	1.60
427.00	15.53
707.50	10.41
707.50	11.24
697.00	10.42
697.50	10.09
697.50	13.30

A partir de esta tabla podemos concluir que el error relativo del dispositivo en comparación con el valor que debe medir puede ser muy grande, lo que lo hace ineficaz en algunos casos.

5.1.2 Comparación del Error Relativo versus el Consumo

Luego de realizar un pequeño análisis a la tabla 5, logramos detectar que puede existir una correlación negativa entre el error relativo y el valor medido por el dispositivo.

Para comprobar esto realizamos la prueba de correlación lineal de Pearson con los datos de la tabla anterior, dando como resultado un coeficiente de -0.74, lo que corresponde a una relación negativa moderada entre las variables. La siguiente gráfica muestra la relación entre ambas variables.

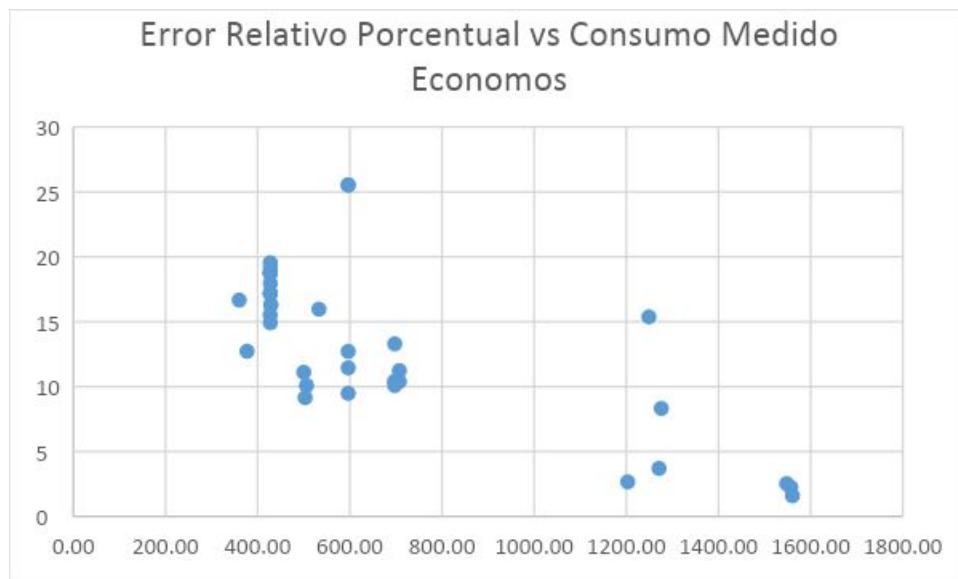


Ilustración 21. Gráfico de Correlación de Pearson entre Error Relativo Porcentual y Medición del Dispositivo.

Con esto podemos inferir que entre más alto sean los valores medidos por el dispositivo, más bajo será el error relativo. Esto hace al dispositivo más eficaz y exacto para mediciones grandes.

5.2 Tiempo de Conectividad

Adicionalmente en la etapa de prueba calculamos el tiempo de conectividad del dispositivo desde que se conecta.

El dispositivo después de conectarse tiene que realizar una medición, ejecutar el programa en Python y enviar los valores a la plataforma Xively, para luego, acceder a ellos desde la aplicación Android.

Después de conectarse el dispositivo al cableado, este toma alrededor de dos minutos en actualizar los datos en la aplicación Android. Lo que le da un buen rendimiento de conectividad.

PAGE
1
MERGE
OR
MA

CONSIDERACIONES FINALES

- El sistema de medición implementado permite monitorear de manera local y remota el consumo eléctrico del hogar utilizando una plataforma web orientadas al Internet de las Cosas y a través de un dispositivo móvil.
- Se determinó que el sensor CT cumple con los requerimientos para un sistema de medición y monitorización de consumo eléctrico, ya que utiliza la teoría electromagnética y las señales de voltaje del cable al que es conectado y así mide con cierta precisión la corriente y la potencia que pasan por el mismo.
- El sistema embebido Raspberry Pi puede ser utilizado para interactuar con cualquier sensor electrónico aplicando la ingeniería de hardware y software correspondiente. Además, al usar una distribución especial de Linux como sistema operativo, tenemos todos los beneficios del software libre y no requerir ninguna licencia comercial, así abaratando costos.
- El monitoreo de consumo eléctrico residencial permite al usuario contar con información segura, asequible y en tiempo real, en comparación con los sistemas tradicionales. El tener registros de consumo cada 2-3 minutos puede permitir construir gráficas estadísticas para poder apreciar mejor el consumo eléctrico en el día a día dentro del hogar y así tener bases en la toma de decisiones futuras, y le da al usuario la información necesaria para ahorrar dinero.
- El Internet de las Cosas como nueva tendencia tecnológica, es la gran oportunidad para monitorear y controlar remotamente dispositivos de toda clase, lo que abre un gran campo de nuevas oportunidades. En un par de años la mayoría de los dispositivos tradicionales que tenemos en nuestros hogares estarán conectados al internet.

- La investigación, desarrollo e implementación en nuevas tecnologías de sistemas embebidos resulta ser fundamental para el desarrollo de nuevos métodos de medición, monitoreo y / o control de parámetros en dispositivos, no sólo a nivel residencial, si no a futuro verse aplicado dentro del campo industrial.
- Los antiguos medidores mecánicos residenciales pueden ser fácilmente reemplazados por el sistema domótico de medición y monitoreo ya que utilizan la misma infraestructura existente.
- No se cuenta con una infraestructura de comunicación propia entre las partes del sistema, tampoco con servidores propios que almacenen la información y la pongan a disposición. Debido a que se utiliza una plataforma web privada, en este caso Xively, no se tiene el acceso y control completo del sistema en sí, pues no permite al administrador un control total y ni permite la solución de errores si ocurre algún fallo de comunicación o el servidor no está disponible.
- Es totalmente viable desarrollar una aplicación para dispositivos móviles Android para monitorear, visualizar y controlar los datos y la información captada por los sensores electrónicos. La aplicación nos permite ver en tiempo real los datos sin importar el lugar ni el momento, siempre y cuando tengamos una conexión a internet. También es posible desarrollar la misma aplicación para dispositivos iPhone y iPad.

PAGE
1
MERGE
OR
MA

RECOMENDACIONES

- El cálculo del consumo eléctrico del dispositivo se hizo mediante la ecuación de potencia utilizando el valor de corriente medido y un valor de voltaje constante para todas las mediciones. En algunos casos el voltaje en algunos hogares tiende a oscilar, lo que hace que las mediciones sean menos exactas. Para evitar esto, y el problema de tener que cambiar la ecuación dependiendo del voltaje que utilice el artefacto que quiera medirse, se pueden utilizar sensores de voltaje. En este caso el resultado del consumo será producto de dos valores medidos mediante sensores, dándole más exactitud al dispositivo.
- Tener una red de internet WiFi o de cableado Ethernet cerca del dispositivo para evitar problemas de conectividad.
- La Raspberry Pi puede ser vulnerable a intentos de infiltración, por lo que deben utilizarse contraseñas de autenticación, configurar el firewall correctamente, ya sea el del sistema operativo Raspbian o el de área local.
- Diseñar una aplicación capaz de dar reportes más detallados utilizando además, las tarifas para el consumo eléctrico de la zona.
- Al momento de culminar este proyecto investigativo salió al mercado un nuevo modelo de Raspberry Pi, el mismo viene integrado con un adaptador para conexión WiFi, lo que disminuye la cantidad de componentes que son necesarios adquirir para desarrollar proyectos de este tipo.
- Detectar un algoritmo capaz de detectar anomalías en el consumo para enviar alarmas al usuario.

- Utilizar procesos de gestión de calidad y pruebas más rigurosas para detectar deficiencias y proponer mejorías en el sistema.
- Es posible desarrollar una aplicación en el lenguaje Objective C para dispositivos móviles iPhone y iPads, para así alcanzar a un mayor número de dispositivos conectados con el proyecto.

PAGE
1
MERGE
OR
MA

REFERENCIAS BIBLIOGRÁFICAS

Asociación Española de Domótica. (2008). La Domótica Hoy. *Cómo ahorrar energía instalando domótica en su vivienda*, 28.

Asociación Española de Domótica. (2009). ¿Qué ofrece la domótica al sector hotelero? *CEDOM*, 48.

Asociación Española de Domótica. (2009). LA DOMÓTICA: SOLUCIONES PARA UN HOGAR ACCESIBLE. *Soluciones domóticas para un hogar accesible.*, 41.

ASOCIACIÓN ESPAÑOLA DE DOMÓTICA E INMÓTICA. (Mayo de 2009). *CEDOM*. Obtenido de Sobre Domótica: <http://www.cedom.es/sobre-domotica/que-es-domotica>

Báez, M., Borrego, Á., Cruz, L., González, M., Palomero, D., & Saucedo, M. (s.f.). *Introducción a Android*. Universidad Complutense de Madrid. Madrid: E.M.E. Editorial.

Basterra, Berte, Borello, Castillo, & Venturi. (2016). *Android OS Documentation*.

Benito, M. (Marzo de 2013). *JAVA y Android, una relación de amor-odio*. Obtenido de El Android Libre: <http://www.elandroidelibre.com/2013/03/java-y-android-una-relacion-de-amor-odio.html>

Broque, B. (8 de Marzo de 2015). *Arduino vs. Raspberry Pi: Mortal enemies, or best friends?* Obtenido de Digital Trends: <http://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/#ixzz3zuKOnR1Y>

Byous, J. (2005). *Java Technology. The Early Years*. California: Sun Developer Network.

Carrasco, M. (4 de Julio de 2013). *Xively, la nube especializada en Internet de las Cosas*. Obtenido de Internet de las Cosas: <http://www.internetdelascosas.cl/2013/07/04/xively-la-nube-especializada-en-internet-de-las-cosas/>

Casado Mansilla, D., & López de Armentia, J. (19 de Enero de 2015). *Revista Ingeniería*. Obtenido de Facultad de Ingeniería. Universidad de Deusto.: <http://revistaingenieria.deusto.es/tag/iot/>

Corvis, A. (15 de Noviembre de 2013). *Tutorial Raspberry Pi: Conexión a Xively (Internet Of Things)*. Obtenido de Geeky Theory: <https://geekytheory.com/conectando-la-raspberry-pi-a-xively-internet-of-things/>

Cristobal, R., Francisco, V., & Carlos, d. C. (2010). *Domótica e Inmótica. Viviendas y edificios inteligentes*. Madrid: RA-MA EDITORIAL.

Doutel, F. (24 de Agosto de 2015). *Raspberry Pi frente a Arduino: ¿quién se adapta mejor a mi proyecto maker?* Obtenido de Xataka: <http://www.xataka.com/makers/raspberry-pi-frente-a-arduino-quien-se-adapta-mejor-a-mi-proyecto-maker>

Dowey, A. (2012). *Think Python*. Nedham, Massachusetts: Green Tea Press.

ECMA International. (2013). *The JSON Data Interchange Format*. Génova: ECMA.

Espeso, P. (18 de Agosto de 2015). *Xataka*. Obtenido de De Cupcake a Marshmallow, así han sido las versiones de Android a lo largo de su historia:

<http://www.xatakamovil.com/sistemas-operativos/de-cupcake-a-marshmallow-asi-han-sido-las-versiones-de-android-a-lo-largo-de-su-historia>

Evans, B. W. (2007). *Arduino Programming Notebook*. San Francisco: Creative Commons.

Fernández, M. (16 de Mayo de 2008). La Domótica aplicada al Ahorro y la Eficiencia Energética. *Asociación Española de Domótica*. Barcelona, España: Asociación Española de Domótica.

Fernández, A. (21 de Julio de 2013). ¿Qué es Raspberry Pi y para qué sirve? ABC.

Fundación de la Energía de la Comunidad de Madrid. (2007). La Domótica como Solución de Futuro. *Cómo ahorrar energía instalando domótica en su vivienda*, 161.

Fundación de la Innovacion Bankinter. (2011). *El internet de las cosas. En un mundo conectado de objetos inteligentes*. España.

Galicia, C. (12 de Mayo de 2014). *Arduino o Raspberry Pi, ¿cuál es la mejor herramienta para ti?* Obtenido de Hacedores: <http://hacedores.com/arduino-o-raspberry-pi-cual-es-la-mejor-herramienta-para-ti/>

International Energy Agency. (2015). *Energy Balances of OECD Countries*. Paris: IEA Statistics.

Jackson, C. (2011). *Learning to program using Python*. CreateSpace Independent Publishing Platform.

Jones, D. M. (2015). *Python for complete beginners*. CreateSpace Independent Publishing Platform.

Jordan, W. (19 de Junio de 2015). Crece consumo y demanda de energía. *La Prensa*.

López, A. C. (2013). *Guía Raspberry Pi*. Creative Commons.

Madrid Network. (2013). *Internet de las Cosas: Objetos Interconectados y Dispositivos Inteligentes*. Madrid: Audiovisual de Madrid.

Maestro, J. A. (Abril de 2009). Domótica e Inmótica. *Diseño tecnológico. Electrónica y ocio*. Madrid, España: Universidad Antonio de Nebrija.

Mansilla, D., & López, J. (19 de Enero de 2015). *Revista Ingeniería*. (A. Zubillaga, Editor) Recuperado el 2015 de Diciembre de 7, de Facultad de Ingeniería de la Universidad de Deusto: <http://revistaingenieria.deusto.es/el-internet-de-las-cosas-y-la-sostenibilidad-medioambiental/>

Mejía, O. Á. (2012). *Android*. Madrid: UAM.

Monnk, S. (2013). *Raspberry Pi Cookbook*. Sebastopol, California: O'Reilly Media Inc.

Oracle. (2008). *Java Fundamentals. Java. A Beginners Guide*. California, Estados Unidos: Oracle.

Orghidan, R. (2012). Domótica. *Ingeniería en Informática de la Universidad de Girona*. Gerona, España: Universidad de Girona.

Pérez, E. (10 de Diciembre de 2012). *Todo sobre x86 y ARM: Diferencias, ventajas y para qué sirve cada uno*. Obtenido de Omicrono: <http://www.omicrono.com/2012/12/te-explicamos-todo-sobre-x86-y-arm-diferencias-ventajas-y-para-que-sirve-cada-uno/>

Richardson, M., & Wallace, S. (2012). *Getting Started With Raspberry Pi*. Sebastopol, California: O'Reilly Media Inc.

Rocamora, E. P. (2008). *Diseño y Simulación de un Sistema Domótico para una Vivienda Unifamiliar*. Cartagena, España: UNIVERSIDAD POLITÉCNICA DE CARTAGENA.

Rodriguez, A., & Casa, M. (2016). *Instalaciones Domóticas*. Barcelona: Marcombo.

Rossum, G. v. (13 de Enero de 2003). The Making of Python. (B. Venners, Entrevistador)

Sáez, D. (2006). Domótica. *Actualidad TIC. Universidad Politécnica de Valencia*, 5.

Samaniego, E. (2011). *Ingeniería de Sistemas Robóticos. Aplicaciones sobre Arduino*. Panamá: Universidad Tecnológica de Panamá.

Sanza, E. (Agosto de 2015). *Muy Interesante*. Obtenido de Qué es el Internet de las Cosas:
<http://www.muyinteresante.es/curiosidades/preguntas-respuestas/que-es-el-qinternet-de-las-cosas>

Schmidt, M. (2012). *Raspberry Pi. A Quick Starting Guide*. Dallas, Texas: Pragmatic Press.

Sims, G. (25 de Noviembre de 2014). *ARM vs x86. Key Differences*. Obtenido de Android Authority:
<http://www.androidauthority.com/arm-vs-x86-key-differences-explained-568718/>

Torres, J. J. (Octubre de 2014). *HiperTextual*. Obtenido de Qué es y cómo funciona el internet de las cosas?: <http://hipertextual.com/archivo/2014/10/internet-cosas/>

van Rossum, G., & Drake, F. L. (2000). *Guía de aprendizaje de Python*. Santa Clara, California: BeOpen.com.

Yahoo! Developer Network. (6 de Enero de 2010). *Using JSON (JavaScript Object Notation) with Yahoo! Web Services*. Obtenido de Yahoo! Developer Network: <http://web.archive.org/web/20100106010113/http://developer.yahoo.com/common/json.html>

ANEXOS

Código Java de Clase para mostrar Datos en la Aplicación Android

```
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class Economos extends AppCompatActivity {

    // URL del feed del dispositivo en xively
    private static String url =
    "https://api.xively.com/v2/feeds/1898014877.json";
    //apikey para acceder a los datos del feed
    private static String API_KEY =
    "9Hw9jnjl64013VhP1sHLVwtEdfJS6A5eMIr6S9kHoX2IokVh";

    //JSON nodos o etiquetas
    //Generales del Dispositivo
    private static final String TAG_TITLE = "title";
    private static final String TAG_STATUS="status";
    private static final String TAG_UPDATED="updated";
    //Generales de cada DataStream
    private static final String TAG_DATASTREAMS="datastreams";
    private static final String TAG_ID="id";
    private static final String TAG_CURRENT_VALUE="current_value";
    private static final String TAG_TAGS="tags";

    //JSONArray del dispositivo
    JSONArray device = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_economos);

        /*Ejecuta la clase que implementa Asyntask para trabajar en
background*/
        new GetContacts().execute();
    }

    /**
     * Clase de Async task to get json by making HTTP call
     */
    private class GetContacts extends AsyncTask<Void, Void, Void> {
```

```

/*Valores del dispositivo*/
private String title;
private String status;
private String updated;
private String power1;
private String power2;
private String temperatura;

/*Respuestas en el XML*/
private TextView mTitle;
private TextView mStatus;
private TextView mUpdated;
private TextView mPower1;
private TextView mPower2;
private TextView mTemperatura;

/*No internet booleano*/
private boolean internet=true;

@Override
protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected Void doInBackground(Void... arg0) {
    // Instancia de la Clase
    HTTPHandle sh = new HTTPHandle();

    String jsonStr = sh.getFeed(url,API_KEY);

    Log.d("Respuesta: ", "> " + jsonStr);

    if (jsonStr != null) {
        try {
            JSONObject jsonObj = new JSONObject(jsonStr);

            /*Generales del dispositivo*/
            title = jsonObj.getString(TAG_TITLE);
            status= jsonObj.getString(TAG_STATUS);
            updated= jsonObj.getString(TAG_UPDATED);

            /*Obteniendo el nodo de datastreams*/
            JSONArray jsondatastream =
            jsonObj.getJSONArray(TAG_DATASTREAMS);

            /*JsonObject de Power 1*/
            JSONObject c = jsondatastream.getJSONObject(0);
            power1=c.getString(TAG_CURRENT_VALUE);

            /*JsonObject de Power 2*/
            c = jsondatastream.getJSONObject(1);
            power2=c.getString(TAG_CURRENT_VALUE);

            /*JsonObject de Temperatura*/
            c = jsondatastream.getJSONObject(2);
            temperatura=c.getString(TAG_CURRENT_VALUE);

        } catch (JSONException e) {
            e.printStackTrace();
        }
    } else {
}

```

```

        internet=false;
        Log.e("ServiceHandler", "Couldn't get any data from the url");
    }

    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);

    //Referencias en el XML
    mTitle=(TextView)findViewById(R.id.titletv);
    mStatus=(TextView)findViewById(R.id.statustv);
    mUpdated=(TextView)findViewById(R.id.updatedtv);
    mPower1=(TextView)findViewById(R.id.power1tv);
    mPower2=(TextView)findViewById(R.id.power2tv);
    mTemperatura=(TextView)findViewById(R.id.temperaturatv);

    //Actualiza datos
    mTitle.setText(title);
    mStatus.setText(status);
    mUpdated.setText(updated);
    mPower1.setText(power1);
    mPower2.setText(power2);
    mTemperatura.setText(temperatura);

    if(internet==false) Toast.makeText(Economos.this,"No se pudo
acceder",Toast.LENGTH_LONG);

    /*Repite cada minuto*/
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            new GetContacts().execute();
        }
    }, 60*1000);
}
}

```

Código Java de Clase que Realiza las Peticiones HTTP a la Plataforma Xively

```
/**  
 * Created by José Herazo on 06/11/2016.  
 */  
  
import java.io.IOException;  
import java.io.UnsupportedEncodingException;  
import java.util.List;
```

```

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;

public class HTTPHandle {

    static String response = null;
    public final static int GET = 1;
    public final static int POST = 2;

    public HTTPHandle() {

    }

    public String getFeed(String url, String apikey) {

        /*Se trata de realizar la conexión*/
        try {
            // Cliente HTTP
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpEntity httpEntity = null;
            HttpResponse httpResponse = null;

            HttpGet httpGet = new HttpGet(url);
            httpGet.addHeader("X-ApiKey", apikey);
            httpResponse = httpClient.execute(httpGet);

            httpEntity = httpResponse.getEntity();
            response = EntityUtils.toString(httpEntity);

            /*Excepciones*/
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return response;
    }
}

```

Código XML de la Interfaz Gráfica Android

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.ghlabspanamagmail.xivelyandroidtesis.Economos"
    android:focusable="true">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text="@string/title"
                android:textColorHint="#09193d"
                android:textSize="24dp" />

            <TextView
                android:id="@+id/titletv"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text=""
                android:textColorHint="#09193d"
                android:textSize="24dp" />
        </LinearLayout>

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_centerHorizontal="true">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:text="@string/status"
                android:textColorHint="#09193d"
```

```
        android:textSize="24dp" />

    <TextView
        android:id="@+id/statustv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text=""
        android:textColorHint="#09193d"
        android:textSize="24dp" />
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text="@string/updated"
            android:textColorHint="#09193d"
            android:textSize="24dp" />

        <TextView
            android:id="@+id/updatedtv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text=""
            android:textColorHint="#09193d"
            android:textSize="24dp" />
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text="@string/power1"
            android:textColorHint="#09193d"
            android:textSize="24dp" />

        <TextView
            android:id="@+id/power1tv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text=""
            android:textColorHint="#09193d"
            android:textSize="24dp" />
    </LinearLayout>
```

```
        android:textColorHint="#09193d"
        android:textSize="24dp" />
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text="@string/power2"
            android:textColorHint="#09193d"
            android:textSize="24dp" />

        <TextView
            android:id="@+id/power2tv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text=""
            android:textColorHint="#09193d"
            android:textSize="24dp" />
    </LinearLayout>

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text="@string/temperatura"
            android:textColorHint="#09193d"
            android:textSize="24dp" />

        <TextView
            android:id="@+id/temperaturatv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:text=""
            android:textColorHint="#09193d"
            android:textSize="24dp" />
    </LinearLayout>

</LinearLayout>
</RelativeLayout>
```

Código Python del Script que realiza las Mediciones en la Raspberry PI

```
#!/usr/bin/python
# coding: utf-8

# Código para leer el valor del sensor de temperatura
# de la Raspberry PI y enviar los valores leídos a la
# plataforma xively

import os
import xively
import serial
import datetime

# Variables del dispositivo dadas por la plataforma
xively
FEED_ID = "1898014877"
API_KEY =
"9Hw9jnjl64013VhP1sHLVwtEdfJS6A5eMIr6S9kHoX2IokVh"

# Inicializar Cliente API
api = xively.XivelyAPIClient(API_KEY)

# Funcion para obtener el valor de la temperatura del
SoC
def getTemp():
    res = os.popen('vcgencmd measure_temp').readline()
    return(res.replace("temp=", "").replace("'C\n", ""))

# Funcion para obtener el valor del voltaje del nucleo
de la RPI
def getVolts():
    res = os.popen('vcgencmd measure_volts
core').readline()
    return(res.replace("volt=", "").replace("V\n", ""))

# Funciones para crear los Canales en xively
(Datastreams)

# Funcion para crear canal de datos de Temperatura
def create_datastreams_temp(feed):
    try:
        datastream = feed.datastreams.get("Temperatura")
```

```

        return datastream
    except:
        datastream = feed.datastreams.create("Temperatura", tags="C")
        return datastream

# Funcion para crear canal de datos de Voltaje
def create_datastreams_volts(feed):
    try:
        datastream = feed.datastreams.get("Voltaje")
        return datastream
    except:
        datastream = feed.datastreams.create("Voltaje",
tags="volts")
        return datastream

# Funcion para crear canal de datos de Power1
def create_datastreams_power1(feed):
    try:
        datastream = feed.datastreams.get("Power1")
        return datastream
    except:
        datastream = feed.datastreams.create("Power1",
tags="Watts")
        return datastream

# Funcion para crear canal de datos de Power2
def create_datastreams_power2(feed):
    try:
        datastream = feed.datastreams.get("Power2")
        return datastream
    except:
        datastream = feed.datastreams.create("Power2",
tags="Watts")
        return datastream

feed = api.feeds.get(FEED_ID)

#Crea canales llamando a las funciones
canalvolts= create_datastreams_volts(feed)
canaltemp= create_datastreams_temp(feed)
canalpower1= create_datastreams_power1(feed)
canalpower2= create_datastreams_power2(feed)

#Se leen los valores del sistema
temp = getTemp()

```

```

volts = getVolts()

#actualizar valores de los canales
canaltemp.current_value = temp
canalvolts.current_value = volts

#Lectura de los valores de los sensores CT
ser = serial.Serial('/dev/ttyAMA0', 38400, timeout=1)
response = ser.readline()
z = response.split(" ")

#Calculo de las potencias
if len(z)>=6:
    #potencias decodificadas
    p1=float((float(z[3])* 256.0 + float(z[2])))
    p2=float((float(z[5])* 256.0 + float(z[4])))
    #consumo recalculado a 120 V
    power1=p1*0.5
    power2=p2*0.5
    canalpower1.current_value = power1
    canalpower2.current_value = power2

#Asignar fecha y hora de actualizacion
canaltemp.at = datetime.datetime.utcnow()
canalvolts.at = datetime.datetime.utcnow()
canalpower1.at = datetime.datetime.utcnow()
canalpower2.at = datetime.datetime.utcnow()

#Se envia la peticion de actualizacion a servidor
xively
canaltemp.update()
canalvolts.update()
canalpower1.update()
canalpower2.update()

```

Fotografías del dispositivo en funcionamiento



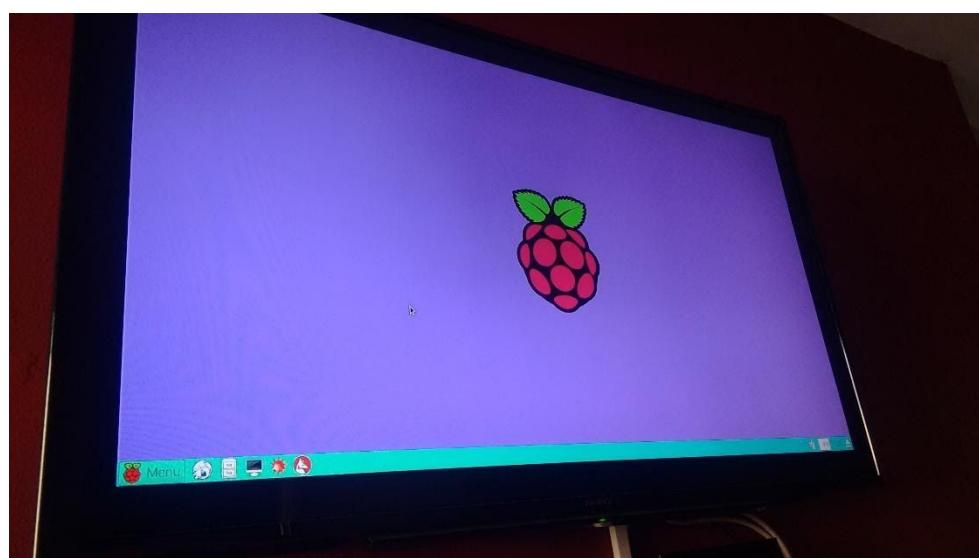
Anexo 1. *Raspberry Pi con todos sus componentes conectados y funcionando.*



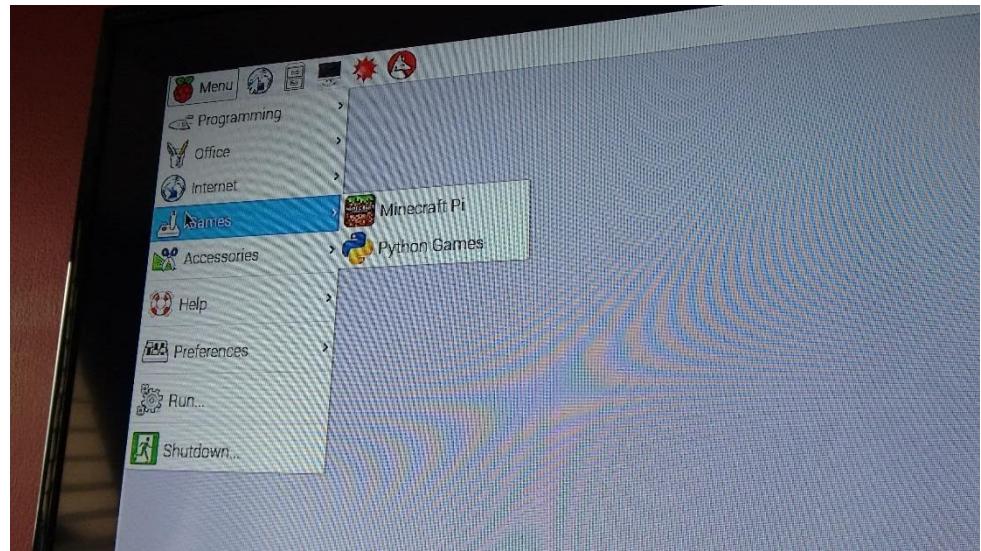
Anexo 2. *Raspberry Pi dentro de su carcasa.*



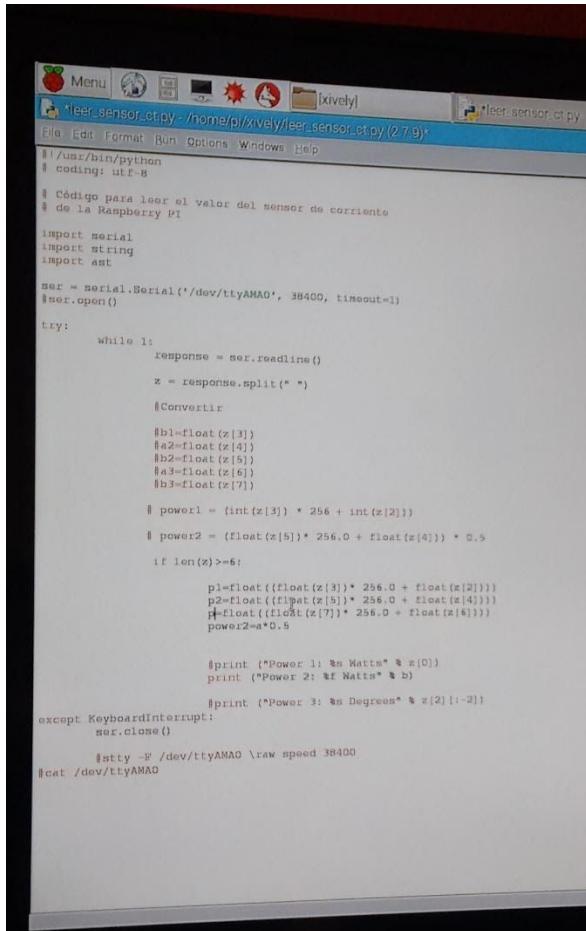
Anexo 3. Primera puesta a prueba del dispositivo.



Anexo 4. Pantalla de inicio del sistema operativo Raspbian.



Anexo 5. Menú de programas dentro del sistema operativo Raspbian.



```
#!/usr/bin/python
# coding: utf-8

# Código para leer el valor del sensor de corriente
# de la Raspberry Pi

import serial
import string
import ast

ser = serial.Serial('/dev/ttyAMA0', 38400, timeout=1)

try:
    while 1:
        response = ser.readline()
        z = response.split(" ")
        #Convertir
        #b1=float(z[3])
        #b2=float(z[4])
        #b3=float(z[5])
        #b4=float(z[6])
        #b5=float(z[7])

        # power1 = (int(z[3])) * 256 + int(z[2])
        # power2 = (float(z[5]))* 256.0 + float(z[4]) * 0.5
        if len(z)>=6:
            p1=float((float(z[3])* 256.0 + float(z[2])))
            p2=float((float(z[5])* 256.0 + float(z[4])))
            p3=float((float(z[7])* 256.0 + float(z[6])))
            power2=p3*0.5

            #print ("Power 1: %s Watts" % z[0])
            print ("Power 2: %f Watts" % b)
            #print ("Power 3: %s Degrees" % z[2][:-1])
except KeyboardInterrupt:
    ser.close()

#stty -F /dev/ttyAMA0 raw speed 38400
#cat /dev/ttyAMA0
```

Anexo 6. Código escrito en Python para leer la data proveniente del sensor CT.

The screenshot shows a terminal window titled "Python 2.7.9 Shell" running on a Raspberry Pi. The window displays a series of power measurements in Watts for two sensors, labeled Power 1 and Power 2. The data is as follows:

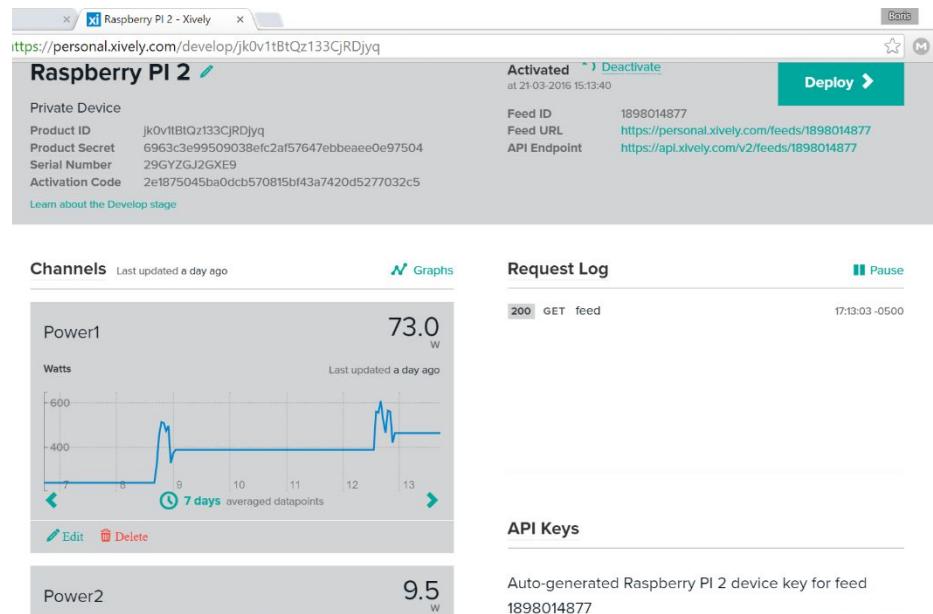
```
Power 1: 77.265000 Watts
Power 2: 77.265000 Watts
Power 2: 78.275000 Watts
Power 2: 77.770000 Watts
Power 2: 77.770000 Watts
Power 2: 78.275000 Watts
Power 2: 78.275000 Watts
Power 2: 78.275000 Watts
Power 2: 77.265000 Watts
Power 2: 78.275000 Watts
Power 2: 77.770000 Watts
Power 2: 77.770000 Watts
Power 2: 78.275000 Watts
Power 2: 77.770000 Watts
Power 2: 76.255000 Watts
Power 2: 77.770000 Watts
Power 2: 77.770000 Watts
Power 2: 78.275000 Watts
Power 2: 76.760000 Watts
Power 2: 77.770000 Watts
Power 2: 77.265000 Watts
Power 2: 76.760000 Watts
Power 2: 78.275000 Watts
Power 1: 14.5 Watts
Power 2: 75.500000 Watts
Power 1: 17.0 Watts
Power 2: 77.500000 Watts
Power 1: 14.5 Watts
Power 2: 78.500000 Watts
Power 1: 14.5 Watts
Power 2: 76.000000 Watts
Power 1: 19.5 Watts
Power 2: 76.000000 Watts
```

Anexo 7. Primeras mediciones obtenidas por el sensor CT, visualizadas ejecutando el código en Raspbian.



Anexo 8. Probando el dispositivo y comparando valores con un medidor convencional.

Capturas de Pantalla del Sistema

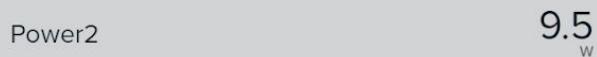
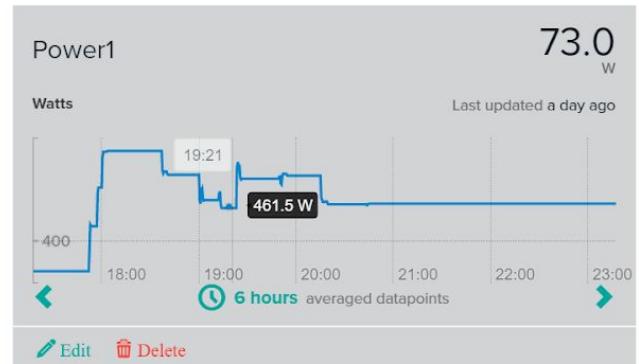


Anexo 9. Plataforma Xively interpretando la data.

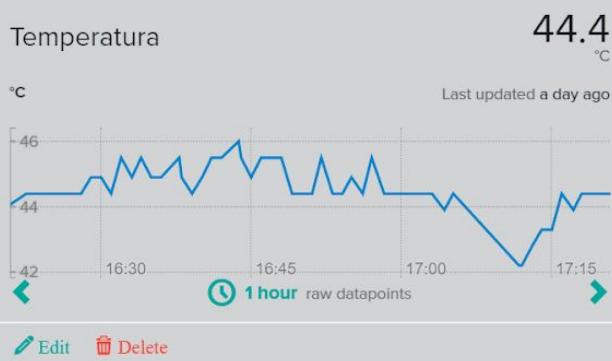


Anexo 10. Plataforma Xively interpretando la data.

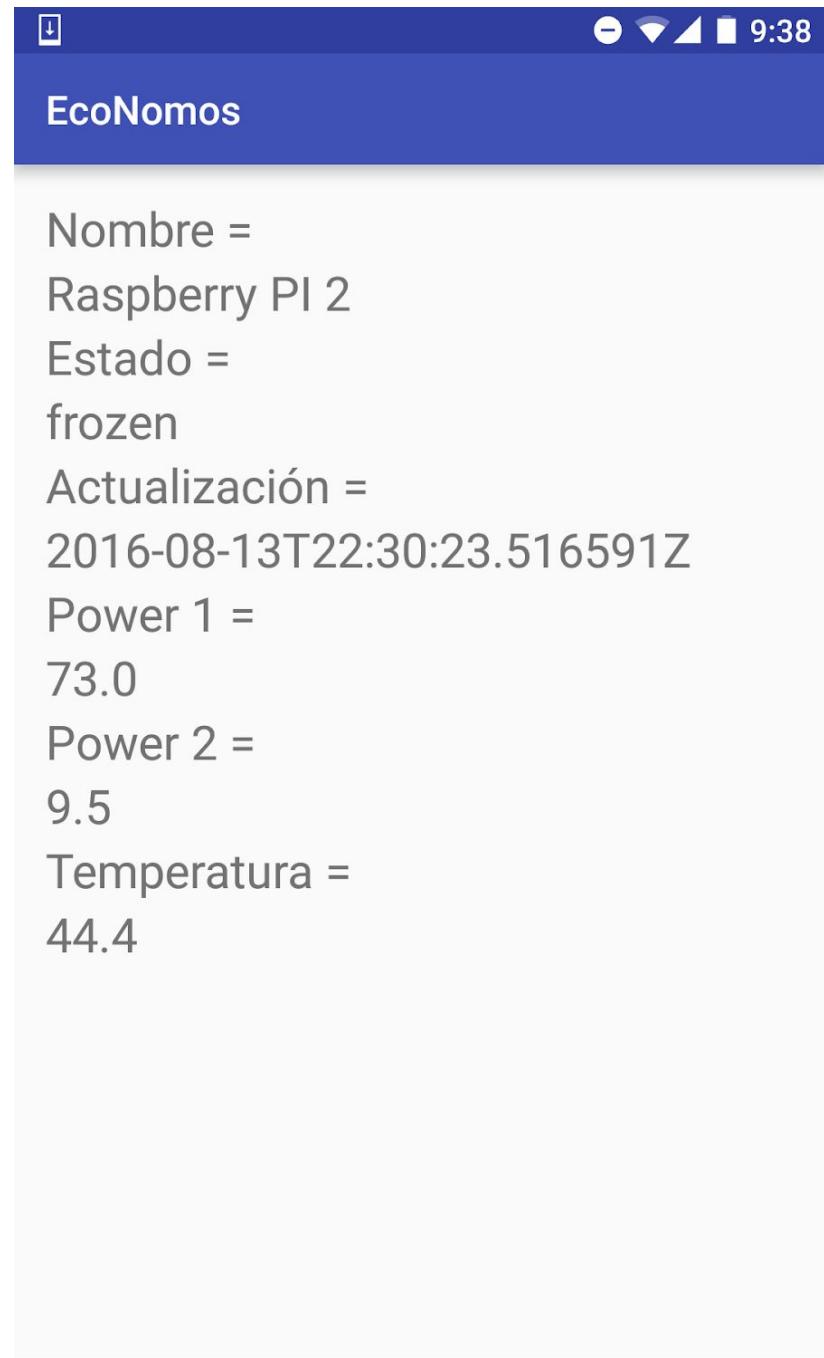
Channels Last updated a day ago  **Graphs**



Anexo 11. Plataforma Xively interpretando la data.



Anexo 12. Plataforma Xively interpretando la data.



Anexo 13. Aplicación Android en funcionamiento, mostrando los datos obtenidos por el sensor CT.