UNIVERSIDAD
MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ORIHUELA

Master Universitario Oficial en Automatización y Telecontrol para la gestión de Recursos Hídricos y Energéticos

APLICACIÓN DE LAS TECNOLOGÍAS IOT EN UN SISTEMA DE RIEGO EN PARCELA

TRABAJO FIN DE MASTER

AUTOR:
JORGE BOLUDA SEGURA

DIRECTORES:
ANTONIO RUÍZ CANALES
JOSÉ MIGUEL MOLINA MARTÍNEZ
MARTIN JOHN OATES
ANTONIO FERNANDEZ LOPEZ

Junio 2017

Índice

1. INTRODUCCIÓN 5

2. OBJETIVOS 6

3. MATERIAL Y MÉTODOS 6

4. DESARROLLO DEL TRABAJO 7

4.1. Internet of Things (IoT) 7
4.1.1. Arquitectura IoT 9
4.1.2. Aplicaciones web: SaaS 10
4.1.3. Teoría de redes y Wi-Fi 12

4.2. Plataformas Software para IoT 16
4.2.1. ThingSpeak 17
4.2.2. Carriots 18
4.2.3. Electric Imp 23
4.2.4. Blaulabs 25
4.2.5. Thinking Things 26
4.2.6. Cuadro comparativo de plataformas IoT 29

4.3. Dashboards 31
4.3.1. Freeboard 31
4.3.2. Clicdata 32
4.3.3. Dashzen 33
4.3.4. MyDevices Cayenne 34

4.4. Dispositivos IoT 34
4.4.1. Arduino 35
4.4.2. Waspmote 36
4.4.3. Spark 37
4.4.4. ESP8266 38

4.5. IoT en un sistema de riego en parcela 39
4.5.1. Características del sistema de riego en parcela 40
4.5.2. Problemas para aplicar tecnología IoT 40
4.5.2.1. Alimentación de los dispositivos 41
4.5.2.2. Conexión a Internet 43
4.5.3. Selección de tecnología IoT 47

4.5.3.1. Selección de plataforma IoT 49
4.5.3.2. Selección de dashboard 51
4.5.3.3. Selección de dispositivo 51
4.5.4. Pruebas de integración 52
4.5.4.1. Programación de Arduino 52
4.5.4.2. Adquisición y gestión de datos con Carriots 54
4.5.4.3. Integración con FreeBoard 65

5. CONCLUSIONES 72

6. BIBLIOGRAFÍA 1

ANEXO I: ÍNDICE DE FIGURAS 4

ANEXO II: PROGRAMA ARDUINO PARA CONEXIÓN CON CARRIOTS 5

1. Introducción

La necesidad de los últimos años de estar constantemente conectados y poder cuantificar cualquier cosa, da lugar a lo que hoy día conocemos como Internet de las Cosas. El Internet de las cosas o "Internet of Things" (IoT) se refiere a la interconexión de dispositivos empleando la infraestructura actual de internet [1].

Tener esta gran red de objetos significa que muchas tareas e intercambios de información pueden automatizarse y compartirse.

La nube o computación en la nube ("The Cloud" o "Cloud Computing") ser refiere a la propuesta tecnológica de programación que permite ofrecer servicios de computación a través de internet [2].

Nos encontramos ante una nueva revolución tecnológica, dónde objetos cotidianos del día a día con una función específica, evolucionan gracias al IoT, pasando a estar conectados y dotar de nuevas funcionalidades a estos objetos, pudiéndolos controlar y administrar desde tabletas, ordenadores o teléfonos móviles.

Los usos y aplicaciones que se pueden dar son muy variadas, y toca prácticamente todos los ámbitos de la vida humana, desde la vivienda, ciudad, salud, industria, agricultura, consumo, etc.

El ámbito de aplicación del presente proyecto se enmarca en la agricultura inteligente. A pesar de que le está costando arrancar en este medio, probablemente por ser un área en la que la evolución tecnológica nunca ha sido una prioridad, sí que se está intentando revolucionar la forma en la que trabajan los agricultores.

La agricultura inteligente se convertirá en el campo de aplicación más importante en los países predominantemente agrícolas.

Actualmente en España existe un gran número de pequeñas explotaciones agrícolas que comienzan a automatizar sus instalaciones destinadas al riego de parcelas (ya sea en comunidades de regantes o en explotaciones particulares). A pesar de la gran inversión que esto supone para el pequeño agricultor, es frecuente que dichas explotaciones no cuenten con un sistema de telecontrol que les permita conocer el estado de sus instalaciones en todo momento sin necesidad de estar físicamente en la parcela.

Como se expondrá en este proyecto, existe una multitud de soluciones IoT de coste reducido que pueden ofrecer muchas ventajas al agricultor aumentando su producción y minimizando sus costes.

Aunque una instalación esté automatizada localmente es posible que aparezcan problemas derivados de roturas de tuberías, falta de agua en impulsión que podría dañar las bombas, riegos innecesarios tras lluvias abundantes, problemas de alimentación eléctrica u otras averías en el propio sistema de automatización, que obligan al agricultor a acudir frecuente a las instalaciones para comprobar que todo funciona como se espera.

Además, gracias a la arquitectura de las plataformas IoT, es posible interconectar diferentes explotaciones para compartir datos entre ellas. Por ejemplo, un agricultor que cuente con varias parcelas de riego cercanas podría instalar una estación meteorológica en una de ellas, los datos de la estación se enviarían a la plataforma IoT y esta se encargaría de enviar las órdenes necesarias a los sistemas de riego de las distintas parcelas, por ejemplo para anular el riego tras una lluvia abundante.

2. Objetivos

El objetivo principal de este proyecto, consiste en dar una visión global del Internet de las Cosas, aplicando este concepto al campo de la agricultura y proponiendo una solución concreta que podría ser aplicada en la gestión de un sistema de riego en parcela.

3. Material y métodos

En primer lugar se analizarán algunas de las principales plataformas software que proporcionan servicios de IoT, con la finalidad de escoger aquella que se considere más

adecuada para nuestro proyecto particular.

La solución global propuesta para la monitorización y explotación de los datos de nuestro sistema de riego deberá cumplir los siguientes requisitos:

- ✓ Gestión centralizada de diferentes equipos y sensores
- ✓ Almacenamiento de datos
- ✓ Escalable
- ✓ Multiplataforma
- ✓ Reporting (historicos)
- ✓ Alarmas
- ✓ Gestión de eventos (actuación ante cierta alarmas o superación de umbrales)
- ✓ Low Cost
- ✓ Monitorización y explotación de los datos de forma sencilla

Una vez seleccionada la tecnología se realizarán pruebas reales de integración, empleando equipos de bajo coste que cuenten con las siguientes capacidades:

- ✓ Conexión a Internet
- ✓ Capacidad de programación
- ✓ Lectura de sensores de campo
- ✓ Actuación sobre elementos de campo (relés, válvulas, etc.)

También se expondrán los principales conceptos relacionados con los sistemas IoT, como son los DashBoard que podrían ser utilizados como parte de nuestra solución para conseguir una visualización de datos sencilla por parte del usuario.

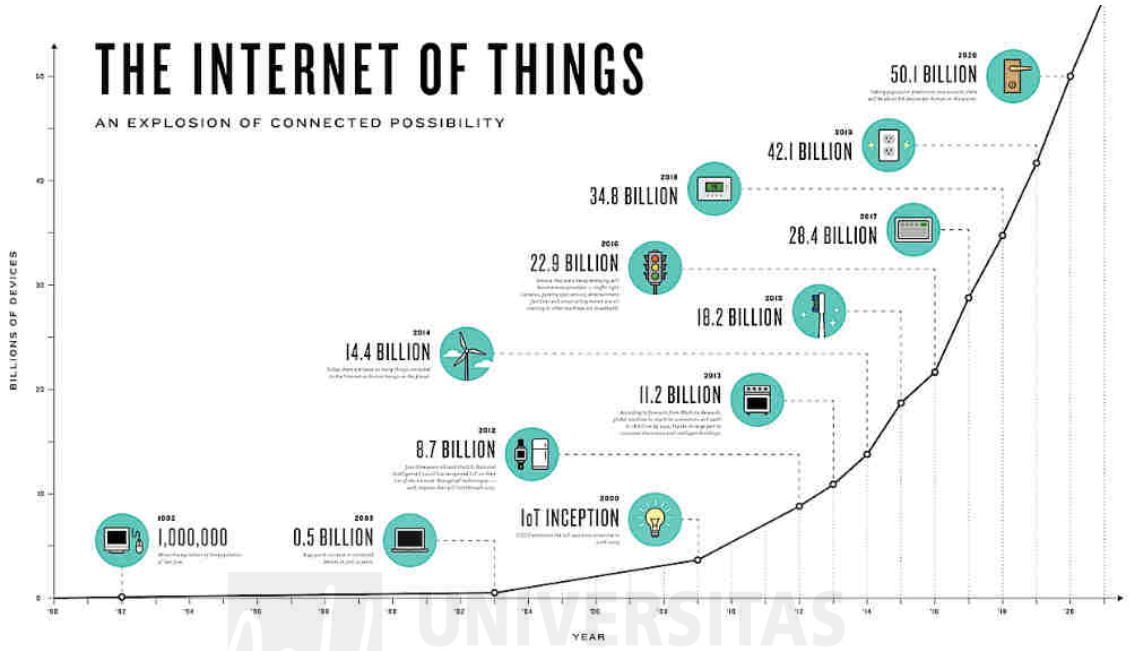
Por último, se expondrán algunas ideas y evolutivos que podrían ser aplicados a este proyecto.

4. Desarrollo del trabajo

4.1. Internet of Things (IoT)

Se prevé un crecimiento de dispositivos conectados mucho mayor que el de PCs o

Smartphones. Se estima que para 2020 habrá unos 50 billones de dispositivos conectados a Internet. Este es el motivo de que algunas veces se hable de "Internet of Everythings", abreviado IoE.

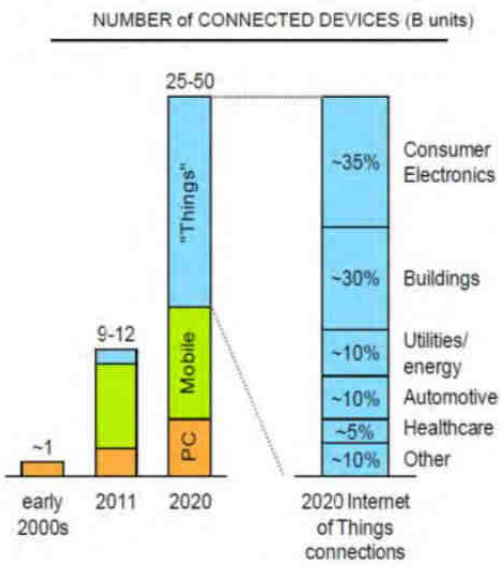


Gráfica de crecimiento de apartados conectados a internet

IoT o IoE ofrecen conectividad avanzada entre dispositivos, sistemas y servicios lo cual va más allá de las comunicaciones máquina a máquina [3]. Por definición el "Internet of Everything" incluye tres diferentes tipos de conexiones [4]:

- Máquina a máquina (M2M)
- Persona a máquina (P2M)
- Persona a persona (P2P)

Además existen estimaciones de porcentaje de dispositivo conectados a internet según los diferentes sectores de la industria. Algunas fuentes estiman que un 65% de los dispositivos conectados a internet serán de electrónica de consumo o edificios online [5].



Estimación distribución del IoT según sectores industriales

4.1.1. Arquitectura IoT

Los dispositivos pueden conectarse con otros dispositivos o aplicaciones a través de internet. Estas aplicaciones pueden ser de diferentes tipos:

- ✓ Aplicaciones móviles
- ✓ Aplicaciones web
- ✓ Aplicaciones de escritorio para PC
- ✓ Aplicaciones propias de otros dispositivos

Los dispositivos pueden conectarse a internet a través de algún componente hardware que permita la conexión por cable o inalámbrica a internet. Este hardware puede ser:

- ✓ Hardware propio del dispositivo
- ✓ Módulo hardware externo
- ✓ Módulo hardware con otro tipo de comunicación (ZigBee, Bluetooth,etc.) más otra interfaz hardware que permita la conexión a internet. Esta interfaz podría ser un dispositivo que actúe de concentrador.

En la arquitectura de la computación en la nube se distinguen tres capas:

- **SaaS, software como servicio.** Es la capa más alta. Caracteriza una aplicación completa ofrecida como un servicio, evitando así la necesidad de instalar la aplicación en los dispositivos del cliente (PC, Smartphone, etc.).
- **PaaS, plataforma como servicio.** Es la capa central. Proporciona una plataforma computacional y un conjunto de software como servicio, de tal forma que permite a los usuarios desarrollar, ejecutar y gestionar aplicaciones sin necesidad de construir y mantener la infraestructura típica del desarrollo de aplicaciones.
- **IaaS, infraestructura como servicio.** Es la capa inferior. A veces HaaS, hardware como servicio, ofrece computadoras físicas o, más a menudo, máquinas virtuales.

Normalmente en una arquitectura IoT, se dispone de una plataforma web (SaaS) que se utiliza como puente para comunicar la aplicación cliente de usuario y el dispositivo, y para el almacenamiento de los datos enviados, visualización de los mismos, gestión de usuarios, sistema de alertas, etc. [6]

Capas de una arquitectura IoT

4.1.2. Aplicaciones web: SaaS

Una aplicación web es proporcionada por un servidor web y utilizadas por usuarios que se conectan desde cualquier punto vía clientes web, principalmente navegadores web. El servidor web distribuye la información a sus clientes a medida que estos la solicitan.

Las aplicaciones web son populares debido a la ubicuidad de los navegadores y su uso multiplataforma, es decir, puede ejecutarse el mismo código en diferentes sistemas operativos.

Arquitectura de una aplicación web

Una aplicación web está basada en el modelo cliente / servidor donde el servidor es un servidor web [7].

Cliente

El código se suministra al cliente y es interpretado por el propio cliente que normalmente es un navegador web. Este código se compone de código HTML y CSS para la estructura y los estilos, y lenguaje script (Ajax, Flash, JavaScript, jQuery)

Servidor

El servidor es el que se encarga de gestionar la base de datos y ejecutar los cálculos necesarios según las solicitudes que se envían desde alguno de los clientes. Según las peticiones del cliente el servidor generará el código necesario para enviarle al cliente.

Base de datos

La base de datos es necesaria para una aplicación web. Se encarga de almacenar los datos necesarios para el correcto funcionamiento de la aplicación web.

Términos: front-end y back-end

Front-end y back-end son términos que se refieren a la separación de intereses entre una capa de presentación y una capa de acceso a datos, respectivamente.

En diseño de software, el front-end es la parte de software que interactúa con el / los usuarios (código de cliente) y el back-end es la parte que procesa la entrada desde el front-end (código del servidor). [8].

Diseño web responsivo

El diseño web responsivo, es una filosofía de diseño y desarrollo cuyo objetivo es adaptar la apariencia de las páginas web al dispositivo que se está utilizando para visualizarlas [9].

Diseño web responsivo

Resulta conveniente utilizar el diseño web responsivo para las aplicaciones web, ya que de ese modo pueden ser accesibles a través de multitud de dispositivos como tabletas, smatphones, libros electrónicos, portátiles, PCs, etc.

4.1.3. Teoría de redes y Wi-Fi

Se hace necesario un repaso a la teoría de redes básica. En este apartado además se explica de forma genérica diferentes protocolos utilizados para intercambio de archivos, así como para seguridad en redes Wi-Fi.

Modelo TCP/IP

Existen diversos modelos de descripción de protocolos de red que determinan el funcionamiento general de las redes. Normalmente se toma como referencia el modelo OSI compuesto por 7 capas de abstracción, aunque el más utilizado es el modelo TCP/IP que hace uso de 5 capas del modelos OSI. El modelo TCP/IP es el que se utiliza para el protocolo 802.11 (Wi-Fi).

En el modelo TCP/IP se describen un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo se conecte a una red. [10]

Las capas que componen el modelo TCP/IP son las siguientes:

Capas del modelo TCP/IP

Cada una de las capas define una serie de características necesarias para la conexión de un dispositivo a una red:

- **Capa de aplicación.** Define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario. Estos servicios utilizan la capa de transporte para enviar y recibir datos. Existen varios protocolos de capa de aplicación. Algunos ejemplos son: HTTP, FTP, Telnet, DNS, SNMP

- **Capa de transporte.** Garantiza que los paquetes lleguen en secuencia y sin errores, al intercambiar la confirmación de la recepción de los datos y retransmitir los paquetes perdidos. Los protocolos de esta capa pueden ser TCP, UDP y SCTP

- **Capa de red.** Acepta y transfiere paquetes para la red. Esta capa incluye el protocolo de Internet (IP), el protocolo de resolución de direcciones (ARP) y el protocolo de mensajes de control de Internet (ICMP).

- **Capa de enlace.** Identifica el tipo de protocolo de red del paquete, en este caso TCP/IP. Proporciona también control de errores y estructuras.

- **Capa física.** Especifica las características del hardware que se utilizará para la red. Entre otras cosas, especifica las características físicas del medio de comunicaciones.

HTTP (Hypertext Transfer Protocol)

Se trata de uno de los protocolos de la capa de aplicación, en la que se centra el objetivo del presente proyecto. HTTP es el protocolo usado en cada transacción de la World Wide Web.

Una petición HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, el tipo de dato retornado o el código de estado.

HTTP define 8 métodos, algunas veces definidos como “verbos” que indican la acción que desean que se efectúe sobre el recurso identificado:

- **GET:** Obtiene información del servidor. Obtenemos tanto la cabecera de la página como el cuerpo del mensaje o el contenido de la página web.

- **HEAD:** Obtiene la cabecera de la página. Es lo mismo que el método GET, pero en este caso sólo nos devuelve la cabecera.

- **POST:** Empleado para enviar información al servidor. Se suele utilizar para enviar información desde formularios como usuario y password que no queremos que sea visible.

- **PUT:** Envía un objeto al servidor. Sirve para crear o registrar información en el

servidor.

- **DELETE:** Solicita al servidor que borre el recurso indicado en la URI.

- **OPTIONS:** Devuelve los métodos habilitados o soportados por el servidor.

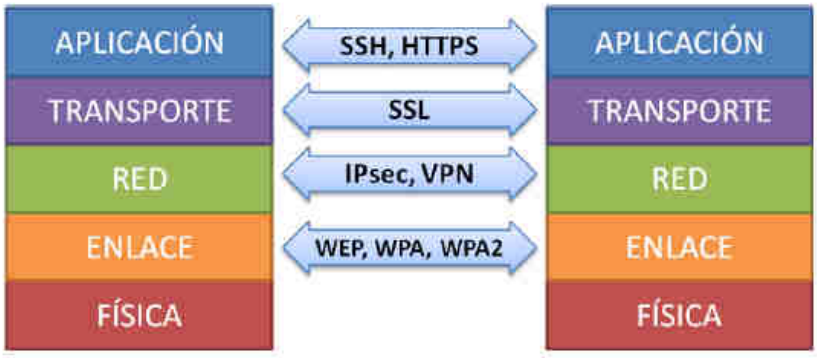
- **TRACE:** Nos permite ver los mensajes que se envían entre el cliente y servidor. Por ejemplo se utiliza para ver la petición que ha llegado al servidor para comprobar si un elemento intermedio (proxy, gateway, tunel, etc.) la ha modificado.

- **CONNECT:** Para ver si se tiene acceso a un servidor. Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL. [11]

Seguridad en redes

Es importante indicar que la información enviada por el protocolo HTTP, descrito en el apartado anterior, se transmiten sin encriptación y por tanto es visible por todos los usuarios de la red.

Por ello hay diferentes mecanismos de seguridad en redes. A continuación se nombran algunos protocolos de seguridad según la capa de modelo TCP/IP:



Mecanismos de seguridad en las distintas capas TCP/IP

Los protocolos WEP, WPA y WPA2 son algoritmos de encriptación para la capa de enlace. Se encargan de la encriptación de la trama Wi-Fi.

A nivel de la capa de red tenemos el conjunto de protocolos IPSec y a nivel de transporte los protocolos SSL y TLS.

Por último, en la capa de aplicación encontramos los protocolos SSH y HTTPs. [12]

Protocolo SSH

Es un protocolo cliente-servidor que permite la conexión segura con máquinas remotas para abrir sesiones y ejecutar comandos, crear túneles o reenviar puertos TCP y conexiones para escritorio remoto. Además puede transferir ficheros mediante los protocolos asociados SFTP y SCP.

Protocolo HTTPS

Se basa en el protocolo HTTP, se puede decir que es la versión segura de HTTP. Este protocolo utiliza SSL/TLS para crear un canal cifrado y de esta forma se consigue que la información sensible no pueda ser utilizada por un atacante.

4.2. Plataformas Software para IoT

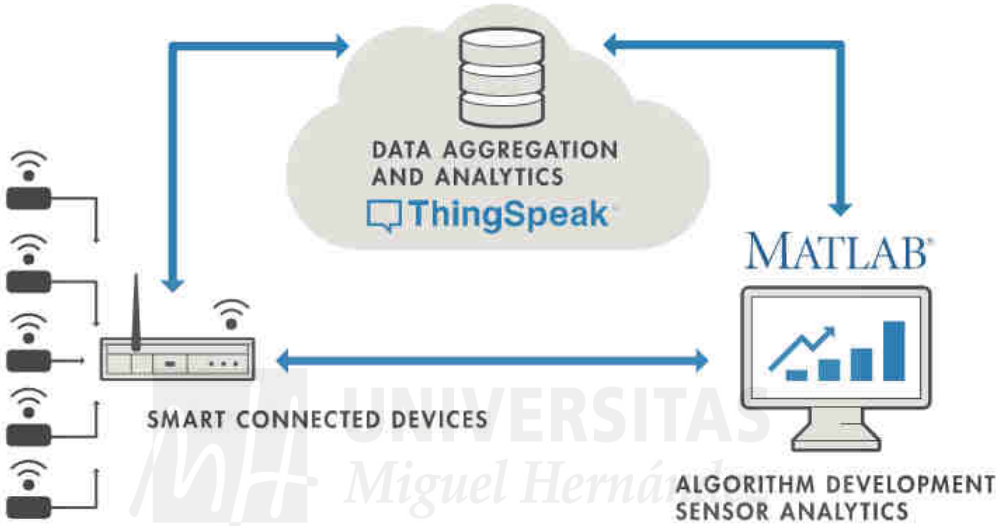
Existen actualmente gran variedad de plataformas web para IoT. Todas estas plataformas ofrecen un servicio de “puente” entre el dispositivo electrónico y la aplicación de control. Normalmente también ofrecen almacenamiento y otros servicios. [13]

En este punto nos vamos a centrar en el estudio de las principales plataformas software que hay actualmente en el mercado. Algunas de estas plataformas son Open Source, otras propietarias, en algunas encontraremos un ecosistema IoT, esto quiere decir que la propia plataforma nos ofrecerá software y hardware para desarrollar nuestra aplicación.

En cada plataforma nos centraremos, en conocer sus características principales, que hardware es el idóneo para cada plataforma, sus ámbitos de aplicación y su propósito general, bien si este es orientado a la educación o por el contrario más orientado a un ámbito profesional.

4.2.1. ThingSpeak

ThingSpeak es una plataforma abierta de aplicaciones, que se caracteriza por ser una plataforma Open Source con una API para almacenar y recuperar datos de los objetos usando el protocolo HTTP sobre Internet o vía LAN (Local Area Network).



Características principales

API

Un punto importante a la hora de desarrollar cualquier proyecto es encontrar un API disponible de forma sencilla para que el desarrollador tenga los mecanismos necesarios para el desarrollo de la aplicación.

En este caso, ThingSpeak dispone de una **API** la cual está disponible en GitHub para su descarga en un servidor propio. Es totalmente abierta, por lo que también se puede modificar su código fuente original y así contribuir a la comunidad con nuevas características, un principio básico en toda plataforma Open Source.

Plugins

Para extender la funcionalidad del sitio también se nos brinda la

oportunidad de desarrollar plugins. Estos nos ofrecen la posibilidad de crear aplicaciones de forma nativa en nuestra plataforma ThigSpeak.

Soporta HTML, CSS y JavaScript como lenguajes de programación.

Integración

Uno de los puntos fuertes en cualquier plataforma IoT, es que permita una amplia integración con diversos dispositivos Hardware y software. En este caso ThingSpeak permite la integración de su plataforma con:

- ✓ Arduino
- ✓ Raspberry Pi
- ✓ IoBridge / RealTime.io
- ✓ Electric Imp
- ✓ Móbiles / Aplicaciones web
- ✓ Redes Sociales
- ✓ Análisis de datos con MATLAB

ThingSpeak también permite la integración con Twitter permitiendo enviar actualizaciones de status a Twitter o incluso escuchar los hashtags para controlar un dispositivo conectado a ThingSpeak.

Ámbitos de aplicación

Esta plataforma reúne a una comunidad que sobre todo se inicia en el mundo del IoT, por lo que principalmente nos encontramos con proyectos que son "prototipos" orientados al mundo del **Smart Home.** [14]

4.2.2. Carriots

Carriots es una plataforma Española en la nube que da un servicio PaaS orientada a proyectos IoT y máquina a máquina (M2M).

Carriots no es una plataforma “open source” como en el caso anterior, aun así podemos darnos de alta y registrar un máximo de 2 dispositivos de forma totalmente gratuita pero con alguna restricción. Para registrar más de 2 dispositivos y hacer uso de todas las funcionalidades que ofrece la plataforma sin ningún tipo de restricción hay que pagar, aunque el precio no es desorbitado siendo este de 2€ por dispositivo extra. También ofrece un servicio para conectar nuestros dispositivos a nuestra nube privada.

GRATIS	CORPORATE	LITE	NUBE PRIVADA BAJO DEMANDA
GRATIS (NO HACE FALTA TARJETA)	2 €* AL MES POR DISPOSITIVO	0,50 €* AL MES POR DISPOSITIVO	Contacte CON NOSOTROS
PARA PRUEBAS Y PROTOTIPOS	DISPOSITIVOS QUE ENVÍAN HASTA 1 MB POR DÍA ESTABLECIENDO MUCHAS CONEXIONES	MUCHOS DISPOSITIVOS QUE ENVÍAN POCO VOLUMEN DE DATOS. EJ. SMART METER, MANTENIMIENTO REMOTO	PARA CONEXIONES ILIMITADAS, ALMACENAMIENTO DE DATOS, USO PRIVADO Y USO PERSONALIZADO
Número mín. dispositivos **1**	Número mín. dispositivos **11**	Número mín. dispositivos **100**	Número mín. dispositivos **Contacte con nosotros**
Número max. dispositivos **2**	Número max. dispositivos **Ilimitado**	Número max. dispositivos **Ilimitado**	Número max. dispositivos **Ilimitado****
API KEYS **4**	API KEYS **100**	API KEYS **10**	API KEYS **Ilimitado****
Max. tramas aceptadas **500 tramas por día** **10 tramas por minuto**	Max. tramas aceptadas **1500 x num. dispositivos por día** **50 x num. dispositivos por minuto** + INFO	Max. tramas aceptadas **25 x num. dispositivos por día** **5 x num. dispositivos por minuto** + INFO	Max. tramas aceptadas **Ilimitado****

Max. tamaño de tramas **5 KB**	Max. tamaño de tramas **10 KB**	Max. tamaño de tramas **5 KB**	Max. tamaño de tramas **Ilimitado****
Max. almacenamiento de tramas **5000 KB por día**	Max. almacenamiento de tramas **1 MB x num. dispositivos por día** + INFO	Max. almacenamiento de tramas **100 KB x num. dispositivos por día** + INFO	Max. almacenamiento de tramas **Ilimitado****
Max. datos almacenados **3 meses**	Max. datos almacenados **1 año**	Max. datos almacenados **1 año**	Max. datos almacenados **Ilimitado****
Max. API requests **1000 por día** **100 por minuto**	Max. API requests **1000 x num. dispositivos por día** **100 x num. dispositivos por minuto** + INFO	Max. API requests **100 x num. dispositivos por día** **10 x num. dispositivos por minuto** + INFO	Max. API requests **Ilimitado****
SMS API **5 SMS por día** **1 SMS por minuto**	SMS API **Gratis 5 SMS por día** **>6 SMS 0.1€* por unidad.**	SMS API **Gratis 5 SMS por día** **>6 SMS 0.1€* por unidad.**	SMS API **Contacte con nosotros**
Email API **100 email por día** **10 email por minuto**	Email API **Gratis 100 email por día** **>100 0.50€* por millar**	Email API **Gratis 100 email por día** **>100 0.50€* por millar**	Email API **Contacte con nosotros**
SDK Http Request (salida) **1000 req. por día**	SDK Http Request (salida) **1000 req. x num. dispositivos por día** + INFO	SDK Http Request (outbound) **25 req. x num. dispositivos por día** + INFO	SDK Http Request (salida) **Ilimitado****
Soporte básico: Atención vía email Foro de desarrolladores	**Soporte Corporate:** Atención telefónica Atención vía Email en menos de 24h ¡Tiempo de respuesta garantizado!	**Soporte Lite:** Atención telefónica Atención vía Email en menos de 24h ¡Tiempo de respuesta garantizado!	**Soporte premium:** Diferentes opciones disponibles **Contacte con nosotros**
Acuerdo de Nivel de Servicio NO	**Acuerdo de Nivel de Servicio** Diferentes planes disponibles **Contacte con nosotros**	**Acuerdo de Nivel de Servicio** Diferentes planes disponibles **Contacte con nosotros**	**Acuerdo de Nivel de Servicio** Diferentes planes disponibles **Contacte con nosotros**

	GRATIS	CORPORATE	LITE	NUBE PRIVADA
Gestión de dispositivos	✔	✔	✔	✔
Uso ilimitado de reglas	✔	✔	✔	✔
Uso ilimitado de listeners	✔	✔	✔	✔
Uso ilimitado de triggers	✔	✔	✔	✔
Motor de aplicaciones SDK	✔	✔	✔	✔
Exportación de datos	✔	✔	✔	✔
Alarmas a medida	✔	✔	✔	✔
Log de depuración	✔	✔	✔	✔
Nivel jerárquico customers		✔	✔	✔
Gestión de API KEY		✔	✔	✔
Gestión de usuarios		✔	✔	✔

Limitaciones y precios de Carriots según el tipo de cuenta de usuario

Características Principales

Carriots es una plataforma que se caracteriza sobre todo por una gran compatibilidad de hardware, una API muy completa y documentada con una gran integración con aplicaciones de terceros. A continuación detallamos sus puntos más importantes:

API

Carriots nos ofrece una API propia basada en REST para poder comunicarnos con la plataforma y gestionar los datos de una forma sencilla. REST se caracteriza principalmente por la existencia de recursos que pueden ser accedidos utilizando un identificador global. Para la manipulación de estos recursos, la comunicación de red se usa el estándar HTTP.

Respecto a los mecanismos de seguridad de la REST API destaca sobre todo el uso de API Keys se caracteriza por un Token que sustituye al clásico user+pass, Checksum con clave simétrica y por último el ya de sobra conocido HTTPS.

Integración

Como venimos comentando anteriormente, sin duda un punto importante en este tipo de plataformas es evaluar la integración que nos ofrece con sistemas externos. En el caso que nos ocupa la plataforma Carritos nos ofrece esta integración a través de su API REST, el PUSH de datos y con peticiones HTTP/s o sockets. Con estos mecanismo podremos tener integración con bases de datos, ERPs, CRM, data warehouse, etc.

Carriots nos ofrece ya mecanismo integrados que podrán ser accesibles mediante su SDK. La lista de mecanismos integrados son: DropBox, Twitter, Mailing, SMS internacionales y Sockets.

En cuanto a equipos hardware Carriots ofrece un amplio catálogo de hardware compatible. Algunos de los equipos soportados son:

- ✓ Arduino
- ✓ Raspberry Pi
- ✓ Beagle Bone

- ✓ Fez Cerbuino
- ✓ Cubie Board
- ✓ TST Gate
- ✓ TST Mote
- ✓ CloudGate
- ✓ Nanode
- ✓ Electric IMP
- ✓ TST Light
- ✓ Mobile Devices C4Max
- ✓ Mobile Devices C4EVO
- ✓ Mobile Device OBD Ongle
- ✓ Electric IMP April Development Board
- ✓ We500
- ✓ Satel Dataloggers DL170,DL171 y OWA31IETH
- ✓ Otro Hardware (Siempre y cuando pueda comunicarse con su API REST)

Ámbito de Aplicación.

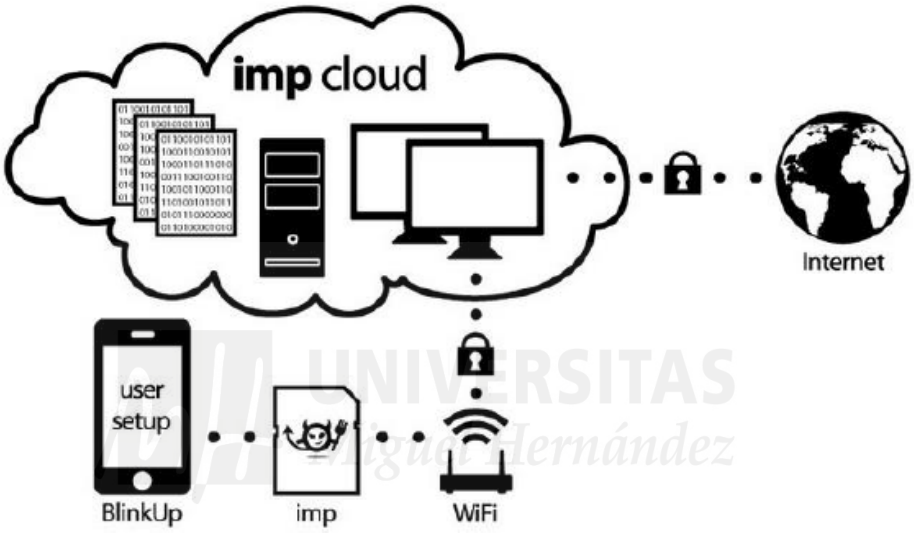
Estamos ante una plataforma que dispone de una amplia oferta en este punto, con diversos proyectos y distintos casos de usos, los ámbitos más importantes se recogen a continuación: [15]

- Smart City
- Smart Energy
- Smart Oil
- Smart Agiculture
- Smart Buildings
- Smart Retail
- Smart Banking
- Smart Consumer
- Smart Logistic

4.2.3. Electric Imp

Esta plataforma abarca tanto soluciones hardware como software en forma de servicios. Se caracteriza por proveer soluciones finales con total integración hardware, sistemas operativos, APIs y servicios en la nube, proporcionando seguridad y rapidez.

Se caracteriza sobre todo por tener un ecosistema de productos completo y variado:



Características principales

ImpOs es el core de la plataforma Electric Imp que dispone de una Api abierta

Cloud es la pieza central de la plataforma Electric Imp para conectar nuestros dispositivos a través de internet, nos ofrece un servicio Saas con almacenamiento de datos, visualización de reportes, etc.

API

El uso de una API abierta permite la personalización de nuestra plataforma y adaptarla a nuestras necesidades, en este caso Electric Imp ha elegido el lenguaje Squirrel. Se trata de un Api que hace un tratamiento de objeto muy variados. Esta API dispone de los métodos necesarios de seguridad para proteger las comunicaciones entre internet y el dispositivo.

Hardware

Electric Imp comenzó con la implementación de los módulos Wi-FI cuya función era de actuar como un Gateway para conectar dispositivos a Internet sin importar el hardware del dispositivo.

Actualmente, en el mercado dispone de tres módulos:

- ✓ **Imp001**: Fue el primer módulo creado se compone de un procesador, Wifi y un antena. A diferencia de los demás módulos este incluye una ranura para un tarjeta SD.
- ✓ **Imp002**: Es una evolución del módulo anterior de un tamaño más reducido.
- ✓ **Imp003**: Este módulo es el que mayor rendimiento Wifi tiene y más sencillo es su integración con la plataforma Electric Imp. A diferencia de los demás módulos este no incluye antena. Tiene un tamaño diez veces menor que su predecesor imp002.

Cada módulo esta optimizado para un consumo mínimo de energía, esto permite que puedan operar durante años con tan solo un par de baterías AA.

Además de estos módulos, Electric Imp ofrece una buena variedad de productos relacionados con su ecosistema.

Integración

Electric Imp ofrece una solución extremo a extremo haciendo posible que sea fácil conectar cualquier dispositivo a internet.

Ámbitos de aplicación

La plataforma Electric Imp ofrece diversas soluciones en el mundo del IoT, pero sobre todo se centra en los siguientes ámbitos: [16]

- Smart Logistic
- Smart Home
- Smart Consumer
- Smart Industry

4.2.4. Blaulabs

Empresa Española con sede en Barcelona, es una plataforma SaaS modular dedicada al IoT, centrándose en la monitorización y análisis en tiempo real de los dispositivos y sensores.

Es una plataforma desde la cual podemos capturar cualquier tipo de datos haciendo uso de la API abierta que se nos facilita. Otros de los puntos importantes, es su base de datos Time series capaz de gestionar altos volúmenes de datos en tiempo real.

Es una solución completamente ampliable y modular, es totalmente flexible permitiendo un gran número de configuraciones entre las que destacan dashboards con diferentes gráficos, informes, analíticas, etc.

Características principales

La plataforma Blaulabs es una plataforma modular, el conjunto de esos módulos se agrupan en torno a Blaulabs Suite cuya línea de negocio se basa en dar soluciones tipo Smart City, Gestión Energética, Smart Manufacturing, etc.

API

A estas alturas poco nos debe sorprender que se use una API basada en REST. Desde la propia web nos facilita información de cómo utilizar esta API, se usa los lenguajes Python y Java.

Hardware

Dado el negocio al que va orientado, el catálogo de hardware compatible es bastante limitado, algunos ejemplos son los siguientes:

- Satel OWA31IETH
- Satel SenNet DL160

- Satel SenNet DL161
- GridPoint
- GridAgent
- GridLink

Ámbitos de aplicación

Esta plataforma tiene una línea de negocio muy marcada, principalmente se centra en ***energía, Smart city e infrastructuras***.

Ofrece soluciones completas para estos tipos de negocios:

- ***BlauEnergy 4.0***: Orientada a la gestión de la energía, a través de la captación y almacenamiento de datos mediante sensores. La finalidad de esta solución es mejorar la eficiencia energética, reducir costes y reducir emisiones.
- ***Smart City*****:** El módulo Smart City busca gestionar de forma eficiente los servicios que se prestan en la ciudad. Con este módulo se pretende dotar de inteligencia a toda esa infraestructura básica para poder desarrollar un modelo de ciudad más eficiente.
- ***Infraestructuras inteligentes*****:** Solución orientada tanto empresas públicas como privadas, que ofrece una gestión más eficiente de los sistemas de transporte, redes eléctricas (Smart Grid), agua y telecomunicaciones. [17]

4.2.5. Thinking Things

Estamos ante una plataforma del internet de las cosas que nos ofrece tanto hardware como software. Thinking Things es la apuesta de Telefónica para el internet de las cosa. Utiliza la red GSM consiguiendo, de esa forma, una conectividad global.

Características principales

Se basa en unos módulos compuestos básicamente de una batería, un sensor y un módulo de conectividad GSM:

Módulos de Thinking Things

También ofrece una plataforma software con una interfaz muy fácil de usar. La conectividad es una novedad respecto a otras plataformas, a través de la red móvil y con una SIM M2M se obtiene conectividad casi en cualquier parte.

API

También se trata de una API basada en REST, que permite la integración software con otras aplicaciones.

Hardware

Sin duda es la base de esta plataforma, por originalidad, sencillez y facilidad de uso. El hardware que proporciona es escalable y modular, en la base está la

batería, la cual puede recargarse mediante un microUSB. Encima del módulo de batería se encuentra conectado el módulo de conectividad, este módulo es el core que envía los datos de todos los módulos a la nube. Como comentamos al principio usa la red móvil y trae una tarjeta SIM. Y justo encima del módulo de conectividad se encuentran conectados módulos con sensores.

Ámbitos de aplicación

Es una plataforma de reciente aparición que de momento dispone de algunos dispositivos con capacidad de comunicación GSM, control de posición GPS, sensores de presencia y control ambiental: temperatura, humedad y nivel de luz. Por ello, actualmente su ámbito de aplicación se limita a proyectos de **Smart Home** y **Smart Logistic**. [18]

4.2.6. Cuadro comparativo de plataformas IoT

NOMBRE	Hardware	Ámbito	Ventajas	Inconvenientes
Thingspeak	Arduino, IoBridge, RealTime.io, Raspberry Pi, Electric Imp	Smart Home	Open Source. Integración redes sociales	Funcionalidades limitadas, orientado a prototipos, pruebas y desarrollo propio de soluciones
Carriots	Arduino, Raspberry Pi, Electric Imp, Beagle, TST, Industrial	City, Energy, Oil, Agiculture, Buildings, Retail, Banking, Consumer, Loguistic	Integración redes sociales. Hw compatible. Ámbitos de aplicación. Proporciona gran cantidad de recursos para construir aplicaciones propias.	Se trata de un servicio Paas. Desde hace poco tiempo (2015) ofrece posibilidad de crear Dashboards, aunque limitados y de pago.
Electric Imp	Electric Imp	Loguistic, Consumer, Home, Industry	HW compatible. Ámbitos de aplicación. Ecosistema propio.	Complejidad elevada.

Bluelabs	SATEL Grid	City, Energy, Manufacturing	Especialistas en Smart City. Solución Saas completa.	HW limitado.
Thinking Things	Propietario	Smart Home	Fácil configuración e integración. Escalabilidad	Poca variedad Sensores. Solo es posible usar su HW.

4.3. Dashboards

Uno de los requisitos mencionados en los métodos del presente proyecto es que la solución seleccionada debe contar con una interfaz de usuario sencilla que permita monitorizar y explotar los datos del sistema de riego.

Como se ha comentado en el apartado anterior, en el mercado existen varias soluciones gratuitas (o versiones gratuitas con alguna limitación) que permiten diseñar sencillos paneles de control de usuario.

Estas soluciones reciben el nombre de "Dashboards" (tablero de instrumentos), consisten en una representación sencilla de los principales indicadores que intervienen en un proceso (KPIs), el concepto es similar al cuadro de mandos de un vehículo.

Están orientados a la toma de decisión para la optimización de un proceso, aunque para un análisis más exhaustivo se requieren de otras herramientas más complejas.

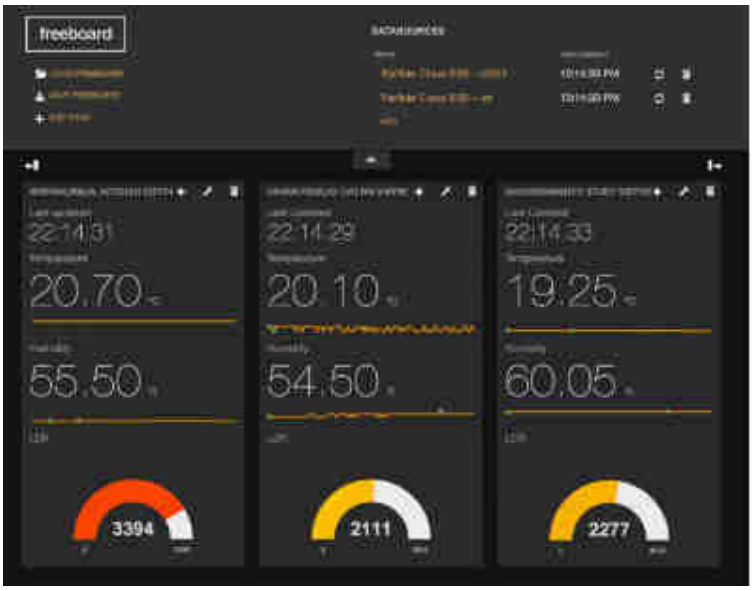
A continuación se describen algunos dashboards con versiones gratuitas, prestando atención a las siguientes características: [19]

- Número de KPIs
- Segmentación y contexto (relevancia)
- Visualización sencilla
- Capacidad de integración y de enlazar información proveniente de distintas fuentes de datos

4.3.1. Freeboard

Freeboard es un dashboard open-source en tiempo real, desarrollada para IOT y otras aplicaciones web híbridas. Se caracteriza por su sencillez a la hora de crear aplicaciones con widgets de arrastrar y soltar.

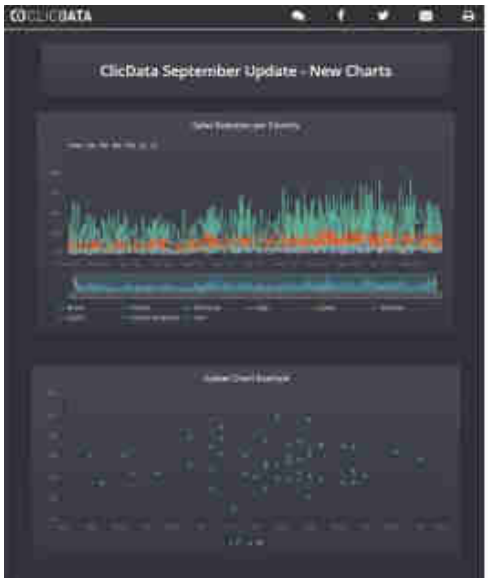
Freeboard proporciona mecanismos de acceso a las API de otras aplicaciones y ofrece una URL única para acceder al panel de control desde cualquier navegador web. Además utiliza un diseño web responsivo (definido en *0 Diseño web responsivo*) que facilita su visualización desde cualquier dispositivo. [20]



Ejemplo panel FreeBoard

4.3.2. Clicdata

ClicData ofrece una solución SaaS intuitiva para crear Dashboard, incluye una gran cantidad de indicadores de arrastrar y soltar en el espacio de trabajo permitiendo la elaboración de paneles de control muy completos.



Ejemplo panel Clicdata

Este Dashboard tiene la capacidad de conectarse a un gran número de sistemas y bases de datos tales como Google Analytics, Facebook, Salesforce, Oracle, MySQL y archivos en Dropbox, Google Drive, etc. Aunque la versión gratuita solo permite recibir datos de ficheros de texto o archivos Excel. [21]

4.3.3. Dashzen

En este caso se trata de un Dashboard gratuito que ofrece la posibilidad de crear paneles públicos y privados. También cuenta con un gran número de widgets que además son completamente personalizables y permiten mostrar imágenes, RSS Feed, URLs, predicciones meteorológicas, etc.

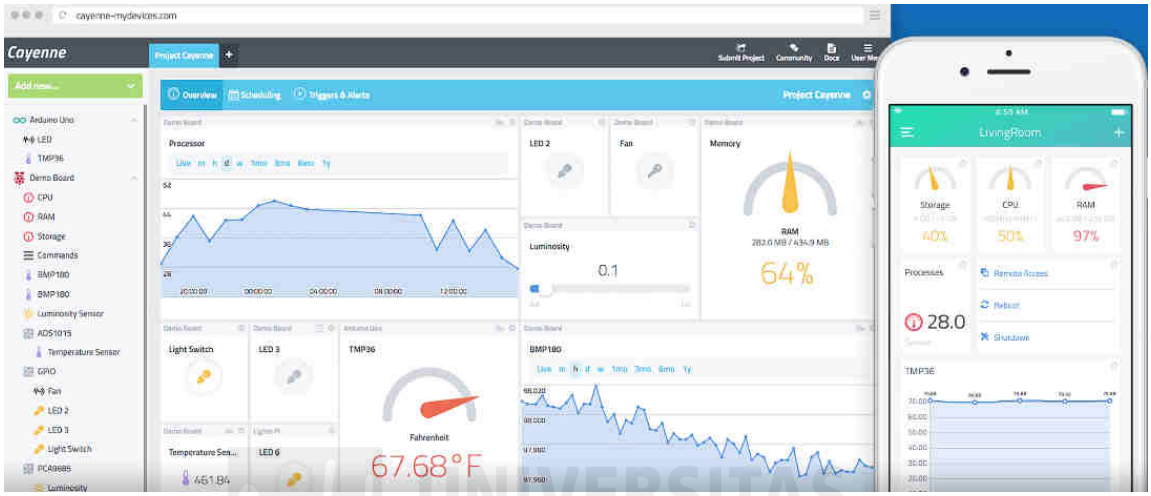
Soporta los siguientes servicios: [22]

- Google analytics
- Twitter
- Facebook
- Chartbeat
- Salesforce
- Stripe
- StackExchange
- Github social coding

Ejemplo panel Dashzen

4.3.4. MyDevices Cayenne

Cayenne es un dashboard que nos permite crear nuestro panel de control directamente desde el Smartphone. Es tan sencillo como descargar su aplicación y empezar a añadir y configurar elementos.



Ejemplo panel Cayenne

Proporciona librerías para Arduino y Raspberry Pi que permiten una rápida integración de estos dispositivos, así como añadir, gestionar y controlar sensores y actuadores. La gran contra de este proyecto es que solo se puede añadir los widgets predeterminados que nos proporciona la aplicación. [23]

4.4. Dispositivos IoT

A continuación se describen algunos equipos de bajo coste que existen actualmente en el mercado y que podrían ser utilizados en la solución adoptada.

4.4.1. Arduino

Arduino es una plataforma de hardware libre, sin duda la más popular del mercado con una amplia comunidad que hace de esta solución la más apropiada para iniciarse en el mundo del hardware libre.

Se basa en una placa con un microcontrolador, con entradas y salidas analógicas y digitales, además posee un entorno de desarrollo propio.

La placa de Arduino se caracteriza por el uso de un microcontrolador Atmel AVR, siendo el Atmel 328 uno de los más utilizados. El modelo Arduino UNO rev3, una de las placas más utilizadas por la comunidad open- hardware, cuenta con este microcontrolador:

Detalle de placa Arduino Uno

Otra característica fundamental a la hora de adquirir una placa Arduino es su bajo coste y su puesta en funcionamiento. En el mercado encontramos kits de iniciación con una gran documentación.

Arduino es multiplataforma, es decir, admite los sistemas operativos más populares que hay en la actualidad, tales como Windows, Mac Os X y Linux.

A la hora de adquirir una placa Arduino nos encontramos con una gran oferta de modelos con diferentes características. A continuación se muestra un cuadro comparativo con las placas Arduino oficiales más populares del mercado:

Modelo	I/O digitales	Entradas analógicas	Salidas PWR	UART	Memoria	Precio*
Uno r3	16	6	6	1	32kb	20€
Mini 05	14	6	8	1	32kb	17€
Leonardo	20	12	7	1	32kb	23€
Mega r3	14	6	7	1	256kb	18€
YUN	20	12	7	1	32kb	70€

**Nota*: *Precios orientativos, ya que en internet puedes encontrar una gran variedad de precios.*

La principal novedad que trae la placa YUN respecto a las demás placas es que trae integrada conexión Wifi y Ethernet, por lo que no es necesario adquirir los Shields para dotar de esta funcionalidad a las demás placas. Combina la capacidad de un Arduino Leonardo (basado en el procesador ATMEGA32U4) con un chip Wifi que usa Linino (un MIPS GNU - Linux basada en OpenWRT) aumentando así las posibilidades de programación del dispositivo. [24]

4.4.2. Waspmote

Creada por la empresas española Libelium, estamos ante una plataforma modular open source, cuya finalidad es la de construir redes inalámbricas de bajo consumo.

Prácticamente la plataforma se compone de la placa Waspmote con un microcontrolador de la familia Atmel, memoria, batería, acelerómetro y sockets para ir añadiendo módulos.

A nivel software cuenta con una API y un compilador open source, usa el protocolo de comunicación Zigbee con alcances de hasta 40 Km. Dispone de diferentes módulos de comunicación, tales como Bluetooh, GPS, y GPRS.

Libelium fabrica estos módulos basándose en estándares de hardware abierto, aunque intenta meter un cariz diferenciador dándole un toque de mayor robustez, la facilidad para incorporar diferentes sensores y la posibilidad de operar a largas distancias. La placa Waspmote usa el mismo IDE que Arduino, por lo que el mismo código es prácticamente compatible en ambas plataformas.

A diferencia de Arduino, cuya plataforma está más orientada a la creación de prototipos, Waspmote es un dispositivo especialmente orientado a la creación de redes sensoriales inalámbricas de bajo consumo en entornos reales.

Estos módulos se utilizan especialmente en el ámbito de las Smart City (Smart Parking, monitorización de redes de alcantarillado, medidas de calidad del aire, etc.) y también en el mundo agrícola (medidas de irrigación, crecimiento de las cosechas, detección de sucesos meteorológicos, etc.). [25]

4.4.3. Spark

Se trata de una placa de tamaña reducido que se compone de un módulo WiFi para dotar de conexión a Internet prácticamente a cualquier cosa. Es una empresa que nació gracias a la financiación de la popular plataforma KickStarter.

Se caracteriza por ser un dispositivo fácil de instalar y que se puede programar sin cables, gracias a su conexión WiFi. Combina la simplicidad de un Arduino con todo el poder del chip ARM cortex M3, considerado uno de los mejores módulos disponibles en la actualidad.

Spark Core es un dispositivo que se caracteriza por estar siempre conectado a Internet, una vez adquirido el producto se proporciona acceso a la nube de la compañía Spark, desde la cual podemos realizar desarrollos y actualizaciones de los equipos de forma inalámbrica.

Es una plataforma totalmente Open Source, todos los diseños de firmware y hardware son de código abierto, lo que permite una integración libre en diversos proyectos. Hay que destacar que Spark Core usa estándares abiertos como HTTP, TCP o TLS/SSL.

Si queremos ampliar su funcionalidad existen varios tarjetas shield o escudos para ello, incluso comercializan una shield que permite añadir cualquier escudo que encontremos en el mercado de la plataforma Arduino. [26]

4.4.4. ESP8266

El ESP8266 es un chip Wi-Fi de bajo coste con una pila TCP/IP completa y un microcontrolador, fabricado por una empresa en Shanghai.

El primer chip aparece en los mercados alrededor de agosto de 2014 con el módulo ESP-01, desarrollado por la empresa AI-Thinker. Este pequeño módulo permite a otros microcontroladores (como Arduino) conectarse a un red inalámbrica Wi-Fi y realizar conexiones simples con TCP/IP.

Aunque su diseño tuviera como objetivo inicial servir para las comunicaciones WiFi controladas por una MCUexterna, cuenta con numerosas características (comunicaciones SDIO, SPI, UART, I^2C, algunos pines GPIO, sensor interno de temperatura, etc.) que hacen que sea una solución viable para muchas tareas por sí mismo.

Al igual que en su momento le ocurriera a Arduino, un diseño abierto y la comunidad generada en su entorno propició aportaciones para la mejora de las prestaciones software del módulo. Sin duda la circunstancia diferencial que ha hecho que se popularice en el mundo de la IoT como dispositivo autónomo e incluso como una alternativa a Arduino, ha sido la creación de un firmware que permite programarlo en varios lenguajes como son C y C++, Lua o JavaScript. Su creciente popularidad ha permitido la adaptación de otros entornos con los que poder programarlo usando diferentes lenguajes. En ese sentido Espruino, que sirve para programar en JavaScript (ECMAScript) el ESP8266 es una interesante alternativa.

El diseño de un dispositivo que usa la tecnología de Arduino implica el diseño de una PCB que incluye un microcontrolador programado como un Arduino mientras que el mismo dispositivo usando un ESP8266, por su formato y tamaño, puede estar basado en una placa que incorpore el módulo completo directamente.

Diferentes modelos del módulo ESP8266

Si bien los módulos ESP8266 no pueden competir en número de GPIO con algunas placas Arduino, el carácter de nodo de la mayoría de los equipos IoT y la posibilidad de añadir integrados específicos hacen que no sea un inconveniente especialmente significativo frente a, por ejemplo, las prestaciones como microcontrolador de a las gamas más básicas de Arduino, que tampoco podrían competir con los bajísimos precios de los ESP8266 y menos aún si necesitan al propio ESP8266 para gestionar las comunicaciones WiFi. [27] [28]

4.5. IoT en un sistema de riego en parcela

En los apartados anteriores se ha descrito el funcionamiento y arquitectura de un sistema IoT. Como se ha visto, estas plataformas IoT tienen muchos ámbitos de aplicación. En este apartado se analiza cómo se podría aplicar esta tecnología a una solución concreta en al ámbito agrícola como podría ser un sistema de riego en parcela como el descrito en el apartado de introducción del presente proyecto.

Los principales problemas que plantea una solución de este tipo son dos: alimentación de equipos instalados fuera del cabezal de riego donde se encuentran las bombas de impulsión y la conectividad a internet: requisito fundamental para una solución IoT que puede ser un problema en zonas de grandes extensiones o con poca cobertura.

En los siguientes apartados se propondrán soluciones para estos problemas. Además se definirán los elementos que suelen estar presentes en cualquier sistema de riego y se realizarán pruebas reales con las soluciones escogidas en los apartados anteriores.

4.5.1. Características del sistema de riego en parcela

Generalmente un sistema de riego cuenta como mínimo con los siguientes elementos:

- Toma de agua (normalmente un embalse o balsa de riego)
- Bomba de impulsión
- Filtros de limpieza
- Tanque solución nutritiva (abono)
- Bomba inyectora de abono
- Válvulas con solenoide para controlar el riego de los diferentes sectores

Normalmente estos elementos se encuentran en una caseta donde existe un cuadro eléctrico con un programador de riego. La función del programador consiste en poner en marcha el sistema de forma automática con los parámetros y horarios configurados (en caso de disponer de un medidor de caudal, este control se suele realizar por volúmenes de agua al ser un método más preciso).

Según estos horarios, el controlador de riego activa la bomba de impulsión que saca agua del embalse para regar los diferentes sectores que se hayan planteado en la parcela, abriendo y cerrando las válvulas de los sectores correspondientes y pasando por algunos filtros de limpieza. Al mismo tiempo, según la cantidad de abono parametrizada, el controlador activa la bomba inyectora que añade solución nutritiva al agua impulsada hacia los distintos sectores. Se suele programa el arranque de la bomba inyectora un tiempo después del arranque de la bomba de impulsión y también se para un tiempo antes, para favorecer la limpieza de las tuberías de impulsión.

4.5.2. Problemas para aplicar tecnología IoT

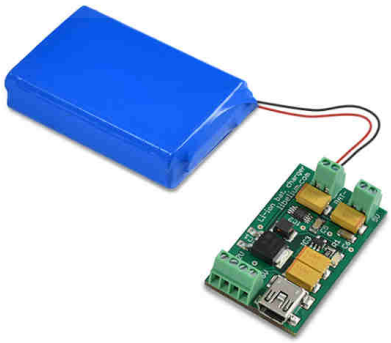
4.5.2.1. Alimentación de los dispositivos

Uno de las dificultades que implica llevar a cabo un proyecto de este tipo es la alimentación de los equipos de campo, ya que es posible que se requiera monitorizar sensores instalados directamente en la zona regable, fuera del cabezal de riego, donde no se dispone de alimentación eléctrica.

Una posible solución sería cablear los sensores y actuadores alojados en esta zona hasta el cabezal de riego que dispone de alimentación eléctrica para alimentar las bombas, variadores de frecuencia, etc. En este caso los dispositivos encargados de recoger la información y actuar sobre los elementos de campo se alojarían también en un cuadro eléctrico ubicado en esta caseta.

Dicho cableado puede ser un problema, ya que pueden existir distancias muy largas y existe una alta probabilidad de rotura al llevar a cabo las labores agrícolas. Por lo que sería recomendable que los equipos dispusiesen de cierta autonomía para evitar esto.

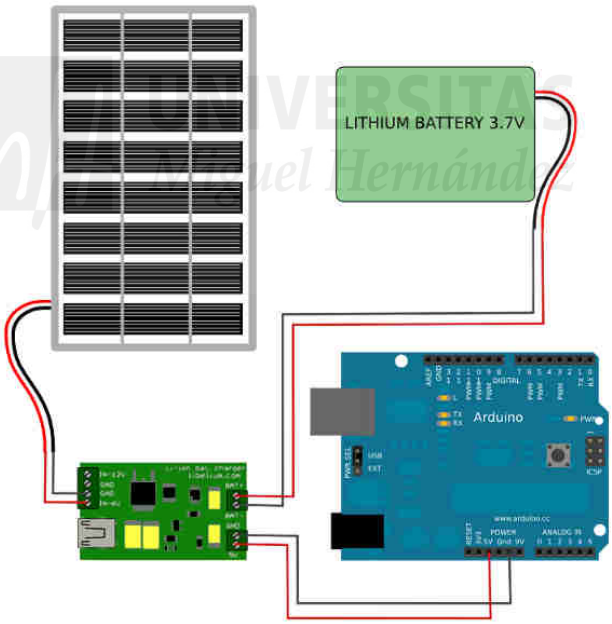
Existen varias soluciones de mercado que solventan este problema. Podemos emplear directamente equipos alimentados mediante baterías y de bajo consumo o diseñar dispositivos propios con dichas capacidades, empleando componentes de mercado como los siguientes:



Módulo regulador de 5V para Arduino con batería Li-Ion de 2300mAh

Panel solar

Empleando un módulo regulador, una batería y un panel solar podríamos dotar a nuestro dispositivo de una total autonomía en cuanto a alimentación se refiere: [29]



Arduino alimentado mediante batería y placa solar

Si bien en la mayoría de los casos sería suficiente con que el dispositivo estuviese alimentado mediante baterías sin necesidad de disponer de paneles solares. Ya que su misión principal sería la de recopilar información de forma periódica y enviarla a la plataforma IoT cuando fuese necesario, se podrían programar estos equipos para que minimizaran su consumo de energía y de esa forma contar con una autonomía de varios años. Para ello existen diversas técnicas:

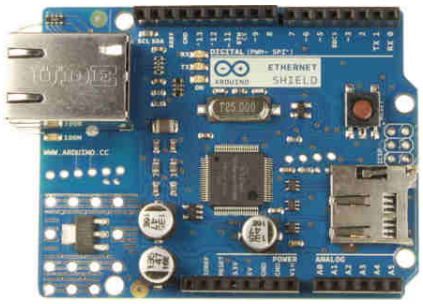
- **Establecer ventanas de comunicación.** De forma que el equipo no enciende el módulo de comunicación fuera de esa ventana. Por ejemplo se podría programar esta ventana un poco antes de la hora de riego, de modo que este se pueda anular si se recibe la orden correspondiente del sistema de control. Podríamos programar nuestro equipo para que, en condiciones normales, registre datos de forma periódica realizando el envío de los mismos una única vez al día a no ser que ocurra algo fuera de lo normal, en ese caso se podría forzar el envío de datos de forma inmediata.

- **Aumentar el tiempo de polling.** En una explotación de regadío no se requiere un tiempo real estricto, es decir, no necesitamos que los datos se registren cada pocos segundos siendo en la mayoría de casos suficiente con disponer de valores actualizados cada hora o incluso varias horas. Cuanto más podamos aumentar este tiempo mayor será la autonomía del dispositivo.

- **Emplear el modo sleep e interrupciones.** Otra opción es hacer que nuestro dispositivo "duerma" mientras no tenga nada que hacer, ahorrando así consumo de batería. En la mayoría de microcontroladores (Arduino y similares) existen funciones que permiten realizar esto. Tenemos la opción de programar que "despierte" cada cierto tiempo (mediante un temporizador o Watch Dog) para tomar las medidas necesarias con el polling que hayamos definido o emplear interrupciones hardware en el caso que simplemente queremos controlar cuando un sensor proporciona un determinado valor (por ejemplo saber si está lloviendo) para que nos envíe la información de forma inmediata.

4.5.2.2. Conexión a Internet

El segundo problema que nos encontramos a la hora de diseñar una arquitectura que permita monitorizar y controlar de forma remota un sistema de riego es la conectividad a Internet.



Shield Ethernet para Arduino

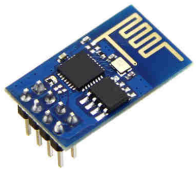
En el cabezal de riego es imprescindible que exista la posibilidad de realizar esta conexión a Internet, aunque sea mediante GPRS, sino la solución propuesta no sería viable.

En tal caso se podría instalar un rúter al que se conectaría los dispositivos IoT de la instalación. Esta conexión se podría hacer directamente mediante cableado UTP para recoger la información y actuar sobre los elementos alojados en el propio cabezal de riego o bien de manera inalámbrica (Wi-Fi). Para recoger la información de los sensores instalados fuera de la caseta (sensores de temperatura, humedad, precipitación, etc.) sí que se recomienda una conexión inalámbrica para evitar los problemas de cableado mencionados en el apartado anterior.

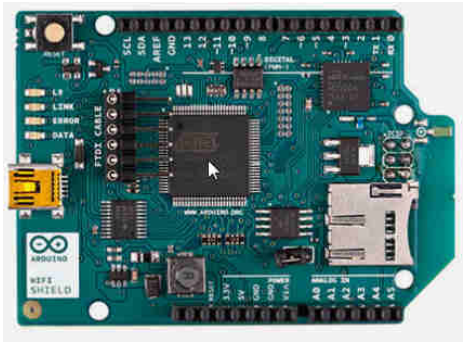
La ventaja de implementar una solución basada en las tecnologías analizadas en este proyecto es la versatilidad y escalabilidad de la solución propuesta. Al igual que en el apartado anterior, en caso de instalar equipos fuera del cabezal de riego podríamos encontrarnos problemas, en este caso de cobertura inalámbrica. Para permitir dicha conexión a internet de los elementos de campo se dispone de diversas alternativas que además podrían combinarse dentro del mismo sistema de control.

Conexión Wi-Fi

Existe una gran variedad de dispositivos de mercado con la capacidad de conexión a una red Wi-Fi, así como varios módulos de ampliación para dotar de esta característica a nuestros dispositivos, que además son muy económicos:



ESP8266



Shield Wifi para Arduino

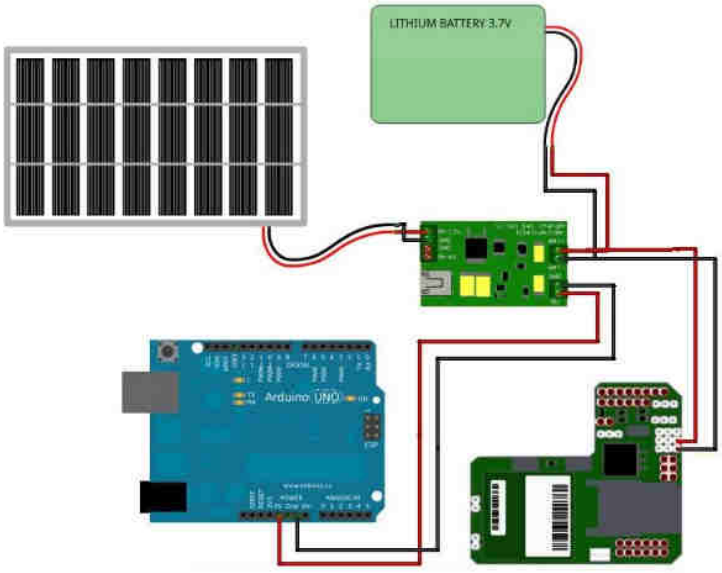
Conexión GSM / GPRS

Es posible que la zona regable cuente con una extensión de terreno demasiado grande como para proporcionar cobertura Wi-Fi a todos los dispositivos instalados. En ese caso una posible solución consistiría en dotar de conectividad a internet a cada uno de los dispositivos encargados de recibir la información de los sensores y de actuar sobre los elementos de campo. Para ello se podrían emplear dispositivos con capacidad de conexión GSM / GPRS o ampliar las capacidades de conexión de nuestros dispositivos con módulos de comunicación como el siguiente:

Módulo de conexión GPRS/GSM para Arduino, Raspberry Pi o Intel Galileo

El inconveniente de emplear equipos con comunicación GSM/GPRS es su elevado coste. Los módulos que proporcionan esta capacidad de conexión son caros y además es necesario un contrato con una operadora de servicios móviles para cada uno de los equipos. Aunque es cierto que el consumo de datos sería muy bajo y que existen contratos especiales y más económicos para estos usos.

También es posible alimentar estos módulos de comunicación empleando los elementos vistos en el apartado anterior: [30]



Arduino y módulo GPRS/GSM alimentado mediante batería y placa solar

Red mallada. Nodos Mesh

Otra opción para resolver el problema de cobertura de nuestro parque de dispositivos de campo sería emplear la tecnología Mesh.

Las redes Mesh, también llamadas redes inalámbricas malladas, redes acopladas o redes inalámbricas de infraestructura, son aquellas redes en las que se mezclan las dos topologías de las redes inalámbricas, la topología Ad-hoc y la topología infraestructura. Básicamente son redes con topología de infraestructura, donde hay un elemento de "coordinación": un punto de acceso o estación base, pero que permiten unirse a la red a dispositivos que, a pesar de estar fuera del rango de cobertura de los puntos de acceso, están dentro del rango de cobertura de alguno de los dispositivos de la red.

Permite que las tarjetas de red de los dispositivos se comuniquen entre sí, independientemente del punto de acceso. Esto quiere decir que los dispositivos pueden no mandar directamente sus paquetes al punto de acceso sino que pueden pasárselos a otras tarjetas de red para que lleguen a su destino. De esta forma se consigue aumentar la cobertura de la red conforme se van integrando nuevos equipos.

Para que esto sea posible es necesario el contar con un protocolo de enrutamiento que permita transmitir la información hasta su destino con el mínimo número de saltos o con un número que aun no siendo el mínimo sea suficientemente bueno. Esta tecnología es resistente a fallos, pues la caída de un solo nodo no implica la caída de toda la red. [31]

4.5.3. Selección de tecnología IoT

Como se ha comentado a lo largo del proyecto, en la actualidad es frecuente encontrarnos sistemas de riego más o menos automatizados que cuentan con un programador de riego (y puede que incluso algunos sensores) para poner en marcha y controlar el funcionamiento del sistema de forma automática según la configuración realizada. Y es habitual que este control automático sea local, es decir, que el explotador no tiene conocimiento de si el sistema está funcionando correctamente a no ser que se encuentre en la propia instalación.

Aunque se podría diseñar y programar un dispositivo IoT de bajo coste que realizase las tareas desempeñadas por el controlador de riego, esto queda fuera del alcance del proyecto. Pero, ante una instalación nueva, sería muy recomendable emplear un dispositivo de control

con la capacidad de conectarse a internet y recibir órdenes de una plataforma IoT.

Suponiendo el caso más común, una instalación con un programador autónomo que funcione en local sin posibilidad de conectarse a Internet, también se podría implantar una arquitectura IoT como las descritas en este proyecto. En cada caso concreto habría que estudiar las posibilidades de comunicación que presenta cada programador (ModBus, CanOpen, etc.), algunos cuentan con entradas digitales físicas de control o con algún bus de comunicaciones que nos permitirían dialogar con él desde otro dispositivo que si cuente con la capacidad de conectarse a la plataforma y, en el peor de los casos, siempre podríamos actuar sobre las salidas del programador anulando sus órdenes o forzando la maniobra de los elementos de campo (bombas, electroválvulas, etc.) con simples relés.

La solución propuesta pasaría por ampliar nuestra instalación con todos los sensores necesarios para controlar la explotación según las necesidades del cliente y condiciones de la parcela. Como podrían ser los siguientes:

- Sensores de nivel en el embalse y en los tanques de nutrientes, para prevenir que las bombas aspiren en vacío
- Sensores de presión y caudal en las tuberías de riego. Con ellos podríamos controlar el correcto funcionamiento de las bombas y detectar posibles roturas de tuberías.
- Sensores de humedad en las zonas de riego para determinar si el riego es necesario.
- Estación meteorológica (pluviómetro, medidor de temperatura, anemómetro, etc.) para controlar las condiciones ambientales de la parcela.
- Analizador de redes para controlar los parámetros eléctricos.
- Sonda multiparamétrica o equipos de medición de calidad del agua para detectar el correcto funcionamiento de los filtros, el estado del agua del embalse o el correcto suministro de nutrientes. Sobre todo, para un sistema de riego en parcela es importante disponer de un sensor de salinidad (o de conductividad).
- Lisímetros u otros sensores para controlar el estado de la producción. [32]

Estos sensores y equipos de medida se conectarían a los dispositivos IoT encargados de enviar la información a la plataforma, además estos equipos disponen de salidas capaces de actuar sobre los elementos de campo cuando así se lo indique el sistema.

Muchos de los equipos mencionados suponen un coste elevado, una de las ventajas de emplear una plataforma IoT como solución de monitorización y control es que son fácilmente escalables, de modo que se podrían ir añadiendo elementos al sistema conforme

se considere necesario.

Otra de las ventajas de este tipo de solución es que la lógica de control se podría programar en la propia plataforma, directamente en la nube, reduciendo la complejidad de la programación del sistema. Los equipos actuarían como meras pasarelas que por un lado recogen y envían información y por otro reciben órdenes y actúan sobre los elementos de campo.

Aunque por seguridad siempre es necesario un control local por si existiese algún problema de comunicación, en nuestro caso ese control local lo realiza el programador de riego convencional. En caso de un fallo de comunicación con los equipos IoT instalados en la parcela, la plataforma contará con la capacidad de informar al agricultor de que el control ha pasado a ser únicamente local o, si se trata de un problema puntual con uno de los equipos IoT, será capaz de identificarlo para proceder a su reparación.

Además de permitir al agricultor una monitorización remota de su instalación de riego, la solución planteada es capaz de enviar alarmas en tiempo real, utilizar la información recogida en otras parcelas cercanas, mejorar las capacidades de un programador de riego convencional modificando el comportamiento automático o deteniendo el sistema de riego si se reciben señales fuera de los parámetros normales e incluso enviar órdenes manuales de forma remota desde un teléfono móvil o cualquier dispositivo conectado a internet.

4.5.3.1. Selección de plataforma IoT

Además de las plataformas analizadas en este proyecto existe una gran variedad de compañías y asociaciones que ofrecen servicios IoT de este tipo, aunque su estudio queda fuera del alcance de este proyecto.

Otros ejemplos de plataformas IoT se citan a continuación: [33] [34] [35]

- Amazon AWs IoT
- Microsoft Azure IoT
- ThingWorkx
- IBM Watson IoT
- Cisco IoT
- Salesforce IoT
- Oracle Cloud

- Predix General Electric
- Xively
- Evrythng
- Vortex Cloud
- Sensor Cloud
- Spark
- Zatar

De las plataformas estudiadas la que se considera que se adapta mejor a las necesidades del presente proyecto es Carriots (vista en el apartado *4.2.2 Carriots)*. Se considera que con esta tecnología sería bastante sencillo gestionar un sistema de riego integrando distintos equipos y sensores en una misma aplicación de usuario, notificando alarmas y eventos, monitorizando los datos del sistema y actuando sobre los elementos de campo si fuese necesario.

Esta plataforma, a pesar de ser de pago, no es excesivamente cara para los servicios que nos ofrece. Al tratarse de una platarforma Paas (plataforma como servicio, definida en el apartado *4.1.1 Arquitectura IoT*) nos ofrece la infraestructura necesaria para construir y mantener nuestras propias aplicaciones, evitándonos la necesidad de tener que invertir en el montaje y mantenimiento de servidores de datos y facilitándonos la tarea de diseño de aplicaciones personalizadas usando los servicios que ofrece la plataforma, por lo que el gasto que supone está justificado.

Carriots proporciona las herramientas necesarias para integrar casi cualquier cosa con la capacidad de conectarse a internet y gestiona el almacenamiento de los datos recibidos, así como el tratamiento de los mismos permitiendo el envío de alarmas sms/mail en el momento en que algún dato se desvía de su comportamiento habitual. También ofrece una aplicación de gestión completa de los dispositivos y sensores que se van integrando en el sistema, donde se configuran los equipos y las reglas necesarias (eventos, alarmas, notificaciones, etc.) permitiendo la escalabilidad de nuestra solución.

Desde hace poco tiempo también ha empezado a ofrecer servicios para la creación de Dashboards propios, estos disponen de funcionalidades algo limitadas para el coste que suponen y existen varias alternativas más económicas e incluso gratuitas para desarrollar nuestra capa de presentación de datos al usuario (como se verá en el apartado *4.3 Dashboards*).

Además con las herramientas que ofrece Carriots se facilita mucho el desarrollo de una aplicación de usuario (ya sea web, móvil, aplicación de escritorio, etc.) si se quisiese realizar

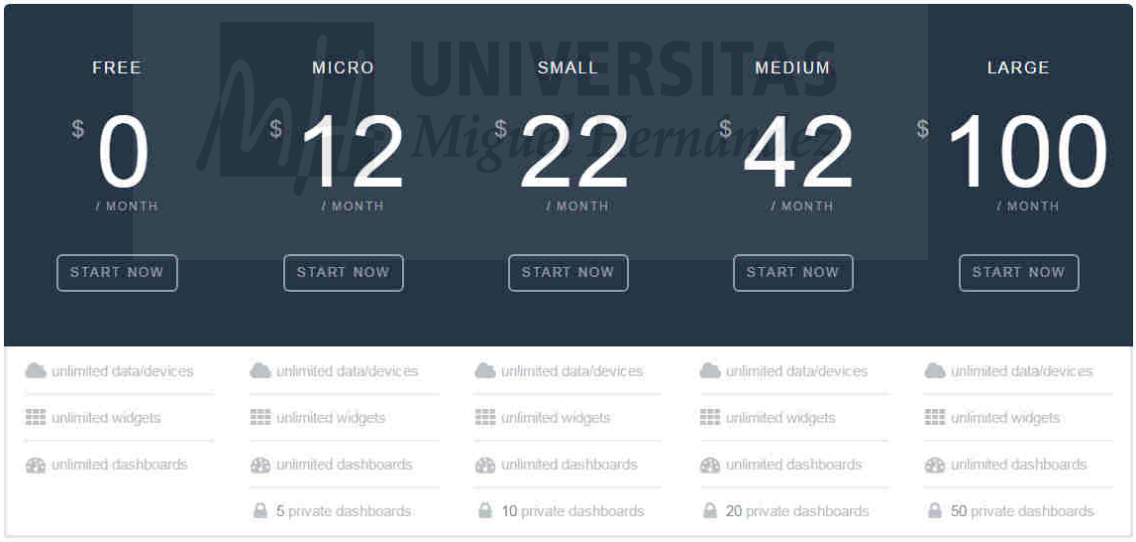
una panel de control de usuario más avanzado (con gestión de usuarios, tramos horarios, etc.).

4.5.3.2. Selección de dashboard

Para llevar a cabo las pruebas de integración de nuestro proyecto se ha seleccionado la tecnología ofrecida por Freeboard.

El motivo es la sencillez de dicho dashboard, tanto a la hora de diseñar el panel de control y presentar los datos al usuario como a la hora de conectar con la API de la plataforma IoT escogida en el punto *4.5.3.1 Selección de Plataforma*

La limitación de la versión gratuita es que no permite crear paneles de usuario privados, aunque estos son bastante asequibles:



Precios FreeBoard

4.5.3.3. Selección de dispositivo

Para las pruebas de integración que se llevarán a cabo en este proyecto se empleará una placa arduino uno rev3 a la que dotaremos de conexión a internet mediante una shield Ethernet.

Todos los dispositivos analizados se programan de manera similar y la mayoría permiten emplear el IDE de Arduino. Es por esto que por precio y sencillez se selecciona esta placa para realizar un prototipo de pruebas. Aunque para una implementación real de la solución planteada en este proyecto, sería necesario diseñar o adquirir un equipo (que podría estar basado en Arduino o no) con mayor robustez frente a condiciones adversas, como podrían ser los módulos diseñados por la empresa Libelium comentados en este apartado.

4.5.4. Pruebas de integración

Tras los análisis realizados en los apartados anteriores se han seleccionado los siguientes componentes para llevar a cabo las pruebas de integración:

- **Arduino** con shield Ethernet como dispositivo IoT al que se conectará un sensor de temperatura 1-wire.
- **Carriots** como plataforma IoT que recibirá y gestionará la información.
- **Freeboard** para realizar un panel de control donde presentar los datos al usuario de la explotación.

En los siguientes apartados se describirá paso a paso como se pueden configurar estos elementos para dialogar entre ellos y generar así un prototipo de la solución propuesta que se podría escalar a una solución real.

4.5.4.1. Programación de Arduino

Con el fin de simular un dispositivo que pudiera encontrarse instalado en una instalación real de riego en parcela, se ha conectado a la placa Arduino un sensor de temperatura 1-wire. Además se ha conectado una shield Ethernet para proporcionarle esta conexión:

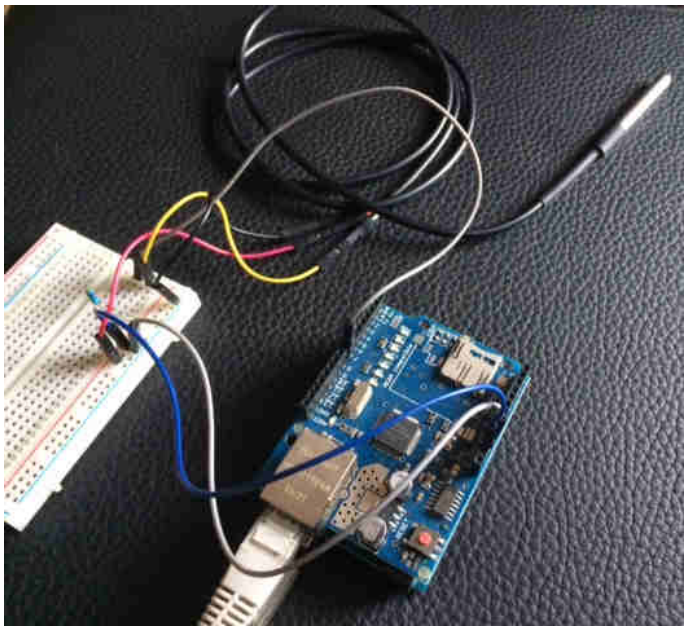


Foto del montaje: conexión de sensor de temperatura 1-Wire a Arduino

Adicionalmente se ha programado el Arduino con la capacidad de medir su propia tensión de alimentación para enviarla a Carriots junto con el valor de temperatura medido. Esto es muy útil para aquellos equipos que se alimenten mediante baterías para poder conocer su estado en todo momento. A continuación se muestran las partes más importantes del programa (véase el código completo en el ANEXO I):

- Se emplean los datos de la cuenta Carritos (en el siguiente apartado se indicará dónde encontrarlos)

```
// Datos la cuenta Carriots
const String APIKEY = "8ba78245b0fdaf7bab27ecc241e0c6498036e8e0b9ca3dcdeb1d597c9972821f"; // Carriots apikey
const String DEVICE = "MiArduinoUno@jboluda.jboluda"; // Id_developer
```

- Se configura la tarjeta de red

```
// Dirección MAC de la shield ethernet conectada al Arduino
byte mac[] = {  0x90, 0xA2, 0xDA, 0x00, 0x79, 0xE9 };

IPAddress ip(192,168,1,177);       // Dirección IP asignada
IPAddress server(82,223,244,60);  // Dirección IP de api.carriots.com
```

- Se leen los sensores conectados al Arduino (en nuestro caso temperatura y tensión de alimentación), se compone la trama JSON con estos datos y se envía a Carriots con la frecuencia deseada

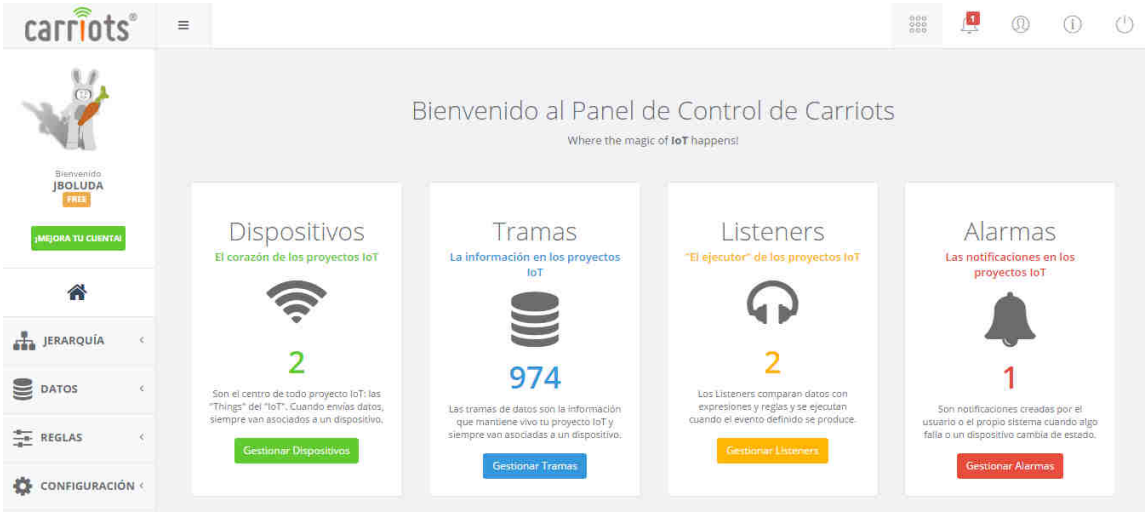
```
// Se compone la trama json que se enviará a carriots con los datos leídos de temperatura y voltaje
String json = "{\"protocol\":\"v2\",\"device\":\""+DEVICE+"\",\"at\":\"now\",\"data\":{\"temperatura\":\""+celsius+"\",\"tension\":\""+voltaje+"\"}}";

// HTTP request
client.println("POST /streams HTTP/1.1");
client.println("Host: api.carriots.com");
client.println("Accept: application/json");
client.println("User-Agent: Arduino-Carriots");
client.println("Content-Type: application/json");
client.print("carriots.apikey: ");
client.println(APIKEY);
client.print("Content-Length: ");
int thisLength = json.length();
client.println(thisLength);
client.println("Connection: close");
client.println();

client.println(json);
```

Una vez cargado el programa, podemos conectarnos al Arduino por su puerto USB para comprobar que todo funciona como esperamos (monitorizando el puerto serie): [36]

4.5.4.2. Adquisición y gestión de datos con Carriots

En este apartado se describen las herramientas que nos proporciona la plataforma escogida, Carriots, para configurar y gestionar los datos de nuestro ecosistema IoT.

En primer lugar, una vez nos hemos registrado (de forma gratuita) en la web de Carriots, obtenemos acceso a un panel de control de usuario muy completo:

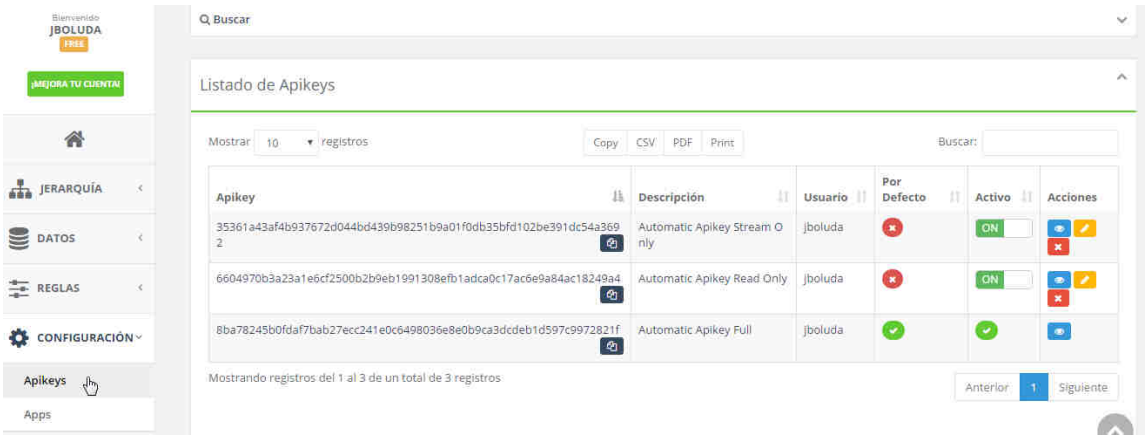


Pantalla inicial del panel de control de Carriots

Configuración de Apikeys

Como ya se ha mencionado en el presente proyecto las Apikeys son empleadas por la REST API como mecanismo de seguridad, sustituyendo al cásico usuario y contraseña.

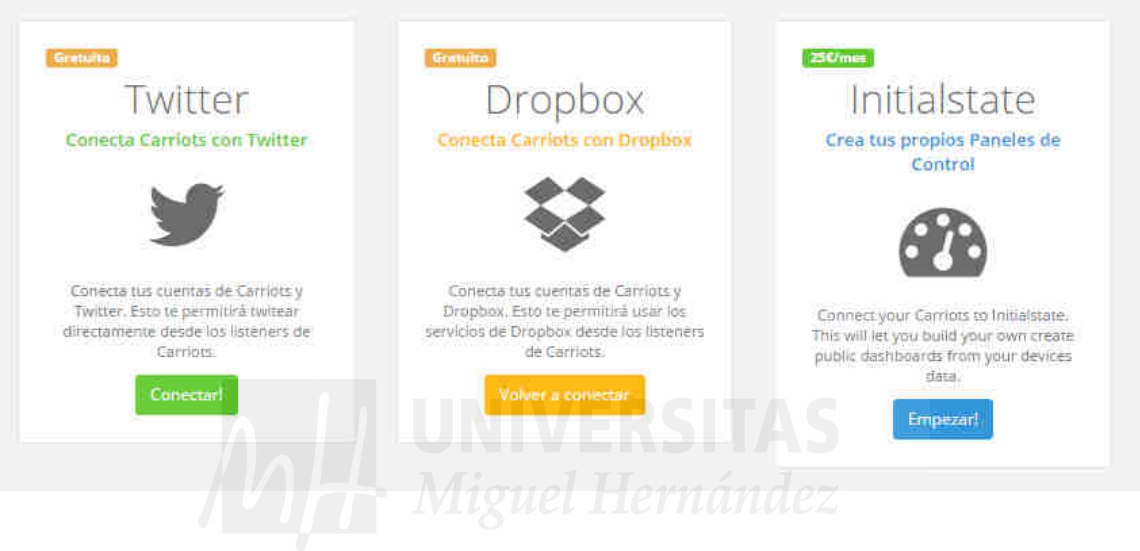
Accediendo al apartado de configuración mostrado en la siguiente imagen podemos ver y copiar la apikey que nos interese (solo lectura, solo escritura o control total) para emplearla en las llamadas que se realicen a los servicios proporcionados por la REST API:



Listado de Apikeys

En el apartado anterior se puede observar cómo se emplea la apikey en el código implementado para las pruebas con Arduino.

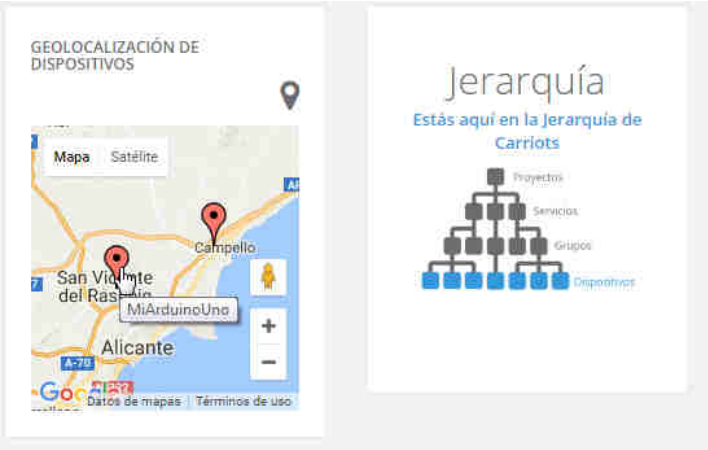
Desde este menú de configuración también se nos permite conectar Carriots con dos aplicaciones de forma gratuita: Twiter y Dropbox, de modo que podemos emplear los servicios ofrecidos por estas en los "listeners" de Carriots que se definen a continuación:



Desde este menú también podemos acceder a la creación de paneles de control de usuario, dashboards, pero se trata de un servicio de pago y existen varios servicios gratuitos en el mercado que permiten programar paneles similares.

Gestión de jerarquías y dispositivos

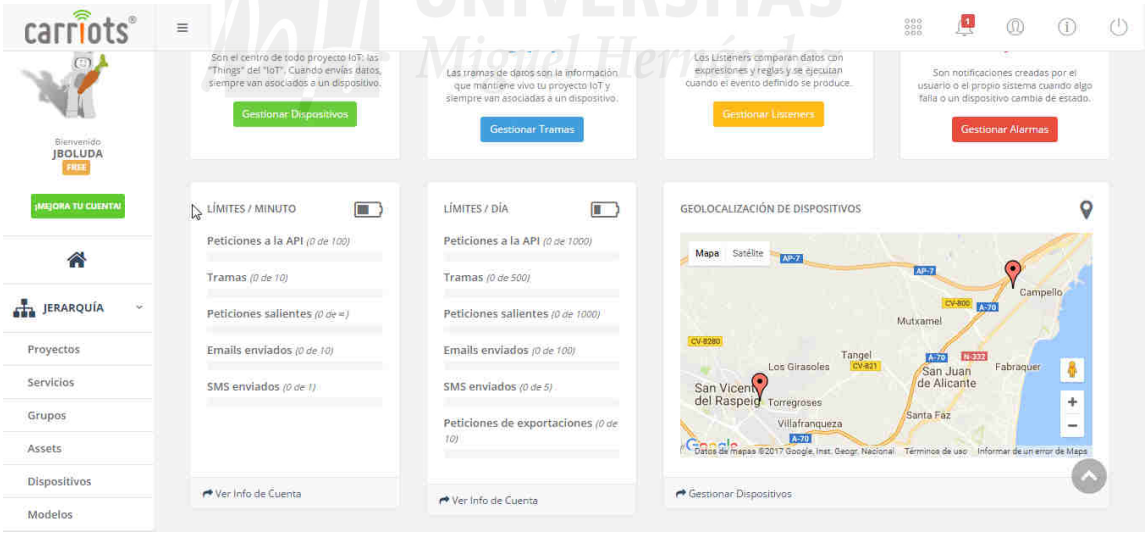
El panel de control Carriots permite gestionar varios proyectos, ordenando los dispositivos por servicios y grupos dentro de estos servicios:



Geolocalización de dispositivos y Jerarquía de Carriots

Además permite hacer otras subdivisiones (Assets) y crear modelos o plantillas de dispositivos que usemos frecuentemente.

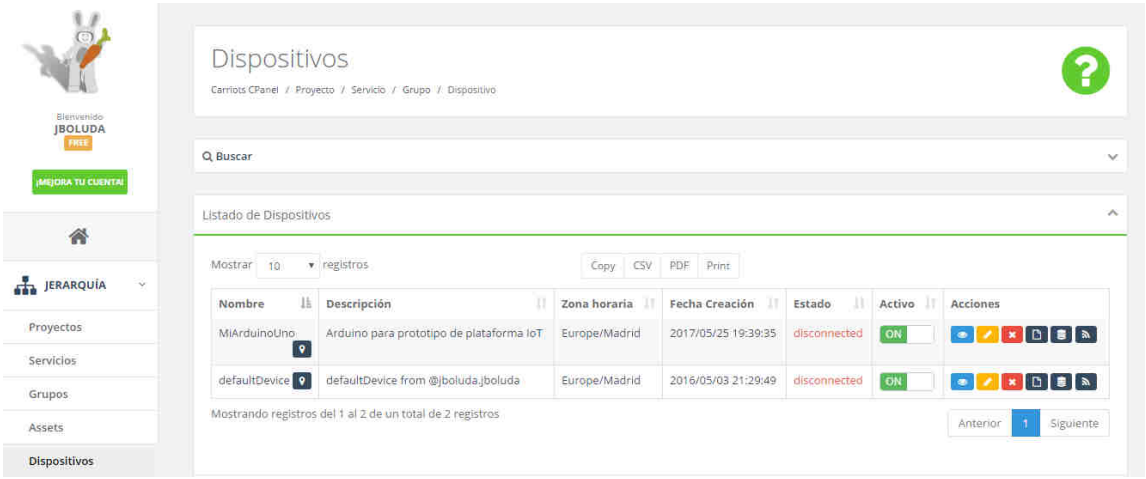
Cuando configuramos un dispositivo desde el panel de control podemos asignarle una ubicación física, esto nos permitirá localizar nuestros equipos de un vistazo sobre un mapa:



Esta organización en jerarquía nos permitirá, por ejemplo, configurar reglas que se apliquen sólo a un determinado grupo de dispositivos dentro de nuestro proyecto, a un servicio o a un dispositivo concreto.

Desde esta parte del panel de control podremos ver el estado de las restricciones de nuestras descripciones, si se ha alcanzado alguno de los límites establecidos por minuto o por día según la cuenta que tengamos.

También nos permite conocer el estado actual de nuestros dispositivos:

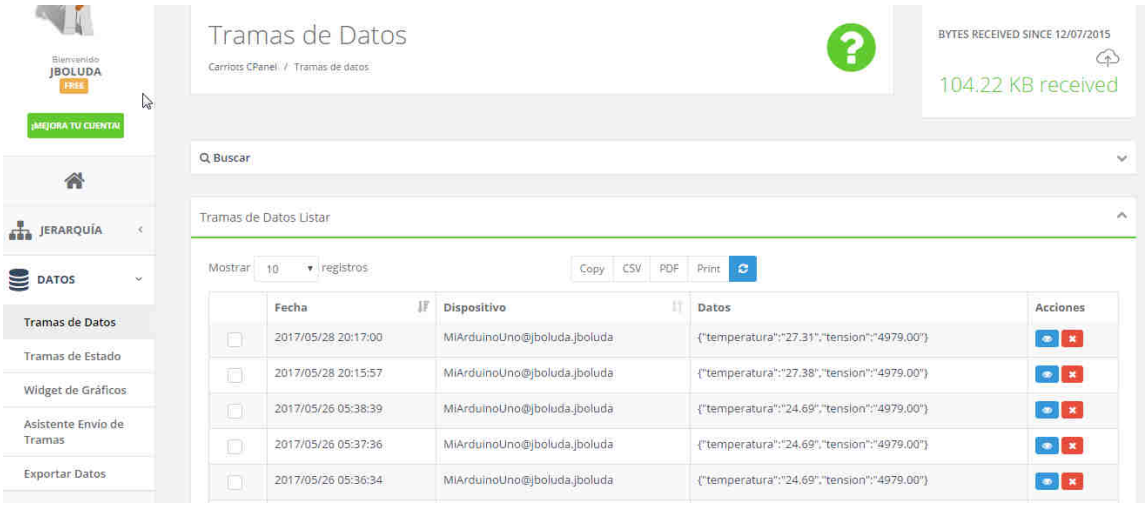


Listado de dispositivos

Toda esta información, además de estar disponible desde el panel de control, Carriots la proporciona mediante su API REST, de modo que podríamos desarrollar nuestro propio panel de control de usuario con las características que nos interese.

Gestión de datos

Desde el siguiente menú del panel de control tendremos acceso a la visualización de los datos recibidos desde nuestros dispositivos, con su “timestamp” y su valor:



Tramas de datos recibidas en Carriots

Además nos ofrece asistentes para exportar los datos recibidos, generación de trama JSON y widgets de gráficos. Por ejemplo, este último asistente nos permite generar un código html que se podría emplear para crear una página web donde visualizar estos datos en un gráfico:


```
<div id="carriots-graph"></div>
<script src="https://www.carriots.com/js/graphs.js"></script>
<script>
var carriots = {
'apikey' : '6604970b3a23a1e6cf2500b2b9eb1991308efb1adca0c17ac6e9a84ac18249a4',
'mime' : 'image/svg+xml',
```

```
        'type' : 'line',
        'tz'   : 'Europe/Madrid',
        'axis[x]' : 'at',
        'axis[y]' : 'temperatura',
        'data[0]' : 'temperatura',
        'query[device]': 'MiArduinoUno@jboluda.jboluda',
        'query[at_from]': '2017-02-26 00:00',
        'query[at_to]': '2017-05-27 00:00',
        'query[sort]': '_id',
        'query[order]': 'desc',
        'query[max]': '40'
        };
    </script>
```

Gráfica generada con el configurador de Carriots

Reglas y listeners

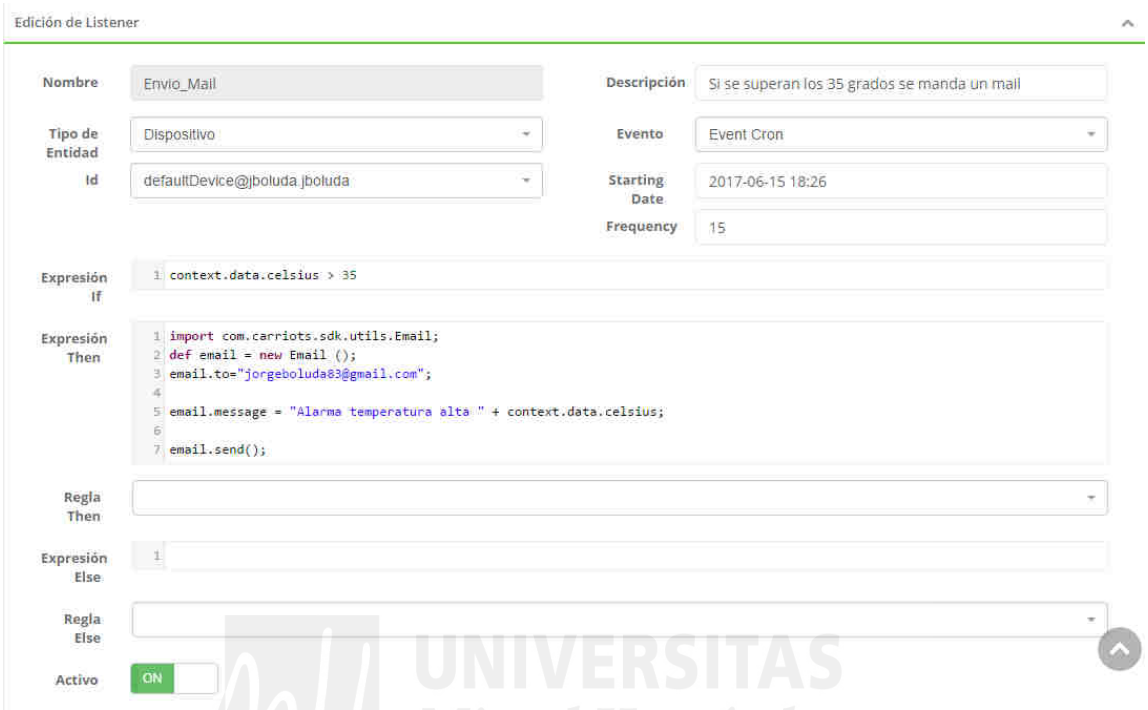
Desde el menú "Reglas" podremos configurar los "listeners" de Carriots. Gracias a esto podremos dotar a nuestro sistema de cierta lógica de control sin necesidad de que esta esté programada en los dispositivos de campo. De ese modo, podríamos cruzar datos de diferentes dispositivos para realizar una determinada acción

Listeners definidos en Carriots

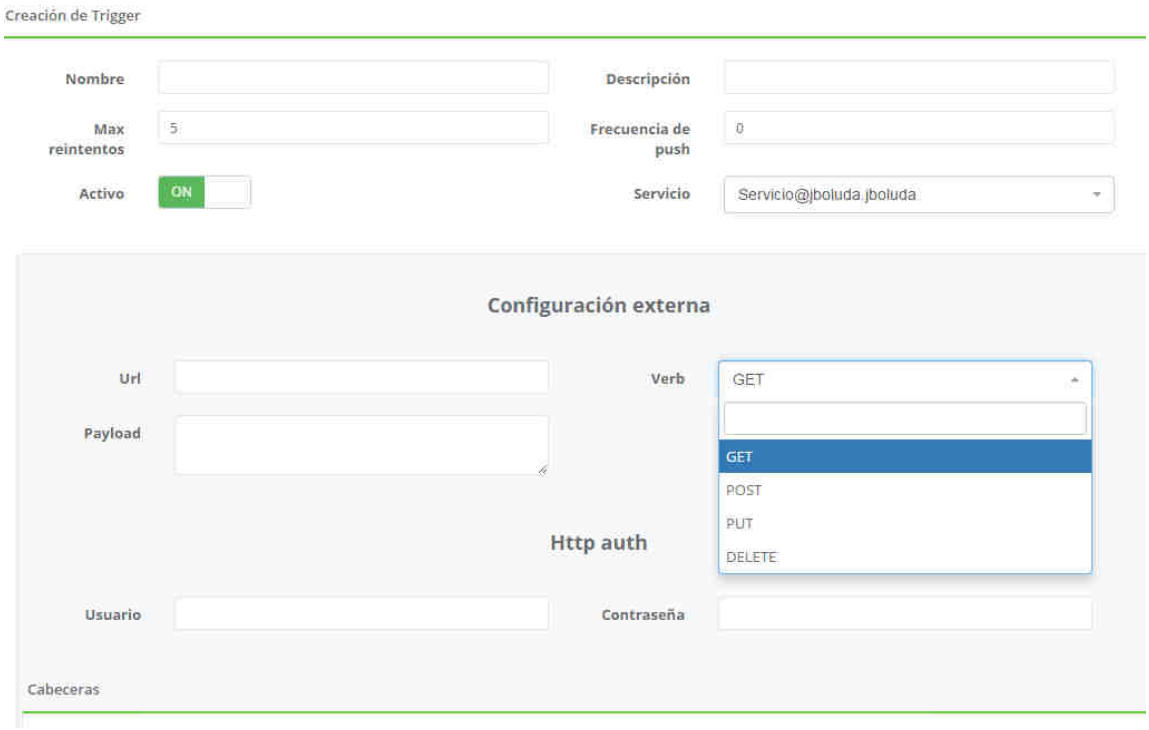
También tenemos la opción de crear reglas o funciones que utilicemos de forma habitual para no tener que escribir el mismo código varias veces cuando configuremos los "listeners". Estas reglas se pueden usar, por ejemplo, para almacenar el código necesario para el envío de un correo electrónico:

De modo que cuando configuremos un "listener" asociado a un equipo concreto podremos escribir el código que queremos que se ejecute cuando se cumpla una determinada

condición o bien especificar que queremos que se ejecute alguna de las reglas que hemos creado desde el panel de control:



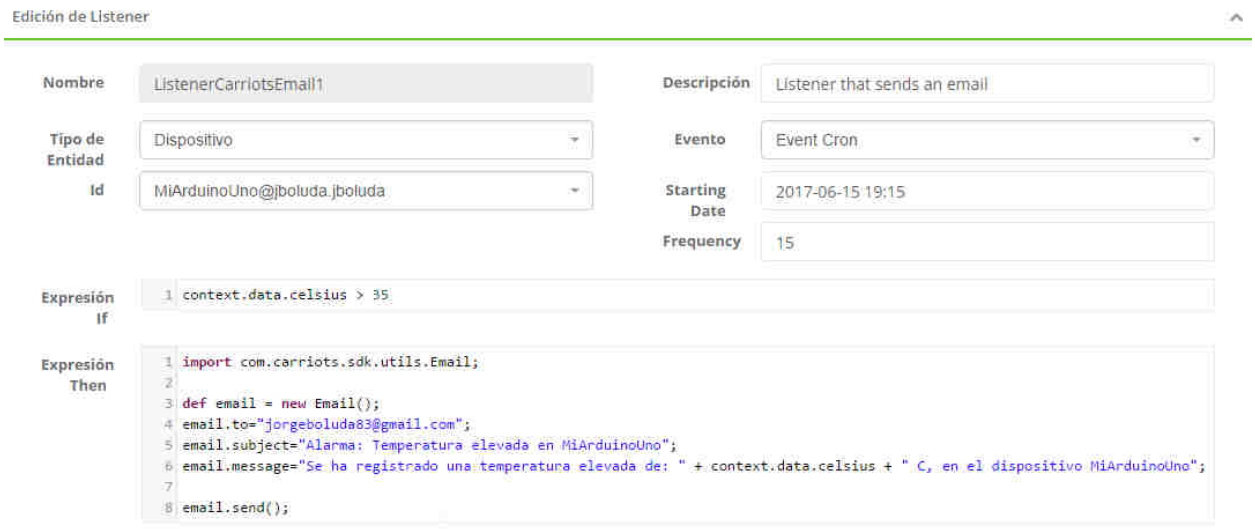
Por último este menú también nos proporciona la capacidad de crear “Triggers”. Con estos podremos enviar órdenes a nuestros dispositivos para solicitarles o enviarles información:



Envío de alarmas E-MAIL y SMS

A modo de ejemplo, se han realizado algunas pruebas desde la plataforma Carriots a la que se ha conectado el Arduino con sensor de temperatura mostrado en el apartado anterior, realizando el envío de una alarma de temperatura elevada tanto por correo electrónico como por SMS.

Para realizarlo por correo electrónico basta con configurar el siguiente “listener”:



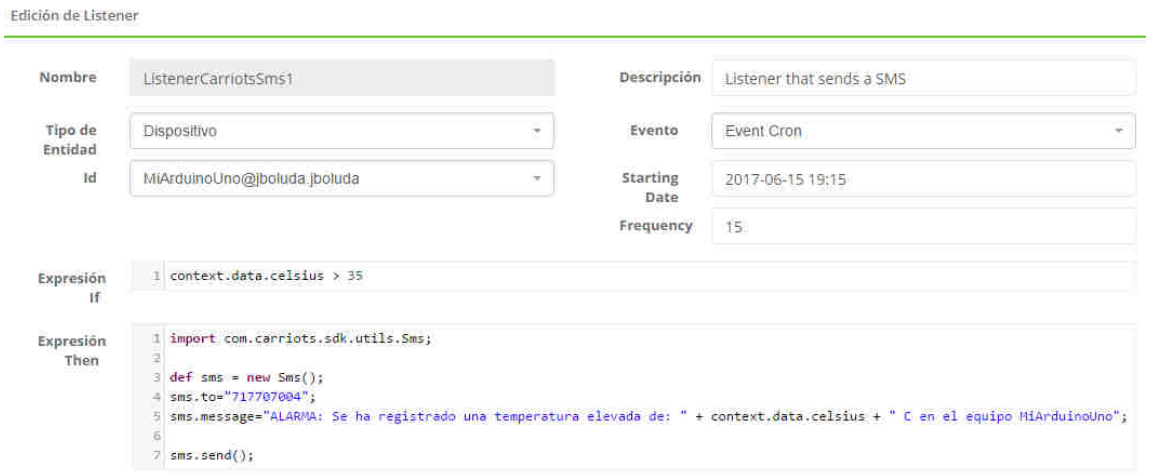
Cuando se cumpla la condición de "superados 35 ºC" que se comprueba con una frecuencia de 15 minutos, se enviará el correo electrónico.

El corre recibido es el siguiente:

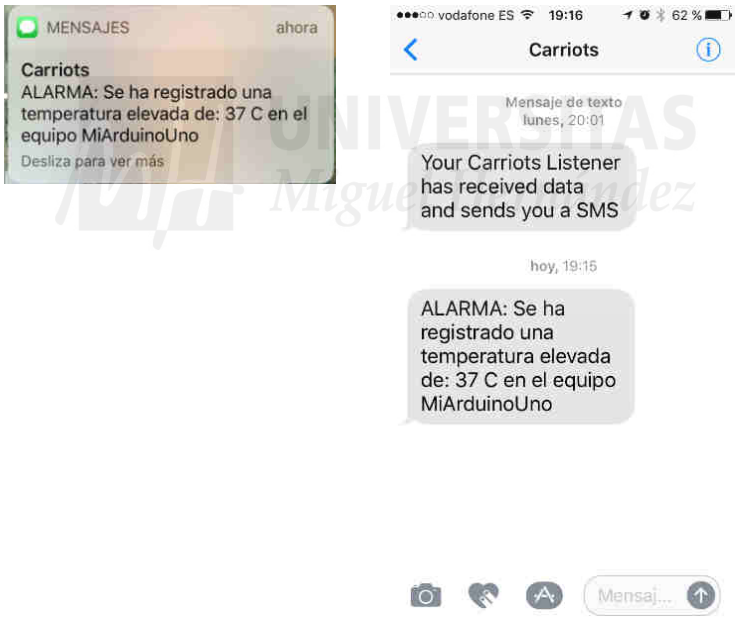
En lugar de un evento periódico ("Event Cron") también se pueden configurar otros eventos, como "cada vez que se reciba un dato" o "cada vez que cambie el estado de un dispositivo", con este último podríamos, por ejemplo, notificar cuando un equipo ha dejado de comunicar.

También, empleando las reglas mencionadas anteriormente podríamos crear diferentes destinatarios de correo para agilizar la configuración de las notificaciones en caso de que existan varios destinatarios posibles.

Podemos enviar la misma notificación por SMS en lugar de E-Mail, con las mismas peculiaridades comentadas, simplemente cambiando el código del "listener" de la siguiente forma:



El SMS recibido en el móvil indicado es el siguiente:



4.5.4.3. Integración con FreeBoard

Como se ha comentado en el apartado anterior, desde Carriots se podrían configurar paneles de usuario (Dashboard) pero estos son de pago y, aunque la plataforma en sí parece muy completa y nos proporciona muchas posibilidades, esta parte es relativamente nueva y no aporta grandes ventajas frente a otros paneles existentes en el mercado, algunos de ellos

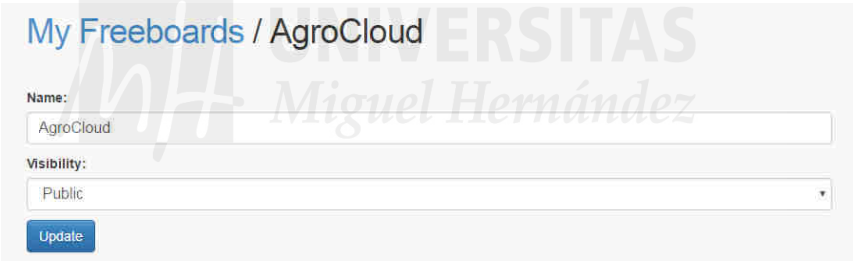
gratuitos como es el caso de FreeBoard.

FreeBoard es un Dashboard muy sencillo, en caso de ejecutar un proyecto real con mayores necesidades de representación y acceso a los datos por parte del usuario, Carriots ofrece muchas herramientas (empezando por su API REST) para realizar un programa o interfaz de usuario a medida.

A continuación se detallan los pasos que se han seguido para vincular FreeBoard con Carriots y mostrar los datos de temperatura y tensión registrados por el equipo Arduino de pruebas.

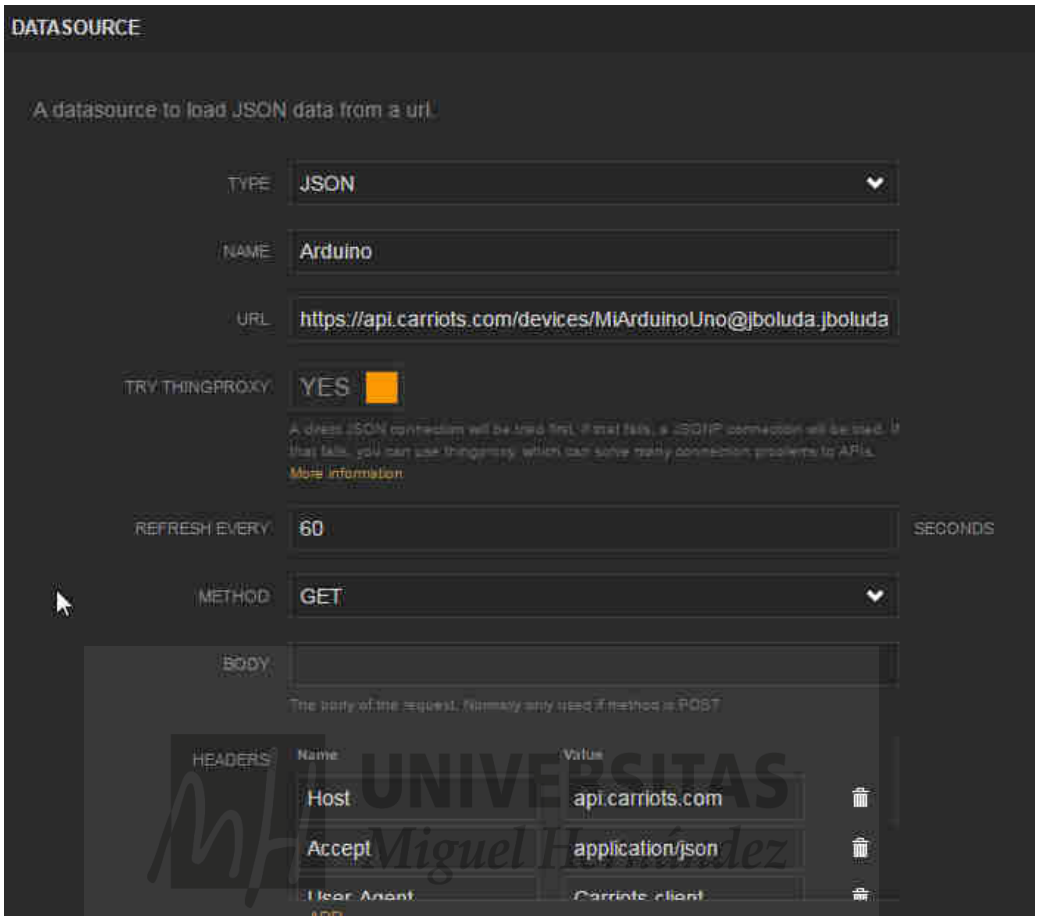
Creación de proyecto nuevo

Tras registrarnos en FreeBoard podremos acceder a nuestra cuenta y crear tantos proyectos como queramos. Recordar que la única limitación de la cuenta gratuita es que no podremos crear paneles privados (todos será públicos).



Añadimos un origen de datos

Para poder leer la información almacenada en Carriots deberemos crear un vínculo con su API REST. La forma de hacerlo con FreeBoard es relativamente sencilla, tan solo deberemos añadir un origen de datos con la siguiente configuración:



Como se observa en la imagen anterior, para conectarnos a la API RES de Carriots desde FreeBoard deberemos indicar la URL que nos proporciona acceso a nuestro proyecto en Carriots, en nuestro caso es la siguiente:

https://api.carriots.com/devices/MiArduinoUno@jboluda.jboluda/streams/?order=-1&max=1

Con esta URL estamos indicando a FreeBoard que queremos conectarnos al dispositivo "MiArduinoUno" de nuestro proyecto creado en Carriots para leer sus datos (streams), ordenándolos por fecha de más reciente y solicitando solo el último valor recibido.

También se indica que se desea utilizar el método GET para solicitar datos de tipo JSON y una serie de cabeceras que contienen información necesaria para poder establecer la conexión (entre ellas nuestra ApiKey de desarrollo):

- *Host - api.carriots.com*
- *carriots.apiKey - 8ba78245b0fdaf7bab27ecc241e0c6498036e8e0b9ca3dcdeb1d597c9972821f*
- *Accept - application/json*

- *User-Agent - Carriots-client*
- *Content-Type - application/json*

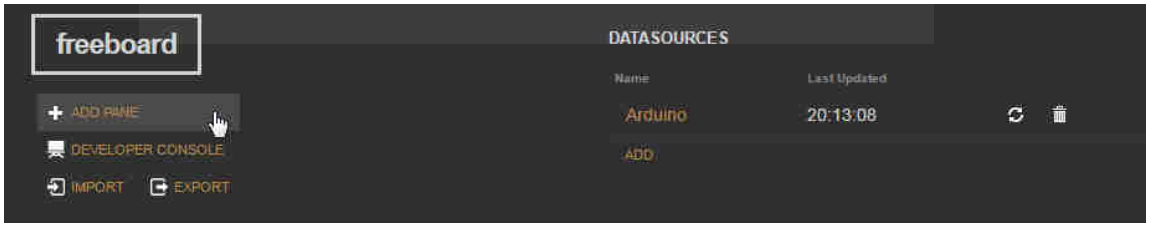
Lógicamente podremos añadir tantos orígenes de datos como necesitemos para enlazar con nuestro panel de usuario cada uno de los dispositivos de nuestro proyecto.

Configuramos el panel de usuario

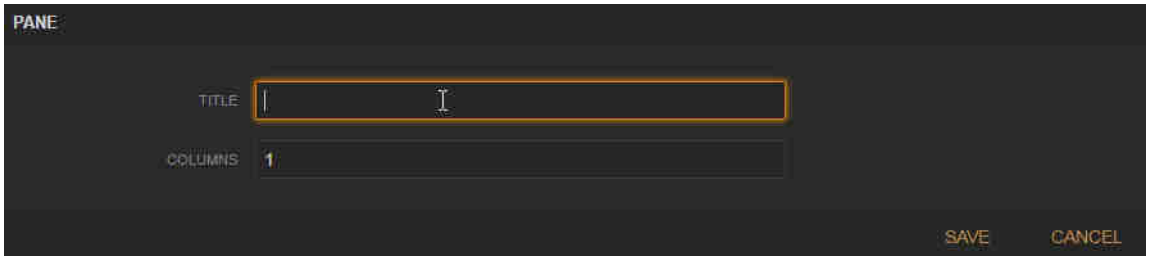
Una vez creado el origen el origen de datos para el Arduino de prueba, se procede a componer el panel de usuario añadiendo los widgets que nos ofrece esta solución, componiendo dicho panel de forma sencilla, arrastrando y soltando cada uno de los componentes en la posición que nos interese.

También nos ofrece una consola de desarrollo donde podríamos programar scripts y widgets personalizados.

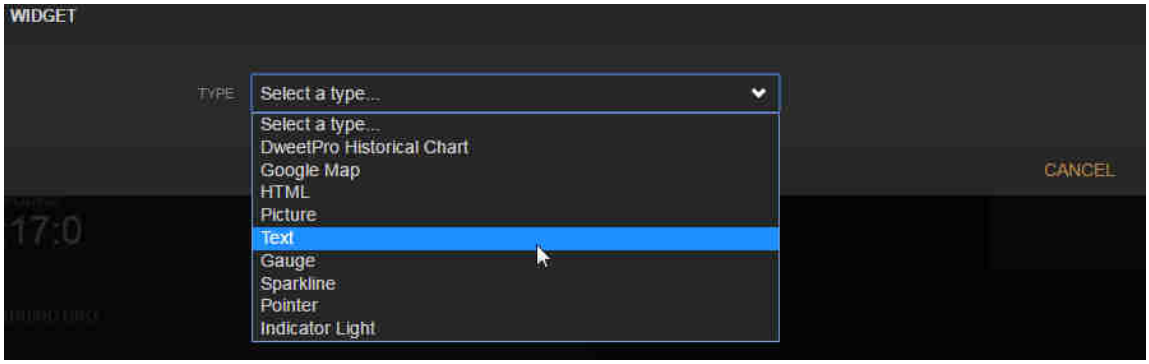
En primer lugar se añade un panel donde albergar widgets:



Se proporciona el título y las columnas que va a ocupar dicho panel:



Posteriormente se añaden los widgets que nos interesen al panel:



Por ejemplo podemos añadir un widget de tipo texto para mostrar el valor de temperatura registrado por el Arduino:

En lugar de indicar un origen de datos en el campo “value” es posible introducir código JavaScript, por ejemplo en nuestro ejemplo se ha realizado esto para mostrar la fecha del último valor recibido, para ello se emplea este código:

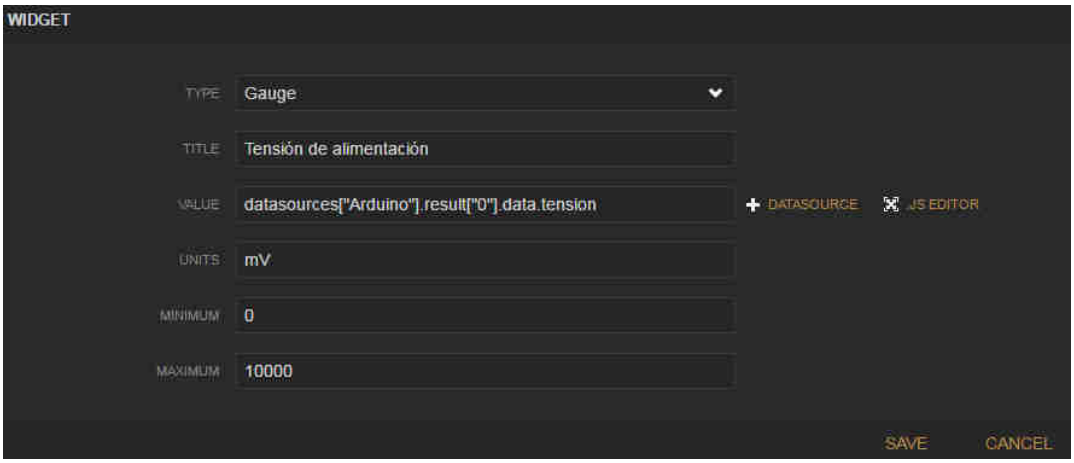
*var date=new Date(datasources["Arduino"].result["0"].at*1000);*

var months=["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];

return (date.getDate())+'/'+(date.getMonth()+1)+'/'+date.getFullYear().toString().substr(2,2)+'\t' +date.getHours()+':'+date.getMinutes()+':'+date.getSeconds();

Para elaborar el prototipo de panel de control de usuario, además del widget de texto se han

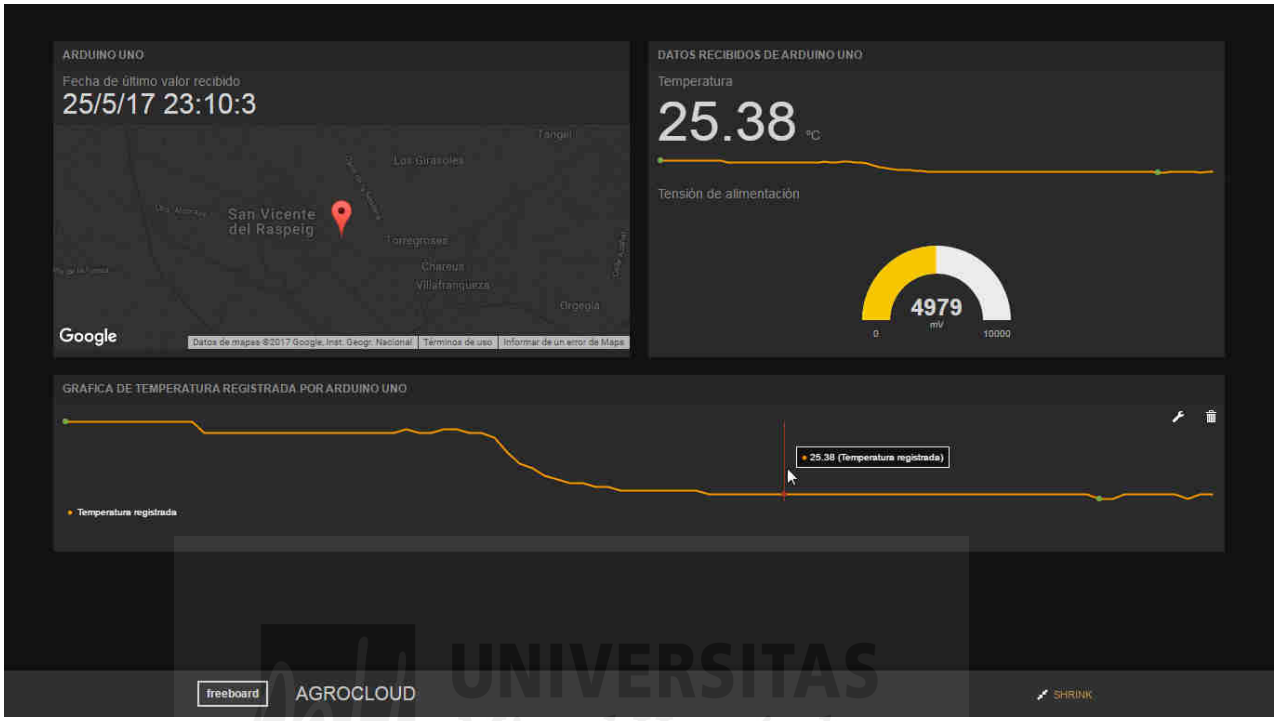
empleado otros como el de tipo “gauge” para representar el valor de tensión de alimentación:



El tipo “Google Map” para mostrar la ubicación del Arduino de pruebas:

O el tipo “Sparkline” para mostrar la curva de temperatura:

Utilizando estos paneles, funcionalidades y widgets se ha compuesto el siguiente panel de control de usuario para mostrar los datos del proyecto:



Panel de control diseñado con Freeboard

Este panel de control será accesible desde cualquier navegador web simplemente introduciendo la URL proporcionada por FreeBoard:

https://freeboard.io/board/63DpJJ

Otro widget interesante que ofrece FreeBord es el de “Picture” que nos permite ingresar la URL de una cámara (por ejemplo para monitorizar nuestra parcela):

5. Conclusiones

Se ha comprobado que para la gestión de un sistema de riego en parcela es viable utilizar una solución IoT formada por Carriots, Arduino y Freeboard. Además se han analizado otras tecnologías que también podrían formar parte de la solución.

El núcleo del sistema de la solución escogida es la plataforma Carriots, a la que se puede conectar casi cualquier dispositivo con la capacidad de acceder a Internet y, a su vez, es capaz de almacenar los datos procedentes de las diferentes fuentes para servirlos a otras aplicaciones mediante llamadas a su API Rest.

Para las pruebas se ha empleado una placa de Arduino como prototipo. Para llevar esta solución a una explotación real sería recomendable emplear equipos más robustos, sobre todo para aquellos equipos que se deban ubicar fuera del cabezal de riego. Aunque estos pueden seguir estando basados en Arduino deben contar con la capacidad de soportar condiciones meteorológicas adversas, como ejemplo se han mostrado los desarrollados por la empresa Libelium. Estos equipos podrían ubicarse en hornacinas, estando así más protegidos y seguros ante posibles actos vandálicos.

Por último, en cuanto a la parte de presentación de datos al explotador del sistema de riego, se ha mostrado como Freeboard puede ser una solución sencilla y gratuita para monitorizar una instalación de riego en tiempo real. Aunque, según las necesidades del explotador, se podría diseñar una aplicación de supervisión y control mucho más avanzada y completa que incluyese mejoras como las siguientes:

- Gestión de usuarios y notificaciones
- Calendarios de riego
- Control remoto automático y manual de la instalación
- Configuración de tramos horarios para regar cuando la energía es más barata
- Histórico de alarmas y eventos
- Visualización de gráficas e informes más potentes
- Etc.

Para implementar todo esto se podría hacer un desarrollo web responsivo, como se ha definido en el apartado 0 Diseño web responsivo. Como se ha descrito en este proyecto, Carriots ofrece herramientas que simplificarían mucho el desarrollo de una aplicación con estas características (gracias a su API, SDK y el resto de servicios que ofrece), quedando totalmente resuelta la parte de comunicación con los equipos de campo.

6. Bibliografía

Internet of Things (IoT):

[1] https://es.wikipedia.org/wiki/Internet_de_las_cosas

[2] https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube

[3]https://rodrigoconz.com/2015/01/22/internet-de-las-cosas-una-forma-sencilla-de-entender-la-oportunidad-tecnologica/

22 enero, 2015 por rodrigoconz

[4]http://www.enertic.org/ActualidadDesarrollo?param1=1448&param2=Claves%20de%20la%20evoluci%C3%B3n%20de%20las%20nuevas%20tecnolog%C3%ADas%20en%20los%20pr%C3%B3ximos%20a%C3%B1os

20 de diciembre, 2013 por enerTIC

[5] https://opentechdiary.wordpress.com/tag/internet-of-things/

16 agosto, 2015

[6] https://www.genbetadev.com/programacion-en-la-nube/entendiendo-la-nube-el-significado-de-saas-paas-y-iaas

31 de agosto, 2012 por Txema Rodríguez

[7] https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

[8] https://es.wikipedia.org/wiki/Front-end_y_back-end

[9] https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable

[10] http://docs.oracle.com/cd/E19957-01/820-2981/ipov-10/index.html

[11] https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[12] http://www.isa.uniovi.es/docencia/redes/protocolos_seguros.pdf

Presentación Universidad Oviedo Junio 2009

[13] http://postscapes.com/internet-of-things-software-guide#protocols

Plataformas IoT:

[14] https://thingspeak.com/

[15] https://www.carriots.com/

[16] https://electricimp.com/

[17] http://www.blaulabs.com/es/

[18] https://iot.telefonica.com/thinking-things

Dashboards:

[19] http://tristanelosegui.com/2014/10/27/que-es-y-para-que-sirve-un-dashboard/

[20] https://freeboard.io/

[21] https://www.clicdata.com

[22] https://www.dashzen.com/

[23] https://mydevices.com/

Dispositivos IoT:

[24] http://www.arduino.cc/

[25] http://www.libelium.com/es

[26]https://www.kickstarter.com/projects/sparkdevices/spark-core-wi-fi-for-everything-arduino-compatible/posts

[27] https://polaridad.es/compara-arduino-esp8266/

[28] https://www.sparkfun.com/products/13678

Otros:

[29] https://www.cooking-hacks.com/documentation/tutorials/arduino-solar/

[30] https://www.cooking-hacks.com/forum/viewtopic.php?f=35&t=3066

[31] http://www.maswifi.com/blog/2012/05/redes-mesh-que-son-y-como-funcionan/

[32] Apuntes de la asignatura "Automatización de comunidades de regantes: infraestructura hidráulica y sistemas de bombeo" del presente Máster.

[33] https://dzone.com/articles/iot-software-platform-comparison

4 febrero, 2016 por Miyuru Dayarathna

[34] http://iotfunda.com/top-15-popular-iot-platforms-2016-complete-list/

17 agosto, 2016 por Nivedita Agnihotri

[35] http://logic.sysbiol.cam.ac.uk/teaching/Node-RED_IoT-platform_Test.pdf

Abril, 2014 por @BorisAdrian

[36] Jornada práctica del Máster sobre Arduino impartida por Martin John Oates (2017)

ANEXO I: Índice de figuras

Gráfica de crecimiento de apartados conectados a internet ____ *8*
Estimación distribución del IoT según sectores industriales ____ *9*
Capas de una arquitectura IoT ____ *10*
Arquitectura de una aplicación web ____ *11*
Diseño web responsivo ____ *12*
Capas del modelo TCP/IP ____ *13*
Mecanismos de seguridad en las distintas capas TCP/IP ____ *15*
Limitaciones y precios de Carriots según el tipo de cuenta de usuario ____ *20*
Módulos de Thinking Things ____ *27*
Ejemplo panel FreeBoard ____ *32*
Ejemplo panel Clicdata ____ *32*
Ejemplo panel Dashzen ____ *33*
Ejemplo panel Cayenne ____ *34*
Detalle de placa Arduino Uno ____ *35*
Diferentes modelos del módulo ESP8266 ____ *39*
Módulo regulador de 5V para Arduino con batería Li-Ion de 2300mAh ____ *41*
Panel solar ____ *42*
Arduino alimentado mediante batería y placa solar ____ *42*
Shield Ethernet para Arduino ____ *44*
ESP8266 ____ *45*
Shield Wifi para Arduino ____ *45*
Módulo de conexión GPRS/GSM para Arduino, Raspberry Pi o Intel Galileo ____ *46*
Arduino y módulo GPRS/GSM alimentado mediante batería y placa solar ____ *46*
Precios FreeBoard ____ *51*
Foto del montaje: conexión de sensor de temperatura 1-Wire a Arduino ____ *53*
Pantalla inicial del panel de control de Carriots ____ *55*
Listado de Apikeys ____ *55*
Geolocalización de dispositivos y Jerarquía de Carriots ____ *57*
Listado de dispositivos ____ *58*
Tramas de datos recibidas en Carriots ____ *59*
Gráfica generada con el configurador de Carriots ____ *60*
Listeners definidos en Carriots ____ *61*
Panel de control diseñado con Freeboard ____ *71*

ANEXO II: Programa Arduino para conexión con Carriots

```
#include <SPI.h>
#include <Ethernet.h>
#include <OneWire.h>

OneWire  ds(7); // en el pin7 se conectará el sensor 1-wire
float celsius;
float voltaje;

// Datos la cuenta Carriots
const String APIKEY = "8ba78245b0fdaf7bab27ecc241e0c6498036e8e0b9ca3dcdeb1d597c9972821f";
// Carriots apikey
const String DEVICE = "MiArduinoUno@jboluda.jboluda"; // Id_developer

// Dirección MAC de la shield ethernet conectada al Arduino
byte mac[] = {  0x90, 0xA2, 0xDA, 0x00, 0x79, 0xE9 };

IPAddress ip(192,168,1,177);     // Dirección IP asignada
IPAddress server(82,223,244,60);  // Dirección IP de api.carriots.com

EthernetClient client; // Inicia la libreria

void setup() {
  Serial.begin(9600);           // Se inicia el puerto serie
  Serial.println(F("Starting"));
  Ethernet.begin(mac);        // Se inicia el puerto serie
  delay(1000);                // Esperamos 1 segundo para dar tiempo a que se establezca la conexión
}

void loop() {
  //Variables para la lectura 1-Wire:
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];

 //Leemos la temperatura 1-Wire
 if ( !ds.search(addr)) {
    ds.reset_search();
    delay(250);
  }
  ds.reset();
```

```
  ds.select(addr);
  ds.write(0x44, 1);
  delay(1000);
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  int16_t raw = (data[1] << 8) | data[0];
  byte cfg = (data[4] & 0x60);
  if (cfg == 0x00) raw = raw & ~7;
  else if (cfg == 0x20) raw = raw & ~3;
  else if (cfg == 0x40) raw = raw & ~1;

  //Se guarda el valor de temperatura que se mandará a Carriots
  celsius = (float)raw / 16.0;
  Serial.print("Temp = ");
  Serial.print(celsius);
  Serial.println(" degrees C");
  delay(500);

  //Leemos el voltaje de alimentacion del arduino
  #if defined(__AVR_ATmega32U4__) || defined(__AVR_ATmega1280__) ||
defined(__AVR_ATmega2560__)
    ADMUX = _BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #elif defined (__AVR_ATtiny24__) || defined(__AVR_ATtiny44__) || defined(__AVR_ATtiny84__)
    ADMUX = _BV(MUX5) | _BV(MUX0);
  #elif defined (__AVR_ATtiny25__) || defined(__AVR_ATtiny45__) || defined(__AVR_ATtiny85__)
    ADMUX = _BV(MUX3) | _BV(MUX2);
  #else
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  #endif
  delay(2); // Wait for Vref to settle
  ADCSRA |= _BV(ADSC); // Start conversion
  while (bit_is_set(ADCSRA,ADSC)); // measuring
  uint8_t low  = ADCL; // must read ADCL first - it then locks ADCH
  uint8_t high = ADCH; // unlocks both
  long result = (high<<8) | low;
  result = 1125300L / result; // Calculate Vcc (in mV); 1125300 = 1.1*1023*1000
  Serial.print("Vcc = ");
  Serial.print(result);
  Serial.println(" mV");

  //Se guarda el valor del voltaje para mandarlo a Carriots
  voltaje = result;
```

```
  //Llamada a la función que envía las tramas a Carriots
  sendStream();

  delay(10000); //añadido para dar tiempo a leer la respuesta sin tener que esperar al siguiente minuto

  // Si hay datos recibidos a través de la conexión de red se imprimen por el puerto serie (para debug)
  while (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
  if (!client.connected()) {
     client.stop();
  }

  //Retardo para no llegar al límite de 1000 tramas por día (al usar la cuenta gratuita de Carriots)
  delay(50000);
}

void sendStream()
{
  if (client.connect(server, 80)) {   // Si la conexión es correcta
    Serial.println(F("connected"));
    // Se compone la trama json que se enviará a carriots con los datos leídos de temperatura y voltaje
    String                                    json                                    =
"{\"protocol\":\"v2\",\"device\":\""+DEVICE+"\",\"at\":\"now\",\"data\":{\"temperatura\":\""+celsius+"
\",\"tension\":\""+voltaje+"\"}}";
    // HTTP request
    client.println("POST /streams HTTP/1.1");
    client.println("Host: api.carriots.com");
    client.println("Accept: application/json");
    client.println("User-Agent: Arduino-Carriots");
    client.println("Content-Type: application/json");
    client.print("carriots.apikey: ");
    client.println(APIKEY);
    client.print("Content-Length: ");
    int thisLength = json.length();
    client.println(thisLength);
    client.println("Connection: close");
    client.println();

    client.println(json);
  }
  else {
    // Si la conexión falla
    Serial.println(F("connection failed"));
  }
}
```