TFC

Integración de Redes Telemáticas

IoT con Raspberry pi

Ingeniería Técnica de Telecomunicaciones

TELEMÁTICA

Alumno: ***Carlos García Muelas***
Consultor: ***Jose López Vicario***

INDICE

1 Introducción

El internet de las cosas se basa en sensores, en redes de comunicaciones y en una inteligencia que maneja todo el proceso y los datos que se generan. Los sensores son los sentidos del sistema y, para que puedan ser empleados de forma masiva, deben tener bajo consumo y coste, un reducido tamaño y una gran flexibilidad para su uso en todo tipo de circunstancias.

Debido a las propiedades de la raspberry pi (bajo consumo, coste y reducido tamaño), el título y el enfoque del proyecto será: IoT (**I**nternet **O**f **T**hings) con raspberry pi.

En los siguientes apartados del proyecto cuando se haga referencia al internet de las cosas nos referiremos al concepto con el acrónimo IoT (**I**nternet **O**f **T**hings).

2 Descripción del proyecto

El internet de las cosas es la conexión de objetos de uso habitual o cotidiano a internet. La finalidad es la interconexión de diferentes objetos que utilizamos con cierta frecuencia, con el objetivo de hacernos la vida más fácil, por ejemplo en la administración eficiente de la energía o en potenciar nuestra propia seguridad.

El proyecto realiza un estudio de las diferentes plataformas software existentes, describiendo sus características, puntos fuertes y débiles así como su ámbito de aplicación. No sólo se analizará las plataformas software existentes en la red, a las cuales se enviarán los datos recogidos por nuestros sensores, si no que analizaremos la forma de crear nuestra propia plataforma en un servidor propio (usando la raspberry pi), de modo que tengamos el control total de los datos y no tengamos límite en cuanto al número de sensores representables.

Las plataformas hardware son la otra parte del internet de las cosas, analizaremos las diferentes plataformas existentes con propiedades de bajo consumo, coste y tamaño.

Estas plataformas son las que nos van a permitir recoger la información de los sensores e interactuar con las plataformas software para la representación final de los datos al usuario.

Estudiaremos la interacción del sistema hardware-software con las personas por medio de envío de mensajes vía telegram o whatsapp, de forma que nos notifique una posible alarma, o yendo más allá, podamos apagar remotamente un equipo con un mensaje de twitter.

Por último el proyecto contendrá un caso real de representación se sensores en una plataforma software. Para ellos usaremos como hardware la raspberry pi, junto con sensores que acoplaremos a dicho hardware. Los sensores serán de diferentes tipos (humedad, temperatura, vigilancia, etc) para su posterior monitorización.

3 Objetivos y motivaciones del proyecto

Los principales objetivos del proyecto son:

- Conocer el concepto de internet de las cosas, aplicaciones reales y su futuro.
- Analizar las diferentes plataformas software para IoT.
- Analizar las diferentes plataformas hardware para IoT.
- Realizar un caso real, usando la raspberry pi y una serie de sensores (temperatura, humedad, cámara, etc) que enviará por un lado los datos a una plataforma software, para que el usuario pueda monitorizarlos remotamente y por otro lado interactuará con aplicaciones de mensajería instantánea como Whatsapp y Telegram.

La principal motivación de realizar este proyecto es sacar el máximo partido a la raspberry pi que ya tenía y la usaba únicamente para tareas locales. El poder integrar la raspberry pi en el IoT a través de sensores y su interacción con aplicaciones de mensajería estilo whataspp además de la subida de datos de esos sensores a plataformas web de IoT amplía con creces las funcionalidades que tenía anteriormente, como centro multimedia o owncloud por destacar alguna de ellas.

No obstante la raspberry pi no está diseñada para el IoT, de ello sus limitaciones, aunque existen los módulos de expansión a arduino que potencian su funcionalidad y de esta manera obtener multitud de aplicaciones para IoT.

Sistemas operativos como Contiki y TinyOS son sistemas operativos de código abierto desarrollados para el IoT. Entre su hardware compatible no se encuentran ni la raspberry pi ni arduino. Estos sistemas operativos son muy potentes, en el caso Contiki se trata de una máquina virtual Ubuntu Linux que se ejecuta en el reproductor de VMWare y tiene Contiki y todas las herramientas de desarrollo, compiladores y

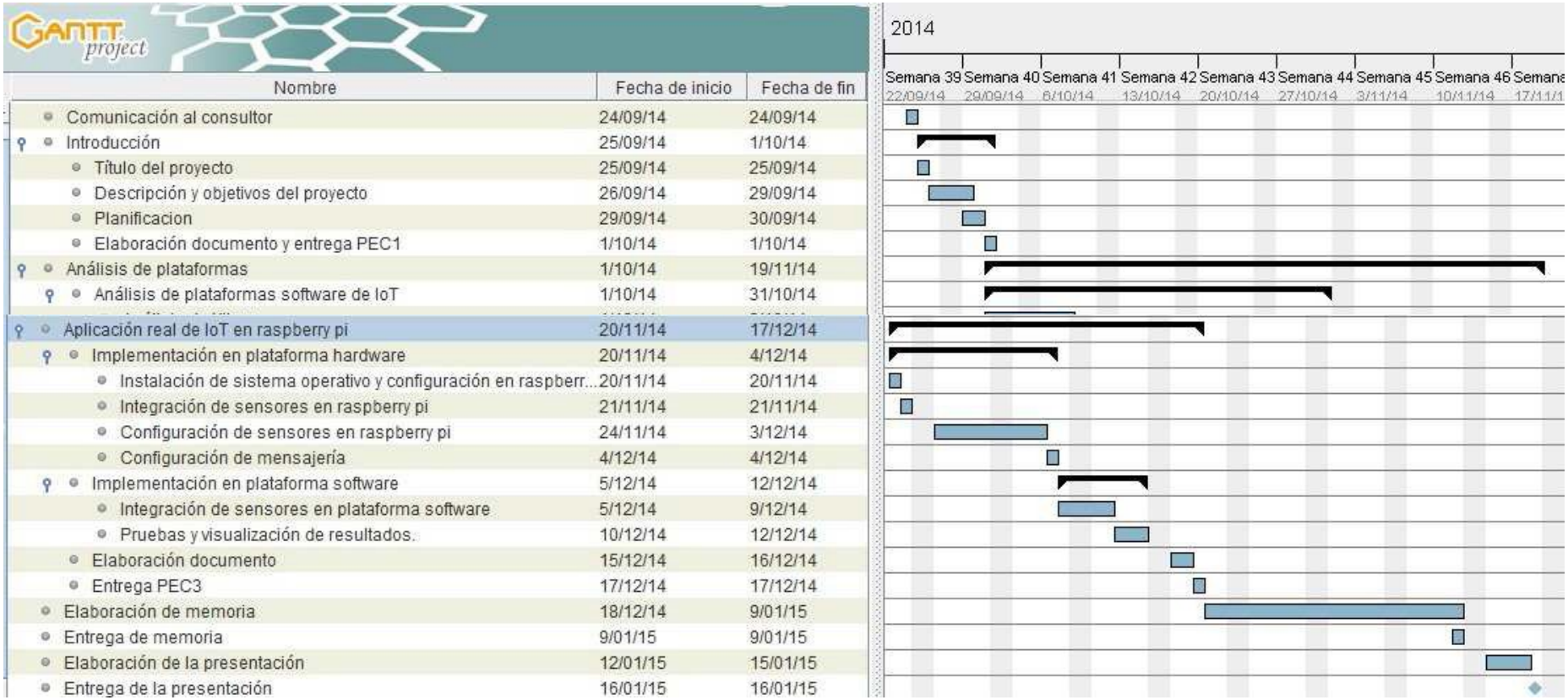
simuladores utilizados en el desarrollo de Contiki instalado.

Con respecto a TinyOS, las mayores diferencias con respecto a Arduino son:

- TinyOS y sus APIs están diseñados para un funcionamiento de bajo consumo; Arduino no lo es.
- Arduino se centra en la detección, la computación, y IO digital; TinyOS también apoya estas abstracciones añadiendo un excelente soporte para redes inalámbricas.
- TinyOS está escrito en nesC, un dialecto C diseñado para el sistema operativo, mientras que Arduino es en C. Mientras en TinyOS los programadores pueden escribir aplicaciones en C, el sistema operativo central está en nesC.

Estos sistemas operativos a pesar de ser muy potentes y diseñados exclusivamente para IoT no nos sirven para la plataforma hardware que hemos seleccionado, la raspberry pi, por lo cual el proyecto es un reto, ya que vamos a realizar con estas plataformas tanto software como hardware no orientadas a IoT, monitorizaciones de datos de sensores en plataformas software de IoT, teniendo siempre en cuenta las limitaciones que tenemos.

4 Diagrama de Gannt



Nombre	Fecha de inicio	Fecha de fin
Comunicación al consultor	24/09/14	24/09/14
Introducción	25/09/14	1/10/14
Título del proyecto	25/09/14	25/09/14
Descripción y objetivos del proyecto	26/09/14	29/09/14
Planificacion	29/09/14	30/09/14
Elaboración documento y entrega PEC1	1/10/14	1/10/14
Análisis de plataformas	1/10/14	19/11/14
Análisis de plataformas software de IoT	1/10/14	31/10/14
Aplicación real de IoT en raspberry pi	20/11/14	17/12/14
Implementación en plataforma hardware	20/11/14	4/12/14
Instalación de sistema operativo y configuración en raspberr...	20/11/14	20/11/14
Integración de sensores en raspberry pi	21/11/14	21/11/14
Configuración de sensores en raspberry pi	24/11/14	3/12/14
Configuración de mensajería	4/12/14	4/12/14
Implementación en plataforma software	5/12/14	12/12/14
Integración de sensores en plataforma software	5/12/14	9/12/14
Pruebas y visualización de resultados.	10/12/14	12/12/14
Elaboración documento	15/12/14	16/12/14
Entrega PEC3	17/12/14	17/12/14
Elaboración de memoria	18/12/14	9/01/15
Entrega de memoria	9/01/15	9/01/15
Elaboración de la presentación	12/01/15	15/01/15
Entrega de la presentación	16/01/15	16/01/15

5 Conceptos de IoT

5.1 Introducción a IoT

El Internet de las Cosas (IoT) consiste en la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas en inalámbricas. Dado su tamaño y coste, los sensores son fácilmente integrables en hogares, entornos de trabajo y lugares públicos. De esta manera, cualquier objeto es susceptible de ser conectado y estar presente en la Red.

El IoT implica que todo objeto puede ser una fuente de datos. Esto está empezando a transformar la forma de hacer negocios, la organización del sector público y el día a día de millones de personas.

El Internet de las Cosas es una realidad muy presente que está evolucionando. Millones de dispositivos están siendo conectados entre sí a través de distintas redes de comunicación. Pequeños sensores permiten medir desde la temperatura de una habitación hasta el tráfico de taxis en una ciudad. A diario, cámaras de vigilancia velan por la seguridad en los edificios y los paneles del metro nos indican el tiempo que falta hasta la llegada del siguiente tren. Incluso en las multas de tráfico existe poca intervención humana. Cada vez más objetos están siendo integrados con sensores, ganando capacidad de comunicación, y con ello las barreras que separan el mundo real del virtual se difuminan. El mundo se está convirtiendo en un campo de información global y la cantidad de datos que circulan por las redes está creciendo exponencialmente.

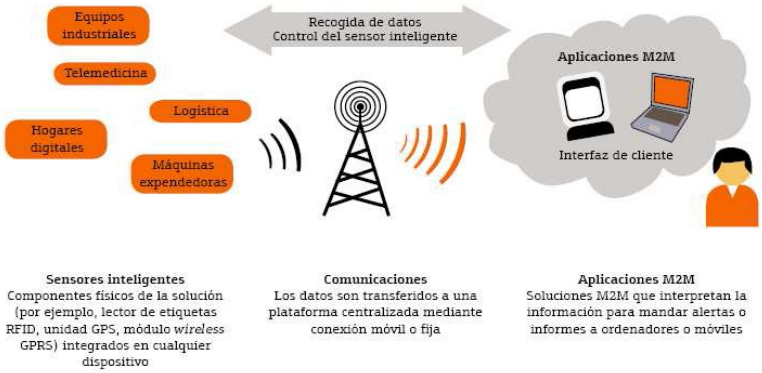
El Internet de las cosas (IoT, por sus siglas en inglés) no es una idea nueva. A principios de los años noventa, Mark Weiser, director científico del Xerox Palo Alto Research Center, introdujo el concepto de computación ubicua, que abogaba por un futuro en el que la computación desaparecería de nuestra vista, es decir, que formaría parte integral de nuestra vida diaria y resultaría transparente para nosotros. Weiser no acuñó el ,término Internet de las Cosas, que se atribuye al ,Auto-ID Center del Massachusetts Institute of Technology (MIT) a finales de los años noventa.

El concepto de Internet de las cosas fue propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999, donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores.

Esta evolución ha seguido el patrón marcado por el visionario Gordon Moore, ,cofundador del fabricante de microprocesadores Intel. Moore formuló su famosa predicción, conocida a nivel mundial como «Ley de Moore», en 1965, refinándola en 1975. En ella establece que el número de transistores que contiene un chip se duplica cada dos años aproximadamente. Bien sea porque Moore fue capaz de , predecir el , futuro o porque los fabricantes de procesadores fijaron sus palabras como un objetivo a largo plazo, la Ley de Moore se ha venido cumpliendo durante los últimos cuarenta años. Funciones que décadas atrás requerían de un ordenador del tamaño de una habitación son hoy día realizadas con facilidad por simples dispositivos electrónicos del tamaño de una gota de agua.

El tamaño, el coste y el consumo de energía del hardware se han reducido drásticamente, por lo que ahora es posible fabricar dispositivos electrónicos diminutos a un coste muy reducido. Estos pequeños dispositivos, junto con la expansión de las redes de comunicación, permiten incorporar inteligencia y conexión a los objetos del mundo real y están transformando lo que era una red global de personas en una red global de todas las cosas.

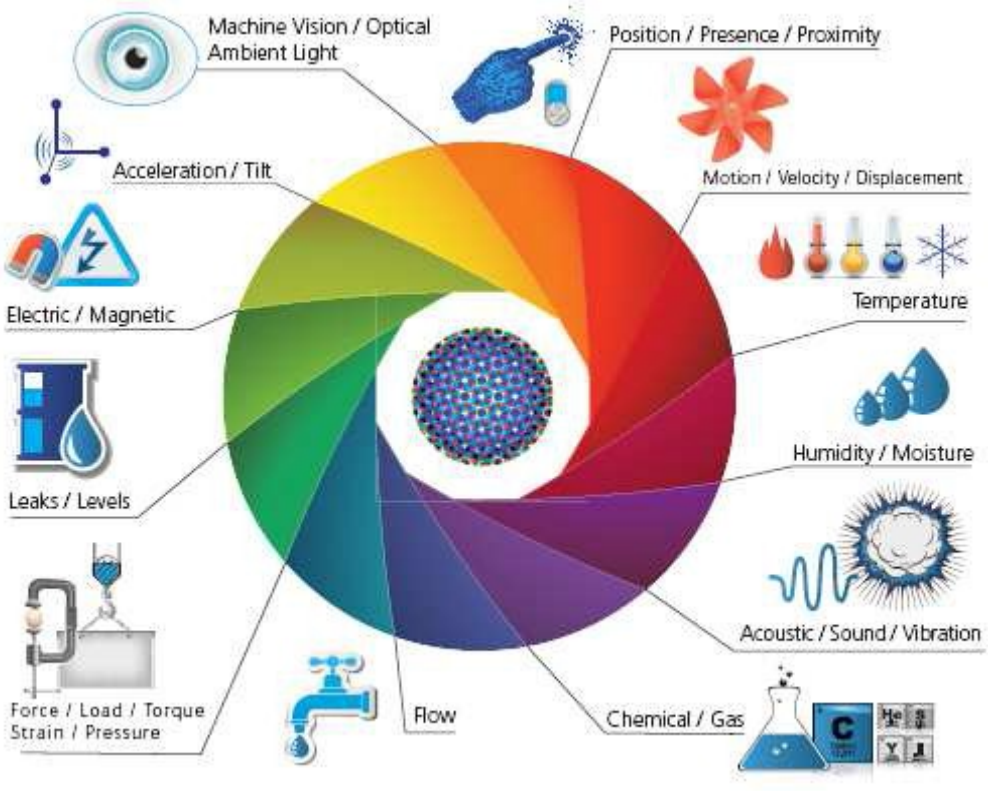
Las etiquetas RFID (radio frequency identification, en español, identificación por radiofrecuencia) son pequeños dispositivos, similares a una pegatina, que pueden ser adheridos a un producto, persona o animal para almacenar información relevante y dinámica. Mediante radiofrecuencia, la información viaja a un ordenador o dispositivo móvil con acceso a Internet. Dicha información puede ser recibida por un usuario para su interpretación. También existe la posibilidad de que el extremo final sea otra máquina que interprete los datos y actúe según parámetros preestablecidos.



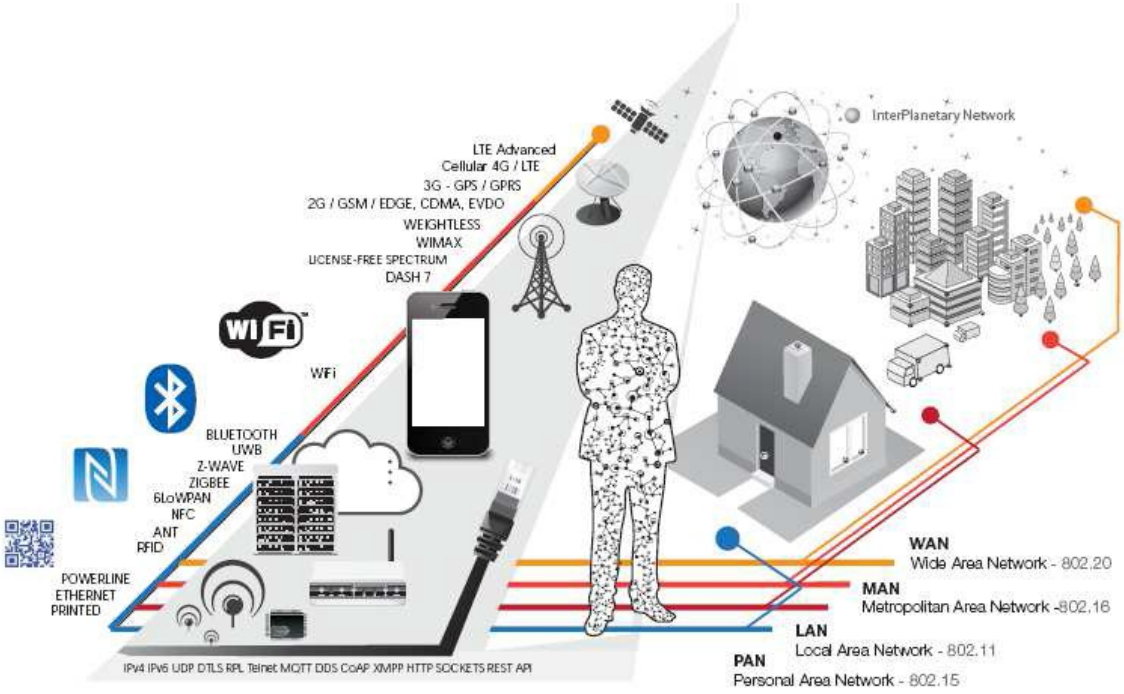
¿Porqué ahora está teniendo el boom?, digamos que intervienen diferentes factores: popularización de placas de hardware libre, el abaratamiento de sensores, mejora en las comunicaciones y la existencia de plataformas IoT.

En las siguientes tres figuras, se resume el IoT:

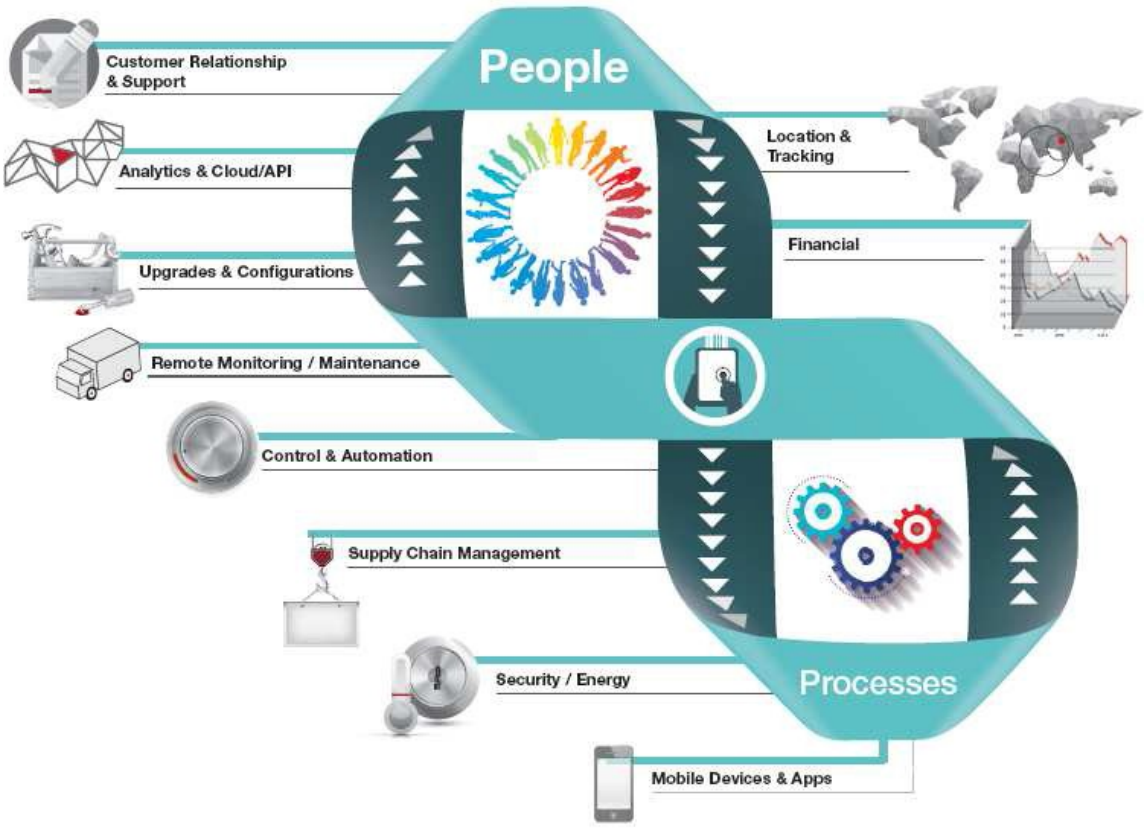
1- **Sensores y actuadores**: estamos dando a nuestro mundo un sistema nervioso digital. Ubicación de datos utilizando sensores GPS, los ojos y las orejas con cámaras y micrófonos, junto con los órganos sensoriales que pueden medir todo, desde la temperatura a cambios de presión.



2- **Digitalización**: estas entradas se digitalizan y se colocan en las redes.



3- **Personas & Procesos** : estas entradas en la red se pueden combinar en sistemas bidireccionales que integran datos, personas, procesos y sistemas para la mejora en la toma de decisiones.



5.2 Aplicaciones reales

En este apartado se va a explorar la aplicación de sensores y dispositivos tanto en el uso sobre nuestro cuerpo, automatización de hogares, ciudades inteligentes o en los sectores de la logística, la salud, el medio ambiente y los consumidores. Veremos en cada campo aplicaciones reales.

Una ciudad inteligente en la que los teléfonos móviles abren puertas, los sensores detectan fugas en las cañerías de agua y las vallas publicitarias cambian sus anuncios según el perfil de consumidor de las personas que pasan. El IoT se ha aplicado a todo tipo de industrias, como la sanitaria, agrícola, logística o de suministros, permitiendo conectar todo tipo de máquinas para monitorizarlas y controlarlas de manera inteligente. El objetivo común de todas es el incremento en eficiencia, la reducción de costes, la mejora en la toma de decisiones, el ahorro energético y la protección medioambiental.

5.2.1 El Internet de la Logística

Cualquier empresa puede utilizar etiquetas RFID en su cadena de suministro. Si a esto añadimos una aplicación que permita visualizar en un móvil la información agregada de todos los productos de toda la cadena en un momento determinado, se estará reduciendo tiempo y recursos destinados al seguimiento de las operaciones de una empresa. También se puede controlar de manera remota la temperatura de ciertos procesos productivos mediante sensores.

La empresa de transportes danesa Container Centralen anunció que utilizaría la tecnología de sensores para que los agentes hortícolas puedan seguir el progreso de sus envíos a medida que avanzan por la cadena de suministro, de los productores a los mayoristas y minoristas, en cuarenta países en Europa. Al transportar flores y plantas de maceta, que son muy sensibles al medio ambiente, es importante vigilar las condiciones y el clima durante el viaje. En esencia, todos los responsables de logística podrían hacer uso de información al instante de las condiciones meteorológicas y de tráfico para planificar las rutas de sus camiones y aviones. De esta manera, aumentaría la eficiencia de su negocio al reducir costes y evitar un ajuste constante de la planificación de su actividad.

El Internet de las Cosas también llega al espacio. La NASA utiliza más de quinientos sensores en los motores de sus naves para reunir información acerca de casi todos los aspectos de sus vuelos. Los ingenieros del proyecto dicen que mostrarán el trabajo de los motores en condiciones reales y pueden servir para crear modelos individualizados que eviten fracasos en el futuro.

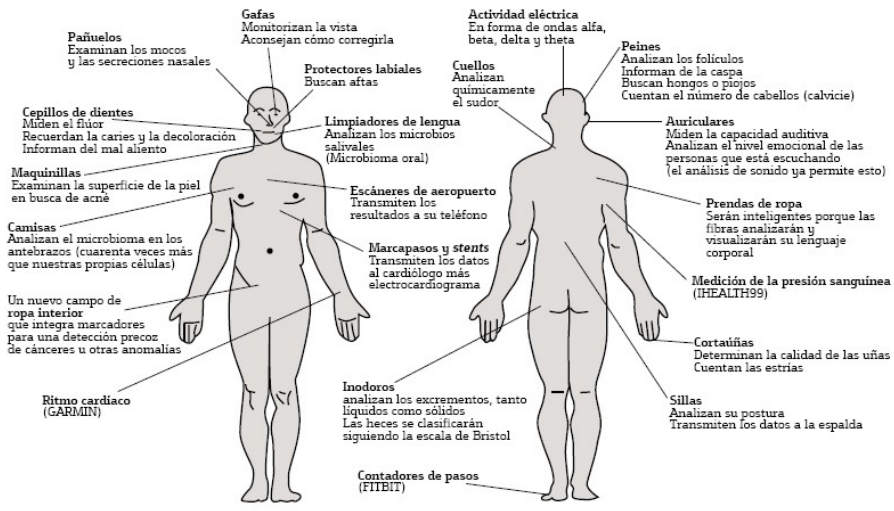
5.2.2 El Internet de la Salud

Normalmente uno va al médico cuando se encuentra mal o cuando se fuerza, una vez al año, a hacerse un chequeo médico. ¿Y si hubiera una manera de controlar nuestro cuerpo para garantizar que, en el momento en que sucediera algo anómalo, se nos informara de ello. Un diagnóstico precoz es muchas veces la solución a enfermedades que pueden resultar mortales. Sería como tener un termostato corporal: en el momento en que detectara un problema, se lanzaría un aviso o, en algunos casos, desencadenaría el suministro de una dosis medicinal.

Los sensores del Internet de las Cosas hacen esto posible. Un ejemplo sería una pegatina que se adhiere al pecho para evitar ataques al corazón. Monitoriza la actividad y los niveles de fluidos en el pecho y con sólo sustituir la pegatina por una nueva cada semana, estará realizando un seguimiento continuo por conexión wifi.

La empresa Telcare ha sacado al mercado el primer glucómetro móvil que permite transmitir los resultados de un análisis a un centro médico para recibir asistencia instantánea on-line. El dispositivo cuenta con una pantalla de datos con conexión wifi y una ranura en la que se inserta una tira de papel con una gota de sangre. De esta manera se evitan desplazamientos innecesarios, la saturación de los centros médicos y los altos costes asociados.

El uso de la nanotecnología permite monitorizar distintas partes de nuestro cuerpo y los datos pueden ser enviados a través de Internet para su diagnóstico. Gafas que revisan tus ojos y aconsejan correcciones, cuellos de camisa que analizan el sudor o cascos que miden la actividad cerebral, todo es posible en el Internet de la Salud.



5.2.3 El Internet del Medio Ambiente

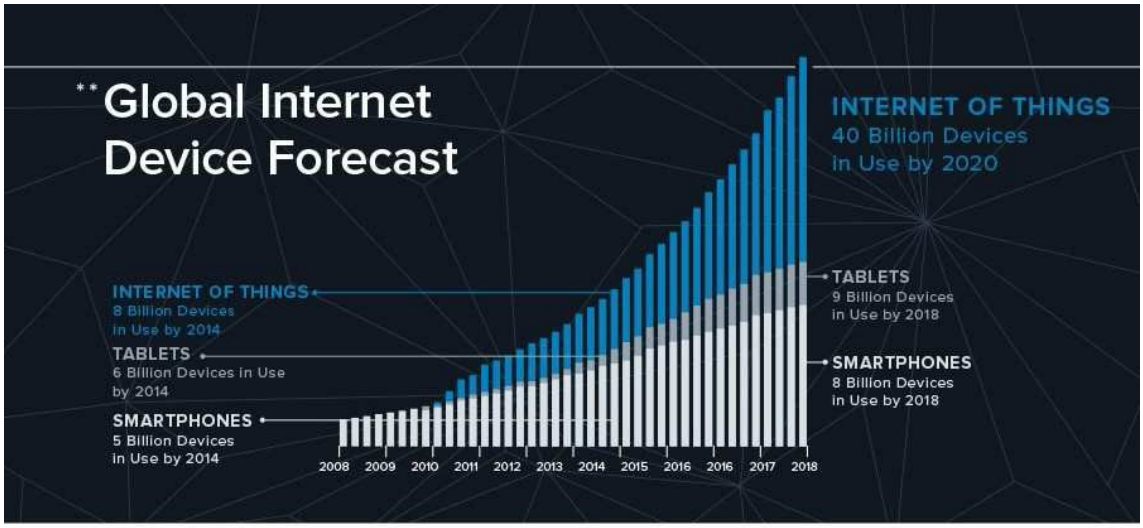
Los edificios inteligentes constituyen el mejor ejemplo de la aplicación de Internet a un objetivo medioambiental. En Estados Unidos, los edificios consumen el 70% de toda la electricidad, de la cual un 50% se malgasta. Además, un 50% del agua que consumen también es derrochada. Para subsanar este tipo de situaciones, se dota a muchos edificios de smart grid, una red que permite optimizar la generación y el consumo de energía gracias a una serie de medidores inteligentes que eligen las mejores franjas horarias entre empresas eléctricas y discriminan entre horarios de consumo. El resultado es un consumo más sensato y económico.

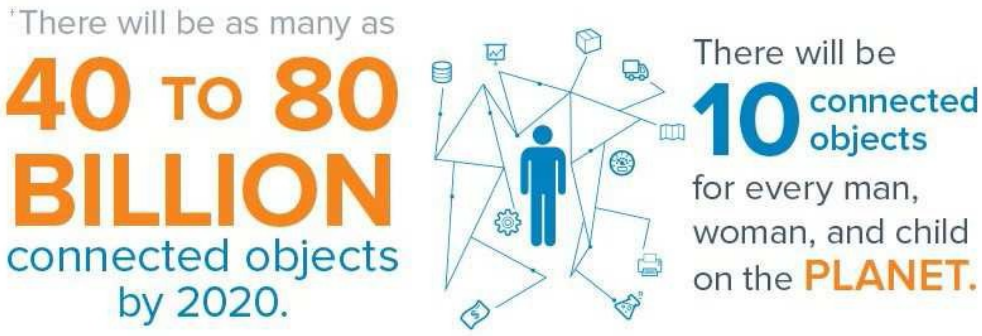
Ante la imperativa necesidad de controlar el consumo de manera más eficiente, se están creando espacios de oficinas verdes en el complejo **GreenSpaces** de Delhi (India) o el proyecto **Smart IPv6 building**, con un programa piloto en Ginebra (Suiza).

5.3 El futuro de IoT

RnRMarketResearch.com realizó un estudio[(1)] de mercado de Internet de las Cosas que proporciona un análisis de la industria y las previsiones para 2019 y 2020 para la IoT, la comunicación M2M, la tecnología de la IoT y aplicaciones. Estos informes pronostican que el Internet de las Cosas (IoT) y Máquina a Máquina (M2M) valdrán en el mercado $ 498,92 Billones en 2019. Se espera que el valor de Internet de las Cosas en el mercado pueda llegar a 1423.09 Billones de dólares en 2020.

Con los nuevos inventos tecnológicos y su perfecta integración en nuestro día a día, es evidente que el concepto de la Internet de las Cosas ya está sentando las bases para un futuro prometedor. Pocas tendencias de la tecnología y de las aplicaciones están actuando sobre todo como futuros capacitadores de IoT que pueden ayudar en la configuración del futuro de la IoT. Algunas de estas tendencias tecnológicas fundamentales que se espera que tengan un gran impacto en la evolución de la IoT son IPV6, la proliferación de sensores, la computación en nube, los datos grandes, y los estándares de comunicación más rápidos.





6 Análisis de plataformas software

6.1 Xively

6.1.1 Introducción

Pachube, que fue creada en 2007 por el artista londinense Usman Haque, era una plataforma que ponía a disposición de cualquiera poder subir a la nube datos de cualquier sensor (por ejemplo, el nivel de humedad y temperatura de tu ciudad, el nivel de ruido del parque, etc.), y de esta manera construir tus propias aplicaciones basadas en estos datos. Quizás la página alcanzó su máxima popularidad tras el desastre nuclear que ocurrió en Fukushima, porque mucha gente consultaba los datos subidos por un sensor de radiación al portal, y de esta manera podía contrastar la información que daba el sensor con los datos que ofrecía el gobierno japonés. En definitiva, a Pachube se la consideraba un "Internet de las Cosas", un concepto en el que todo está conectado a internet, como un zapato o una cafetera.

La empresa LogMeIn, conocida por ofrecer programas de control remoto e interconexión de ordenadores, compró a Pachube por $15m en Julio de 2012 y rebautizó el portal a Cosm, y ahora, tras su alianza con ARM, ha vuelto a rebautizar el portal para llamarlo Xively (https://xively.com/).

Xively está habilitado para trabajar en cualquier plataforma importante que se pueda imaginar. Ya sea un desarrollo en Linux, Android, un sistema operativo Embedded C (en tiempo real o no), Arduino, imp eléctrica, etc.

Esta plataforma IoT proporciona la posibilidad de subir datos Smart Objects u objetos utilizando las librerías que ellos proporcionan. Para utilizarlas se debe usar la APIKey que te asignan cuando te registras o crear una nueva. Con ella se puede limitar el uso de ciertas acciones REST a un dispositivo. Dan un total de diez librerías , entre las que cabe destacar la de Arduino, Java y Android. Por medio de su uso, un usuario puede subir a su perfil los datos de diferentes objetos. A este nuevo dispositivo puede añadirse disparadores, para que, tras cumplirse una determinada condición realice una petición POST a un servicio http usando un método REST para comunicarle la acción.

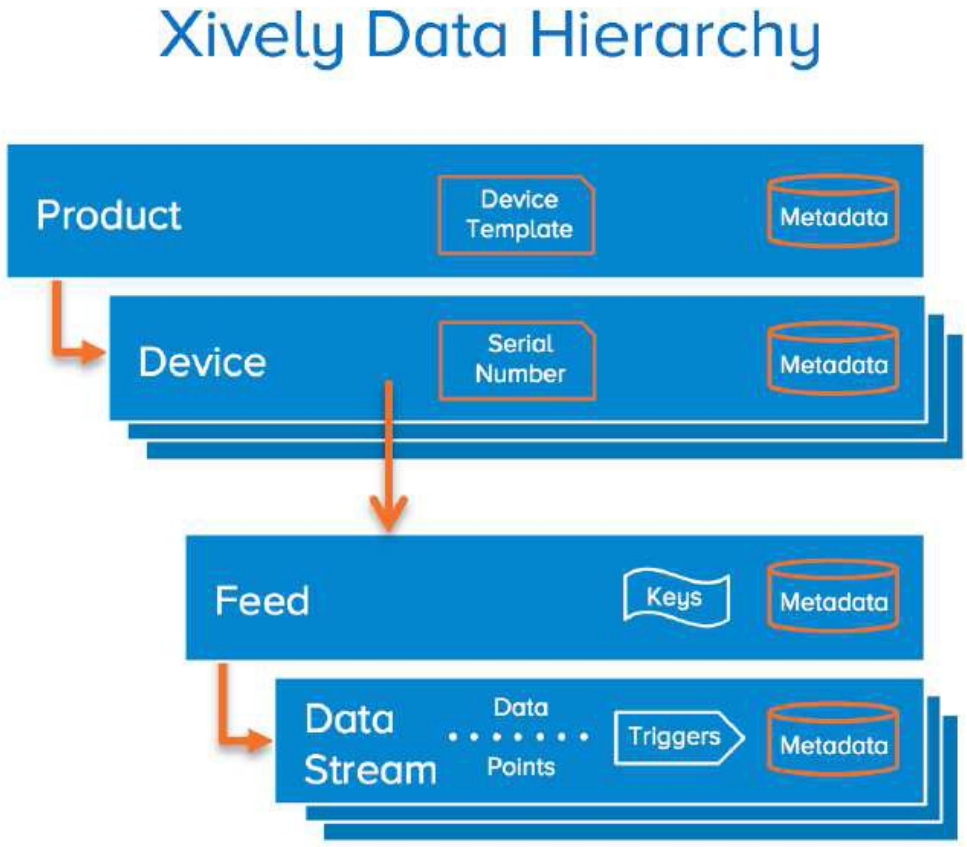
6.1.2 Hardware compatible

En la siguiente tabla figura el hardware compatible con Xively.

TSmarT	Embedded C	WiFi
SimpleLinks	Linux, Android	WiFi
IMX53QSB	Linux, Android	Ethernet
Hitex OM13031	FreeRTOS	Ethernet
Beaglebone	Linux, Android	Ethernet
RX62N	FreeRTOS	Ethernet
Raspberry Pi	Linux	Ethernet
Android PC 8750	Android	Ethernet
LPCXpresso	FreeRTOS	Ethernet Optional
ARM mbed LPC1768	Embedded C/C++	Ethernet
Zolertia Z1	Embedded C	6loWPAN
PIC32 Ethernet Kit	Embedded C	Ethernet
Arduino Uno	Arduino	Ethernet, WiFi, Cellular
Arduino Due	Arduino	Ethernet, WiFi, Cellular
Arduino Ethernet	Arduino	Ethernet
Arduino Mega 2560	Arduino	Ethernet, WiFi, Cellular
Arduino Leonardo R3	Arduino	Ethernet, WiFi, Cellular
RedBack 1.0	Arduino	WiFi
DiamondBack 1.0	Arduino	WiFi

6.1.3 Jerarquía de datos en Xively

Xively organiza sus datos en diferentes capas. La capa inferior está asociada a los datos o metadatos capturados por dispositivos y subido a la plataforma. Las capas superiores encapsulan y organizan los datos con el fin de crear aplicaciones y fabricar productos.



El nivel más alto de la jerarquía corresponde a los productos. Son entidades abstractas que abarcan dispositivos, y representan las aplicaciones y servicios basados en los datos proporcionados por dispositivos. Uno de los productos puede combinar servicios y aplicaciones de varios dispositivos. La definición de producto se especifica con el nombre del producto, una descripción y otros parámetros opcionales. El segundo nivel está reservado para los dispositivos que son objetos físicos individuales que proporcionan y recuperan datos desde y hacia la plataforma. Se identifican por un número de serie que puede consistir en cualquier combinación de caracteres alfanuméricos. Cualquier dispositivo se corresponde con un feed que representa a todos los tipos de datos que un dispositivo es capaz de gestionar (por ejemplo, un dispositivo de Arduino puede tener un sensor de temperatura y un sensor de luz).

Los feeds pueden ser públicos o privados; feeds públicos significa que el dispositivo y sus datos están a disposición de cualquier usuario en los términos de la licencia Creative Commons CCO 1,0 licencia universal; por el contrario, feeds privados se refieren que los datos del dispositivo sólo están disponibles para el desarrollador o personas seleccinadas.

Estos tipos de datos se llaman "Datastreams" que se identifican de forma única, y representan los flujos de datos que van hacia la plataforma y viceversa (por ejemplo, un dispositivo Netduino que envía continuamente valores de temperatura). Xively llama punto de datos a cada valor archivado en un tiempo discreto de una corriente de datos.

6.1.4 Xively API

API Xively actúa como una interfaz para leer y escribir datos al servicio de cloud Xively y está basado en los principios REST. La mayoría de las peticiones HTTP son JSON, XML y CSV (JSON es el formato por defecto). API Xively se puede llamar de forma segura a través de IPv4 o IPv6 a través de HTTP, HTTPS (es el protocolo por defecto), sockets o websockets y protocolos MQTT. Cada solicitud requiere autenticación a través de una clave de API. Además, es compatible con OAuth, por lo que las entidades de terceros pueden comunicarse con el API.

El API Xively volverá códigos de estado HTTP apropiados para cada solicitud.

HTTP request	Description
200 OK	Request were successfully processed
400 Bad Request	This reply would be due to a malformed JSON document.
401 Unauthorized	The requester (HTTP client) does not include the authentication field or this is invalid.
403 Forbidden	The authentication field is valid but there is no access to the resource.
404 Not found	The API does not match the requested resource with any of the available ones.
406 Not acceptable	It due to the wrong document type
422 Unprocessable entity	Xively could not create a Feed because the payload was malformed.
500 Internal Server Error	The server could not fulfill the request due to an internal code.
503 No server error	It is used when Xively servers are overloaded

API Xively gestiona 7 recursos para llevar a cabo las operaciones para interactuar con la plataforma: productos, dispositivos, claves, feeds, triggers, flujos de datos y puntos de datos. Las operaciones pueden ser clasificadas en almacenamiento, lectura y escritura.

a) Los datos almacenados se pueden borrar de la siguiente manera: puntos de datos individuales, múltiples puntos de datos mediante la especificación de un rango, y todo el datastream.
b) La lectura puede hacerse a través de tres recursos: feeds, datastreams y puntos de datos. Xively API permite leer un sólo feed, un sólo datastream, una gama de puntos de datos históricos, y todos los canales.
c) La escritura a datastreams se puede lograr mediante dispositivos, aplicaciones y los servicios de las siguientes maneras: un sólo datapoint a un único flujo de datos, un solo datapoint a múltiples datastreams, múltiples puntos de datos a un único flujo de datos y múltiples puntos de datos a múltiples datastreams.

En el ANEXO I figuran las tablas que muestran las acciones que se pueden realizar para cada recurso. Todas las acciones están formato JSON por defecto, sin embargo, algunas de ellas son capaces de responder en formatos alternativos, como XML, CSV y PNG.

Las peticiones de triggers soportan JSON y XML. Los triggers se utilizan para el envío de notificaciones mediante POST HTTP a una dirección URL cuando un Datastream monitorizado satisface una condición preconfigurada. Los siguientes datos indican varios ejemplos de respuestas HTTP en diferentes formatos. Cada acción realizada supone una solicitud y respuesta HTTP. Los siguientes datos indican dos peticiones con sus respuestas; primero crear un dispositivo en formato JSON , y segundo leer un solo Datastream en JSON, XML, CSV y formato de datos PNG.

1) Crea un nuevo dispositivo para cada número de serie dado. Una vez creado, los dispositivos estarán en un estado de pre-activación en espera. El número de serie es un identificador único asignado a cada dispositivo, accesible para el código de activación que se ejecuta en el dispositivo. Este ID puede ser un valor generado y programado en el dispositivo durante el proceso de fabricación, o puede ser una identidad específica del dispositivo tal como una dirección MAC o el número de serie del procesador. Un número de serie puede ser cualquier cadena arbitraria

compuesto por caracteres alfanuméricos, así como los siguientes caracteres: Más (+) Menos (-) por comas (,) guión bajo (_) y puntos (:)

Body

JSON XML CSV

POST /v2/products/PRODUCT_ID_HERE/devices

```
{
  "devices": [
     {"serial": "abc123"},
     {"serial": "321cba"}
   ]
}
```

Response

Parameters

Status Code	201 CREATED

Headers

Location	https://api.xively.com/v2/products/h5wT2IlOMrd09r_qd1jm/devices

2) Leer un Datastream, el cuerpo de la solicitud puede ser en JSON, XML, CSV, y PNG, respectivamente. Debido a la naturaleza de los datos, Xively ofrece para responder los datos solicitados directamente en un formato gráfico PNG.

Request

Parameters

Method	GET
Base URL	https://api.xively.com
API Endpoint	/v2/feeds/FEED_ID_HERE/datastreams/DATASTREAM_ID

Headers

X-ApiKey	API_KEY_HERE

Body

JSON XML CSV PNG

GET /v2/feeds/FEED_ID_HERE/datastreams/DATASTREAM_ID.json

Response

Parameters

Status Code 200 OK

Headers

No Headers

Body

JSON XML **CSV** PNG

```
2013-05-06T00:30:45.694188Z,500
```

Response

Parameters

Status Code 200 OK

Headers

No Headers

Body

JSON XML CSV PNG


```
{
    "id":"example",
    "current_value":"500",
    "at":"2013-05-06T00:30:45.694188Z",
    "max_value":"500.0",
    "min_value":"333.0",
    "version":"1.0.0"
}
```


Response

Parameters

Status Code 200 OK

Headers

No Headers

Body

JSON XML CSV **PNG**

Response

Parameters

Status Code 200 OK

Headers

No Headers

Body

JSON **XML** CSV PNG


```
<?xml version="1.0" encoding="UTF-8"?>
<eeml
    xmlns="http://www.eeml.org/xsd/0.5.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="0.5.1" xs
    <environment updated="2013-05-05T19:40:08.859383Z" created="2013-03-29T1
        <data id="example">
            <current_value at="2013-05-06T00:30:45.694188Z">500</current_valu
            <max_value>500.0</max_value>
            <min_value>333.0</min_value>
        </data>
    </environment>
</eeml>
```

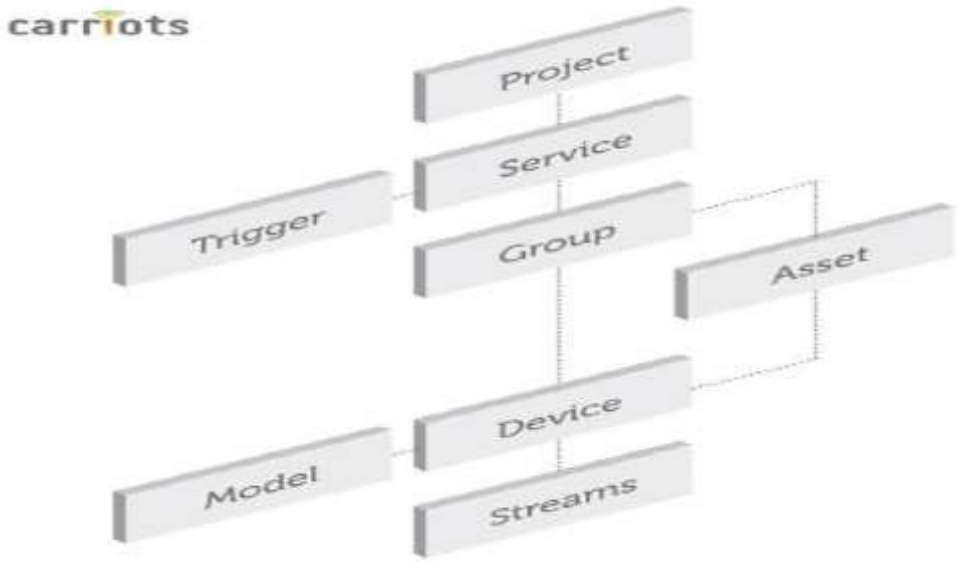
6.2 Carriots

6.2.1 Introducción

Carriots (Carriots, 2013) es sinónimo de Transporte de la Internet de las cosas, y tiene como objetivo proporcionar una plataforma M2M para la integración fácil, rápida y barata de los dispositivos de aficionados y / o sistemas complejos de las empresas. La plataforma Carriots fue creada en 2010 para gestionar los datos de cientos de millones de dispositivos, y en octubre de 2013 se puso en marcha la versión comercial. La versión beta da a los desarrolladores de forma gratuita la posibilidad de gestionar hasta 10 dispositivos.

6.2.2 Modelo jerárquico de Carriots

La plataforma Carriots permite a los dispositivos y a los sistemas individuales la comunicación entre ellos mediante cualquier combinación. No importa la complejidad del sistema o la simplicidad del dispositivo: la plataforma Carriots ve todos como "dispositivos".



La jerarquía Carriots es "Proyecto" -> "Servicio" -> "Grupo" -> "Device". Un desarrollador pertenece a una cuenta de Carriots. Dentro de la cuenta, el desarrollador puede crear varios proyectos genéricos (por ejemplo, un cuenta podría ser para una

ciudad y que podría tener proyectos como vigilancia ambiental o gestión del tráfico. Cada proyecto proporcionará servicios (por ejemplo, a raíz de la mencionada ejemplo, el proyecto vigilancia ambiental podría proporcionar servicios relacionados con la calidad del aire y condiciones básicas). Estos servicios serán alimentados por datos llamados streams y prestados a la plataforma a través de dispositivos. Los dispositivos tienen características de grupo que se pueden agrupar en un Modelo. La plataforma Carriots ofrece una amplia gama de opciones, tales como la transferencia de los servicios de datos recibidos a sistemas externos a través de disparadores asociados a dichos servicios.

6.2.3 Hardware compatible en Carriots

Prototipado:

ARDUNIO, RASPBERRY PI. BEAGLE BONE, FEZ CERBUINO, CUBIE BOARD, NANODE, ELECTRIC IMP.

Industriales:

TST GATE, TST MOTE, CLOUD GATE, TST LIGHT, MOBILE DEVICES C4MAX, MOBILE DEVICES C4EVO, MOBILE DEVICES OBD Dongle.

6.2.4 Carriots API

El corazón de la plataforma tecnológica Carriots es su SDK Carriots; bibliotecas internas basadas en Scripting Groovy las cuales usuarios importan para el desarrollo de software en dispositivos externos para ejecutar listeners y reglas. Los dispositivos interactúan con la plataforma a través de la API REST Carriots. Las peticiones y respuestas HTTP soportan formatos JSON y XML.

Carriots pueden controlar la cuenta de Twitter de un usuario o la cuenta de Dropbox, incluso se pueden enviar datos desde la plataforma hacia Ducksboard o Zoho Reports. Las siguientes tablas especifican en detalle la mayoría de los parámetros de las peticiones HTTP para la gestión de los recursos de la plataforma: proyectos, servicios, grupos, activos, dispositivos, modelos, secuencias de datos, estado flujos de datos, triggers, reglas, listeners y alarmas.

HTTP Status	Descripción
200 OK	Request of the action performed was successfully processed
400 Bad Request	Request is not valid likely due to a malformed JSON body.
401 Unauthorized	Error in the actions “Create a data stream” and “Create a status stream”
404 Not found	The API does not match the requested resource with any of the available ones.
503 No server error	Carriots’ services are unavailable

En el ANEXO II figuran las tablas que muestran las acciones que se pueden realizar para cada recurso. Todas las acciones están formato JSON por defecto, sin embargo, algunas de ellas son capaces de responder en formatos alternativos, como XML.

La siguiente figura muestra la cabecera de la pericón HTTP y su respectiva respuesta HTTP en formato JSON de la acción: **Create a device**


```
{
  "type":"Arduino",
  "id_group":"Grupo@carriots",
"networking":{
"telephone_number":"620145282",
"sim_card":"234582385934578964",
"carrier":"vodafone",
"type":"gprs",
"conn_address":null
},
"name": "newdevice",
"frequency_stream": 5,
"firmware":"db01.v2.7.4.0.0.4_BEEEE",
"sensor":"thermometer",
"description":"Tarjeta Arduino",
"frequency_status":5,
"checksum":"8e0f1692c01e3333d0cdefe6840c5d61ffb7e3b7",
"enabled":true,
"time_zone":"Europe/Madrid"
}
```

```
{
"code":"2001",
"message":"Device created",
"details":{
"type":"Arduino",
"id_group":"5075366928ddcbe564000055",
"networking":{
"telephone_number":"620145282",
"sim_card":"234582385934578964",
"carrier":"vodafone",
"type":"gprs",
"conn_address":null
},
"name":"newdevice",
"frequency_stream":5,
"firmware":"db01.v2.7.4.0.0.4_BEEEE",
"sensor":"thermometer",
"description":"Tarjeta Arduino",
"frequency_status":5,
"checksum":"8e0f1692c01e3333d0cdefe6840c5d61ffb7e3b7",
"enabled":true,
"time_zone":"Europe/Madrid",
"id_developer":"newdevice@carriots",
"status":"ok",
"created_at":1355967817,
"owner":"carriots",
"_id":"50d26d495c5d75b81a00000d"
}
}
```


6.3 ThingSpeak

6.3.1 Introducción

ThingSpeak (ThingSpeak, 2013) es una plataforma web impulsada por ioBridge, Inc., (ioBridge, 2013) una empresa que ofrece soluciones de cosas basadas en la Web para los fabricantes, profesionales y los usuarios ocasionales. ThingSpeak ofrece servicios para la recolección de datos en tiempo real de los dispositivos, procesamiento de los datos, visualizaciones de datos para una amplia gama de aplicaciones tales como monitoreo de sensores, control de la energía, el seguimiento de ubicación geográfica o la interconexión con las redes sociales. Proporciona una API de código abierto para la conexión a la infraestructura ThingSpeak que es apoyada por más de 500 servidores.

6.3.2 Hardware compatible

Arduino, Raspberry pi, Electric Imp, Freetonics, Netduino y Xbee wíreless networks.

6.3.3 ThigSpeak API

ThingSpeak API proporciona una interfaz para el intercambio de información entre los dispositivos físicos, entidades ThingSpeak y plataformas de terceros como Twitter, Prowl y Twilio. La plataformaThingSpeak permite que cualquier dispositivo físico con socket TCP pueda enviar peticiones HTTP a la API y almacenar o recuperar cualquier tipo de dato.

La plataforma ThingSpeak almacena los datos en "Canales ThingSpeak". Los canales se representan como una interfaz web donde se publican los datos almacenados. Pueden ser configurados para ser públicos para que otras personas puedan verlos o privados, sólo se puede acceder mediante el registro en ThingSpeak.com con cuenta de usuario. Las aplicaciones cliente están integradas en los dispositivos físicos que pueden leer y escribir a un canal ThingSpeak mediante peticiones HTTP a la API de ThingSpeak. Cada entrada o feed es etiquetada con una ID de entrada única y se almacenan con una fecha y un time stamp. Escribir en un canal requiere una clave de escritura para asegurar que sólo las aplicaciones autorizadas pueden acceder a sus datos. Los datos numéricos pueden ser procesados como escala de tiempo, promedio, mediana, suman, y el redondeo. Los canales soportan los formatos JSON, XML, CSV.

Action performed	Method	Endpoint
Updating a channel	POST	/update
Retrieving channel feeds	GET	/channels/(channel_id)/feed.(format)
Retrieving the last entry in channel feed	GET	/channels/(channel_id)/feed/last.(format)
Retrieving a field feed	GET	/channels/(channel_id)/field/(field_id).(format)
Retrieving the last entry in a Field Feed	GET	/channels/(channel_id)/field/(field_id)/last.(format)
Retrieving status updates	GET	/channels/(channel_id)/status.(format)

6.3.4 Chart API

Se utiliza para mostrar los datos almacenados en los canales ThingSpeak en gráficos. Estos gráficos se denominan "charts" y pueden presentar datos numéricos y puede ser embebidos en sitios web externos. Los gráficos pueden ser estáticos o dinámicos. Los gráficos dinámicos visualizan la variación de datos en tiempo real.

6.3.5 Thing Tweet

Twitter da a sus usuarios la opción de actualizar su cuenta a través de una autenticación abierta. Sin embargo, esta opción no está diseñada para dispositivos con capacidades de procesamiento limitadas dichas como Arduino o Netduino. ThingSpeak la API resuelve el problema con ThingTweet APP. Ejecuta un Twitter proxy que envía actualizaciones de estado a Twitter a través de las llamadas ThingSpeak API.

6.3.6 ThingHTTP

ThingHTTP permite la conectividad de dispositivos con restricciones a cualquier servicio web como Prowl y Twilio usando HTTP a través de una red o de Internet. Esta aplicación elimina la necesidad de que los dispositivos implementen el protocolo para tratar con cada servicio web. Además, es posible incorporar un "Parse String" dentro de la petición HTTP a ese servicio, para evitar la codificación de un analizador sintáctico en los dispositivos restringidos.

6.3.7 TweetControl

API de Twitter Streaming permite monitorizar flujos en tiempo real. Hacer esto requiere un servidor dedicado con proceso de larga duración que esté preguntando constantemente a Twitter para las actualizaciones de estado. ThingControl App hace qla tarea mencionada. A continuación, el desarrollador puede concentrarse en determinar el stream y "hasttag" específico y poder realizar una acción de control tal como el envío de una solicitud ThingHTTP a terceras partes.

6.3.8 React

La plataforma ThingSpeak procesa streams de datos y permite la posibilidad de establecer acciones de trigger cuando se cumpla una condición con respecto a esos datos en un canal ThingSpeak. Los tipos de condición dependen de los datos específicos a monitorizar; datos de los sensores, textos, etc. Cuando se cumple una condición React puede desencadenar una solicitud ThingHTTP a una tercera entidad o publicar un Tweet con ThingTweet App.

Las siguiente figura muestra un ejemplo de petición HTTP para la creación de un canal.

Example POST:

```
POST https://api.thingspeak.com/channels.json
     api_key=XXXXXXXXXXXXXXXX
     name=My New Channel
```

The response will be a JSON object of the new channel, for example:


```
{
  "id": 4,
  "name": "My New Channel",
  "description": null,
  "metadata": null,
  "latitude": null,
  "longitude": null,
  "created_at": "2014-03-25T13:12:50-04:00",
  "elevation": null,
  "last_entry_id": null,
  "ranking": 15,
  "username": "hans",
  "tags": [],
  "api_keys":
  [
    {
      "api_key": "XXXXXXXXXXXXXXXX",
      "write_flag": true
    }
  ]
}
```


6.4 Comparativa plataformas

6.4.1 Comparativa

Como hemos podido ver en el análisis todas las plataformas son bastante similares a la hora de tratar la información , usando cada una de ellas su API correspondiente.

Veíamos que cada una usaba un formato , algunos de ellos coinciden en todas las plataformas, como el JSON, XML, y el formato PNG sólo lo tenía Xively.

El corazón de la plataforma tecnológica Carriots es su **SDK Carriots**; bibliotecas internas basadas en Scripting Groovy que los usuarios importen para el desarrollo de software en los dispositivos externos para ejecutar oyentes y reglas. Los dispositivos interactúan con la plataforma a través de la API REST Carriots. Las peticiones y respuestas HTTP soportan formatos JSON y XML.

Carriots pueden controlar la cuenta de Twitter del usuario o cuenta de Dropbox, incluso se pueden enviar los datos a partir de la plataforma hacia Ducksboard o Zoho Reports.

Xively API La mayoría de las peticiones HTTP soportan JSON, XML y CSV (JSON es el formato por defecto) . Al igual que los anteriores , terceras entidades pueden comunicarse con el API.

ThingSpeak también interactúa con plataformas de terceros como Twitter, Prowl y Twilio .Los canales soportan los formatos JSON, XML, CSV.

6.4.2 Tabla Comparativa

	Xively	Carriots	Thingspeak
Formatos	JSON, XML,CSV,PNG	JSON,XML	JSON,XML,CSV
Plataformas de terceros	USO DE OAuth PARA CONEXION CON TERCERAS APLICACIONES	TWITER, DROPBOX DUCKSBOARD, ZOHO REPORTS	TWITTER, PROWL, TWILIO
Pros	- ORIGEN PACHUBE - UNA DE LAS PRIMERAS PLATAFORMAS - MUCHA DOCUMENTACION EN LA RED	USO DE LIBRERIA DE CLICARRIOTS	INTERFAZ DE PRESENTACION DE LOS CANALES MUY AMIGABLE
Contras	HASTA 10 SENSORES	POCA DOCUMENTACIÓN EN LA RED	- POCA DOCUMENTACION EN LA RED - FALTAN FUNCIONALIDADES - MENOS HARDWARE COMPATIBLE

7 Análisis de Plataformas Hardware en IoT

7.1 Raspberry Pi

Raspberry Pi es un ordenador de placa reducida o placa única de bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) Video Core IV, y 512 MB de memoria RAM

(aunque originalmente al ser lanzado eran 256 MB). El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa.

En cuanto al SO El Raspberry Pi usa mayoritariamente sistemas operativos basados en el núcleo Linux. Raspbian, una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi, se lanzó durante julio de 2012 y es la distribución recomendada por la fundación para iniciarse.

La SBC de la Raspberry Pi Foundation es la auténtica y oficial, aunque hayan salido competidores. Esta placa surgió como estímulo para las escuelas y el acercamiento de las ciencias de la computación a más personas. El primer germen apareció en 2006, aunque era una placa basada en microcontrolador Atmel ATmega644 similar a los de Arduino. Al ser abierta, tanto sus esquemas y los datos de diseño están disponibles para su descarga, y es aquí donde radica su mayor éxito, junto con su precio asequible

Pero no sería hasta 2009 cuando se creó la Fundación Raspberry Pi en Caldecote, South Cambridgeshire (Reino Unido) y sus fundadores son: Eben Upton, David Braden, Jack Lang, Pete Lomas, Alan Mycroft y Robert Mullins.

El cofundador Upton, director técnico del departamento sobre arquitectura ASIC de Broadcom, quería llevar a los niños y entusiastas la misma filosofía del ordenador Acorn BBC Micro de 1981. Y su proyecto sería apadrinado por la Universidad de Cambridge y la compañía Broadcom (creadora de los SoCs integrados en la placa).

Dos años más tarde de aparecer la fundación, se fabricarían las primeras placas prototipo para luego comenzar a venderse años después. Su éxito fue tal que se agotó y las manufactureras no daban abasto para sacar al mercado tantos dispositivos como demanda. El primer lote se fabricaría en Taiwan y China, unas 10.000, para luego trasladar la producción al Reino Unido. Se fabrican miles al día gracias a una fábrica de Sony en Pencoed, Gales.

Las placas certificadas por la The Raspbery Pi Foundation son:

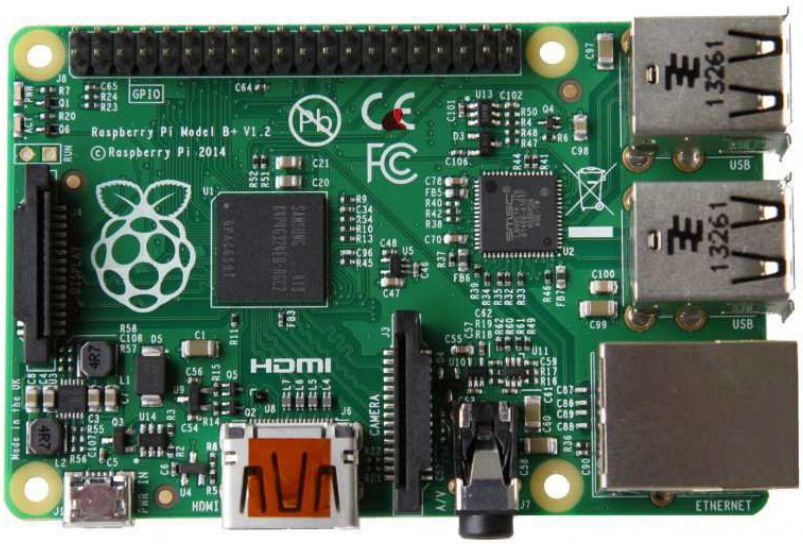
Raspberry Pi Model A : apareció con un precio de unos 25 dólares de coste en fábrica, con un SoC Broadcom BCM2835 que integra una CPU, GPU, DSP, RAM y controlador

USB. Este chip es común a todos los modelos, tanto B como B+. La CPU es una ARM1176JZF-S de la familia ARM11 y con un set de instrucciones ARMv6. Trabaja a 700Mhz, pero no está sola dentro del SoC, también incluye una GPU VideoCore IV a 250Mhz encargada de los gráficos y el inicio. Esta GPU soporta OpenGL ES como API gráfica. Además se complementa con una memoria SDRAM de 256MB que comparten CPU y GPU. Por otro lado, dispone de una conexión HDMI para una pantalla externa y un Jack RCA, Jack para audio, ranura para SD y MMC, pines GPIO, conexión de alimentación por microUSB de 5v o por los cabezales GPIO. El peso y dimensiones son parecidos en las tres placas oficiales.

Raspberry Pi A+ : sus características son similares a la nueva B+, excepto que su tamaño es más pequeño (solo mide 65 x 56 mm) y que su memoria RAM se ha reducido a 256 MB. También ha cambiado su conectividad, que ahora rescinde de su conexión Ethernet. Igual que ocurría con la Raspberry Pi A y Raspberry Pi B iniciales, la A+ esmás barata. Al igual que ocurría con la B+, esta placa es más eficiente energéticamente que las anteriores.

Raspberry Pi Model B : con un precio de unos 35 dólares de coste en fábrica, las únicas diferencias con el modelo A son los 512MB SDRAM incluidos en el SoC, dos puertos USB y la integración de un adaptador Ethernet para conexión a red. El consumo, al tener más elementos sube de los 300mA (1,5w) hasta los 700mA (3,5w). El resto de características son comunes. Es el modelo más vendido (más de 3 millones de ventas).

Raspberry Pi Model B+ : es la actualización del modelo B, cuyas ventajas son la integración de más pines GPIO (pasando de los 26 de los modelos anteriores a los 40, respetando el esquema de patillaje anterior), un consumo inferior (600mA, 3w), inclusión de 4 puertos USB on-board y una mejora en la fuente de alimentación para reducir el ruido y mejorar el sistema de sonido. Por el resto de características es idéntica a la B. Lo más práctico es lo de los puertos USB, que se echan mucho en falta en los modelos anteriores, aunque puede ser solucionado con un hub. Todo esto manteniendo el mismo precio del Model B, debido a que la tecnología ha madurado y permite su manufactura a un menor coste.



7.1.2 Accesorios para la Raspberry Pi

El gran éxito y la comunidad que se ha formado alrededor de este invento han hecho que se disponga en el mercado de diversos accesorios o módulos para la Raspi. Son elementos que agregan funcionalidades a la placa base, como los shields o escudos de Arduino.

7.2 Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Proccesing / Wiring y el cargador de arranque que es ejecutado en la placa.

Desde octubre de 2012, Arduino se usa también con microcontroladoras CortexM3 de ARM de 32 bits, que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas que compilen sin cambios en las dos plataformas. Eso sí, las microcontroladoras CortexM3 usan 3,3V, a diferencia de la mayoría de las placas con AVR que generalmente usan 5V. Sin

embargo ya anteriormente se lanzaron placas Arduino con Atmel AVR a 3,3V como la Arduino Fio y existen compatibles de Arduino Nano y Pro como Meduino en que se puede conmutar el voltaje.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Proccesing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador.

El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el Prix Ars Electrónica de 2006.

Un poco de historia: Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia). En ese tiempo, los estudiantes usaban el microcontrolador BASIC Stamp, cuyo coste era de 100 dólares estadounidenses, lo que se consideraba demasiado costoso para ellos. Por aquella época, uno de los fundadores de Arduino, Massimo Banzi, daba clases en Ivrea.

El nombre del proyecto viene del nombre del *Bar di Re Arduino* (Bar del Rey Arduino) donde Massimo Banzi pasaba algunas horas. En su creación, contribuyó el estudiante colombiano Hernando Barragán, quien desarrolló la tarjeta electrónica Wiring, el lenguaje de programación y la plataforma de desarrollo. Una vez concluida dicha plataforma, los investigadores trabajaron para hacerlo más ligero, más económico y disponible para la comunidad de código abierto (hardware y código abierto). El instituto finalmente cerró sus puertas, así que los investigadores, entre ellos el español David Cuartielles, promovieron la idea. Banzi afirmaría años más tarde, que el proyecto nunca surgió como una idea de negocio, sino como una necesidad de subsistir ante el inminente cierre del Instituto de diseño Interactivo IVREA. Es decir, que al crear un producto de hardware abierto, éste no podría ser embargado.

Posteriormente, Google colaboró en el desarrollo del Kit Android ADK (Accesory Development Kit), una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android para que el teléfono controle luces, motores y sensores conectados de Arduino.

Para la producción en serie de la primera versión se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul, debía ser Plug and Play y que trabajara con todas las plataformas informáticas tales como MACOSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVRAE, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos.

En el año 2005, se incorporó al equipo el profesor Tom Igoe, que había trabajado en computación física, después de que se enterara del mismo a través de Internet. Él ofreció su apoyo para desarrollar el proyecto a gran escala y hacer los contactos para distribuir las tarjetas en territorio estadounidense. En la feria Maker Fair de 2011 se presentó la primera placa Arduino 32 bit para trabajar tareas más pesadas.

A la hora de seleccionar la placa para nuestro proyecto tenemos que tener esto muy presente para no llevarnos sorpresas. Puede que nos interese una placa compatible por ciertas cualidades del hardware que no posee Arduino o por cuestiones de licencias y sin embargo querer que sea compatible con el entorno de desarrollo Arduino IDE. En otras ocasiones puede que simplemente se desee compatibilidad en cuanto a los shields pero se tiene la necesidad de emplear otro compilador (AVR Studio, Makefiles,…).

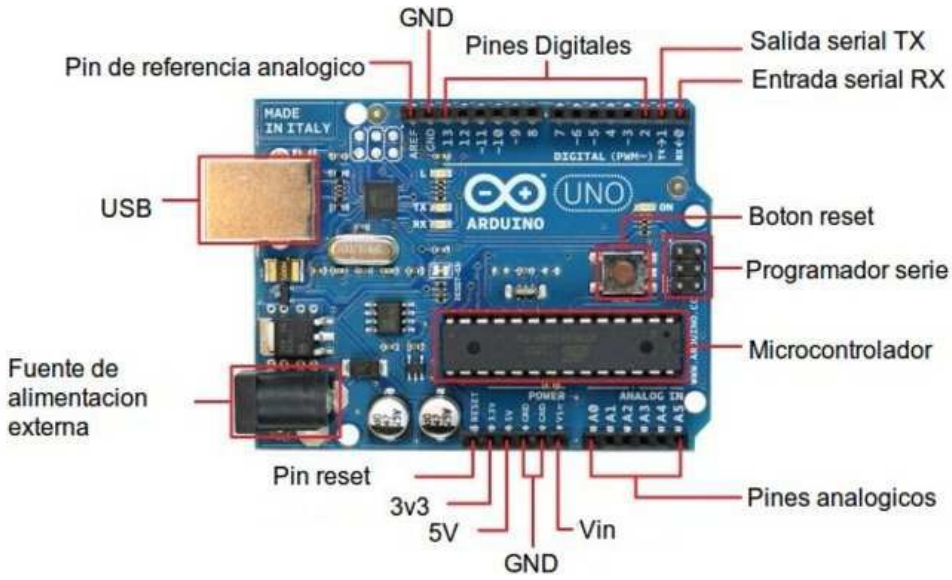
Lo principal que debemos saber es que tipo de proyectos vamos a implementar. Con esto nos da una idea de la cantidad de pines analógicos y digitales (normales y de tipo PWM o modulados por ancho de pulso para simular una salida analógica) que necesitamos para nuestro trabajo. Este primer escrutinio nos permite descartar algunas placas más simples que no tengan suficientes pines o, al contrario, descartar las de mayor número de ellos para reducir los costes puesto que con menos pines nos conformamos.

También podemos deducir el tamaño de código que vamos a generar para nuestros sketchs. Un programa muy largo, con muchas constantes y variables demandará una antidad mayor de memoria flash para su almacenamiento, por lo que se debe elegir una

placa adecuada para no quedarnos cortos.

La RAM será la encargada de cargar los datos para su inmediato procesamiento, pero no es uno de los mayores escollos, puesto que esto solo afectaría a la velocidad de procesamiento. La RAM va ligada al microcontrolador, puesto que ambos afectan a la agilidad de procesamiento de Arduino.

Por último, en cuanto al voltaje, no importa demasiado a nivel electrónico, excepto en algunos casos, para tener en cuenta la cantidad de tensión que la placa puede manejar para montar nuestros circuitos. Esto no supone mayor problema, puesto que una placa de Arduino podría trabajar incluso con tensiones de 220v en alterna con el uso por ejemplo de relés. Pero cuando queremos prescindir de una fuente de alimentación externa, hay que tener en cuenta que este es el voltaje que se puede manejar. Y entre otras cosas marcar el límite para no destruir la placa con sobretensiones no soportadas. Pero no confundas el voltaje al que trabaja el microcontrolador y al que funcionan los periféricos de la placa.



De entre las placas oficiales puedes encontrar multitud de modelos. Todos especialmente pensados para un fin, compatibles con los shields y módulos oficiales, así como con Arduino IDE. Vamos a intentar detallar las principales características de una de ellas:

- **Arduino UNO R3 :** es la plataforma más extendida y la primera que salió al mercado, por ello nos podemos basar en esta para hacer la comparativa con el resto de placas. Todas las características de esta placa

estarán implementadas en casi todas las placas restantes, a excepción de algunas que ya veremos. Se basa en un microcontrolador Atmel ATmega320 de 8 bits a 16Mhz que funciona a 5v. 32KB son correspondientes a la memoria flash (0,5KB reservados para el bootloader), 2KB de SRAM y 1KB de EEPROM. En cuanto a memoria es una de las placas más limitadas pero no por ello resulta insuficiente para casi todos los proyectos que rondan la red. Las salidas pueden trabajar a voltajes superiores, de entre 6 y 20v pero se recomienda una tensión de trabajo de entre 7 y 12v. Contiene 14 pines digitales, 6 de ellos se pueden emplear como PWM. En cuanto a pines analógicos se cuenta con hasta 6. Estos pines pueden trabajar con intensidades de corriente de hasta 40mA.

7.3 INTEL GALILEO Gen 2

La placa de desarrollo Intel® Galileo Gen 2 es la primera de una familia de placas de desarrollo y para prototipos certificada por Arduino basada en la arquitectura Intel y diseñada específicamente para creadores, estudiantes, educadores y entusiastas de la electrónica. La placa Intel Galileo Gen 2 proporciona a los usuarios un entorno de desarrollo de software y hardware de código totalmente abierto. Además, complementa y amplía la linea de productos Arduino para ofrecer funciones informáticas más avanzadas a los usuarios ya familiarizados con las herramientas de prototipo Arduino. Por sus características de diseño, la placa de desarrollo Intel Galileo Gen 2 viene con software, hardware y pines compatibles con una amplia gama de protectores Arduino Uno R3. Por otra parte, permite a los usuarios incorporar llamadas de firmware Linux en la programación para bocetos de Arduino.

Procesador de aplicaciones Intel Quark SoC X1000, 32 bits, un núcleo, un hilo, compatible con la arquitectura de conjunto de instrucciones (ISA) del procesador Intel Pentium, que funciona a velocidades de hasta 400 MHz.
Compatible con una amplia gama de interfaces de E/S estándar de la industria, como la ranura de tamaño completo mini-PCI Express, el puerto Ethernet de 100 Mb, la ranura de microSD, el puerto anfitrión USB y el puerto cliente USB.

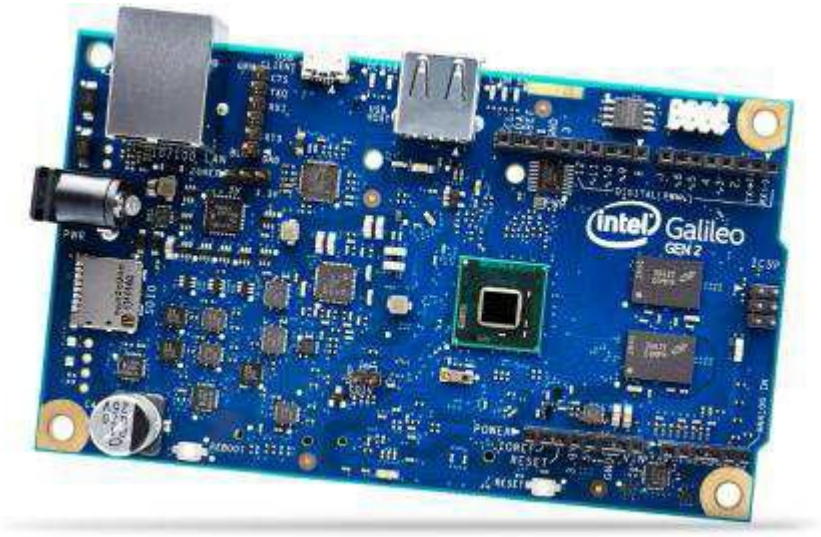
DDR3 de 256 MB, SRAM integrada de 512 kb, memoria NOR Flash de 8 MB y EEPROM de 8 kb de serie en placa; además, admite una tarjeta microSD de hasta

32 GB.
Compatibilidad de hardware y pines con una amplia gama de protectores Arduino Uno R3.

Programable a través del entorno de desarrollo integrado (IDE) de Arduino, compatible con los sistemas operativos anfitriones Microsoft Windows, Mac OS y Linux.
Compatibilidad con la versión Yocto 1.4 Poky de Linux.

Lo nuevo con respecto a la generación 1:

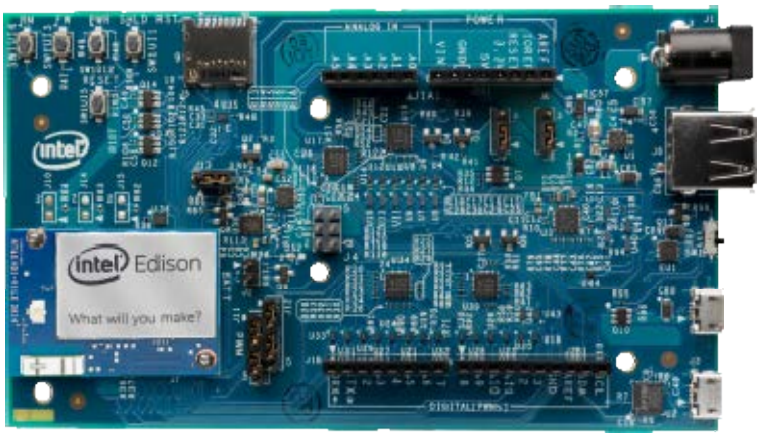
- El colector de 6 pines de 3,3 V USB TTL UART reemplaza el puerto de la consola del conector RS-232 de 3,5 mm para depurar Linux. El nuevo conector de 6 pines puede usarse con el cable de serie USB FTDI estándar (TTL-232R- 3V3) y las populares placas convertidoras de USB a serie. 12 GPIO totalmente nativos para mayor velocidad y una resistencia superior de la unidad.
- Modulación por amplitud de impulsos (PWM) de 12 bits para un control más preciso de servos y una respuesta más fluida.
- UART1 de la consola puede redireccionarse hacia los colectores Arduino en los bocetos, eliminando así la necesidad de una unidad de serie de software en muchos casos.
- Compatible con la alimentación a través de Ethernet (PoE) de 12 V (es necesario instalar el módulo PoE).
- El sistema de regulación de energía se modificó para admitir fuentes de alimentación de 7 a 15 V.



7.4 Intel EDISON

La plataforma de desarrollo Intel Edison es la primera de una serie de plataformas informáticas de uso general, bajo coste y listas para el producto que ayudan a derribar barreras para que emprendedores de cualquier nivel, desde diseñadores profesionales hasta consumidores y empresas, puedan trabajar en Internet de las cosas. La plataforma de desarrollo Intel Edison incluye un robusto conjunto de características en un tamaño pequeño, proporcionando un gran rendimiento, durabilidad y un amplio espectro de soporte para software y E/S.

Con el tamaño de una tarjeta SD (35.5 x 25.0 x 3.9 mm) y con poder suficiente para proyectos entusiastas, Intel pone a la venta Edison, que puede conectarse a varias placas, siendo una de ellas el Arduino. Cuenta con un CPU Atom Silvermont de doble núcleo a 500 MHz con un Quark a 100 MHz. Se ofrece 1 GB de RAM LPDDR, 4 GB de almacenamiento NAND eMMC, conectividad Wi-Fi 802.11a/b/g/n de 2.4 y 5 GHz, así como Bluetooth 4.0, con una antena en la placa. El sistema operativo es Yocto Linux 1.6.



7.5 Comparativas

Cada una tiene sus pros y sus contras, y una plataforma es mejor que otra para una determinada aplicación. Es importante mencionar que Arduino impulsó la tendencia de su uso como microcontrolador, mientras que tanto el Raspeberry Pi y Galileo son microprocesadores.

Microcontroladores VS Microprocesadores: un microcontrolador es un circuito integrado diseñado con el propósito de tareas específicas. Es principalmente usado en productos que requieren un grado de control impuesto por el usuario. Los microprocesadores en cambio, son usados para ejecutar aplicaciones grandes y genéricas.

Arduino: Para principiantes y proyectos de propósito simple.
Es la principal plataforma de la comunidad DIY, ya que es open-source. Es fácil de desarrollar, consume poca energía y es muy simple de usar. Además, está especialmente diseñado para principiantes, de tal forma que cualquiera pueda jugar con el mismo y conectarlo a componentes externos.
En esencia, el Arduino es una plataforma pequeña programable que acepta y almacena código de la computadora convencional. Es capaz de cosas simples, pero buenas, como controlar luces o controlar sistemas de jardinería. La plataforma, el lenguaje de programación y muchos proyectos ya hechos que se pueden encontrar son de distribución libre, dispuestos a ser utilizados para adecuarse a las necesidades.

Su uso es tan sencillo que cualquiera puede usarlo, es decir, no se precisa de

conocimientos muy profundos de programación ni electrónica. Su uso es el punto perfecto de partida para cualquiera que busca iniciarse en la movida electrónica DIY debido a su simplicidad.

Ventajas: el Arduino es relativamente barato para disponer de varias unidades y explotar su uso. Además de su estandarte Arduino Uno, se disponen de muchas variaciones de modelos de Arduino para elegir. Como es de bajo consumo, es ideal para aplicaciones de usos de larga duración, o incluso para uso de baterías. Pero sobre todo, el Arduino tiene una popularidad muy alta, lo que conlleva a una gran facilidad de encontrar apoyo, documentación sobre proyectos particulares, tutoriales, etc. Además, presenta flexibilidad para distintos tipos de interfaces.

Desventajas: Es una plataforma para principiantes. Si bien tiene una amplia proyección de uso y aplicaciones como se nombró en las ventajas, aún toma tiempo acostumbrarse a usar algo sin interfaz gráfica. Debido a lo barato y pequeño que es, normalmente el Arduino no puede manejar diferentes procesos al mismo tiempo, lo cual hace que no sea bueno para proyectos que requieren mayor poder de cómputo.
El Arduino es mejor para: proyectos de propósito simple. Por ejemplo, un sistema en el que el secador de ropas envía un mensaje de texto cuando estas están listas, o un sistema de video para timbres. Su uso es más práctico y mejor aprovechado para interactuar con objetos en el mundo real. Si la aplicación requiere conexión a Internet, tener un display multi-touch y completa automatización, el Arduino probablemente no funcionaría.

Raspberry Pi: Para proyectos de multimedia complejos o basados en Linux.
El Raspberry Pi ha sido un icono para DIY (Do It Yourself) desde un principio. Esencialmente, es una pequeña computadora que corre Linux desde una tarjeta SD, de la que se puede correr todo tipo de proyectos DIY. En pocas palabras, es una computadora Linux de bajo consumo, de tal forma que en principio puede hacer lo que hace una máquina Linux a un costo más bajo. Con 2 puertos USB y la salida HDMI, puede usarse como cualquier computadora, lo cual significa que es perfecto para proyectos que requieran un sistema Linux. Es así que Raspberry Pi es ideal para requerimientos de pantalla y especialmente, proyectos que requieran conexión a internet.

Ventajas: Una pequeña computadora trae toda clase de ventajas. El puerto HDMI

puede usarse para conectarse a un televisor y los 2 puertos USB permiten operarlo como a una computadora con teclado y mouse fácilmente. Su procesador gráfico soporta 1080p. También tiene un puerto ethernet para fácil conexión a internet con leves dificultades. Ya que el sistema operativo corre desde una tarjeta SD, este puede cambiarse fácilmente con solo cambiar la tarjeta. Esto es muy útil considerando que se tienen varias opciones para el sistema operativo.

Dado su precio, es poderoso y aun así de fácil uso para principiantes. En cuanto a capacidad de expansión, es importante mencionar que se tiene grandes beneficios gracias a la placa que permite la compatibilidad con los shields de Arduino, ya que sin incluir estas expansiones, el soporte exclusivo para el Pi es muy bajo.

Desventajas: Es muy útil para cualquier proyecto que implique el uso de una computadora, pero a diferencia del Arduino y el BeagleBone, no tiene tantas opciones para interfaces con sensores externos o botones, es decir, orientación a hardware, lo que no lo hace una buena opción para proyectos meramente electrónicos.

El Raspberry Pi es mejor para: proyectos que requieren interfaz gráfica o internet. Ya que sus orígenes se basan en la educación, es también mejor aprovechado para principiantes que buscan un proyecto de computación educativo de bajo costo. Debido a sus varias entradas y salidas, también tiende a ser una plataforma preferida para proyectos multimedia como el XBMC Media Center.

Galileo & EDISON

Se estima que esta plataforma cambiará la forma de ver a los sistemas empotrados. El Edison presenta características robustas en su pequeño tamaño, garantizando buen desempeño, duración y un amplio espectro de soporte software y de periféricos de entrada-salida, para satisfacer las necesidades de inventores, entusiastas y principiantes. Su bajo consumo y tamaño lo hacen ideal para proyectos que requieren mucha capacidad de cómputo, pero no disponen de una mayor fuente de alimentación cercana. Para qué usar el Edison, si bien 500 MHZ ya puede resultar relativamente poco en comparación a un típico smartphone, de 1 GHz o más, Intel promueve su uso para aplicaciones de bajo consumo, por ejemplo, para el internet de las cosas. A diferencia del Raspberry Pi , el Edison no tiene salida de video. Aún así es un poderoso sistema empotrado. La idea es ser capaz de programar el Edison usando el software Arduino, o

puede crearse programas personalizados en Linux usando C, C++, Python, Pascal, entre otros.

Además del Edison, Intel también está lanzando un gran número de plataformas de expansión, para el Edison. La primera es el Breakout Board, que es una pequeña placa en la que se conecta el Edison, que provee energía, puerto USB on-the-go y una sección de prototipo donde los pins son conectados a los GPIO del Edison. Otra expansión es la placa Arduino. El Edison se conecta a esta y así se tiene compatibilidad con todos los shields de Arduino. Es así que con tales especificaciones, los entusiastas a su vez tienen una amplia gama de oportunidades para crear sus propias expansiones para una aplicación determinada.

7.6 Tabla Comparativa Plataformas Hardware

PLATAFORMAS	Expansion	Finalidad	Pros	Contras
Raspberry Pi	Si con Arduino	Más indicado para multimedia	Ideal para iniciarse Mucha documentacion	Limitado para proyectos de electrónica
Arduino	Si muchos módulos	Proyectos de propósito simple	Mucha documentación	No válido en proyectos que requieren mayor poder de cómputo
Galileo	Si	IoT	Bajo consumo Tamaño	Poca documentación
Edison	Si	IoT	Muchas plataformas de expansión Tamaño	No salida de video Poca documentación

8 Aplicación real de IoT en Raspberry Pi

8.1 Funcionalidades de la aplicación real

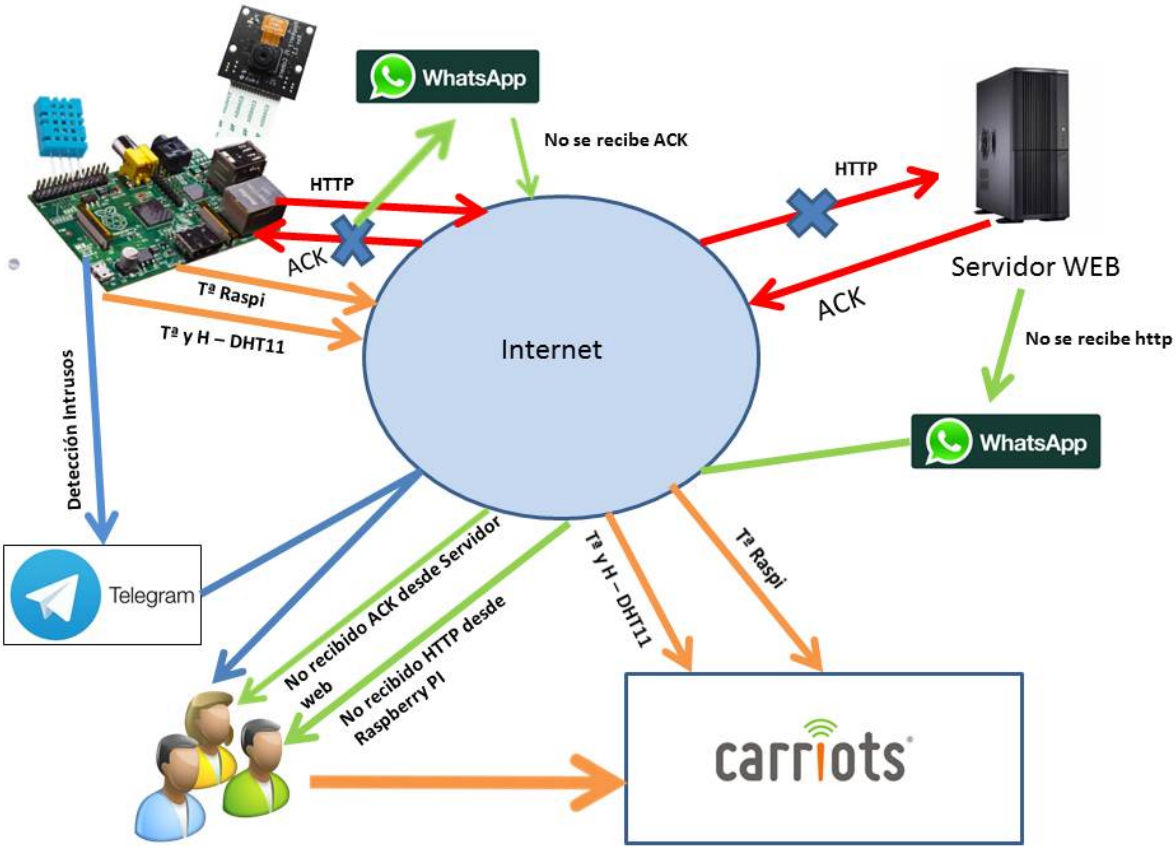
La aplicación real dispondrá de las siguientes funcionalidades:

- Aviso a través de la aplicación de mensajería Whatsapp de posible fallo de suministro eléctrico en casa o posible fallo de red.
- Sistema de video vigilancia con detección de movimiento, aviso con mensaje y fotos a través de TELEGRAM.
- Monitorización de temperatura interna de la Raspberry pi, con subida de datos a plataforma Carriots.
- Monitorización de la temperatura y humedad del domicilio, con subida de

datos a plataforma Carriots.

La idea original era usar Xively como plataforma software de IoT, pero a día de hoy no admite más registros, por lo cual será la plataforma Carriots la usada para la subida de datos de los sensores.

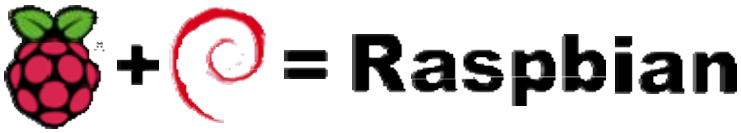
El siguiente esquema representa la solución a implementar:



8.2 Implementación de plataforma hardware.

8.2.1 Instalación y configuración de sistema operativo en raspberry pi

El sistema operativo que instalamos en la raspberry pi será Raspbian lo descargamos desde aquí: Raspbian. Y su página oficial es la siguiente: Web oficial.



El disco duro de la Raspberry es la tarjeta SD**.**

En esa tarjeta instalaremos el sistema operativo, debe ser de al menos 2GB. En nuestro caso usaremos una SD de 8 GB y necesitaremos otro ordenador para preparar la tarjeta. Una vez que tenemos la imagen descargada de Internet , la descomprimimos en una carpeta y copiamos esa imagen o archivo con extensión **".img"** en la SD usando uno de estos programas:

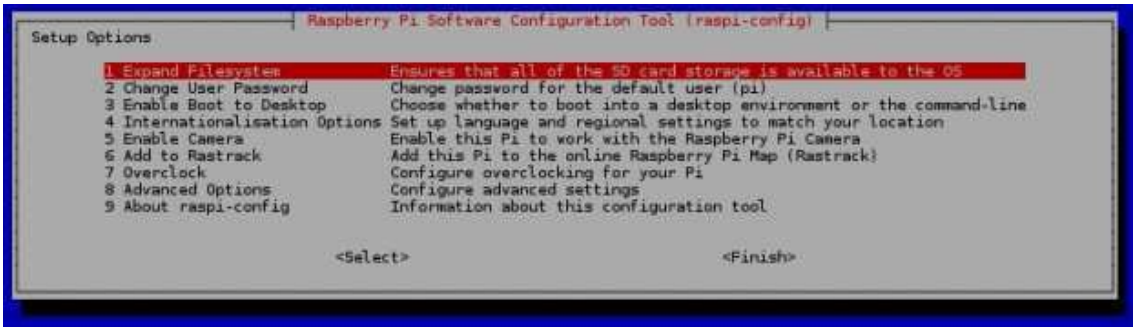
1. Windows – **Win32DiskImage**
2. OSX – **PiWriter**
3. Linux – o con el comando dd o con **USB imagewriter**

Una vez finalizado el proceso de copia del sistema operativo en la tarjeta SD, la introducimos en la raspberry pi y la encendemos. A continuación configuraremos una serie de parámetros básicos.

Se ejecutará un programa llamado raspi-config, este programa podrá ejecutarse manualmente cuando se quiera, con el siguiente comando:

sudo raspi-config

El primer menú que muestra contiene 9 diferentes opciones disponibles, a continuación revisaremos alguna de las opciones disponibles:



Opción 1 – Expandir el sistema de archivos (Expand Filesystem)

Esta opción permite expandir el sistema operativo para que utilice todo el espacio disponible en la tarjeta. Cuando se instala Raspbian "Wheezy" la imagen copiada en la tarjeta solo ocupa 2 GB, por lo tanto es necesario ejecutar esta opción para que todo el espacio de la tarjeta SD sea utilizado.

Opción 2 – Cambiar la contraseña del usuario Pi (Change User Password)

En el Raspberry Pi y en general en sistemas Linux existen diferentes tipos de usuario, los dos que vienen predeterminados por el sistema son los usuarios "*root*" y "*pi*"

El más importante "*root*", tiene acceso privilegiado a todos los archivos, configuraciones y carpetas del sistema. El otro tipo de usuario son los comunes como lo es "*pi*", este viene predeterminado con la contraseña "*raspberry*" por lo tanto cualquier persona podría acceder al sistema. Por eso, es recomendable cambiar la contraseña en esta opción

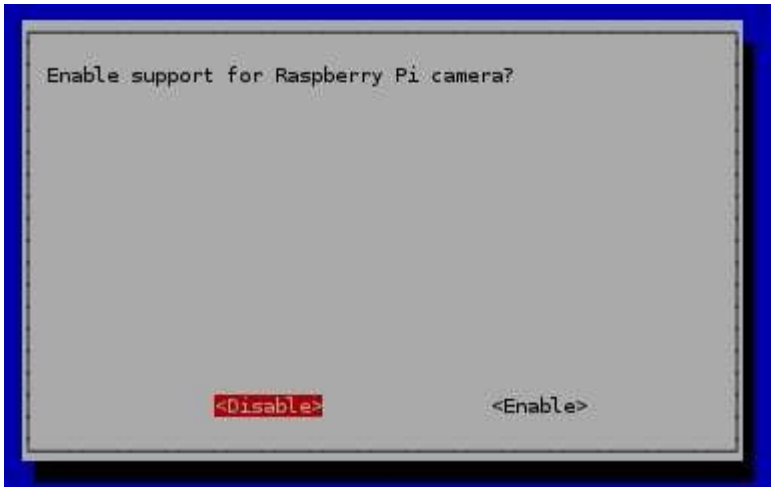
Opción 4. Opciones de internacionalización (Internationalisation Options)

Esta opción permite modificar el lenguaje del sistema operativo, la zona horaria y la distribución de su teclado.

La opción I1 sirve para indicar donde nos encontramos ubicados, esta opción configura el lenguaje del sistema operativo, los caracteres, la denominación de la moneda, etc.

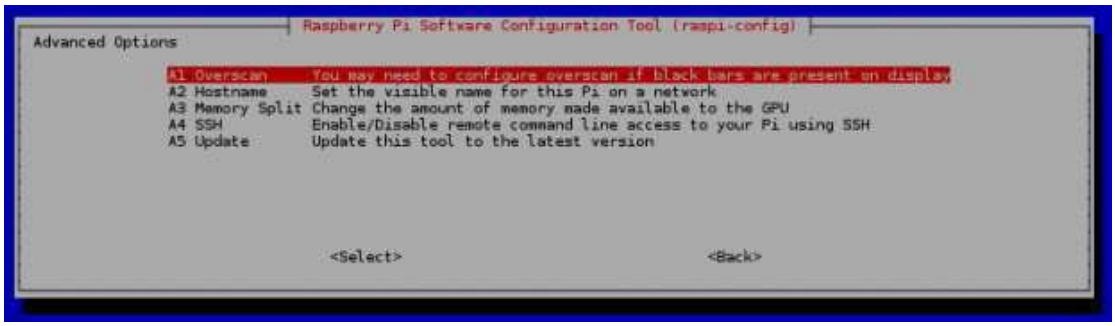
Opción 5 – Activar la cámara (Enable camera)

Esta opción sirve para dar soporte a la cámara de Raspberry Pi, esta opción permite activar el puerto para que haya comunicación entre la CPU y el controlador de la cámara.



Opcion 8. Opciones avanzadas (Advanced Options)

Este opción presenta un otro submenú con las siguientes opciones.



La opcion A1 overscan sirve para borrar las líneas negras en algunos monitores o televisores La opción A2 Hostname, sirve para identificar la Raspberry Pi en la red local. Hay que tener en cuenta que el sistema diferencia mayúsculas y minúsculas.

La opción A4 – Activar SSH (Enable SSH) se utiliza para acceder a la Raspberry Pi remotamente desde un cliente SSH. SSH significa "Secure SHell" el cual es una forma segura de conectarse al Raspberry Pi a través de la red, es recomendable activar esta opción, ya que con esto no se necesitará utilizar ni un monitor, ni teclado, ni mouse adicionales para poder controlar el dispositivo.

Y la última opción A5 – Actualizar (update) se utiliza para descargar una actualización del sistema, si ya estamos conectados a la red se puede ejecutar inmediatamente. Si hay nuevas versiones de las librerías o programas se descargarán e instalarán las últimas versiones.

Con estas configuraciones ya tenemos la raspberry pi preparada para trabajar.

8.2.2 Integración de sensores en Raspberry pi

Sólo dos funcionalidades necesitan la integración de sensores en la Raspberry pi.

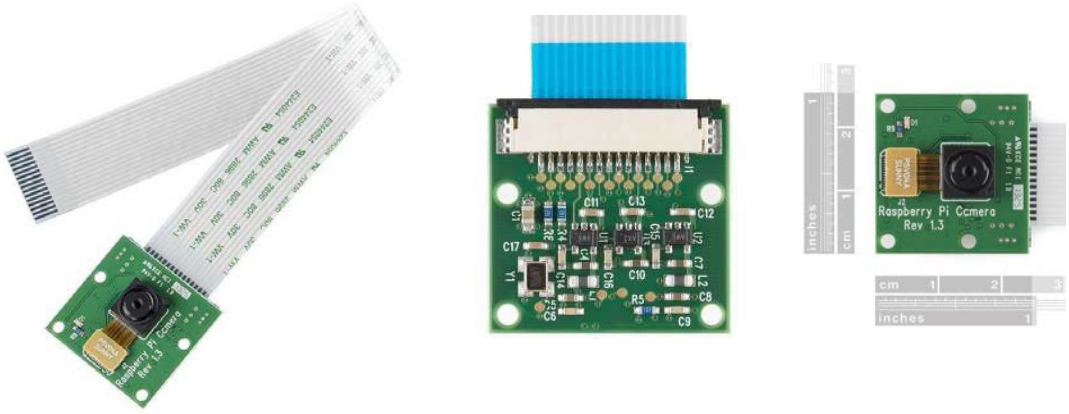
8.2.2.1 Sistema de video vigilancia con detección de movimiento, aviso con mensaje y fotos a través de TELEGRAM.

Para esta función necesitamos la cámara de la Raspberry pi, esta cámara de 5 MP es capaz de capturar vídeo a 1080p (1920x1080) y también imágenes una vez conectada a una placa Raspberry Pi. Para utilizarla, simplemente necesita ser conectada directamente con su cable ribbon al conector SCI (Camera Srial Interfaz) de la Raspberry Pi, y cargar la última versión de Raspbian . Luego una vez reiniciada la Pi, tenemos que activar el soporte de vídeo con raspi-config que ya realizamos en la configuración inicial del Sistema Operativo.

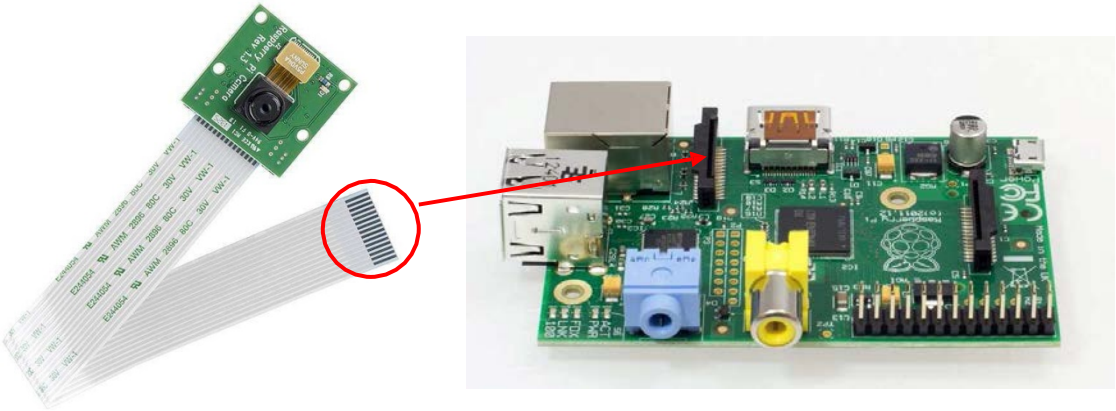
La placa es muy pequeña, unos 25x20x9 mm y pesa apenas 3 gramos por lo que es perfecta para nuestra aplicación o cualquier otra aplicación donde el tamaño sea importante.

En cuanto a imágenes, el sensor es capaz de capturar hasta 2592x1944. Sin embargo en modo vídeo, las resoluciones soportadas son: 1080p30, 720p60 y 640x480p60/90

La cámara es capaz de capturar imágenes y vídeo pero sin sonido, ya que no tiene micro incorporado.



La conexión a realizar será la siguiente:



Una vez conectado el cable al conector SCI quedará así:

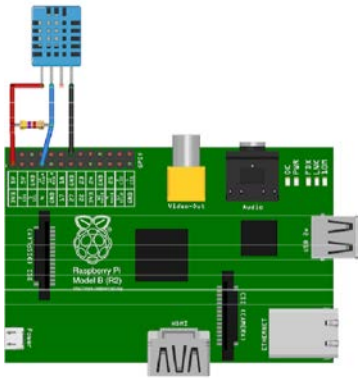
8.2.2.2 Monitorización de la temperatura y humedad del domicilio.

Para esta aplicación necesitaremos el sensor DHT11, una resistencia de 10K, cables y una protoboard para hacer las conexiones con los pines GPIO de la Raspberry pi.
DHT11 es un sensor con un microcontrolador de 8 bits y 2 sensores resistivos encapsulados en una pequeña caja de plástico azul. Existen varios modelos del sensor DHT11 en el mercado, algunos vienen soldados a una placa y otros vienen sueltos para usarse directamente conectados a una breadboard (placa de pruebas). Lo bueno de los modelos que ya vienen soldados a una placa es que nos ahorra tener que montar la resistencia que actúa como pull-up.

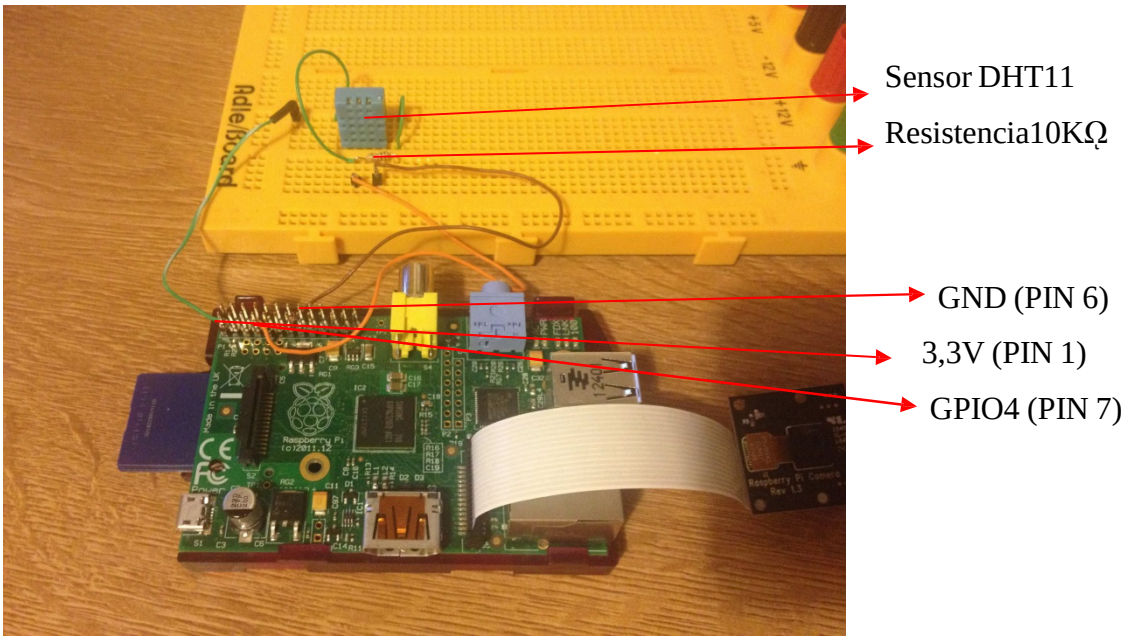
El protocolo de comunicación se realiza a través de un único hilo (Single-Bus) para que la integración en vuestros proyectos sea fácil y rápida pudiéndolo conectar directamente a los pines GPIO de vuestra Raspberry Pi.
La tensión de alimentación del sensor es de 3~5V, se puede conectar a los pines +5V o +3’3V y GND de la Raspberry Pi o alimentarlo desde una batería externa. Su rango de funcionamiento es de 0 a 50ºC para la temperatura y de 20% a 90% de humedad relativa. El pin de comunicaciones para realizar lecturas se puede conectar a cualquier pin GPIO y tiene la capacidad de transmitir la señal hasta 20 metros de distancia.

El sensor DHT11 que se vende suelto no tiene especificados los pines así que hay que fijarse , el sensor esta boca arriba por la parte que tiene agujeros y las patillas están hacia abajo. El pin que esta situado más a la izquierda es el pin VCC (3’3V o 5V). El segundo pin es el que usaremos para las lecturas, en el esquema esta conectado al pin GPIO nº4 de la Raspberry Pi. El tercer pin no se usará . El cuarto pin es GND, lo tenemos que conectar en cualquier pin GND de la Raspberry Pi o de la batería.

Por último hay que conectar una resistencia de 10 KOhm del pin VCC al segundo pin (el que usamos para realizar lecturas), esto actuará como “Pull-up”.



La integración en nuestra Raspberry pi quedaría sí:



8.3 Configuración de sensores

8.3.1 Aviso a través de la aplicación de mensajería Whatsapp de posible fallo de suministro eléctrico en casa o posible fallo de red.

Descripción: esta funcionalidad consiste en un programa hecho en shell-script, el cual mandará una petición HTTP a un servidor web instalado en la casa de un familiar. El servidor web al recibir la petición web nos responderá con un ACK , esta respuesta la trataremos en el script, de tal manera que si no recibimos un ACK, posiblemente el servidor web esté caído, mandando un mensaje vía Whataspp al familiar. De la misma manera, si el servidor web no recibe la petición HTTP que le enviamos, la raspberry Pi situada en casa del familiar nos enviará un mensaje vía Whatsapp de posible fallo de suministro eléctrico o fallo en la red. Configuramos un cron para ejecutar el script cada 5 minutos, ese intervalo será el de comprobación de un posible fallo.

Pasos:

1 – Instalar y configurar Whatsapp en la Raspberry pi

2- Instalar curl para el envío de peticiones HTTP al servidor.

3- Realización del script

4 – Configuración del servidor WEB para recibir las peticiones y responder a ellas

5 - Configuración del cron

1 - Instalar y configurar Whatsapp en la Raspberry pi

Yowsup es una libreria que permite interactuar con los servidores de WhatsApp para enviar y recibir mensajes con nuestra **Raspberry Pi.**

El primer paso será actualizar el sistema de la Raspberry Pi con estos comandos:

sudo apt-get update
sudo apt-get upgrade

instalamos **Python Dateutil**

sudo apt-get install python-dateutil

Ahora clonamos la librería Yowsup desde GitHub a nuestra Raspberry pi

git clone git://github.com/tgalal/yowsup.git

El uso de WhatsApp requiere de un número de teléfono, se puede usar cualquier número de móvil. En mi caso utilizaré el número del móvil del trabajo para la Raspberry pi , hay que validarlo con un mensaje o una llamada.

cd /home/pi/yowsup/src

Ahora editamos el fichero **config.example** para colocar nuestro número.

nano config.example
cc= 34
phone= 346********
id=
password=

Para registrar nuestro número tendremos que ejecutar en la terminal

python yowsup-cli -c config.example --requestcode sms

Tendremos que esperar a que nos llegue un código XXX-XXX. Una vez llegue, procedemos a solicitar nuestra password:

python yowsup-cli -c config.example --register XXX-XXX

La consola nos devolverá algunos valores entre los cuales está **PW**. Apuntamos este valor y lo copiamos al archivo "**config.example**" en el apartado password. Quedando de la siguiente manera:

cc=34
phone=346XXXXXXXX
id=
password=FduF9VRpXXXXXXXXPXhvshDLHA=

Para mandar un mensaje:

python yowsup-cli -c config.example -w -s 34numero "Mensaje"

2- Instalar curl para el envío de peticiones HTTP al servidor

Curl es una herramienta para usar en un intérprete de comandos para transferir archivos con sintaxis URL, soporta FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP. cURL soporta certificados HTTPS, HTTP POST, HTTP PUT, subidas FTP, Kerberos, subidas mediante formulario HTTP, proxies, cookies, autenticación mediante usuario+contraseña (Basic, DIgest, NTLM y Negotiate para HTTP y kerberos4 para FTP), continuación de transferencia de archivos, tunneling de proxy http y muchas otras prestaciones. cURL es open source/software libre distribuido bajo la Licencia MIT.

root@raspberrypi:~# apt-get install curl

3- Realización del script

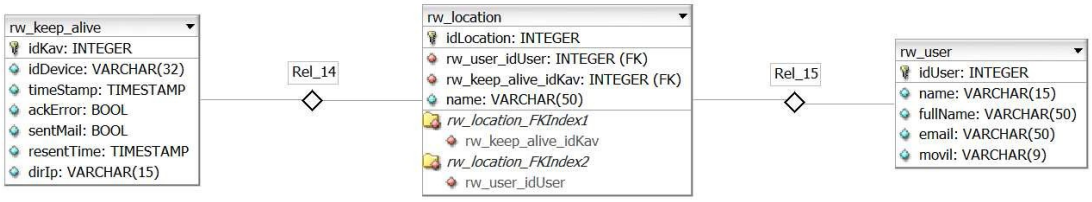
```
pi@raspberrypi: ~/yowsup
  GNU nano 2.2.6                                File: estoyvivo.sh

#!/bin/bash

curl http://www.roncero.es/server/promos/setKAV.php?KAV=f775641eb0804de2864fe04257b0e911>salida.txt
cut -c2-5 salida.txt >salida2.txt
H=$(cat salida2.txt)
echo $H
if [[ $H = "ACK" || $H = "NACK" ]];
then
#python yowsup-cli -c config.example -w -s 346XXXXXXXX "Recibido ACK, Servidor respondiendo"
exit 0
else
python yowsup-cli -c config.example -w -s 346XXXXXXXX "Posible caida de servior roncero.es"
fi
```

Mandamos la petición al servidor mediante curl y lo que me devuelva el servidor lo meto en el fichero salida. Le quito la # de delante, y compruebo el valor que he recibido. Si es ACK o NACK, el servidor remoto está respondiendo, si no recibo esos valores, mando un mensaje de whataspp al móvil del familiar indicándole posible caída de su servidor.

4. Configuración del servidor WEB para recibir las peticiones y responder a ellas

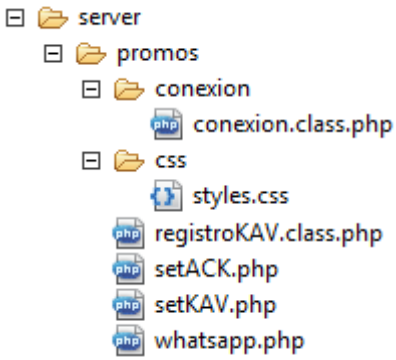
Este sistema registra una trama KeepAlive (KAV) de la raspberry , a fin de detectar una posible caída de tensión en un en el domicilio donde está situada.
En la Raspberry se programa el envío de una trama KAV al servidor, cada 5 minutos.
El sistema comprueba si han pasado más de 6 minutos desde el envío de alguna trama desde la raspberry, y en caso afirmativo, envía la notificación correspondiente y activa el flag de notificado (sentMail).

MODELO DE DATOS
El siguiente esquema muestra el modelo de datos implementado en mySql.



ARCHIVOS PHP

Este sistema está implementado en PHP siguiendo la siguiente estructura de archivos:



conexión.class.php

Clase para la conexión a la base de datos. Se parametriza con el host del servidor mysql, el nombre de la base de datos y el usuario y contraseña de acceso.

registroKAV.class.php

Clase con los procedimientos de inserción, actualización y consulta de la base de datos.

setACK.php (?KAV=< hash >)

Página web para la confirmación por parte del usuario de la recepción de notificación de incidencia.
Recibe un parámetro GET con el identificador de la raspberry sobre la que se hace la confirmación.

setKAV.php (?KAV=< hash >)

Servicio web para el envío de la trama KeepAlive (KAV) por parte del sistema cliente.
Recibe un parámetro GET con el identificador del propio sistema.

whatsapp.php (?num=<destinatario>&msg=<texto>)

Servicio web para el envío de un mensaje mediante Whatsapp.
Recibe un parámetro GET con el número de móvil del destinatario y otro parámetro GET con el texto del mensaje a enviar.

NOTIFICACIONES

Cuando la raspberry realiza la notificación KAV al servidor, se produce el envío de la notificación correspondiente a la raspberry cuya última notificación KAV al servidor, fue realizada hace más de 6 minutos.

En caso de que ya se haya notificado y el cliente no hubiera mandado una confirmación al respecto, el sistema volvería a enviar la notificación cada 5 minutos.

Las notificaciones se realizan por Whatsapp y en caso de fallo en el envío, se enviará la notificación por correo electrónico haciendo uso de la función mail de PHP.

CONFIRMACIONES

Cuando el cliente recibe una notificación, en el cuerpo del mensaje dispondrá de un hipervínculo a la página web setACK.php, para realizar una confirmación de la recepción del mensaje.

5 – Configuración del cron

Añadimos con el usuario pi, la siguiente línea al crontab para que se ejecute el script estoyvivo.sh cada 5 minutos.

crontab –e para editar el cron y añadir la línea

*/5 * * * * python /home/pi/yowsup/src/estoyvivo.sh

crontab –l confirmamos que lo ha guardado

Reiniciamos el servicio con los nuevos los cambios

service cron restart

8.3.2 Sistema de video vigilancia con detección de movimiento, aviso con mensaje y fotos a través de TELEGRAM.

Descripción: este sistema a través del programa motion, nos detectará movimiento, una vez detectado el movimiento la cámara realizará una serie de capturas, configurando el evento **on_motion_detected** haremos que ejecute un programa en python que nos enviará un mensaje junto con las fotos realizadas a través de Telegram y que recibiremos en nuestro móvil avisándonos de la posible intrusión en el domicilio.

Pasos:

1- Instalación y configuración del programa motion
2- Instalación y configuración de telegram
3- Programa en python para el envío de mensaje y fotos al móvil.

1. Instalación y configuración del programa motion

Instalamos motion:

sudo apt-get install motion

La versión actual de motion no soporta el módulo de la cámara de la raspberry pi, tenemos que descargar e instalar una versión especial con soporte para este módulo de la cámara.

cd /tmp

sudo apt-get install -y libjpeg62 libjpeg62-dev libavformat53 libavformat-dev libavcodec53 libavcodec-dev libavutil51 libavutil-dev libc6-dev zlib1g-dev libmysqlclient18 libmysqlclient-dev libpq5 libpq-dev

wget https://www.dropbox.com/s/xdfcxm5hu71s97d/motion-mmal.tar.gz

Descomprimimos el fichero

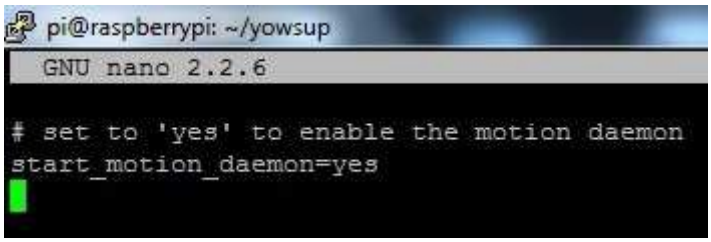
tar zxvf motion-mmal.tar.gz

Después de descomprimir , necesitamos actualizar el motion instalado con el motion con soporte para la cámara.

sudo mv motion /usr/bin/motion

sudo mv motion-mmalcam.conf /etc/motion.conf

Habilitamos el demonio de motion para que siempre esté corriendo, /etc/default/motion



Damos permisos a los siguientes ficheros:

sudo chmod 664 /etc/motion.conf

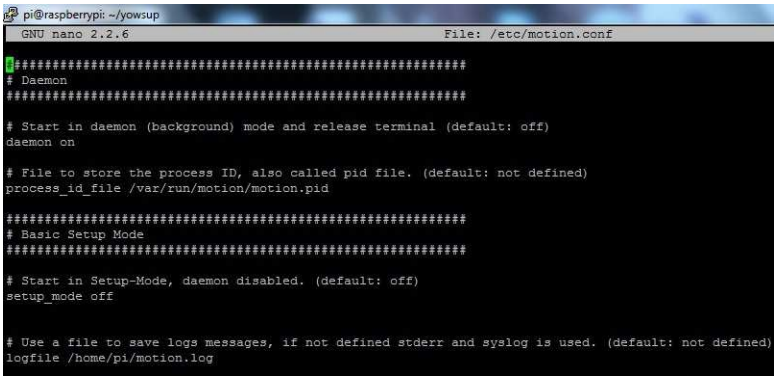
sudo chmod 755 /usr/bin/motion

sudo touch /tmp/motion.log

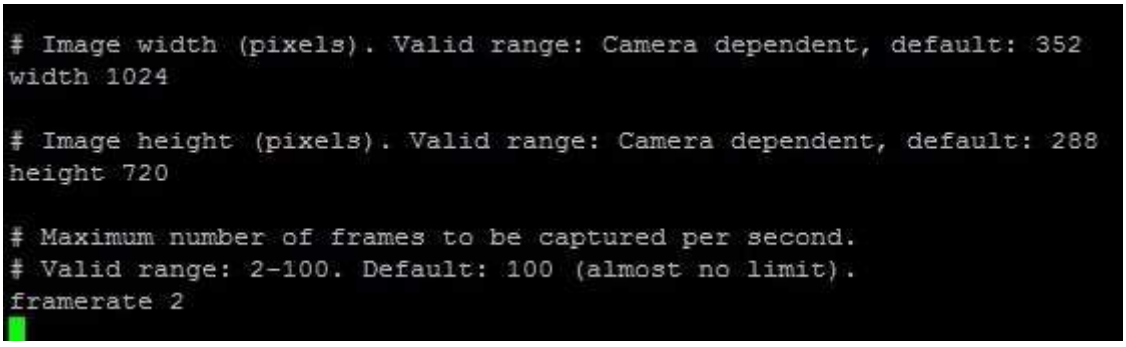
sudo chmod 775 /tmp/motion.log

Abrimos el fichero de configuración y establecemos los siguientes parámetros:

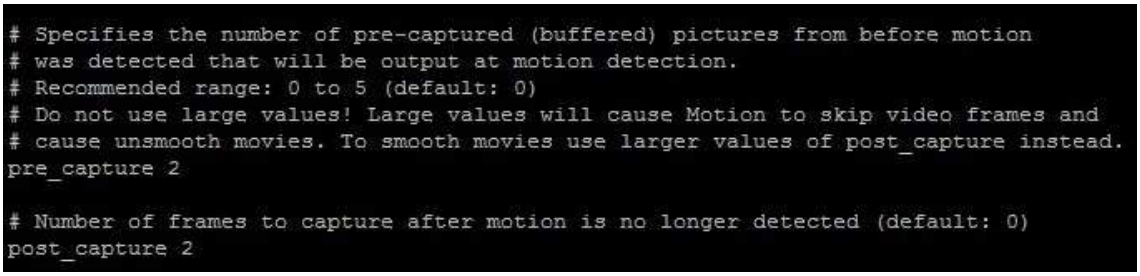
Nos aseguramos que motion está siempre corriendo como un demonio en background y establecemos la ruta del logfile in /home/pi/motion.log



Usamos high quality surveillance video



Capturamos 2 frames antes y después de la detección de movimiento:



Cambiamos el codec a msmpeg4:

```
# Codec to used by ffmpeg for the video compression.
# Timelapse mpegs are always made in mpeg1 format independent from this option.
# Supported formats are: mpeg1 (ffmpeg-0.4.8 only), mpeg4 (default), and msmpeg4.
# mpeg1 - gives you files with extension .mpg
# mpeg4 or msmpeg4 - gives you files with extension .avi
# msmpeg4 is recommended for use with Windows Media Player because
# it requires no installation of codec on the Windows client.
# swf - gives you a flash film with extension .swf
# flv - gives you a flash video with extension .flv
# ffv1 - FF video codec 1 for Lossless Encoding ( experimental )
# mov - QuickTime ( testing )
# ogg - Ogg/Theora ( testing )
ffmpeg_video_codec mpeg4
```

Habilitamos acceso a the live stream desde cualquier lugar

```
# Restrict stream connections to localhost only (default: on)
stream_localhost off
```

Configuramos el path donde almacenar las fotos capturadas:

```
# Target base directory for pictures and films
# Recommended to use absolute path. (Default: current working directory)
target_dir /home/pi/motion
```

Configuramos que cuando un movimiento sea detectado ejecute el programa en python para el envío de fotos.

```
# Command to be executed when a motion frame is detected (default: none)
on_motion_detected /home/pi/tg/enviofotos.py
```

2. Instalación y configuración de telegram

git clone https://github.com/vysheng/tg.git && cd tg

Después de haber clonado el repositorio de GitHub instalamos lo siguiente:

sudo apt-get install libreadline-dev libconfig-dev libssl-dev lua5.2 liblua5.2-dev

Ejecutamos el archivo de configuración y compilamos el programa

./configure

make

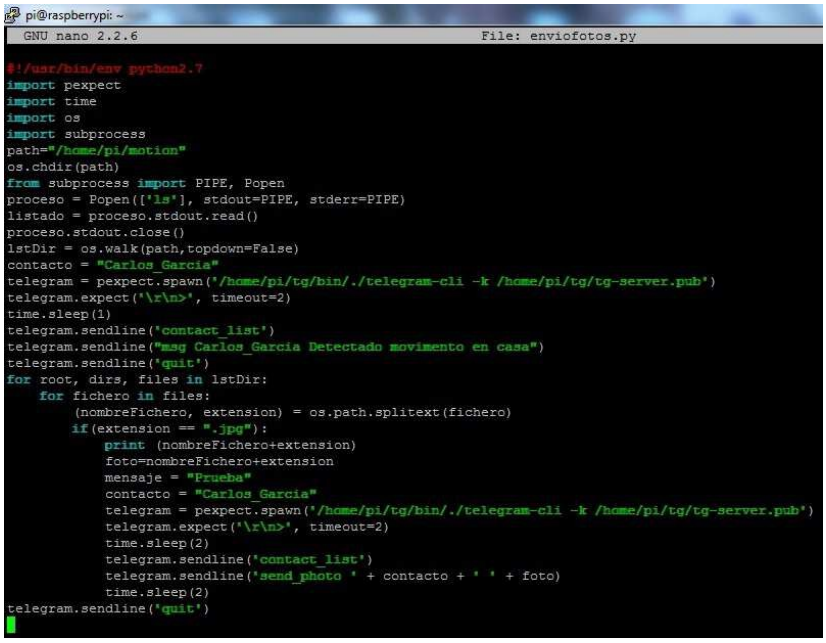
Ejecutamos el programa, pasándole la clave pública

./telegram-cli -k tg-server.pub

La primera vez la aplicación nos pide nuestro numero telefónico incluyendo el signo + y el área del país. Después si aún no estamos registrados nos preguntará si queremos registrarnos, pedirá nuestro nombre y apellido. Posteriormente enviará un código de verificación a nuestro móvil.

Añadimos los contactos

3. Programa en python para el envío de mensaje y fotos al móvil.



Importamos las librerías necesarias, nos situamos en la ruta donde están las fotos, recorremos el directorio buscando los archivos.jpg, si existen, la variable foto toma el valor del nombre del archivo, ponemos la variable contacto con el nombre del contacto al que vamos a enviar las fotos, que previamente fue añadido en telegram, y ejecutamos telegram para enviar las fotos, tantas como haya en el directorio, cuando salga del bucle for salimos de telegram.

8.3.3 Monitorización de temperatura interna de la Raspberry pi

Descripción: debido a que la raspberry pi va a estar conectada 24H/365 es importante monitorizar la temperatura, para ello realizaremos un script que leerá la temperatura de la raspberry pi y la mandará a la plataforma Carriots.

La explicación de la integración en la plataforma Carriots, será en el apartado 6, tener en cuenta que valores como API CARRIOTS y DEVICE_ID que veremos en el script, son obtenidos de la integración de la Raspberry pi en la plataforma.

Pasos:

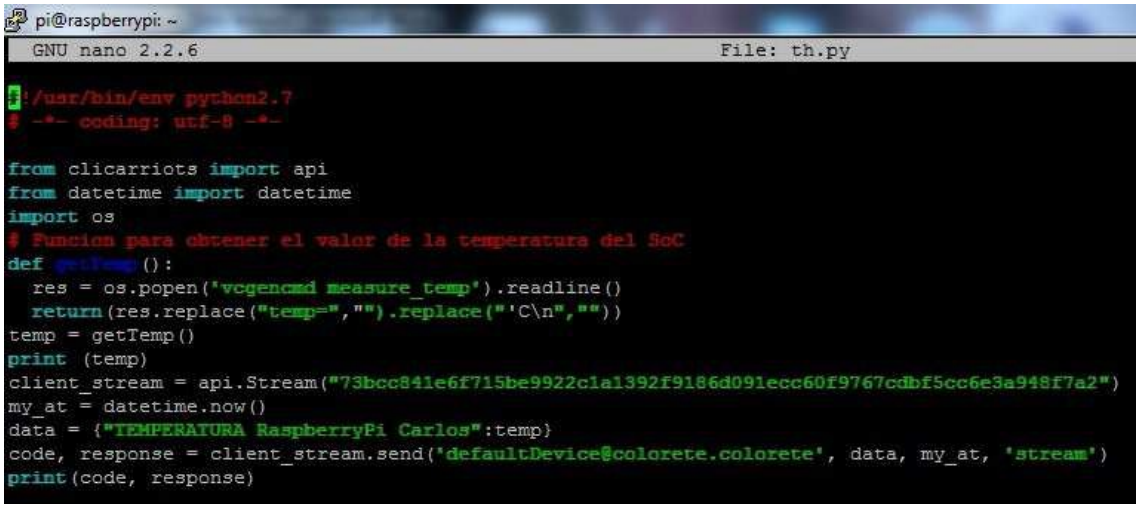
1. Instalación de clicarriots
2. Programa en python para leer y enviar la Tª a Carriots
3. Configuración de cron

1 . Instalación de clicarriots

Instalación de la librería para interactuar con la API de Carriots.

sudo pip install git+https://github.com/sdeancos/clicarriots.git#egg=clicarriots

2. Programa en python para leer y enviar la Tª a Carriots

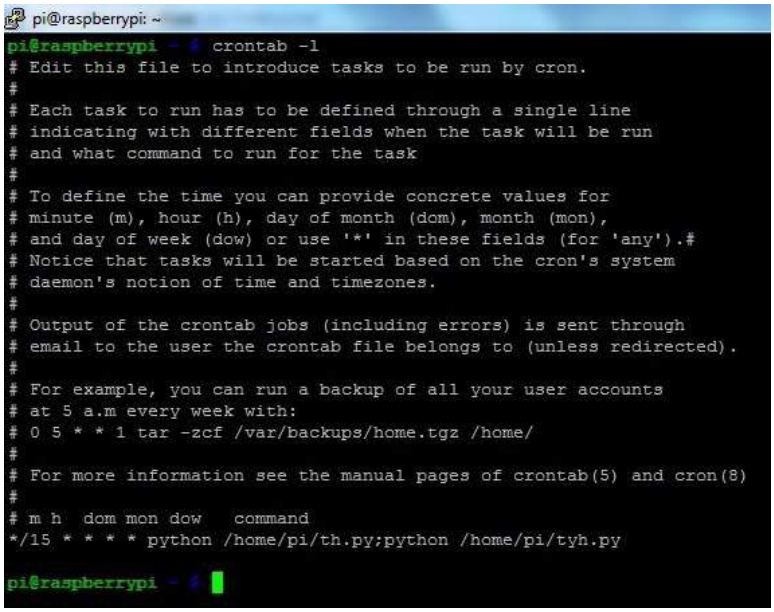


Importamos librerías, definimos la función para obtener la temperatura de la Raspberry pi y la almacenamos en la variable temp, asignamos el valor de nuestra API de Carriots

y definimos los valores del stream que vamos a enviar, como son la fecha y hora , el dato leído y el device asociado en Carriots.

3 . Configuración de cron

Configuramos el cron para que el programa se ejecute cada 15 minutos.



8.3.4 Monitorización de la temperatura y humedad del domicilio, con subida de datos a plataforma Carriots.

Descripción: con la integración del sensor DHT11, podemos leer los valores de temperatura y humedad, a través de un progrma en python leeremos esos valores y los mandaremos a la plataforma Carriots. Tendremos un cron para ejecutar el programa cada 30 minutos.

Pasos:

1 . Programa en pyhton para leer los datos del sensor y subirlos a la plataforma Carriots.

2 . Configuración de cron

1 . Programa en pyhton para leer los datos del sensor y subirlos a la plataforma Carriots

```
pi@raspberrypi: ~
  GNU nano 2.2.6                                        File: tyh.py

#!/usr/bin/env python2.7
# -*- coding: utf-8 -*-
import RPi.GPIO as GPIO
import time
from clicarriots import api
from datetime import datetime
import os
def binadec(string_num): #Función para transformar de binario a decimal
 return str(int(string_num, 2))
data = [] #Definimos data como un array
GPIO.setmode(GPIO.BCM) #Ponemos la placa en modo BCM
GPIO.setup(4,GPIO.OUT) #Configuramos el pin 4 como salida
GPIO.output(4,GPIO.HIGH) #Enviamos una señal
time.sleep(0.025) #Pequeña pausa
GPIO.output(4,GPIO.LOW) #Cerramos la señal
time.sleep(0.02) #Pausa
GPIO.setup(4, GPIO.IN, pull_up_down=GPIO.PUD_UP) #Ponemos el pin 4 en modo lectura
for i in range(0,500): #Lee los bits que conforman la respuesta binaria del sensor
 data.append(GPIO.input(4))
#Definimos variables para cálculo
bit_count = 0
tmp = 0
count = 0
HumedadBit = ""
TemperaturaBit = ""
crc = ""
try:
#El siguiente código lee los bits de respuesta que envia el "
#sensor y los transforma a un número decima.
        while data[count] == 1:
                tmp= 1
                count = count + 1
        for i in range(0, 32):
                bit_count = 0
                while data[count] == 0:
                        tmp = 1
                        count = count + 1
                while data[count] == 1:
                        bit_count = bit_count + 1
```

En esta primera parte se importan las librerías y se configura el pin4 GPIO de la RaSPBERRY PI como salida y se pone en modo lectura, a través de ese pin recibiremos los datos del sensor.

```
pi@raspberrypi: ~
  GNU nano 2.2.6                                File: tyh.py

                if bit_count > 3:
                        if i>=0 and i<8:
                                HumedadBit = HumedadBit + "1"
                        if i>=16 and i<24:
                                TemperaturaBit = TemperaturaBit + "1"
                else:
                        if i>=0 and i<8:
                                HumedadBit = HumedadBit + "0"
                        if i>=16 and i<24:
                                TemperaturaBit = TemperaturaBit + "0"
except:
        print "ERR_RANGE"
        exit(0)
try:
        for i in range(0, 8):
                bit_count = 0
                while data[count] == 0:
                        tmp = 1
                        count = count + 1
                while data[count] == 1:
                        bit_count = bit_count + 1
                        count = count + 1
                if bit_count > 3:
                        crc = crc + "1"
                else:
                        crc = crc + "0"
except: #Errores en el bloque
        print "ERR_RANGE"
        exit(0) #Salimos
Humedad = bin2dec(HumidityBit)
Temperatura = bin2dec(TemperatureBit)
#Mandamos a plataforma Carriots
client_stream = api.Stream("73bcc841e6f715be9922c1a1392f9186d091ecc60f9767cdbf5cc6e3a948f7a2")
my_at = datetime.now()
data = {"TEMPERATURA Casa":Temperatura, "HUMEDAD Casa":Humedad}
code, response = client_stream.send('defaultDevice@colorete.colorete', data, my_at, 'stream')
print(code, response)
```

En esta segunda parte se gestiona los bits recibidos por el sensor tanto para temperatura como para humedad, almacenándose en las variables HumedadBit y TemperaturaBit.

Estos datos están en binario así que los transformaremos a decimal con la función bin2dec. Por último mandamos los datos a la plataforma Carriots.

2. Configuración del cron

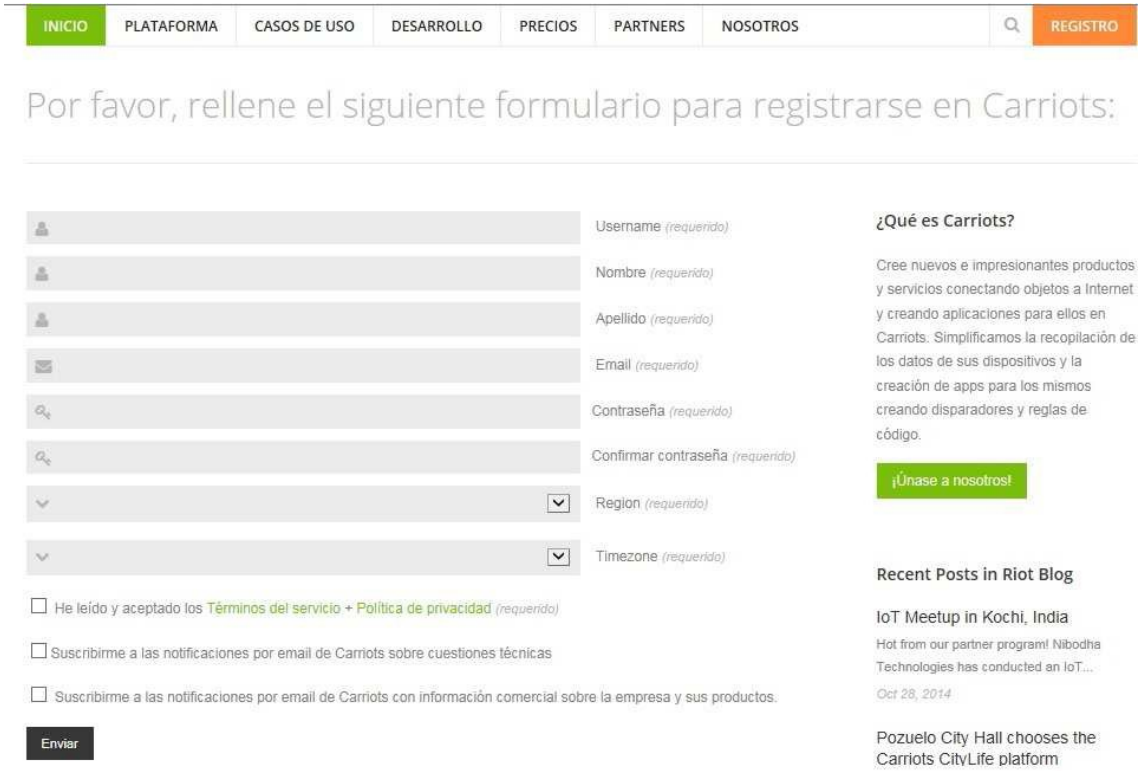
```
pi@raspberrypi: ~
pi@raspberrypi ~ $ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/15 * * * * python /home/pi/th.py;python /home/pi/tyh.py

pi@raspberrypi ~ $
```

8.4 Implementación en plataforma Carriots

8.4.1 Registro y configuración

Lo primero de todo será registrarnos en Carriots (https://www.carriots.com/joinus):

Nos aparecerá la siguiente pantalla, en la cual rellenamos los datos que nos piden:



Para la confirmación del registro, nos mandarán un email al correo y tendremos que hacer click en el enlace que nos mandan.

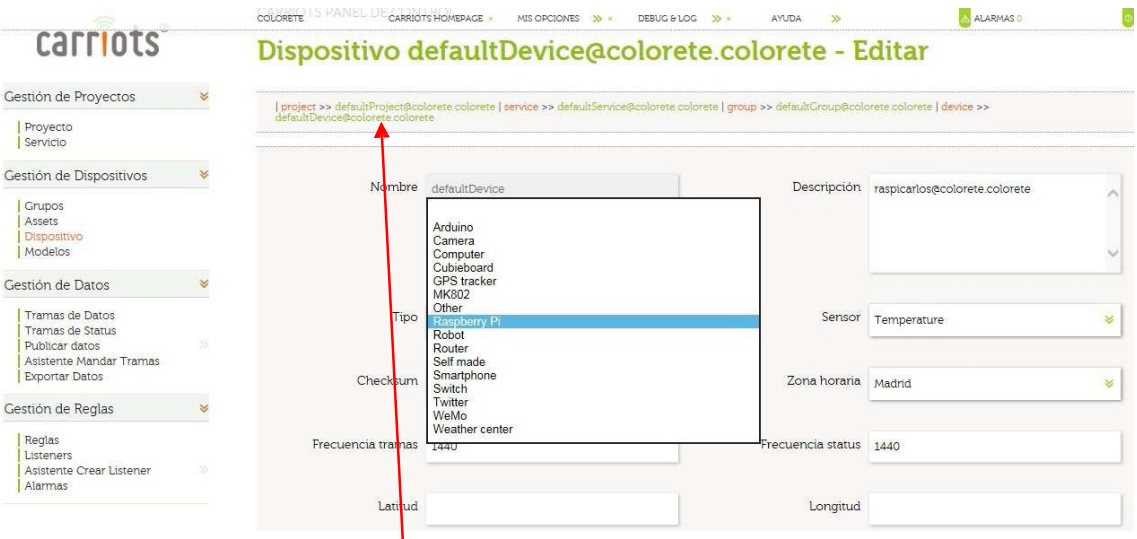
Una vez confirmado el registro hacemos clic en login para entrar en nuestra cuenta.

Accederemos al panel de control de carriots, desde el cual podremos gestionar los dispositivos asociados.

En el menú mis opciones, hacemos click en el submenú MI CUENTA.

En MI CUENTA actualizamos nuestros datos, zona horaria y podemos ver cual van a ser nuestras API Keys, importantísimos para mandar los datos desde la Raspberry pi a Carriots.

A continuación vamos a configurar nuestro dispositivo, por defecto viene uno, usaremos el defaultDevice y simplemente debemos decirle con qué tipo de sensor vamos a trabajar y qué dispositivo es, en nuestro caso la Raspberry PI.

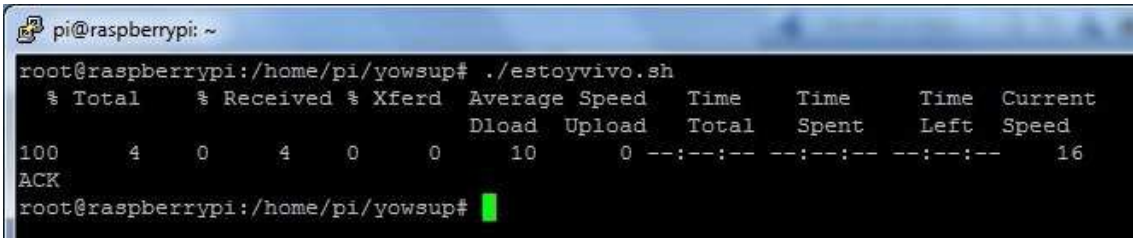


Otro de los parámetros que hemos usado en nuestros programas es el DEVICE_ID, en nuestro caso : defaultDevice@colorete.colorete

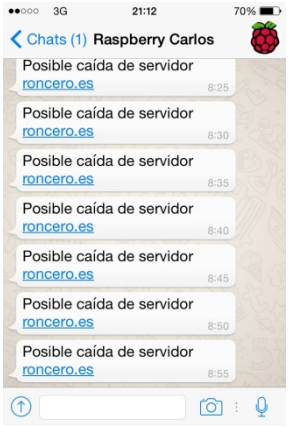
9 Pruebas y resultados

9.1 Pruebas Aviso a través de la aplicación de mensajería Whataspp de posible fallo de suministro eléctrico en casa o posible fallo de red.

Ejecutamos manualmente el script estoyvivo.sh, que envía la petición HTTP al servidor web.

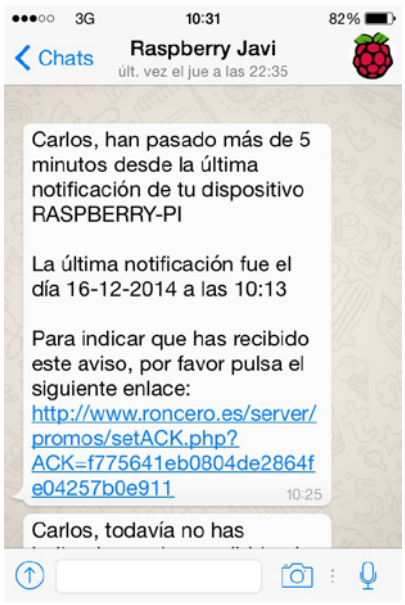


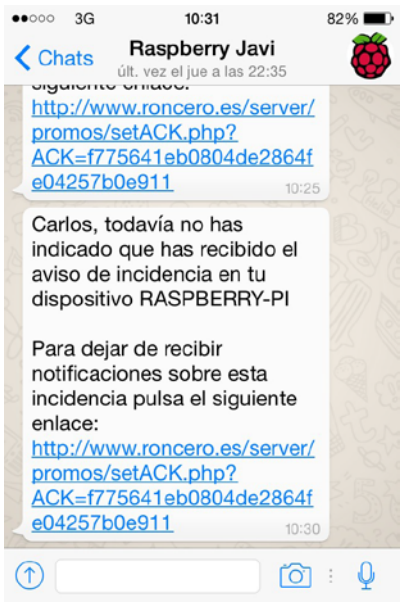
Si todo va bien, recibiremos un ACK y no mandaremos ningún mensaje , en caso de no recibir un ACK, el programa enviará un whataspp indicando una posible caída del servidor.



En el caso de que nuestra raspberry pi no envíe la petición HTTP al servidor web, éste al no recibir pasados 6 minutos desde la última notificación nos mandará un whataspp a través de la raspberry remota a nuestro móvil indicando un posible fallo de suministro eléctrico (no recibida notificación desde la raspberry pi) .

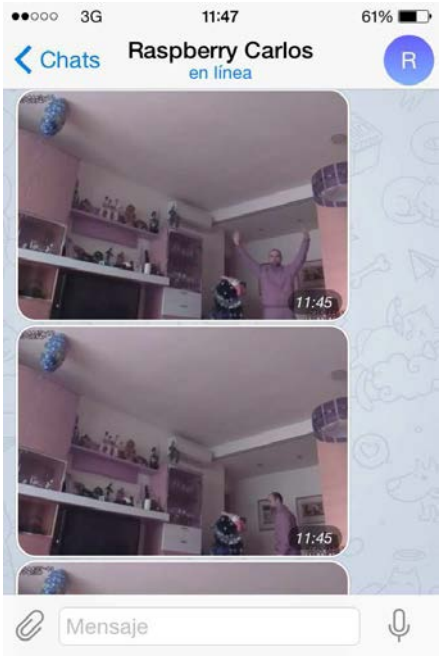
Primero recibimos el mensaje de que han pasado más de 5 minutos desde la última notificación, si nos pulsamos el link para indicar que hemos recibido el mensaje, cada 5 minutos nos estará mandando mensajes, como el siguiente, “ Carlos todavía no has indicado…”, para dejar de recibir notificaciones pinchamos en el link y nos confirma que no se volverán a enviar más avisos.



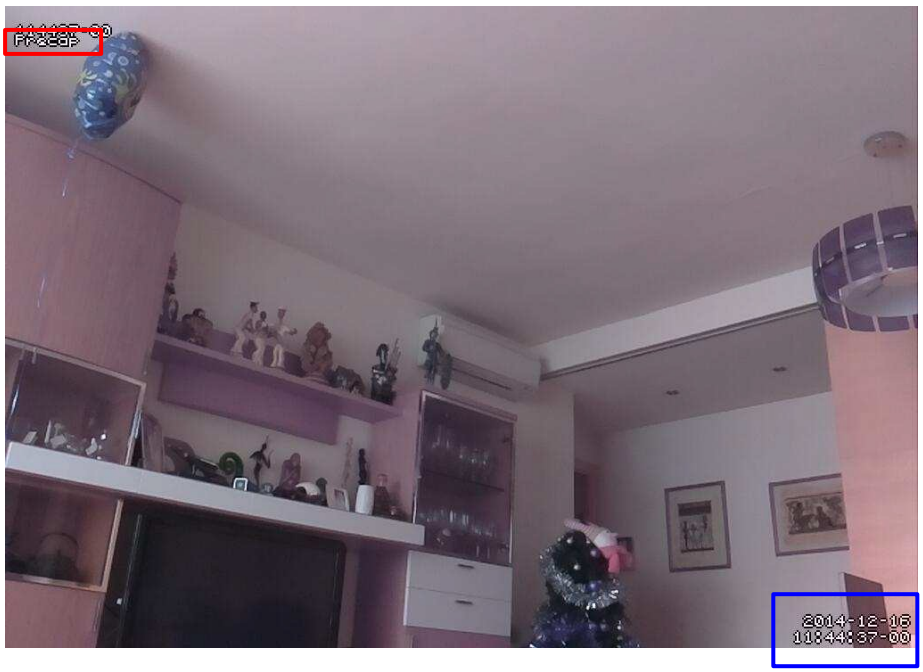


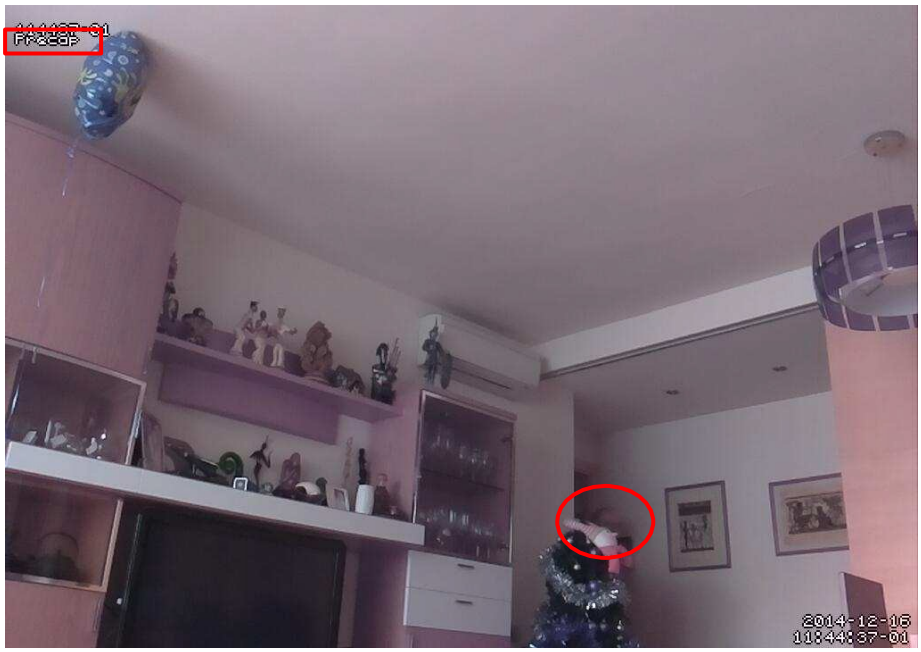
9.2 Sistema de video vigilancia con detección de movimiento, aviso con mensaje y fotos a través de TELEGRAM.

Provocamos una detección y comprobamos como recibimos en nuestro móvil el mensaje y las fotos capturadas.



Si extraemos las fotos podemos observar valores que definimos en /etc/motion.conf, como son las 2 fotos pre-capture y las 2 post-capture. Así como las trigger, que son las capturas del movimiento .También podemos ver la sincronización entre las capturas y el envío al móvil. En cuanto a la sincronización podemos ver que es real, se demora unos segundos más, debido a que en el programa de python tuve que hacer un sleep entre envío y envío de foto para que de tiempo a enviarse la foto antes de iniciar el envío de la siguiente, sin el sleep no se enviaban todas las fotos.


2014-12-16
11:44:37-00


2014-12-16
11:44:37-01

Los trigger:






9.3 Envío a plataforma Carriots

En este apartado comprobamos en la plataforma Carriots, el registro de los streams con los datos tanto de temperatura de la Raspberry Pi como de temperatura y humedad de la casa.

Accedemos a la web con nuestro usuario, y vamos al panel de control. Una vez allí seleccionamos en el menú lateral: **Gestión de Datos / Trama de Datos.**

En la siguiente imagen aparecen las tramas enviadas, **temperatura de la raspberry pi** y los **datos leídos por el sensor dht11**

Las siguientes figuran muestra las tramas enviadas:

Tramas:


```
{
  "_t": "str",
  "_m": {
    "me": "POST",
    "pr": "http",
    "ve": "HTTP\/1.0",
    "ur": "\/streams",
    "ho": "api.carriots.com",
    "he": {
      "Content-Length": "137",
      "Accept-Encoding": "identity",
      "X-Scheme": "https",
      "Host": "api.carriots.com",
      "Accept": "application\/vnd.carriots.api.v2+json",
      "User-Agent": "clicarriots",
      "Connection": "close",
      "X-Real-Ip": "213.37.234.111",
      "Content-Type": "application\/vnd.carriots.api.v2+json"
    }
  },
  "at": 1418762703,
  "device": "defaultDevice@colorete.colorete",
  "protocol": "v2",
  "data": {
    "HUMEDAD CASA": 46,
    "TEMPERATURA CASA": 20
  },
  "id_developer": "9b883777f947a0d48b07091a65030475886822b9772c5cda7f41db0b15b340a1@colorete.colorete",
  "created_at": 1418762703,
  "owner": "colorete"
}
```

Tramas:


```
{
  "_t": "str",
  "_m": {
    "me": "POST",
    "pr": "http",
    "ve": "HTTP\/1.0",
    "ur": "\/streams",
    "ho": "api.carriots.com",
    "he": {
      "Content-Length": "133",
      "Accept-Encoding": "identity",
      "X-Scheme": "https",
      "Host": "api.carriots.com",
      "Accept": "application\/vnd.carriots.api.v2+json",
      "User-Agent": "clicarriots",
      "Connection": "close",
      "X-Real-Ip": "213.37.234.111",
      "Content-Type": "application\/vnd.carriots.api.v2+json"
    }
  },
  "at": 1418762702,
  "device": "defaultDevice@colorete.colorete",
  "protocol": "v2",
  "data": {
    "TEMPERATURA RaspberryPi Carlos": "42.2"
  },
  "id_developer": "0f9dd338ca6dc8c1d340687057ca23fa60d8b5e7bb4a889639721e546b66d99e@colorete.colorete",
  "created_at": 1418762702,
  "owner": "colorete"
}
```


10 Presupuesto del proyecto

Cantidad	Concepto	Precio unidad
1	Tarjeta SD 8Gb	5.39 €
1	Raspberry Pi B	34.19 €
1	Cámara Raspberry PI	28 €
1	Sensor DHT11	1.18 €
1	Cables	1€
1	Resistencia 10k Ohmios	0.04€
1	Protoboard	2.04€
	TOTAL	**71.84 €**

Aunque nos pueda parecer a simple vista un precio excesivo, en realidad las funcionalidades extras que podemos tener ya que es un mini PC son muchas.

11 Conclusiones

En la actualidad el IoT ha tomado una presencia importantísima en el mundo que nos rodea, apenas sin darnos cuenta observamos cada vez más objetos conectados a la red, más sensores que nos proporcionan datos bien meramente informativos (temperatura, humedad) o por el contrario datos importantes como una alarma de intrusión, o como vimos en el capítulo 2 , nos avisa de bajo nivel de azúcar en nuestro organismo, etc.

Este proyecto no sólo nos ha presentado el concepto de IoT, sino que hemos podido analizar plataformas web donde subir esos datos recogidos por los sensores, poder analizarlos para realizar estadísticas, incluso interactuar con esos datos en otros sistemas externos.

También hemos analizado desde un punto de vista "casero" plataformas Hardware bastante económicas que nos permiten realizar cosas muy interesantes.

La Raspberry Pi como pilar de nuestro trabajo y gracias a sus propiedades de bajo consumo, coste y tamaño nos ha proporcionado una primera visión práctica de IoT.

No sólo hemos interactuado con plataformas web subiendo datos de sensores, sino también con aplicaciones de mensajería instantánea, con el fin de poder visualizar los datos o avisos en nuestros teléfonos. Esta última es una gran ventaja de la raspberry pi, el tener un sistema operativo como Linux que puede interactuar con infinidad de aplicaciones externas.

Este trabajo lo considero la versión 1.0, las posibilidades son enormes, incluir más sensores, la integración de la raspberry pi con arduino para poder realizar ya tareas con servo motores serán próximas aplicaciones.

Para finalizar, este trabajo no sólo he disfrutado haciéndolo, he podido ampliar mi concepto sobre IoT, he comprendido el funcionamiento de las API´s de las plataformas Software, y por último me he dado cuenta de la cantidad de posibilidades que ofrece la raspberry pi.

REFERENCIAS

(1) **http://www.rnrmarketresearch.com/the-m2m-iot-wearable-technology-ecosystem-2015-2020-opportunities-challenges-strategies-industry-verticals-and-forecasts-market-report.html**

BIBLIOGRAFÍA

http://www.domodesk.com/a-fondo-que-es-el-internet-de-las-cosas

http://www.iot-spain.com/?lang=es

http://www.libelium.com/es/top_50_iot_sensor_applications_ranking

http://www.strategy-business.com/article/00294?pg=all

http://www.contiki-os.org/

http://www.tinyos.net/

http://www.internetdelascosas.cl/

http://es.container-centralen.com/

https://www.telcare.com/

http://www.raspberrypi.org/

https://github.com/tgalal/yowsup

https://github.com/vysheng/tg

https://github.com/raspberrypi

https://xively.com/

https://www.carriots.com/

https://thingspeak.com/

http://www.intel.es/content/www/es/es/do-it-yourself/edison.html

http://www.intel.es/content/www/es/es/do-it-yourself/galileo-maker-quark-board.html

http://www.arduino.cc/

ANEXO I

Tablas Data Management API Xively

Feeds

Operación	Formato	Punto final	Método
Read	json, xml y csv	/feeds/	GET
Read a range	json, xml y csv	/feeds/ ***feed_id*?*range***	GET
Update	json, xml y csv	/feeds/***feed_id***	PUT
List all	json, xml y csv	/feeds	GET

Datastreams

Operación	Formato	Punto final	Método
Create	json, xml y csv	/feeds/**feed_id**/ datastreams	POST
Read	json, xml ,csv y png	/feeds/**feed_id**/ datastreams/**datastream_id**	GET
Read a range	json, xml, csv y png	/feeds/**feed_id**/ datastreams/**datastream_id**?**range**	GET
Update	json, xml y csv	/feeds/**feed_id**/ datastreams/**datastream_id**	PUT
Delete	json, xml y csv	/feeds/**feed_id**/ datastreams/**datastream_id**	DELETE
List all	json, xml y csv	/feeds/**feed_id**	GET

Datapoints

Operación	Formato	Punto final	Método
Delete (single)	n/a	/feeds/***feed_id***/datastreams/***datastream_id***/datapoints/ ***timestamp***	DELETE
Delete (range)	n/a	/feeds/***feed_id***/datastreams/***datastream_id***/ datapoints?***range***	DELETE

Products

Operación	Formato	Punto final	Método
Create	json	/products	POST
Read	json	/products/ **product_id**	GET

Update	json	/products/ **product_id**	PUT
Delete	n/a	/products/ **product_id**	DELETE
List all	json	/products	GET

Devices

Operación	Formato	Punto final	Método
Create	json	/products/ **product_id**/devices	POST
Read	json	/products/ **product_id**/devices/**serial_number**	GET
Update	json	/products/ **product_id**/devices/**serial_number**	PUT
Delete	n/a	/products/ **product_id**/devices/**serial_number**	DELETE
List all	json	/products/ ***product_id***/devices	GET
Activate	json, csv	/devices/ ***activation_code***/activate	GET

Keys

Operación	Formato	Punto final	Método
Create	json, xml	/keys	POST
Read	json, xml	/keys/***key_id***	GET
Delete	n/a	/keys/***key_id***	DELETE
List all	json, xml	/keys	GET

Triggers

Operación	Formato	Punto final	Método
Create	json, xml	/triggers/	POST
Read	json, xml	/triggers/**trigger_id**	GET
Update	json, xml	/triggers/**trigger_id**	PUT
Delete	n/a	/triggers/**trigger_id**	DELETE
List all	json, xml	/triggers/	GET

ANEXO II

Tablas Data Management API Carriots

Projects

Accion	Formato	Punto final	Método
Create a project	json/xml	/projects	POST
Show a project	json/xml	/projects/[id_project]	GET
List projects (max 1000)	json/xml	/projects	GET
Update project value	json/xml	/projects/[id_project]	PUT
Delete a project	json/xml	/projects/	DELETE
Show services	json/xml	/projects/[id_project]/services/[id_service]/	GET
List services (max 1000)	json/xml	/projects/[id_project]/services/	GET

Services

Accion	Formato	Punto final	Método
Create a service	json/xml	/services/	POST
Show service data	json/xml	/services/[id_service]	GET
List services (max 1000)	json/xml	/services/	GET
Update a service	json/xml	/services/[id_service]	PUT
Delete a service	json/xml	/services/[id_service]	DELETE
Show a group from a service	json/xml	/services/[id_service]/groups/[id_group]/	GET
List group	json/xml	/services/[id_service]/groups/	GET

Groups

Accion	Formato	Punto final	Método
Create a group	json/xml	/groups/	POST
Show a group	json/xml	/groups/-- ID_developer --/	GET
List a group (max 1000)	json/xml	/groups/	GET
Update a group	json/xml	/groups/-- ID_developer --/	PUT
Delete a group	json/xml	/groups/-- ID_developer --/	DELETE
Show an asset from a group	json/xml	/groups/-- Group ID_developer --/assets/-- Asset ID_developer--/	GET
List assets from a group	json/xml	/groups/-- ID_developer --/assets/	GET
Show device from a group	json/xml	/groups/-- Group ID_developer --/devices/-- Device ID_developer --/	GET

Assets

Accion	Formato	Punto final	Método
Create an asset	json/xml	/assets/	POST
Show an asset	json/xml	/assets/-- ID_developer --/	GET
List assets (max 1000)	json/xml	/assets/	GET
Update an asset	json/xml	/assets/-- ID_developer --/	PUT
Delete an asset	json/xml	/assets/-- ID_developer --/	DELETE
Show a device from an asset	json/xml	/assets/-- Asset ID_developer --/devices/-- Device ID_developer --/	GET
List devices from an asset	json/xml	/assets/-- ID_developer --/devices/	GET

Devices

Accion	Formato	Punto final	Método
Create a device	json/xml	/devices/	POST
Show device information	json/xml	/devices/-- ID_developer --/	GET
Update a device	json/xml	/devices/-- ID_developer --/	PUT
List devices (max 1000)	json/xml	/devices	GET
List of device's configuration files	json/xml	/devices/-- ID_developer --/deviceconfigs/	GET
Download device config file	json/xml	/devices/-- Device ID_developer --/deviceconfigs/-- Filename --/-- File version	GET
List streams from a device	json/xml	/devices/-- Device ID_developer --/	GET
Delete streams	json/xml	/devices/-- Device ID_developer --/	DELETE

Models

Accion	Formato	Punto final	Método
Create a model	json/xml	/models/	POST
Show models	json/xml	/models/-- ID_developer --/	GET
List models (max 1000)	json/xml	/models/	GET
Update a model	json/xml	/models/-- ID_developer --/	PUT
Delete a model	json/xml	/models/-- ID_developer --/	DELETE

Datastreams

Accion	Formato	Punto final	Método
Create a data stream	json/xml	/streams/	POST
Show a data stream	json/xml	/streams/-- ID_developer --/	GET
List data streams, max 1000	json/xml	/streams/	GET
Delete a data stream	json/xml	/streams/-- ID_developer --/	DELETE
List status stream (max 1000)	json/xml	/streams/?_t=sta	GET
Delete status stream	json/xml	/streams/-- ID_developer --/	DELETE

Status of datastreams

Accion	Formato	Punto final	Método
Create a status stream	json/xml	/status	POST
Show status stream	json/xml	/streams/-- ID_developer /	GET
List status streams (max 1000)	json/xml	/streams/?_t=sta	GET
Delete a status stream	json/xml	/streams/-- ID_developer --/	DELETE

Triggers

Accion	Formato	Punto final	Método
Create a trigger	json/xml	/triggers/	POST
Show a trigger	json/xml	/triggers/-- ID_developer --/	GET
List triggers (max 1000)	json/xml	/triggers/	GET
Update a trigger	json/xml	/triggers/-- ID_developer --/	PUT
Delete a trigger	json/xml	/triggers/-- ID_developer --/	DELETE

Rules

Accion	Formato	Punto final	Método
Create a rule	json/xml	/rules/	POST
Show a rule	json/xml	/rules/-- ID_rule --/	GET
List rules (max 1000)	json/xml	/rules/	GET
Update a rule	json/xml	/rules/-- ID_rule --/	PUT
Delete a rule	json/xml	/rules/-- ID_rule --/	DELETE

Listeners

Accion	Formato	Punto final	Método
Create a listener	json/xml	/listeners/	POST
Show a listener	json/xml	/listeners/-- ID_listener --/	GET
List listeners (max 1000)	json/xml	/listeners/	GET
Update a listener	json/xml	/listeners/-- ID_listener --/	PUT
Delete a listener	json/xml	/listeners/-- ID_listener --/	DELETE

ANEXO III

Datasheet Cámara

applications

- cellular phones
- toys
- PC multimedia
- digital still cameras

ordering information

- **OV05647-G04A** (color, chip probing, 200 µm backgrinding, reconstructed wafer)

features

- 1.4 µm x 1.4 µm pixel with OmniBSI technology for high performance (high sensitivity, low crosstalk, low noise)
- optical size of 1/4"
- automatic image control functions: automatic exposure control (AEC), automatic white balance (AWB), automatic band filter (ABF), automatic 50/60 Hz luminance detection, and automatic black level calibration (ABLC)
- programmable controls for frame rate, AEC/AGC 16-zone size/position/weight control, mirror and flip, cropping, windowing, and panning
- image quality controls: lens correction, defective pixel canceling
- support for output formats: 8-/10-bit raw RGB data
- support for video or snapshot operations
- support for LED and flash strobe mode
- support for internal and external frame synchronization for frame exposure mode
- support for horizontal and vertical sub-sampling
- standard serial SCCB interface
- digital video port (DVP) parallel output interface
- MIPI interface (two lanes)
- 32 bytes of embedded one-time programmable (OTP) memory
- on-chip phase lock loop (PLL)
- embedded 1.5V regulator for core power
- programmable I/O drive capability, I/O tri-state configurability
- support for black sun cancellation

key specifications

- **active array size:** 2592 x 1944
- **power supply:**
  core: 1.5V ± 5% (with embedded 1.5V regulator)
  analog: 2.6 ~ 3.0V (2.8V typical)
  I/O: 1.7V ~ 3.0V
- **power requirements:**
  active: TBD
  standby: TBD
- **temperature range:**
  operating: -30°C to 70°C (see table 8-2)
  stable image: 0°C to 50°C (see table 8-2)
- **output formats:** 8-/10-bit RGB RAW output
- **lens size:** 1/4"
- **lens chief ray angle:** 24° (see figure 10-2)
- **input clock frequency:** 6~27 MHz
- **S/N ratio:** TBD
- **dynamic range:** TBD
- **maximum image transfer rate:**
  QSXGA (2592 x 1944): 15 fps
  1080p: 30 fps
  960p: 45 fps
  720p: 60 fps
  VGA (640 x 480): 90 fps
  QVGA (320 x 240): 120 fps
- **sensitivity:** TBD
- **shutter:** rolling shutter / global shutter
- **maximum exposure interval:** 1968 x t_{ROW}
- **pixel size:** 1.4 µm x 1.4 µm
- **well capacity:** TBD
- **dark current:** TBD
- **fixed pattern noise (FPN):** TBD
- **image area:** 3673.6 µm x 2738.4 µm
- **die dimensions:** 5520 µm x 4700 µm

ANEXO IV

Datasheet DHT11

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

Detailed Specifications:

Parameters	Conditions	Minimum	Typical	Maximum
Humidity				
Resolution		1%RH	1%RH	1%RH
			8 Bit	
Repeatability			±1%RH	
Accuracy	25℃		±4%RH	
	0-50℃			±5%RH
Interchangeability	Fully Interchangeable			
Measurement Range	0℃	30%RH		90%RH
	25℃	20%RH		90%RH
	50℃	20%RH		80%RH
Response Time (Seconds)	1/e(63%)25℃, 1m/s Air	6 S	10 S	15 S
Hysteresis			±1%RH	
Long-Term Stability	Typical		±1%RH/year	
Temperature				
Resolution		1℃	1℃	1℃
		8 Bit	8 Bit	8 Bit
Repeatability			±1℃	
Accuracy		±1℃		±2℃
Measurement Range		0℃		50℃
Response Time (Seconds)	1/e(63%)	6 S		30 S