Trabajo Fin de Máster realizado por

JUAN LUIS DEL CAMPO COLLAR

para la obtención del título de

Máster en Ingeniería de Automatización e Informática Industrial

"DISEÑO Y DESARROLLO DE UN SISTEMA INTELIGENTE PARA LA GESTIÓN DE UN ACUARIO"

Tutor: **JOSÉ MARÍA ENGUITA GONZÁLEZ**

Junio de 2019

Resumen:

El objetivo de este trabajo fin de grado consiste en la realización de un prototipo a escala que permita la evaluación de las tecnologías IoT (Internet of Things) para el control, monitorización, automatización y supervisión de un acuario, ofreciendo una solución de bajo coste. Se trata de recoger, enviar y recibir datos en tiempo real sin que haya una intervención humana directa. La recopilación, el almacenamiento y la gestión de los datos nos van a permitir monitorizar y automatizar actividades a partir de la información obtenida. Un posterior análisis de la información capturada nos va a facilitar la mejora de la eficacia de nuestro sistema. La gestión de la información junto con la inteligencia artificial (IA) nos permite tomar decisiones, prever hechos y el envío de avisos.

Agradecimientos

Quiero agradecer primero a todas las personas que forman parte del Máster en Ingeniería de Automatización e Informática Industrial de la Universidad de Oviedo, porque de una u otra manera me han proporcionado las herramientas académicas para poder llegar a donde estoy hoy.

A mi esposa e hijo: Ana y Pelayo, quienes han tenido toda la paciencia por el tiempo que estado ausente.

Y en especial, agradecer a mi tutor académico José María Enguita González su tiempo y esfuerzo para la realización de este trabajo.

"El que da, no debe volver a acordarse; pero el que recibe nunca debe olvidar"

Proverbio judío

Contenido

Índice de las ilustraciones

Índice de las tablas

1. Introducción

1.1. Identificación del Trabajo Fin de Master

Título	Diseño y desarrollo de un sistema inteligente para la gestión de un acuario
Tutor	José María Enguita González
Alumno	Juan Luis Del Campo Collar
D.N.I.	09.379.226 - X
Fecha	Mayo de 2019

1.2. Estructura del documento

La memoria del Trabajo de Fin de Master realizado consta de los siguientes apartados:

- **Apartado 1. Introducción**. Es la introducción del trabajo e incluye la identificación del trabajo, estructura del documento, planteamiento, objetivos y alcance.
- **Apartado 2. Análisis y selección de materiales.** Estudio, análisis y selección de los principales elementos (sensores, actuadores, etc.) utilizados en este proyecto.
- **Apartado 3. Desarrollo del proyecto**. Requisitos y modos de funcionamiento del proyecto. Esquemas de conexión de los dispositivos hardware y descripción de las herramientas software utilizadas. Descripción del prototipo realizado.
- **Apartado 4. Pruebas.** Comprobaciones realizadas para ver si el funcionamiento del sistema es correcto.
- **Apartado 5. Planificación y presupuesto.** Se muestran la planificación de las tareas principales necesarias para la realización de este proyecto y el presupuesto del trabajo.
- **Apartado 6. Conclusiones y posibles mejoras.** Se presentan las conclusiones el trabajo y una serie de ampliaciones para realizar en un futuro con el fin de mejorar más el sistema.
- **Apartado 7. Referencias**. Listado de la documentación utilizada para el desarrollo del proyecto.
- **Apartado 8. Anexos**. Programación en Arduino. Configuración de la plataforma Thinger.io. Configuración plataforma IFTTT. Código fuente del Arduino.

1.3. Planteamiento

Actualmente estamos viviendo la 4ª Revolución industrial, la Revolución computacional, en donde la tecnología Cloud, la inteligencia artificial y el Internet de las Cosas (IoT) son los puntales principales. Para el año 2025 está previsto que el número de objetos conectados a internet será de unos 75 billones, como se muestra en la Ilustración 1.

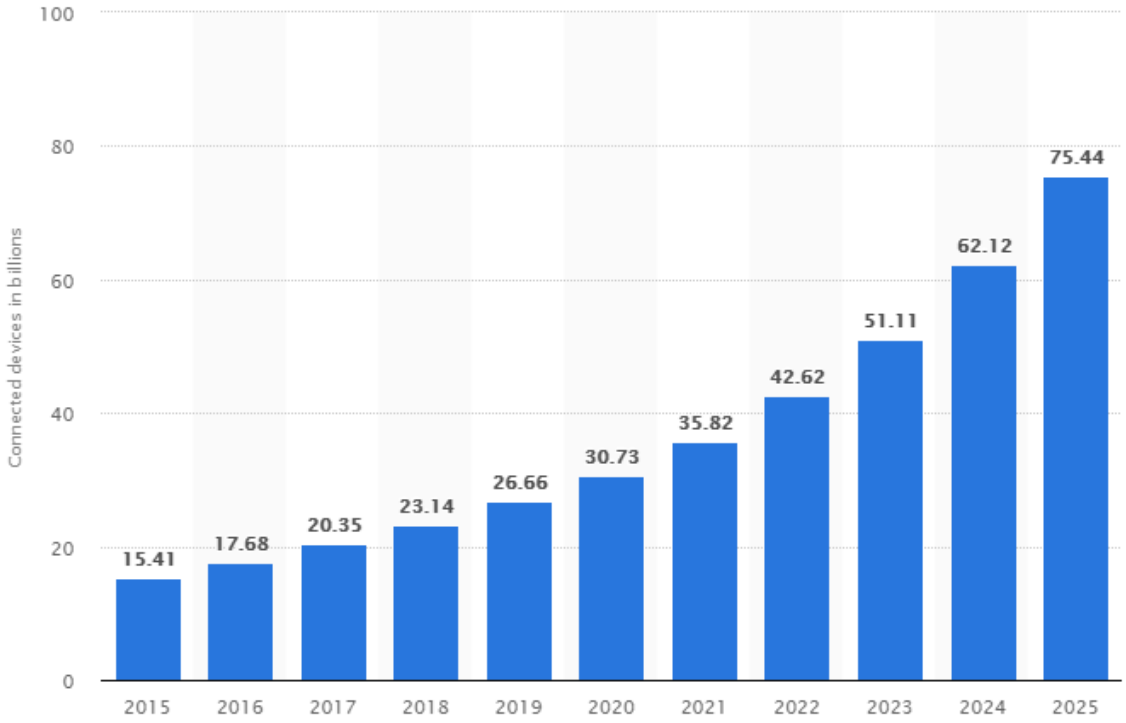


Ilustración 1 Crecimiento del número de dispositivos conectados
Fuente: https://www.hiberus.com/crecemos-contigo/introduccion-al-iot-internet-of-things/

El IoT con una idea de integración total busca conectar dispositivos, incluso personas, en tiempo real y aprovechar esa conectividad para recibir y enviar datos desde y hacia los dispositivos. En este trabajo, queremos estudiar cómo se incorporan los acuarios a este mundo del IoT.

Actualmente, se construyen espectaculares acuarios (1) algunos con más de 30.000 m^3 de agua y miles de animales marinos, como por ejemplo el acuario de Georgia (Atlanta - EEUU https://www.georgiaaquarium.org/) que podemos observar en la ilustración 2.

Ilustración 2 Georgia Acuarium
Fuente: http://canalviajes.com/georgia-aquarium-el-acuario-mas-grande-del-mundo/

Sin embargo, no solo se construyen como centro de atracción turística, sino que ya en muchas instalaciones comerciales o de negocios incorporan un acuario como elemento decorativo para atraer gente, como el AquaDom (https://www.visitsealife.com/en/berlin/), un acuario cilíndrico con un millón de litros de agua salada y que está dentro del hotel Radissom, en Berlín, el cual vemos en la ilustración 3.

Ilustración 3 Berlin's AquaDom
Fuente: *https://www.aplaneticketandreservations.com/2017/01/12/aquadom-sea-life-berlin-germany/*

Pero no solo se construyen grandes acuarios. También tienen un campo de uso en las viviendas particulares donde, además de por su uso decorativo, son usados por la sensación de paz y tranquilidad que transmiten, provocando una sensación muy relajante. Vemos en la ilustración 4 un ejemplo de acuario doméstico.

Ilustración 4 Acuario doméstico
Fuente: *https://www.pinterest.es/pin/401172279293762776/*

1.4. Objetivo

Este trabajo fin de máster tiene como objetivo la realización de un prototipo a escala que permita la evaluación de las tecnologías IoT (Internet of Things) para el control, monitorización, automatización y supervisión de un acuario, ofreciendo una solución de bajo coste. Se trata, no solo de conectar el dispositivo a la red y que muestre los valores de sus sensores. Se necesita recoger, enviar y recibir datos en tiempo real sin que haya una intervención humana directa. Que exista una programación que permita al dispositivo realizar su función, comunicarse y tomar decisiones en función de la información recibida, y que los actuadores reaccionen a esas decisiones. La recopilación, el almacenamiento y la gestión de los datos nos van a permitir monitorizar y automatizar actividades a partir de la información obtenida. Un posterior análisis de la información capturada nos va a facilitar la mejora de la eficacia de nuestro sistema. La gestión de la información junto con la inteligencia artificial (IA) nos permite tomar decisiones, prever hechos y el envío de avisos.

Este proyecto nos permite el estudio de estas nuevas tecnologías que se encuentran en expansión y que se considera de gran importancia en el momento actual, y por otra, para profundizar en otras tecnologías vistas durante la realización del máster.

1.5. Alcance

El alcance de este proyecto es crear un sistema, que se pueda instalar de forma sencilla en una vivienda familiar, que controle los parámetros de un acuario, y que permita al usuario controlar y monitorizar, desde cualquier parte en la que disponga de conexión a internet, el estado del acuario.

Para ello se necesita:

- Diseño, desarrollo e implementación de un sistema inteligente hardware, utilizando un microcontrolador, sensores y actuadores.
- Comunicación con los sensores para recibir la información que estos capturan.
- Comunicación con los actuadores para que puedan ejecutar las órdenes.
- Comunicación entre el hardware y la plataforma de internet.
- Diseñar una interfaz gráfica en la plataforma escogida.
- Gestionar la información, para poder consultar el estado actual del acuario.
- Almacenar los datos recibidos para poder ser procesados y analizados posteriormente.
- Visualizar gráficamente la evolución de los datos.
- Configurar los parámetros principales de control del acuario.
- Ejecutar ordenes de marcha/paro de los actuadores.
- Programación horaria de alguno de los actuadores.
- Automatización del cambio de agua.
- Recibir alertas y manejo remoto del sistema a través del teléfono.

Para la simulación del trabajo vamos a utilizar un acuario casero, que se dispone, de dimensiones 50 x 45 x 30 cm y una capacidad de 60 l. Además, intentaremos utilizar siempre que sea posible, los sensores o actuadores existentes. En la ilustración 5 se muestra el acuario y diversos componentes que se van a utilizar en este proyecto.

Ilustración 5 Acuario a utilizar
Fuente: *Propiedad del autor*

1.6. Antecedentes

Ya en época de los romanos era costumbre tener un Aquarium que se utilizaba para almacenar peces comestibles y así disponer de comida fresca. También disponían de estanques e incluso de contenedores de arcilla con los que transportaban los peces al mercado. Posteriormente, en la Edad Media, los monasterios disponían de estanques y charcas para la cría de peces de agua dulce, mientras que en China y Japón los tenían como peces ornamentales en sus casas y jardines. En Europa se empezaron a poner de moda a partir de mediados del s. XIX. En esta época el principal inconveniente era el cambio constante del agua y fue el naturalista ingles **Sir Henry Hamilton Johnston** (https://en.wikipedia.org/wiki/Harry_Johnston) quien resolvió el problema hacía 1850. Desde ese momento se empezaron a construir los acuarios de salón, inaugurando la Sociedad Zoológica de Londres en 1853 el acuario del jardín de Regent´s Park y en 1862 se inauguró el acuario del bosque de Boloña en París. La aparición de los acuarios en las casas se produce a partir de la Primera Guerra Mundial con la llegada a estas de la electricidad (2). Desde entonces su uso como adorno se ha difundido por todo el mundo, existiendo acuarios en muchos hogares actuales.

La existencia de mascotas en casa hace que nos preocupemos para que tengan una buena calidad de vida, lo que nos obliga diariamente a ocuparnos de su alimentación y de los cuidados necesarios, para que disfruten de buena salud. Por ello, la automatización de algunos de estos cuidados nos descarga de trabajo y nos deja más tiempo libre. En el caso de los acuarios podemos utilizar la tecnología para controlar la iluminación, la alimentación, la temperatura del agua, nivel del agua, el pH, etc. La automatización nos va a permitir ausentarnos del hogar sabiendo que nuestras mascotas van a estar bien cuidadas.

1.7. Sistemas comerciales actuales

Existen diversas empresas que ofrecen productos para el control de acuarios como, por ejemplo:

- **ProFiLux GHL Iberia** (3)**.** Dispone de diferentes gamas de dispositivos para el control total del acuario, dosificadores, iluminación, etc. Conexión WLAN compatible. Servidor web. Cliente de correo electrónico. En la ilustración 6 podemos ver este controlador. Se puede ampliar la información en su página web http://www.profilux.es/#

Ilustración 6 ProfiLux 3.1T
Fuente: http://www.profilux.es/profilux-3-1t/

- **Aquatronica** (4). Permite manejar todos los equipos eléctricos y electrónicos que estén conectados al acuario. Se compone de una unidad de control, una unidad de potencia, interfaces y sensores. Permite controlar el acuario a través de una conexión Wifi. En la ilustración 7 podemos ver este controlador. Se puede ampliar la información en la página web https://www.aquatronica.com/

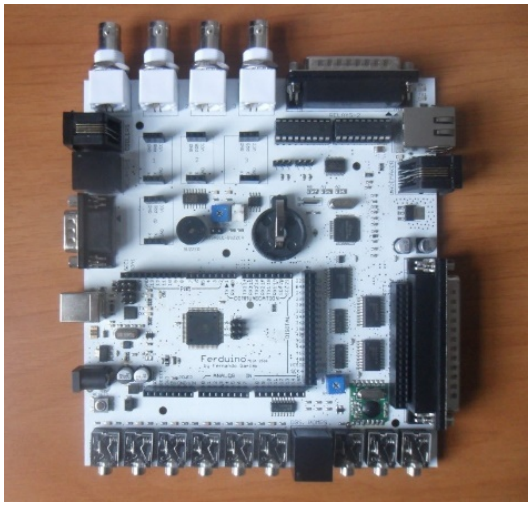
Ilustración 7 Unidad de control ACQ130
Fuente: https://www.aquatronica.com/pc_index.php?L=ita&PAGE=home&OTP=/

- **Neptune** (5)**.** Su nuevo sistema Apex con WIFI permite el control de acuarios y un servicio gratuito en la nube de Apex Fusion. En la ilustración 8 podemos ver este controlador. Su página web es https://www.neptunesystems.com/#

Ilustración 8 Plataforma ApexEL
Fuente: https://www.neptunesystems.com/apexel/

Además de los sistemas comerciales anteriores, existen variedad de proyectos open source entre los cuales destacan Jarduino y Ferduino, los dos sobre el microcontrolador Arduino Mega. Jarduino originalmente es para el control de riego, pero es fácil adaptar el código para el control de acuarios.

En la ilustración 9 vemos un módulo Ferduino. Podemos encontrar más información sobre estos proyectos en https://code.google.com/archive/p/jarduino-aquarium-controller/ y https://www.ferduino.com/



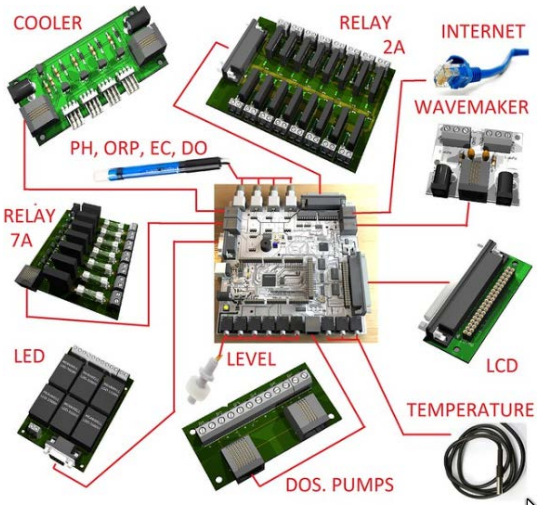


Ilustración 9 Ferduino
Fuente: https://www.ferduino.com/

Teniendo en cuenta lo visto anteriormente, se puede decir que actualmente en el mercado existen diversas soluciones para el control de acuarios. Todas estas marcas ofrecen un control completo del acuario, ofreciendo exactitud y sencillez en la instalación, siendo su principal problema el alto coste, con precios superiores a los 700 €, que incluso en algunos casos se ve incrementado con la adición de módulos opcionales, sensores y actuadores específicos, etc. por lo que estos productos no entran dentro del rango de nuestro proyecto.

En el caso de los sistemas open source las placas rondan los 180 euros y lo que nos proporcionan es un entorno integrado del Arduino mega 2560, el módulo ethernet W5100 y diferentes conectores para bombas, sensores de nivel y de temperatura, que nos facilitan la labor de integrar los distintos componentes pues solo hay que conectarlos. Sin embargo, no ofrecen ninguna capacidad IoT.

Por todo ello, podemos concluir que estos sistemas no se adaptan a nuestras especificaciones.

1.8. Descripción del sistema

El sistema a desarrollar consta de un acuario al que se le dota de sensores, para capturar los valores de los principales parámetros del sistema y de actuadores para su control. La interacción con los sensores y actuadores se realiza a través de un dispositivo IoT. La comunicación bidireccional con la nube se hace a través de un router. La información se almacena en la nube en una plataforma IoT. La plataforma IoT nos va a permitir visualizar, analizar y compartir la información.

En la ilustración 10 podemos ver una descripción gráfica del sistema a desarrollar en este proyecto.

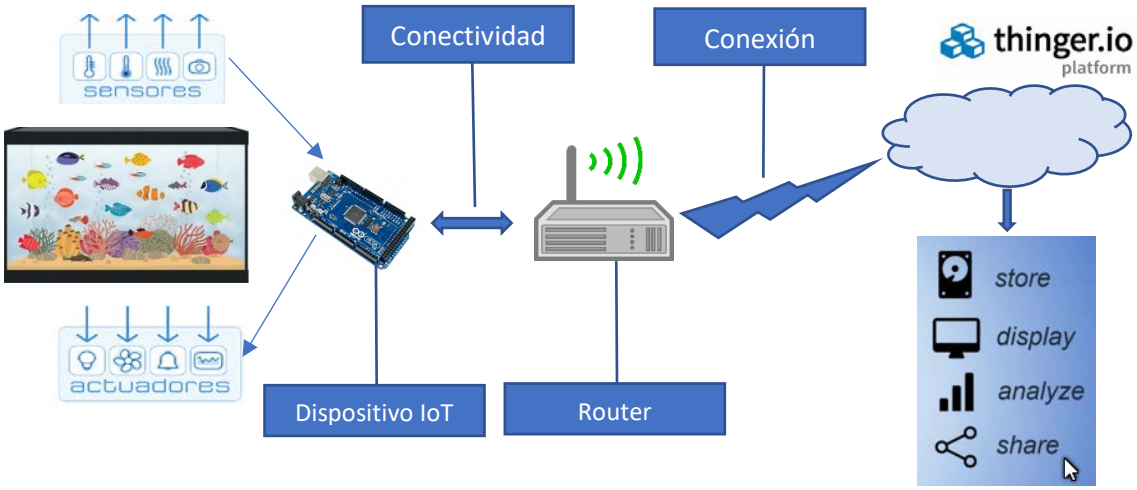


Ilustración 10 Descripción del modelo
Fuente: *Elaboración propia*

En los siguientes apartados se va a ir viendo parte por parte cada uno de los elementos que forman parte de este sistema.

La función principal del sistema es la automatización de las tareas de mantenimiento y control que se realizan en los acuarios para el correcto cuidado de los peces y plantas que viven en él.

Para ello necesitamos conocer los parámetros clave para el correcto mantenimiento del acuario (6), y son:

- **Temperatura:** La temperatura óptima para un acuario de agua dulce ronda por lo general los 24 - 25 ºC, aunque hay que tener en cuenta los peces que se van a introducir pues existen diferencias en función de las especies, pudiendo esta variar desde los 21 a 29 ºC. Una temperatura incorrecta del agua hace que sus colores no sean tan brillantes e incluso pude llegar a debilitar el sistema inmunológico aumentando el riesgo de contraer infecciones bacterianas.
- **pH:** El pH es la abreviatura de potencial de hidrógeno y se expresa en grados. El pH tiene una escala que va de 1 a 14, siendo 1 el valor más ácido y 14 el más alcalino. Los peces tropicales de agua dulce necesitan un pH próximo a 7, pero al igual que ocurre con la temperatura, cada especie puede necesitar un pH específico, normalmente entre 6 y 8.
- **Calidad del agua:** Es necesario la existencia de un buen sistema de filtrado y de cambios del agua que nos garantice una buena calidad del agua.
- **Iluminación:** Para las plantas, la iluminación tanto en calidad como en intensidad es un factor importante. El acuario necesitará de 9 a 12 horas de luz diaria.
- **KH (dureza), GH (dureza total), NH_4 (Amonio), NH_3 (Amoniaco), NO_2 (nitritos) y NO_3 (nitratos):** Es importante medir estos parámetros que nos indican la calidad del agua una vez a la semana (7). El KH es el resultado de los

compuestos de calcio y magnesio con ácido carbónico. Ayudan a estabilizar el pH. El valor ideal de KH es entre 2 y 4º. El GH es la concentración de sales de calcio y magnesio en el agua. Influye en el crecimiento de los peces, plantas, bacterias y microorganismos. El NH_4 y el NH_3 determinan la cantidad de amonio/amoniaco que es altamente tóxico para la vida del acuario. El NO_2 nos determina la cantidad de nitrito el cual es tóxico. El NO_3 determina la cantidad de nitrato (por transformación del nitrito) y es asimilable por las plantas. El análisis semanal de estos parámetros lo haremos manualmente a través de unas tiras reactivas. Por lo que no entran dentro del ámbito de este trabajo.

2. Análisis y selección de materiales

A continuación, vamos a presentar un estudio del mercado actual sobre los materiales que necesitamos para la realización del proyecto.

2.1. Sensores

Los sensores son unos dispositivos que recogen la información del entorno, la transforman en una señal eléctrica y la envían al sistema de control, es decir, reaccionan a los cambios de una variable física y la transforman en una señal eléctrica que transmiten a un dispositivo lógico.

En el acuario vamos a disponer de cinco sensores, tres para medir parámetros claves como son la temperatura (dos sensores) y el pH y dos para controlar el nivel correcto del agua (nivel mínimo y máximo).

En el apartado 2.9. se describen los sensores concretos utilizados en este proyecto.

2.2. Actuadores

Los actuadores nos permiten interactuar con el sistema teniendo en cuenta los datos obtenidos por los sensores. Dado que la mayoría de los actuadores que vamos a utilizar trabajan con tensiones e intensidades superiores a las suministradas por la placa Arduino Mega va a ser necesario utilizar relés.

En el acuario vamos a disponer de ocho actuadores, tres bombas de agua, dos lámparas, un comedor, un calentador y un aireador.

En el apartado 2.10. se describen los actuadores concretos utilizados en este proyecto.

2.3. Dispositivo IoT

Internet es una red de computadoras conectadas a lo largo de todo el mundo. Todo elemento que queramos conectar a internet debe de contar con algún equipo electrónico que le permita la lectura y el envío de datos a internet.

Dado las características del proyecto a realizar la mejor opción es utilizar un microcontrolador / microcomputadora. Estos dispositivos constan de un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Nos permite controlar elementos de entrada/salida. Disponen en su interior de una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos.

Dentro de los microcontroladores encontramos:

Arduino Uno R3 (8)

Se trata de una tarjeta de desarrollo open-source basada en un microcontrolador Atmega328 que dispone de 14 pines de entrada/salida digitales, de las cuales 6 pueden ser utilizadas como salidas PWM, 6 entradas analógicas, velocidad de reloj 16 MHz, 32 K de memoria flash y voltaje de entrada de 7 – 12 V. Se puede comunicar a través del puerto serial a un Pc. En la ilustración 11 podemos ver esta tarjeta. Se puede ampliar información sobre Arduino en su página web https://www.arduino.cc/

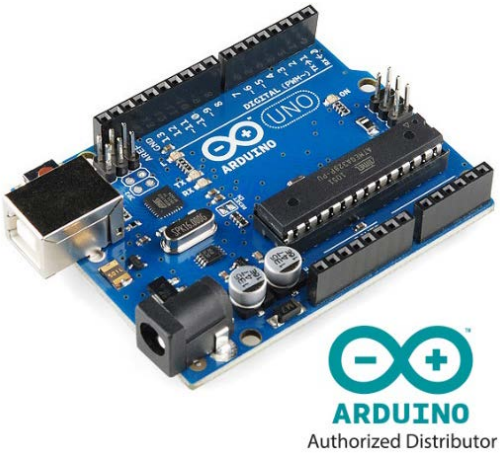


Ilustración 11 Arduino Uno
Fuente: https://arduino.cl/arduino-uno/

Arduino Mega 2560 (8)

Se trata de una tarjeta de desarrollo open-source basada en un microcontrolador Atmega2560 que dispone de 54 pines de entrada/salida digitales, de las cuales 14 pueden ser utilizadas como salidas PWM, 16 entradas analógicas, velocidad de reloj 16 MHz, 256 K de memoria flash y voltaje de entrada de 7 – 12 V. Se puede comunicar a través del puerto serial a un Pc. En la ilustración 12 podemos ver esta tarjeta. Se puede ampliar información sobre Arduino en su página web https://www.arduino.cc/

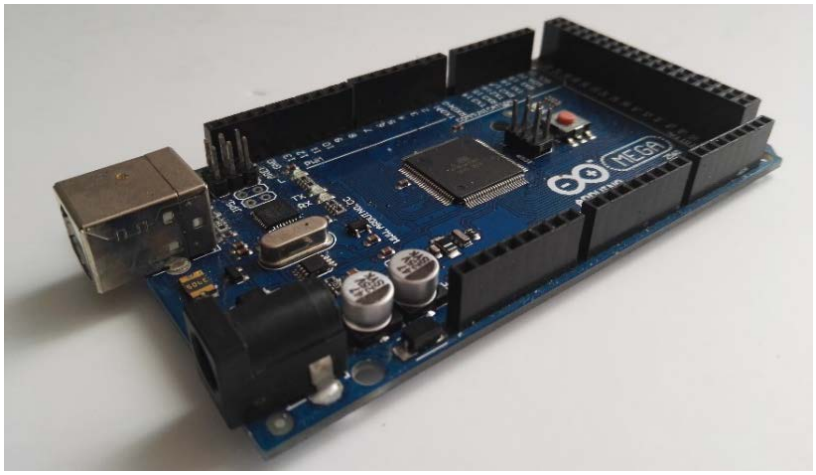


Ilustración 12 Arduino Mega
Fuente: Propiedad del autor

Arduino Yún (8)

Se trata de una tarjeta de desarrollo open-source basada en un microcontrolador Atmega32u4 que dispone de 14 pines de entrada/salida digitales, 7 salidas PWM, 6 entradas analógicas, velocidad de reloj 16 MHz, 32 K de memoria flash y voltaje de entrada de 5V. Tiene un módulo Wifi y salida ethernet. Soporta memoria microSD. En la ilustración 13 podemos ver esta tarjeta. Se puede ampliar información sobre Arduino en su página web https://www.arduino.cc/

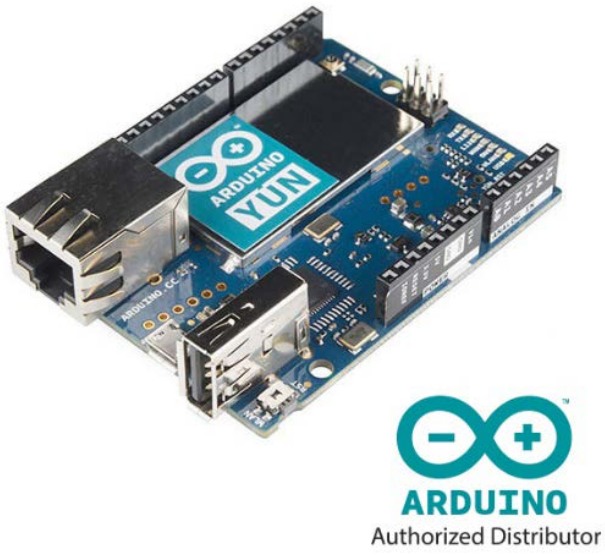


Ilustración 13 Arduino Yun
Fuente: https://arduino.cl/arduino-yun/

D1 mini Pro V2.0.0

Se trata de una placa de desarrollo basada en un microcontrolador ESP8266EX que dispone de 11 pines de entrada/salida digitales, 1 entrada analógica, velocidad de reloj 80/160 MHz, 16 MB de memoria flash y voltaje de entrada de 3,3 V. Dispone de un módulo Wifi. En la ilustración 14 podemos ver esta tarjeta. Se puede ampliar información en su página web https://www.wemos.cc/

Ilustración 14 D1 mini Pro 2.0.0
Fuente: https://wiki.wemos.cc/products:d1:d1_mini_pro#pin

Waspmote

Es una plataforma modular open-source para construir redes de sensores inalámbricas de muy bajo consumo. Basada en un microcontrolador ATmega 1281 que dispone de 8 pines de entrada/salida digitales, 7 entradas analógicas, velocidad de reloj 80/160 MHz. 128 K de memoria flash y voltaje de entrada de 5V. Comunicación inalámbrica con el protocolo Zigbee con alcances de hasta 40Km. Dispone de módulos opcionales para añadir comunicación Bluetooth, GPRS y GPS. Variedad de placas de sensores. Integra en la propia placa un sensor de temperatura y un acelerómetro. En la ilustración 15 podemos ver esta tarjeta. Se puede ampliar información en su página web http://www.libelium.com/

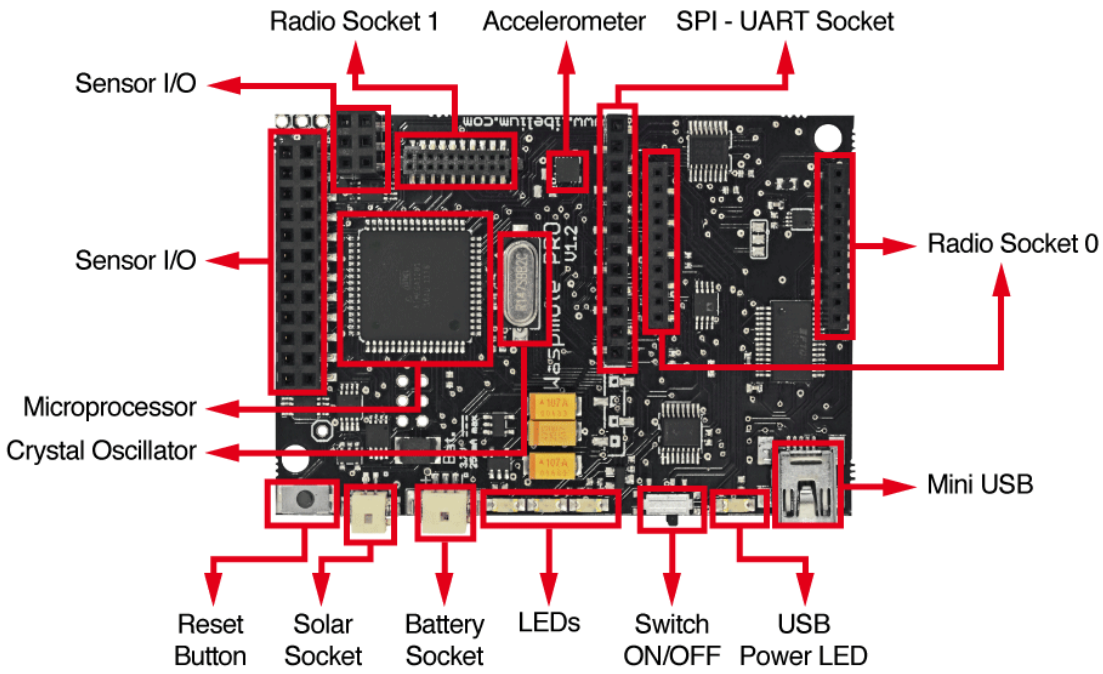


Ilustración 15 Waspmote
Fuente: http://www.libelium.com/products/waspmote/hardware/

Además de las placas mencionadas, existen multitud de empresas con plataformas de microcontroladores que ofrecen una funcionalidad similar como son Parallax Basic Stamp (https://www.parallax.com/microcontrollers/basic-stamp), Netmedia (http://www.basicx.com/), Phidgets (https://www.phidgets.com/?tier=3&catid=2&pcid=1&prodid=3), MIT (http://web.mit.edu/16.070/www/indexhb.html), etc.

Dentro de los microcomputadores encontramos:

Raspberry Pi 3 Model B+

El Raspberry Pi es el micro-ordenador más popular gracias a relación prestaciones/costo. Las prestaciones y características del nuevo Raspberry Pi 3 Modelo B+ son:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4 GHz y 5 GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- GPIO de 40 pines
- HDMI
- 4 puertos USB 2.0
- Puerto CSI y DSI para conectar una cámara y una pantalla táctil
- Salida de audio estéreo y vídeo compuesto
- Micro-SD

En la ilustración 16 podemos ver esta tarjeta. Se puede ampliar información en su página web https://www.raspberryshop.es/

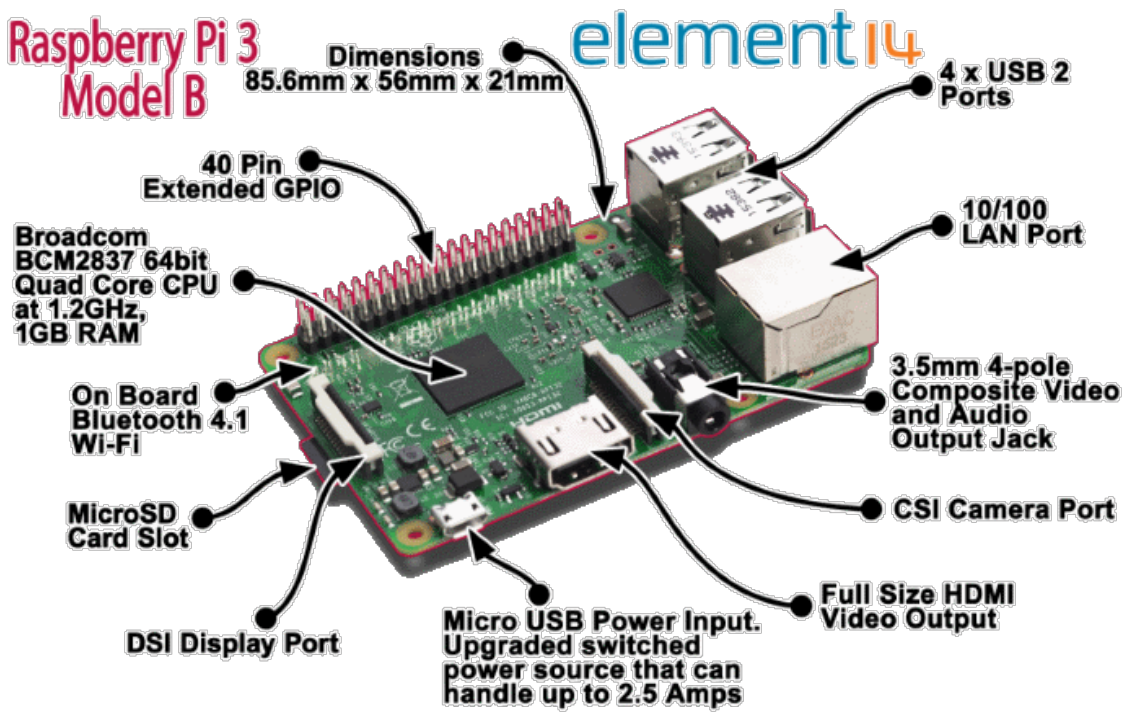


Ilustración 16 Raspberry PI 3 Model B+
Fuente: https://www.raspberryshop.es/raspberry-pi-3.php

Banana Pi m64

Es un dispositivo de plataforma abierta. Sus características principales son:

- 1,2 GHz Quad-Core ARM Cortex A53 de 64 bits
- 2 GB SDRAM DDR3
- Ethernet 10/100/1000 Mbps
- Wifi
- Bluetooth BT4.0
- Micro-SD
- 2 Puertos USB 2.0
- GPIO de 40 pines
- Receptor infrarrojos

En la ilustración 17 podemos ver esta tarjeta. Se puede ampliar información en su página web http://www.banana-pi.org/m64.html

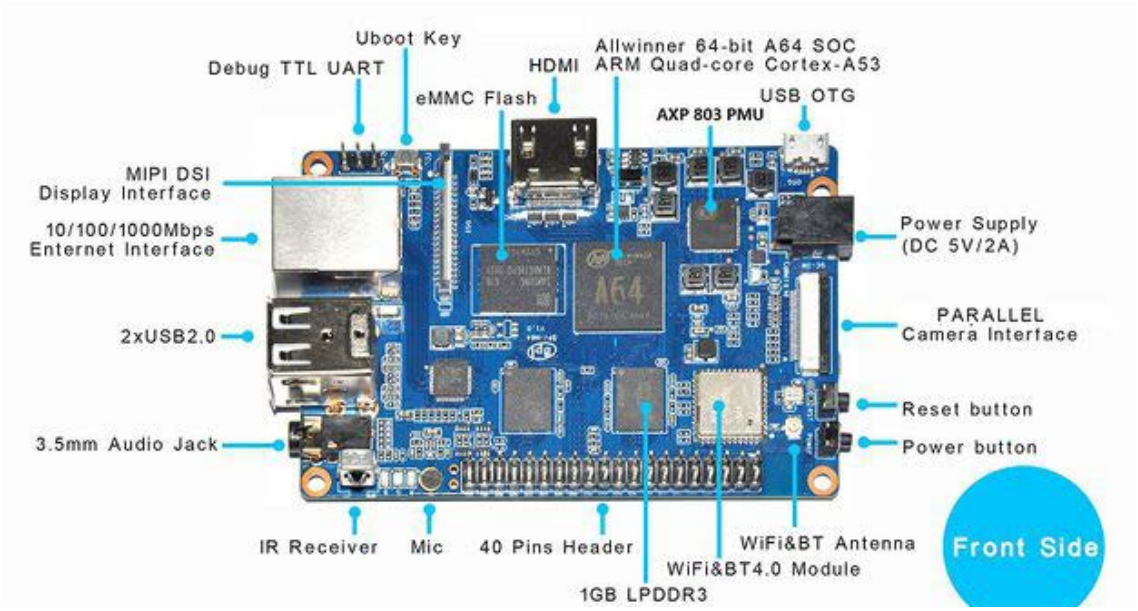


Ilustración 17 Banana PI m64
Fuente: http://wiki.banana-pi.org/File:BPI-M64_interface.jpg

ODROID-C2

Es un dispositivo ARM. Sus características principales son:

- 1,5 GHz Quad-Core Amlogic ARM Cortex A53 de 64 bits
- 2 GB SDRAM DDR3
- Ethernet 10/100/1000 Mbps
- Wifi
- Micro-SD
- 4 Puertos USB 2.0
- GPIO de 40 pines
- Receptor infrarrojos

En la ilustración 18 podemos ver esta tarjeta. Se puede ampliar información en su página web https://www.hardkernel.com/shop/odroid-c2/

Ilustración 18 Odroid-C2
Fuente: https://www.hardkernel.com/shop/odroid-c2/

Asus Tinker Board

Sus características principales son:

- 1,8 GHz Quad-Core Rockchip RK3288 ARM
- 2 GB SDRAM LPDDR3
- Ethernet 10/100/1000 Mbps
- Wifi
- Bluetooth
- Micro-SD
- 4 Puertos USB 2.0
- GPIO de 40 pines

En la ilustración 19 podemos ver esta tarjeta. Se puede ampliar información en su página web https://www.asus.com/us/Single-Board-Computer/Tinker-Board/

Ilustración 19 Asus Tinker Board
Fuente: https://www.asus.com/uk/Single-Board-Computer/Tinker-Board/gallery/

Además de las placas mencionadas, se pueden encontrar otras como la Pine A64 (https://www.pine64.org/), Orange Pi 3 (http://www.orangepi.org/), NanoPi-NEO-Plus2 (http://www.nanopi.org/), etc.

Una vez analizadas la variedad de opciones existentes en el mercado, creemos que para iniciarse en este campo las dos grandes opciones son Arduino y Raspberry pi. En principio, aunque en su aspecto puedan parecer muy similares su funcionalidad es muy distinta. La diferencia principal es que el Arduino es un microcontrolador y la Raspberry pi es un miniordenador. Esto implica que Arduino ofrece menos prestaciones, pero por el contrario simplifica el proceso de aprendizaje. Para este proyecto decidimos utilizar la plataforma de Arduino por considerarla más adaptada para la realización de proyectos de electrónica. Arduino ofrece una gran versatilidad a bajo coste. Además, dispone de multitud de shields que le permiten aumentar su campo de actuación y de un único IDE para todos los Sistema Operativos. Existe para Arduino multitud de software adicional, como por ejemplo software de programación en entornos gráficos, de diseño de circuitos, etc. que aumentan sus posibilidades de uso. Las principales claves del éxito de Arduino que la convierten en la plataforma más extendida se pueden ver en la ilustración 20.

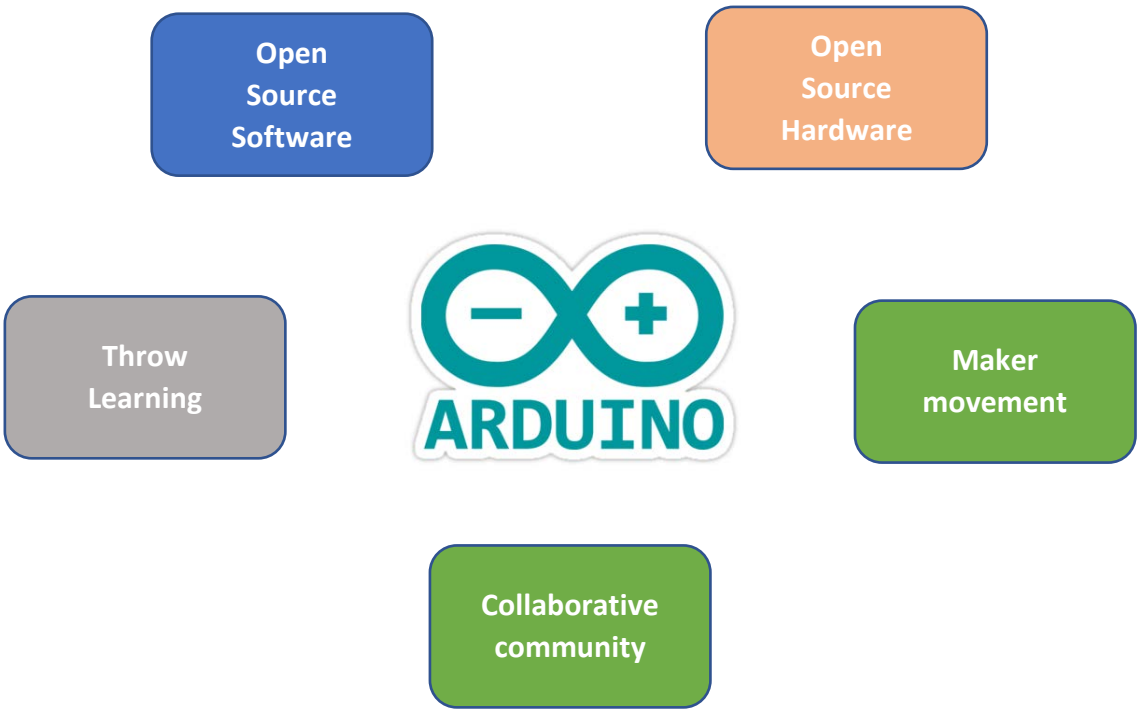


Ilustración 20 Las claves de Arduino
Fuente: Curso experto en internet de las cosas con Thinger y Arduino (UNED)

Dentro de Arduino vemos que existen multitud de modelos, siendo los más utilizados el Arduino Uno y el Arduino Mega. Para este proyecto se ha elegido el Arduino Mega 2560 rev3, que está diseñado para proyectos más complejos, por la cantidad de entradas y salidas que dispone lo cual permite posibles ampliaciones del proyecto en un futuro. En (9) se puede ver un análisis comparativo de las placas Arduino (oficiales y compatibles).

2.4. Comunicaciones con Arduino

Una vez que tenemos los sensores, los actuadores y el Arduino Mega 2560, vamos a ver la forma de comunicarse entre ellos. En la ilustración 21 se muestra la distribución de los puertos de comunicación del Arduino Mega 2560.

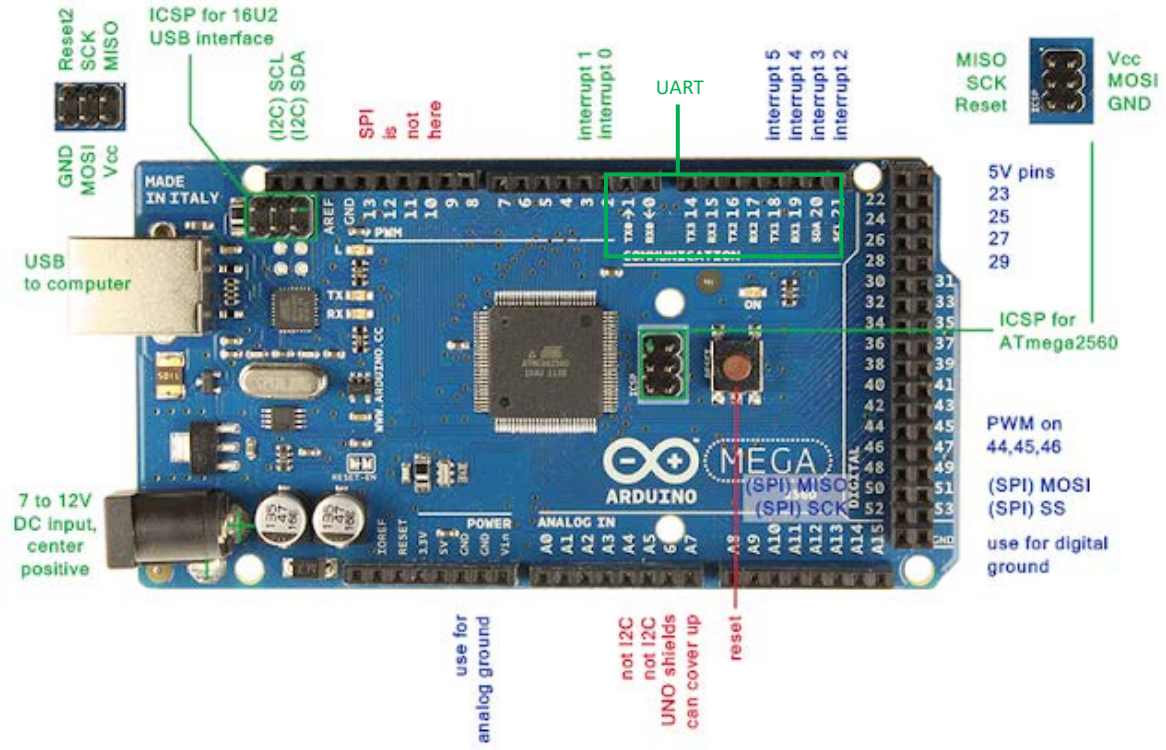


Ilustración 21 Puertos de comunicación del Arduino Mega
Fuente: http://ingenio-triana.blogspot.com/2017/01/comunicacion-con-arduino-usb-serie-i2c.html

Dentro de la comunicación serie, en Arduino Mega tenemos:

UART. Universal Asynchronous Receiver-Transmitter, es el dispositivo que controla los puertos y dispositivos serie. Usa una línea de datos simple para transmitir y otra para recibir datos. Es una comunicación asíncrona y la velocidad de datos está limitada a 2Mbps. Comúnmente, 8 bits de datos son transmitidos de la siguiente forma: un bit de inicio, a nivel bajo, 8 bits de datos y un bit de parada a nivel alto. El Arduino Mega 2560 dispone de cuatro puertos serie (Serial 0: 0 (RX) y 1 (TX); Serial 1: 19 (RX) y 18 (TX); Serial 2: 17 (RX) y 16 (TX); Serial 3: 15 (RX) y 14 (TX)). En este proyecto no se va a utilizar.

USB tipo B. Universal Serial Bus. El USB es utilizado como estándar de conexión de periféricos. La conexión USB la vamos a utilizar para conectarnos al IDE de Arduino que nos permite compilar el código y subirlo a la placa. Dentro del interfaz de Arduino la opción “serial monitor” nos posibilita la visualización de datos procedentes de la tarjeta.

SPI. Es un bus síncrono. Un maestro envía la señal de reloj, y tras cada pulo de reloj envía un bit al esclavo y recibe un bit del esclavo. Las señales que utiliza son SCK para el reloj, MOSI para enviar datos a los periféricos, MISO para enviar datos al maestro y

SS para la selección del esclavo. Comunicación Full Duplex. En la ilustración 22 podemos ver el esquema de esta conexión.

En este proyecto vamos a utilizar el protocolo SPI para la conexión del shield Arduino Ethernet, que utiliza el pin 4 como SS (Slave Select). Para usar el bus SPI en Arduino tenemos la librería "SPI.h" que dispone de las funciones necesarias para su control.

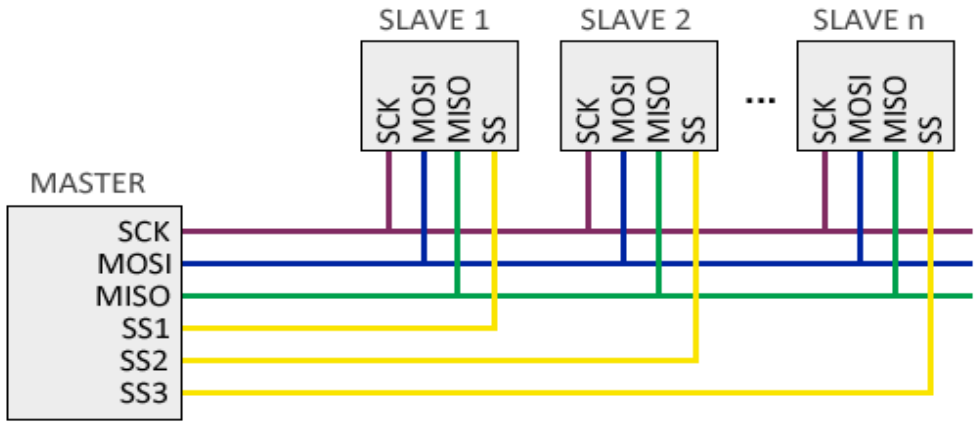


Ilustración 22 Bus SPI
Fuente: https://www.luisllamas.es/arduino-spi/

I²C. Es un protocolo síncrono que solo utiliza dos cables, SCL para el reloj y SDA para el dato (10). El maestro controla un único cable por donde se envía y se reciben los datos y proporciona la señal de control que mantiene sincronizados a los dispositivos. Para el control de los esclavos se utiliza el direccionamiento, por ello los dispositivos conectados al bus tienen una dirección única. Dos o más señales a través del mismo cable pueden causar conflicto, y ocurrirían problemas si un dispositivo envía un 1 lógico al mismo tiempo que otro envía un 0. Por ello el bus es "cableado" con dos resistencias para poner el bus a nivel alto, y los dispositivos envían niveles bajos. Si quieren enviar un nivel alto simplemente lo comunican al bus. A pesar de que se usan dos líneas para transmitir la información es necesaria una tercera línea como referencia (masa). Se limita su uso a entornos de poca interferencia. En la ilustración 23 podemos ver el esquema de esta conexión.

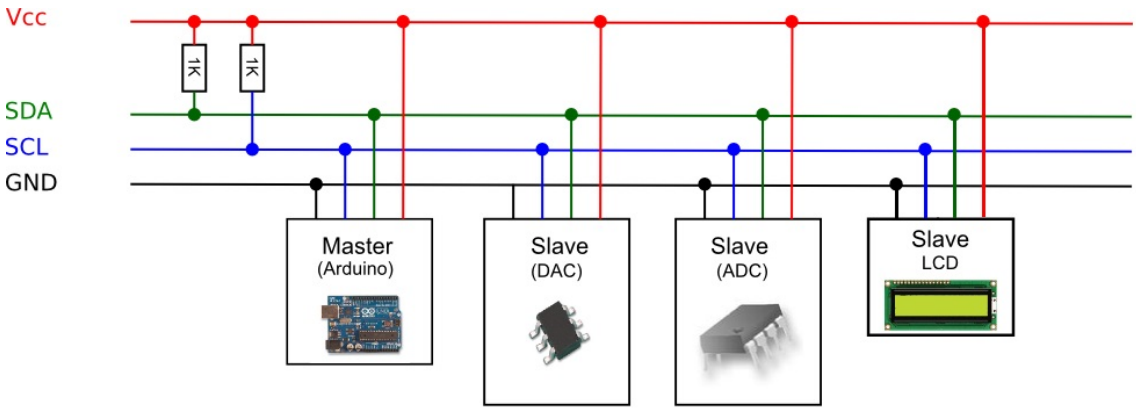


Ilustración 23 Bus I2C
Fuente: https://www.prometec.net/bus-i2c/

Podemos encontrar la especificación y el manual de usuario de I^2C en http://www.nxp.com/documents/user_manual/UM10204.pdf y como usar el bus I^2C en http://www.i2c-bus.org/fileadmin/ftp/i2c_bus_specification_1995.pdf

El protocolo I^2C nos permite conectar varios Arduinos entre sí, ampliando el número de entradas, como vemos en la ilustración 24, o a otros periféricos como la Raspberry Pi.

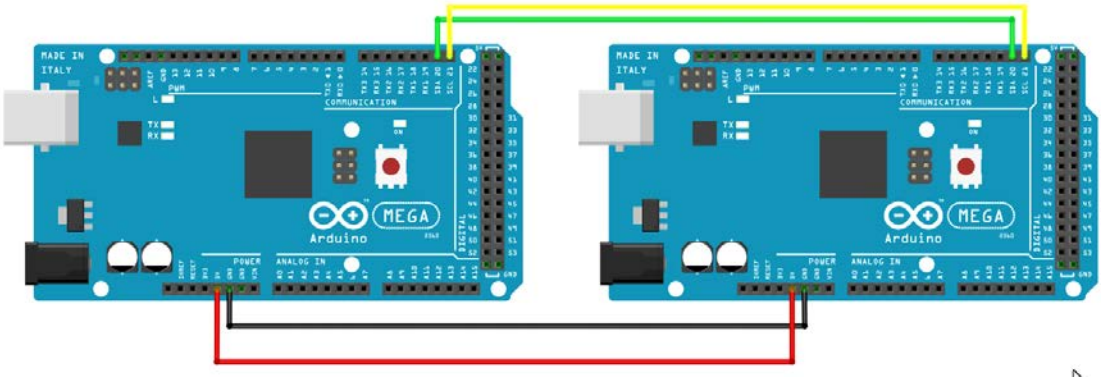
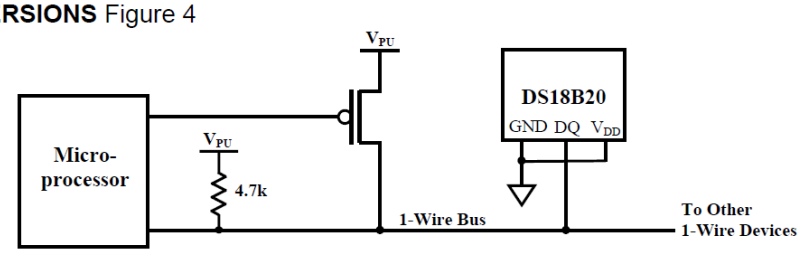


Ilustración 24 Esquema de conexión de dos Arduinos Mega
Fuente: *Elaboración propia*

En este proyecto vamos a utilizar este protocolo I^2C para conectar el display LCD.

1-Wire. Es un protocolo de comunicación serie desarrollado por Dallas Semiconductor. Es un bus donde disponemos de un maestro y varios esclavos en una sola línea de datos que a la vez sirve para alimentar el sensor. También se necesita una línea de referencia (masa) y la línea de datos/alimentación debe de disponer de una resistencia pull-up conectada a alimentación. Podemos escoger entre dos tipos de alimentación, como podemos observar en la ilustración 25.

SUPPLYING THE PARASITE-POWERED DS18B20 DURING TEMPERATURE CONVERSIONS Figure 4



POWERING THE DS18B20 WITH AN EXTERNAL SUPPLY Figure 5

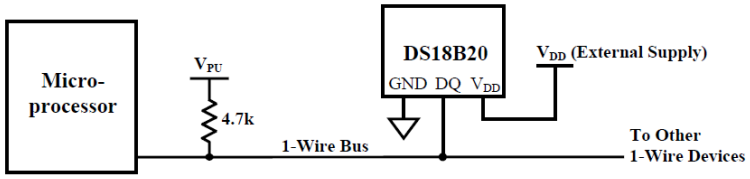


Ilustración 25 Conexión DS18B20
Fuente: DS18B20 Datasheet

En el primer caso se alimenta a los dispositivos en modo parásito, esto quiere decir que solo suministra alimentación mientras se realizan las operaciones de conversión de temperatura. Para ello se deben de puentear las patillas GND y VDD del DS18B20 al GND del Arduino. En el segundo caso, la alimentación de los sensores se realiza mediante una fuente de alimentación externa que va conectada a la patilla VDD del DS18B20 y la patilla GND.

Como en el bus puede haber muchos dispositivos el protocolo incluye el direccionamiento de los mismos empleando un código único de 64 bits de los cuales el byte más significativo indica el tipo de dispositivo, y el último es un código de detección de errores (CRC) de 8 bits.

En este proyecto vamos a utilizar este protocolo 1-wire para conectar los sensores de temperatura DS18B20 al Arduino Mega. Nos permite conectar hasta 100 dispositivos a un único pin y a una distancia máxima de 200 metros.

2.5. Tecnologías de conectividad

Necesitamos un protocolo de comunicación y una infraestructura de red para poder conectar el dispositivo IoT a las plataformas y servicios IoT. Existen diferentes tipos de tecnologías de red, desde las de corto alcance a las más amplias. Podemos encontrar redes de área personal inalámbricas como 6LoWPAN, ZigBee o Bluetooth o una red inalámbrica más grande como Wifi.

Los datos que recogen estas redes se transmiten a la plataforma IoT a través de una red más amplia, Internet. También se pueden utilizar el teléfono móvil para acceder a servicios web remotos que utilizan tecnologías como LTE o 4G. En la ilustración 26 podemos ver el logo de distintos protocolos de comunicación.

Ilustración 26 Logo de diferentes protocolos de comunicación
Fuente: *Elaboración propia*

6LoWPAN (11)**.** (IPv6 over Low power Wireless Personal Area Networks) es un estándar que posibilita el uso de IPv6 sobre redes basadas en el estándar IEEE 802.15.4. Hace posible que dispositivos como los nodos de una red inalámbrica puedan comunicarse directamente con otros dispositivos IP.

Zigbee. La tecnología Zigbee implementa un sistema de comunicación inalámbrico en la banda de 2.4 GHz, está hecho para ser inmune al ruido, y su principal característica es su bajo consumo, su sencillez y su bajo coste. Tiene un alcance de pocas decenas de metros. Está basado en el estándar IEEE 802.15. 4. Se suele utilizar en aplicaciones donde el consumo es un factor importante.

Bluetooth. Bluetooth es una combinación de tecnología Hardware y Software, el Hardware se basa en un chip de radio mientras que los protocolos de seguridad y control se implementan en Software. Es un protocolo de comunicación inalámbrico para la transmisión de datos entre dos dispositivos sin cable, mediante radiofrecuencia, que estén en un radio de 10 metros. Está basado en el estándar IEEE 802.15.1

Redes WiFi. Las redes WIFI o WLAN (Wireless Local Area Network), se basan en el estándar IEEE802.11x (a, b, g) y se han creado específicamente para operar como una red Ethernet sin cable, es una tecnología de estándar abierto que permite la conectividad inalámbrica entre equipos y redes de área local. Efectúa la transmisión por señales de radiofrecuencia que se propagan por el aire. Entre sus ventajas destaca poder utilizar la red dentro de los límites del alcance de la transmisión y la rápida inserción de los dispositivos en la red.

Ethernet. Es la forma estandarizada de poder conectar computadores a través de una red con el fin de poder intercambiar información entre equipos, o compartir el acceso a internet. Es un protocolo de comunicación por cable y es el más utilizado en una red de área local (LAN). La conexión se realiza a través de un conector RJ-45. Está basado en el estándar IEEE 802.3.

GSM. Global System for Mobile comunications. También se la conoce como 2G y supuso un salto de las comunicaciones analógicas a las digitales (12). La banda de frecuencia en la que trabaja el GSM difiere según el territorio. En Europa se utiliza el espectro radioeléctrico de 900 y 1800 MHZ. Los teléfonos que se utilizan se denominan estaciones móviles. Para que una estación sea operativa se necesita una tarjeta SIM, que contiene información sobre el terminal y su usuario. Cada estación móvil tiene un identificador único, el IMEI. Las tarjetas también tienen su propio identificador internacional, con lo que se puede transferir a otro equipo sin perder la información.

GPRS. General Packet Radio Service. El GPRS se basa en el sistema GSM de transmisión de voz, que permitir comunicarse vía satélite, sin necesidad de cables ni conexión física a dos terminales móviles (13). Se utiliza en zonas donde la cobertura 3G y 4G no llega. La diferencia principal entre GSM y GPRS es que la primera estaba orientada a la transmisión de audio y la segunda a la de datos.

3G. Tercera generación de tecnologías de telefonía móvil. Es una tecnología móvil que permite la transmisión de datos, voz y vídeo a una velocidad mínima de 200 Kbit/s y

máxima de 384 y sin cables (14). Otra ventaja con respecto a 2G es que se aumenta el grado de seguridad de las comunicaciones.

4G. Cuarta generación de tecnologías de telefonía móvil. Tiene una velocidad máxima de transmisión de 100 Mbit/s en movimiento y de 1 Gbit/s en reposo (14). La principal diferencia entre el 3G y el 4G es la velocidad de navegación y de descarga de datos en Internet. La cuarta generación es entre cinco y diez veces más rápida que la tercera, y el tiempo de respuesta es entre tres y cuatro veces menor.

LiFi. light Fidelity. Sistema de comunicación inalámbrica de bajo coste y rápido. Consiste en una comunicación inalámbrica que utiliza la luz visible o ultravioleta cercana (UV) e infrarroja cercana (NIR) del espectro electromagnético.

Según el tipo de aplicación se debe de elegir el protocolo de comunicaciones a utilizar. Para este proyecto consideramos que el protocolo que mejor se adapta es el Wifi, que es una conexión muy cómoda para el hogar. Sin embargo, nos hemos decantado por utilizar como protocolo de comunicación Ethernet, por ya disponer del Arduino Ethernet Shield. Además, la conexión Ethernet nos va a permitir un mejor aprovechamiento del ancho de banda.

Para la conexión ethernet necesitamos añadir al Arduino Mega el módulo Arduino Ethernet shield (15), que está basado en el chip ethernet Wiznet W5100 y que se muestra en la Ilustración 27. Este shield provee un conector ethernet estándar RJ45. Dispone de unos conectores que permiten conectar a su vez otras placas encima y apilarlas sobre la placa Arduino.

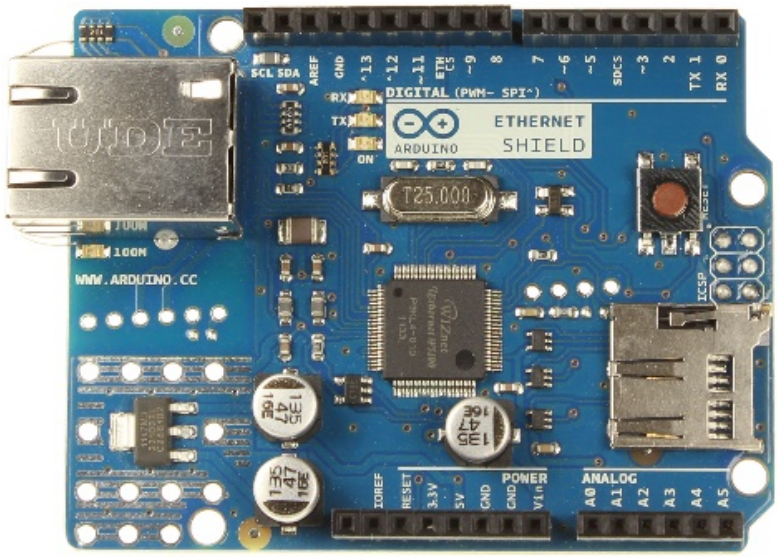


Ilustración 27 Shield Ethernet Arduino
Fuente: https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1

En este proyecto hemos utilizado el Arduino Ethernet Shield V1, que es el modelo que disponemos, pero actualmente está sustituido por el Arduino Ethernet Shield V2.

2.6. Router

Es un dispositivo que proporciona conectividad a nivel de red. Su función principal consiste en enviar o encaminar paquetes de datos de una red a otra, es decir, interconectar subredes, entendiendo por subred un conjunto de máquinas IP que se pueden comunicar sin la intervención de un encaminador (mediante puentes de red o un switch), y que por tanto tienen prefijos de red distintos.

El router que vamos a utilizar es el que proporciona el proveedor de internet, que es el Cable módem Router Wifi con adaptador telefónico modelo Technicolor TC7210, que vemos en la ilustración 28. Este dispositivo, además de su capacidad de proporcionar líneas de telefonía, integra bajo un mismo equipo las funcionalidades de cablemodem, router y punto de acceso inalámbrico con doble banda (2,4 GHz y 5 GHz), proporcionando una solución completa de conexión a Internet. Proporciona prestaciones de cable módem router de datos y voz, de manera cableada (Ethernet) o inalámbrica, para conectar una variedad de dispositivos en el hogar o la oficina pequeña. Además, admite un acceso de alta velocidad a servicios de datos y de voz. ofrece las siguientes ventajas y funciones:

- Conectividad a Internet de banda ancha y alto rendimiento para potenciar su experiencia en línea.
- Adaptador de voz incorporado de dos líneas para telefonía por cable.
- Cuatro puertos Ethernet 1000/100/10BASE-T para conectividad con cables.
- Punto de acceso inalámbrico 802.11n.
- WPS (del inglés Wi-Fi Protected Setup, configuración Wi-Fi protegida), incluido un botón que activa WPS para una configuración inalámbrica simplificada y segura.

Ilustración 28 Router Technicolor TC7210
Fuente: https://blog.telecable.es/internet/telecable-cablerouter-ac-technicolor/

En nuestro proyecto vamos a utiliza uno de los cuatro puertos Ethernet de los que dispone el router para conectarnos al módulo de Arduino y así, establecer la comunicación entre el módulo Arduino y la plataforma de internet que elijamos. La conexión se realiza mediante un cable Ethernet (CAT5/RJ-45) que se conecta un extremo a uno de los puertos del modem router y el otro extremo al conector ethernet del shield Arduino.

Podemos encontrar el manual de usuario de fabricante del cablerouter Technicolor TC7210 en https://www.vodafone.co.nz/cms/documents/1375786333624/ y una guía para la configuración del dispositivo en http://ayuda.telecable.es/inicio/-/wizard/detail/27531/27555/27567/214595335/0/-1/0

2.7. Protocolo de comunicación del Router

Históricamente los dispositivos conectados a internet tenían grandes recursos y utilizaban protocolos muy pesados que impiden poder utilizarse en muchas de las aplicaciones emergentes. Para ellas, se requieren protocolos livianos que no necesiten muchos recursos. En la tabla 1 se muestran diferentes soluciones actualmente utilizadas en aplicaciones IoT para la pila de protocolos de nivel superior.

Protocol	*Transport*	*Messaging*	*2G,3G,4G (1000's)*	*LowPower and Lossy (1000's)*	*Compute Resources*	*Security*	*Success Stories*	*Arch*
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility field area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High- Optional	Military	Bus
DPWS	TCP		Good	Fair	100Ks/RAM Flash ++	High- Optional	Web Servers	Client Server
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low- Optional	Smart Energy Phase 2	Client Server
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High- Optional	Network Monitoring	Client-Server
UPnP		Pub/Subscrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High- Manditory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subscrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High- Optional	CERN	P2P

Tabla 1 Protocolos de comunicación para aplicaciones IoT
Fuente: Curso "Experto en Internet de las Cosas con Arduino – Intercambio de datos"

La “IoT Developer Survey of 2017” revela las tecnologías más populares que se utilizan actualmente en el entorno de la IoT. La ilustración 44 muestra los protocolos más utilizados según esta encuesta. Como se puede observar, los tres protocolos más utilizados son HTTP, MQTT y CoAP.

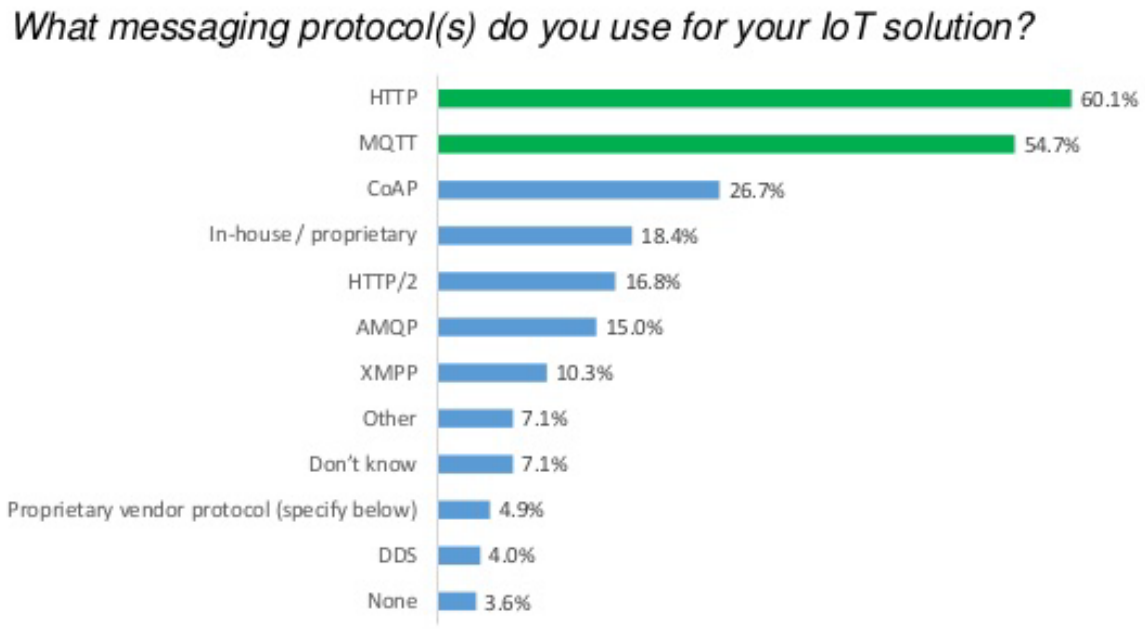


Ilustración 29 Protocolos IoT más utilizados
Fuente: https://www.eclipse.org/lists/iot-wg/pdf05OUMA7E1X.pdf

HTTP es un protocolo de comunicación usado tradicionalmente en aplicaciones de internet, mientras que MQTT y CoAP están más orientados a las aplicaciones IoT. Es preferible el MQTT al CoAP para las comunicaciones de misión crítica porque puede hacer cumplir la calidad del servicio y garantizar la entrega de mensajes. CoAP, por su parte, es adecuado para recopilar datos de telemetría transmitidos desde nodos de baja potencia, como sensores de campo diminutos. En este proyecto se va a utilizar el protocolo MQTT por ser el que mejor se adapta a nuestras necesidades.

A continuación, mostramos las características principales de estos tres protocolos.

- **HTTP/REST**

  Hypertext Transfer Protocol es el protocolo más popular para la comunicación a través de internet. Es un protocolo de intercambio y manipulación de datos. Es un protocolo cliente/servidor sin estado. Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP las podemos ver en la tabla 2.

Comando	Descripción
GET	Solicita el recurso ubicado en la URL especificada
HEAD	Solicita el encabezado del recurso ubicado en la URL especificada
POST	Envía datos al programa ubicado en la URL especificada
PUT	Envía datos a la URL especificada
DELETE	Borra el recurso ubicado en la URL especificada

Tabla 2 Lista de comandos HTTP
Fuente: *https://es.ccm.net/contents/264-el-protocolo-http*

REST (Representational State Transfer) se basa en HTTP para el intercambio de información siendo más ligero, menos engorroso, pero con más limitaciones. En la ilustración 30 vemos el protocolo HTTP.

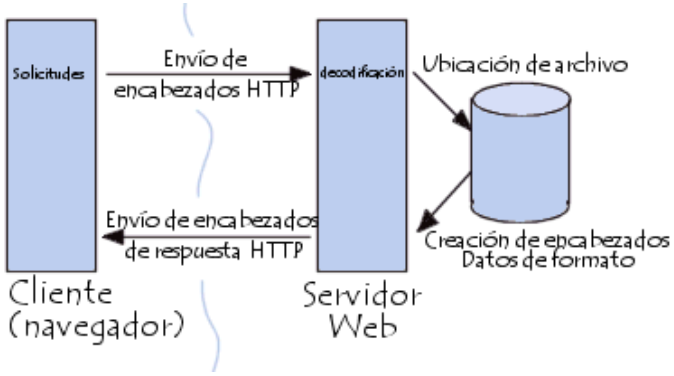


Ilustración 30 Protocolo HTTP
Fuente: http://www.rfwireless-world.com/Tutorials/MQTT-tutorial.html

- **MQTT**

Message Queue Telemetry Transport es un protocolo de mensajería ligero con una arquitectura de publicación/suscripción. Es un protocolo utilizado para la comunicación machine to machine. Está orientado a la comunicación de sensores, por usar poco ancho de banda y por poder ser utilizado por dispositivos con pocos recursos. Es un protocolo open source basado en el paso de mensajes entre múltiples clientes a través de un intermediario central.

Hay dos tipos de entidades en el protocolo MQTT: un intermediario y el cliente. El intermediario se puede considerar como un servidor central que recibe los mensajes de todos los clientes y luego los dirige a los clientes de destino requeridos. Un cliente puede considerarse como cualquier dispositivo que puede interactuar con el intermediario para enviar y recibir mensajes (un sensor, etc.).

El proceso de suscripción/publicación es el siguiente: El cliente se conecta con el agente y puede suscribirse a cualquier mensaje "tema" en el agente. La conexión puede ser una conexión TCP/IP simple o una conexión cifrada TLS para mensajes confidenciales. El cliente publica mensajes bajo un tema enviando el mensaje y el tema al intermediario. Luego, el intermediario reenvía el mensaje a todos los clientes suscritos a ese tema. En la ilustración 31 vemos la arquitectura MQTT.

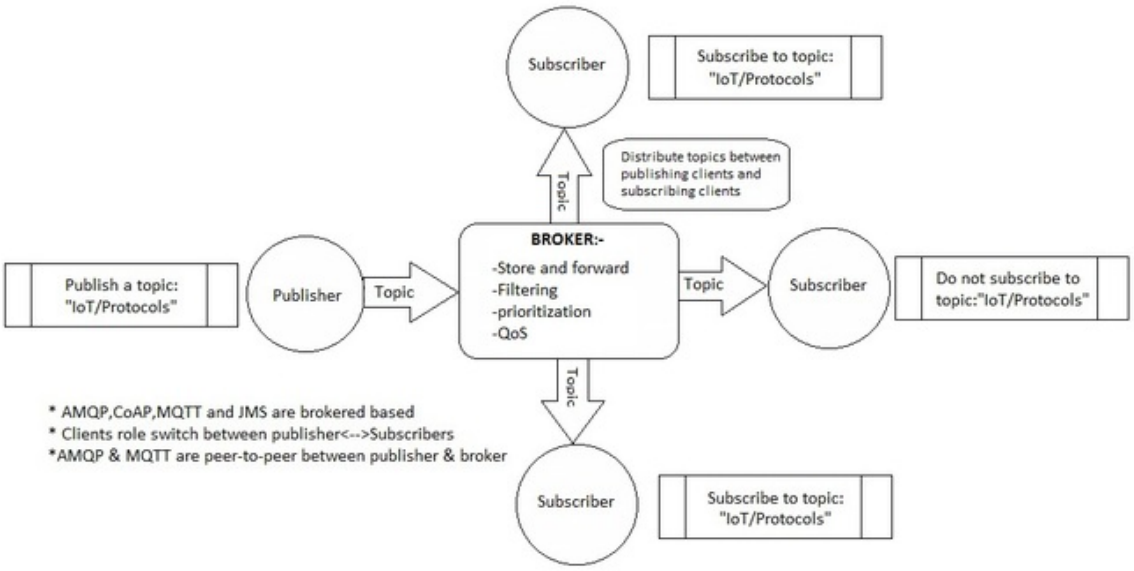


Ilustración 31 Arquitectura MQTT
Fuente: http://www.rfwireless-world.com/Tutorials/MQTT-tutorial.html

Los mensajes MQTT están organizados por temas. Un tema es una cadena UTF-8, que el intermediario utiliza para filtrar mensajes para cada cliente conectado. Consiste en uno o más niveles, separados por una barra inclinada. Un cliente solo puede publicar en un tema individual, pero se le permite suscribirse a múltiples temas.

Las características principales del protocolo MQTT son:

- Utiliza conexiones TCP
- Fue diseñado para un tráfico de red mínimo
- Es apropiado para dispositivos pequeños, ya que tiene una sobrecarga mínima y un formato eficiente.
- Proporciona comunicaciones bidireccionales en redes no confiables, utilizando tres niveles de QoS (calidad de servicio). Ver ilustración 32.

  • QoS 0 (como máximo una vez): es una entrega de mejor esfuerzo. Los mensajes no son reconocidos por el receptor (y como consecuencia, no son reenviados por el remitente). Este nivel de QoS puede proporcionar la misma garantía que el protocolo TCP subyacente.

• QoS 1 (al menos una vez): en este caso, el emisor almacena el mensaje transmitido hasta que recibe un acuse de recibo del receptor. Se garantiza que se enviará un mensaje al receptor al menos una vez (puede entregarse más de una vez).

• QoS 2 (exactamente una vez): garantiza que cada mensaje se reciba solo una vez. Para ese propósito, se usan dos flujos de confirmación, convirtiéndose en el nivel de QoS más lento.

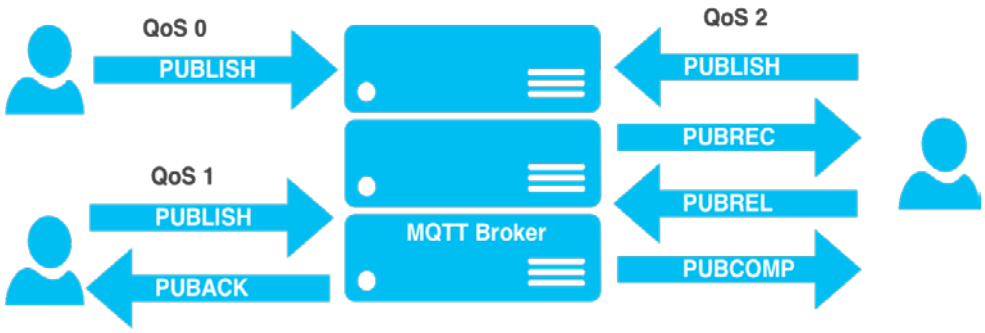


Ilustración 32 Tres niveles de Qos
Fuente: https://aprendiendoarduino.wordpress.com/tag/seguridad-mqtt/

- **CoAP**

Tiene una arquitectura cliente/servidor similar al de HTTP con la diferencia que CoAP realiza el intercambio de mensajes de forma asíncrona por medio del protocolo de transporte UDP. Utiliza REST y soporta las operaciones GET, PUT, POST y DELETE. En la ilustración 33 vemos su arquitectura. Está pensado para ser usado en dispositivos electrónicos sencillos que necesitan comunicarse sobre Internet. Cuando el cliente realiza una petición, mediante una opción que indica el método a utilizar para solicitar un recurso (identificado por una URI), el servidor envía una respuesta con un código que puede incluir una representación de dicho recurso, esta comunicación es asíncrona a través de transporte UDP. Esto se realiza utilizando una capa de mensajes, que soporta una fiabilidad opcional, los mensajes pueden ser: Confirmable (CON), No-Confirmable (NON), asentimientos o reconocimiento (ACK) y Reset (RST). Estos mensajes se encuentran en la cabecera CoAP.

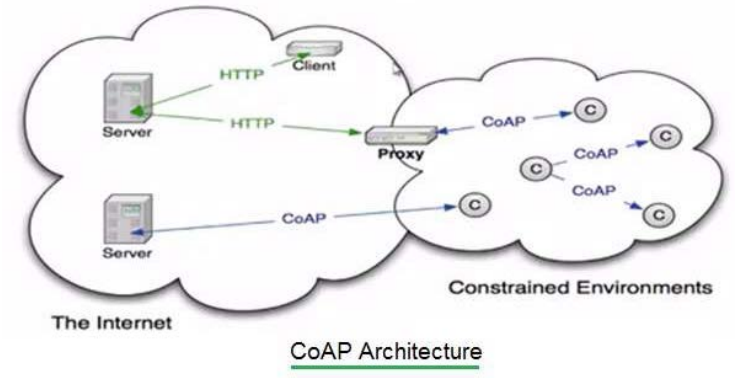


Ilustración 33 Arquitectura CoAP
Fuente: http://www.rfwireless-world.com/IoT/CoAP-protocol.html

Hay dos modos en el protocolo CoAP, como se muestra en la ilustración 34, para el intercambio de mensajes entre el cliente CoAP y el servidor CoAP:

- Sin respuesta separada.
- Con respuesta separada. En este caso el servidor notifica al cliente sobre la recepción del mensaje de solicitud. Esto aumentará el tiempo de procesamiento, pero ayudará a evitar retransmisiones innecesarias.

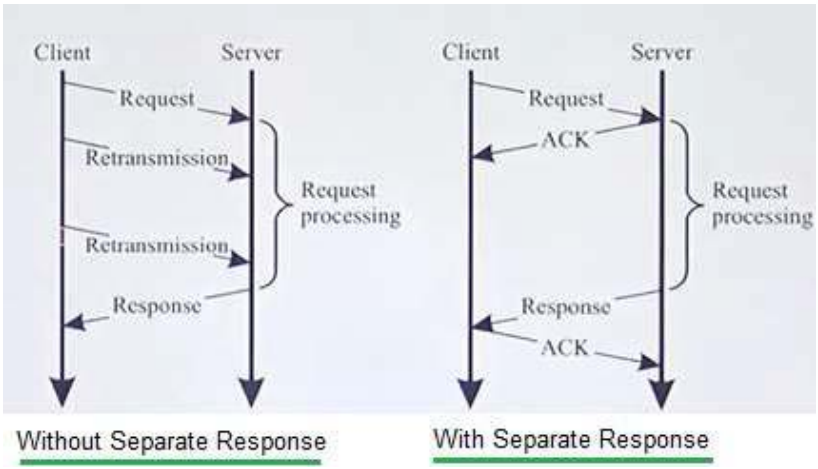


Ilustración 34 Intercambio de mensajes CoAP
Fuente: http://www.rfwireless-world.com/IoT/CoAP-protocol.html

2.8. Plataformas IoT

El propósito de cualquier dispositivo IoT es conectarse con otros dispositivos y aplicaciones IoT para transmitir información.

La plataforma IoT conecta el hardware, puntos de acceso y bases de datos con el usuario. Sus principales funciones son:

- Garantizar la transmisión de datos y la interacción con los dispositivos.
- Almacenar los datos obtenidos de los dispositivos en la nube.
- Ejecutar acciones inteligentes en base a los datos obtenidos por los dispositivos.
- Visualizar los datos obtenidos por los dispositivos.
- Integración con otros sistemas

Las plataformas IoT nos va a permitir visualizar los datos, generar eventos, etc.

En la actualidad existen múltiples plataformas de IoT. En (16) podemos ver el artículo "Top 20 IoT Platforms in 2018" donde se realiza un estudio de diferentes plataformas IoT. En la ilustración 35 podemos ver el logo de varias plataformas IoT que se citan en el artículo.

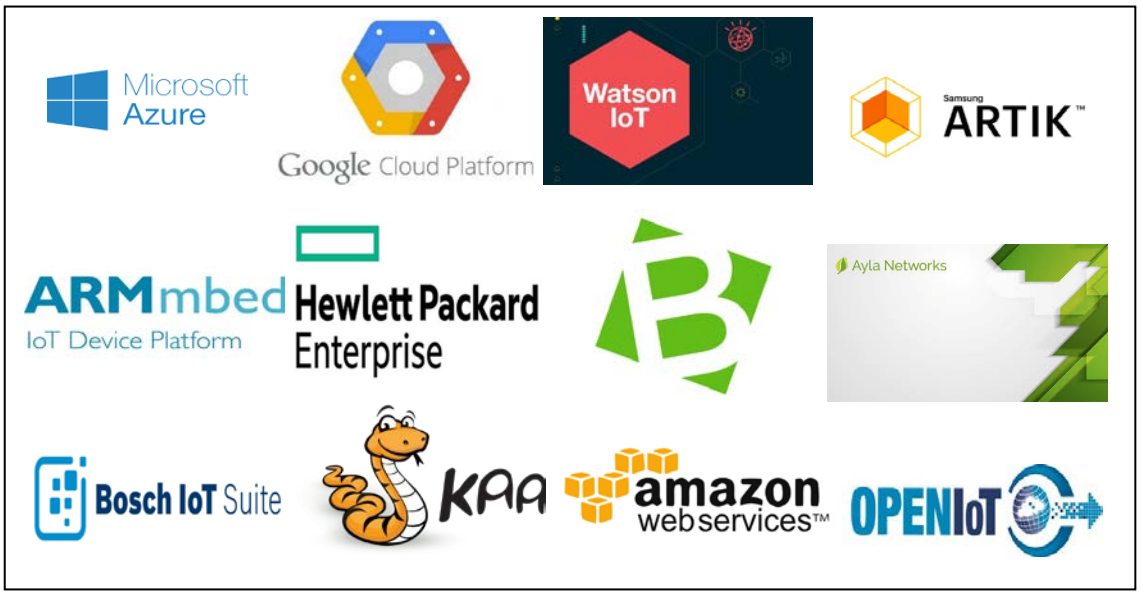


Ilustración 35 Varias Plataformas IoT
Fuente: Elaboración propia

Cada día aparecen nuevas plataformas IoT con lo que es difícil hacer un análisis de todas ellas debido al gran volumen existente. Para facilitar el análisis podemos hacer una clasificación en función del coste y el sector al que van destinadas (17).

- **Plataformas orientadas a pequeñas empresas**. Plataformas que permiten su uso de manera gratuita, pero con limitaciones en cuanto al número de mensajes enviados y de dispositivos conectados.
- **Plataformas que ofrecen servicios globales**. También ofrecen servicios gratuitos o versiones de prueba. Son el paso intermedio entre las plataformas orientadas a IoT y las plataformas de las grandes empresas. Alguna de ellas son TheThingsIo, IFTTT Maker, Particle, Firebase Google, etc.
- **Plataformas de grandes empresas** como Google, Amazon, Microsoft, IBM, Autodesk, etc. Están orientadas al sector industrial y a grandes proyectos de IoT.
- **Plataformas de código abierto**. Dan acceso al código sin restricciones. No son recomendables para usuarios que no sean expertos en tecnología del lado del servidor (Apache, NodeJS, etc.). Algunas son Zetta, OpenHaB, Node-RED, Kaa, OpenIoT, Kura, etc.

Para la elaboración de nuestro proyecto nos interesan las plataformas pertenecientes al primer grupo, plataformas orientadas a pequeñas empresas, por estar orientadas a la conectividad de objetos y por existir, dentro de este grupo, varias plataformas preparadas para una fácil integración con proyectos realizados en la plataforma Arduino. Muchas de estas plataformas son totalmente configurables y permiten su uso gratuito con ciertas restricciones. Dentro de este grupo tenemos plataformas como Samsung Artik Cloud, aREST Framework, Tinger.io, Arduino Cloud, Cayenne mydevices y ThingSpeak. En la ilustración 36 vemos el logo de estas plataformas IoT.

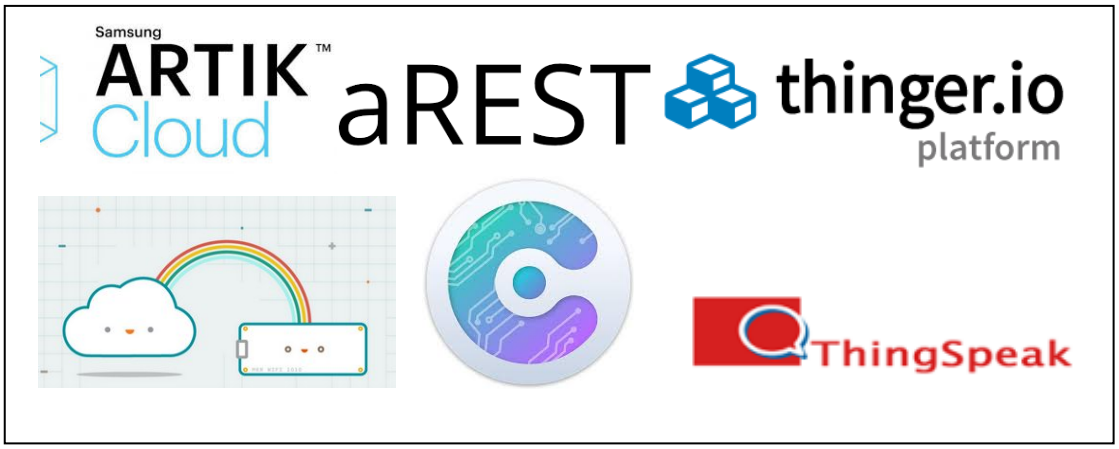


Ilustración 36 Plataformas IoT orientadas a pequeñas empresas
Fuente: Elaboración propia

En (17) se encuentra una evaluación sobre plataformas IoT para usar con proyectos de Arduino y las plataformas que recomiendan para la iniciación en el mundo del IoT. En la ilustración 37 vemos un resumen del estudio.

Ilustración 37 6 Plataformas IoT para Arduino
Fuente: https://programarfacil.com/podcast/proyectos-iot-con-arduino/

De estas seis plataformas se prueban las plataformas Tinger.io y Cayenne.

- **Thinger.io**

Ofrece una infraestructura de nube escalable lista para usar para conectar millones de dispositivos. Puede controlarlos con la consola de administración fácil de usar, o integrarlos en la lógica de su negocio con su API REST (18).

Es una plataforma española de código abierto, con una programación muy sencilla y que dispone de una librería en el repositorio oficial de Arduino. Una de las mayores ventajas es que soporta IFTTT, lo que nos permite comunicación con cualquier cosa que te puedas imaginar. Tiene una consola de administración muy buena, como podemos apreciar en la ilustración 38, y existe abundante documentación.

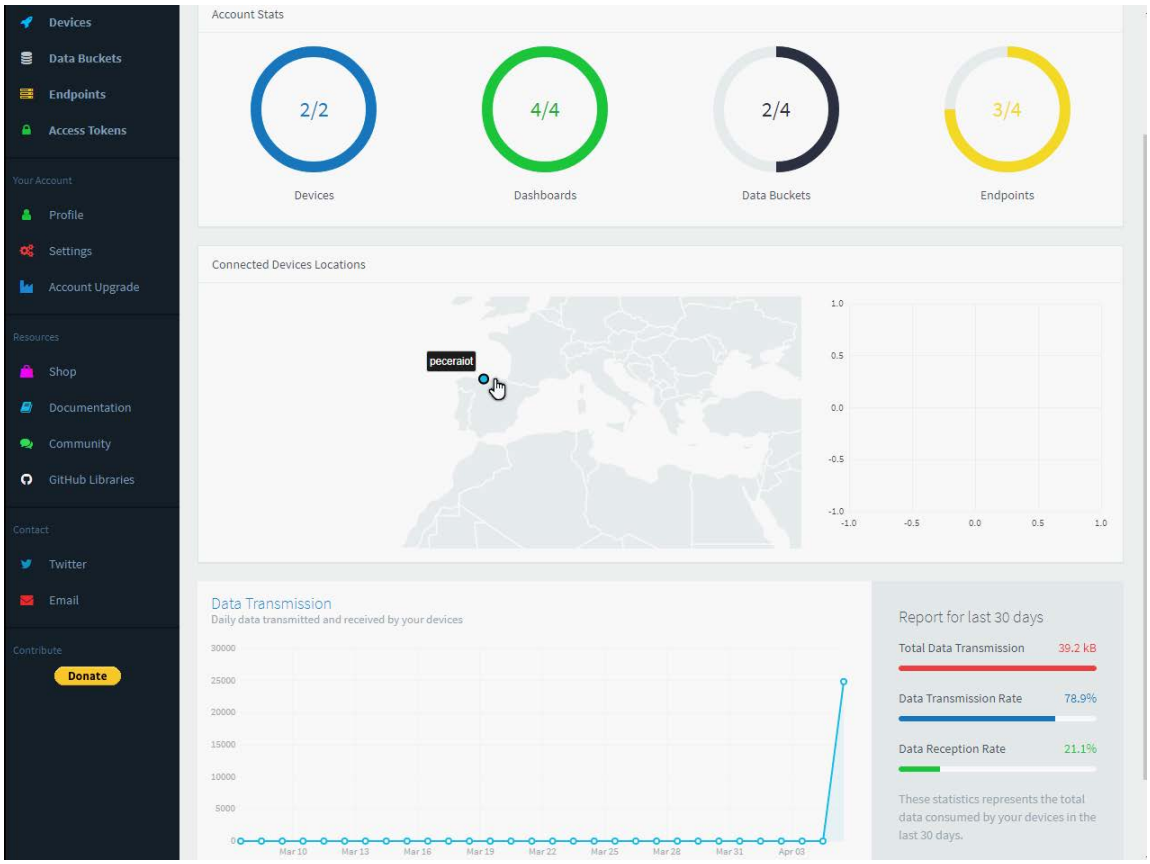


Ilustración 38 Consola de Thinger.io
Fuente: Captura de pantalla

Las principales características de la plataforma son: gestión completa del dispositivo e interacción API; puntos finales para interactuar con servicios de terceros o enviar notificaciones; cubos de datos para almacenar y exportar información; administración de acceso para otorgar acceso a sus dispositivos y datos desde otras aplicaciones; y cuadros de mando para la visualización de datos en tiempo real.

En la ilustración 39 podemos ver un ejemplo de Dashboard realizado con esta plataforma.

Ilustración 39 Ejemplo Dashboard de Thinger.io
Fuente: Captura de pantalla

Thinger.io dispone de distintos tipos de cuentas y tarifas con distintas capacidades, que podemos ver en la ilustración 40, para cubrir las distintas necesidades de los usuarios.

Ilustración 40 Tipos de cuentas y precios de Thinger.io
Fuente: https://thinger.io/

- **Cayenne my devices**

Cayenne myDevices es una plataforma de prototipado de dispositivos del IoT. Una de las mayores ventajas que tiene es que es una solución visual, arrastrar y soltar, es decir, nos permite configurar el sistema sin programar. La ventaja está a la vista, no hace falta programar ni una línea de código del lado del servidor. Sin embargo, como ocurre con este tipo de herramientas, nos limita bastante ya que tenemos que adaptarnos a lo que nos ofrecen (19). En la ilustración 41 vemos la consola principal de Cayenne.

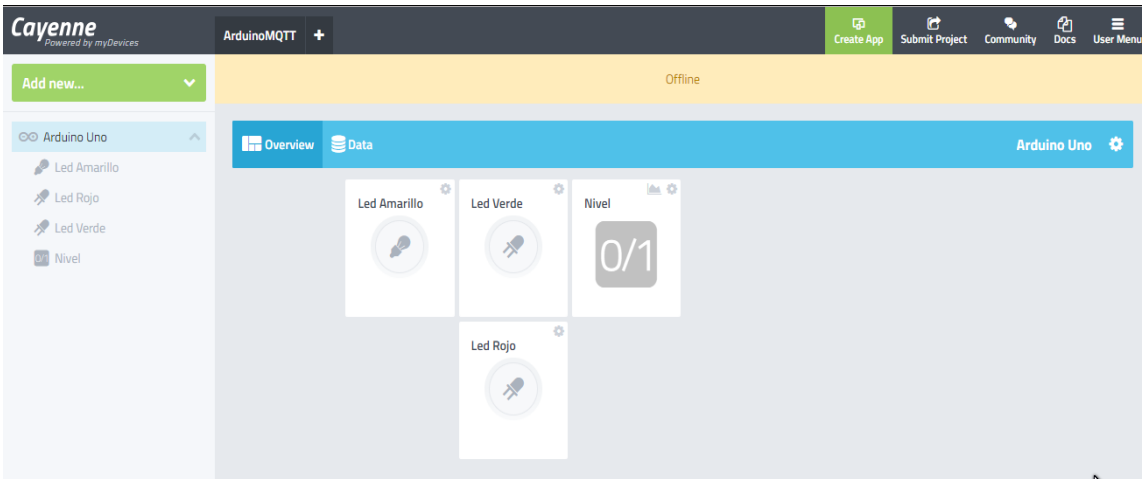


Ilustración 41 Consola de Cayenne
Fuente: Captura de pantalla

Todo se comunica a través de una API MQTT y la única limitación que existe es a la hora de hacer llamadas a la API. Cada cliente sólo puede enviar 60 mensajes por minuto e intentar 50 conexiones cada 10 minutos por IP. Estas limitaciones no afectan al número de dispositivos conectados, pero si al total de mensajes enviados o de intentos de conexión. En la ilustración 42 podemos ver un ejemplo de Dashboard realizado con esta plataforma.

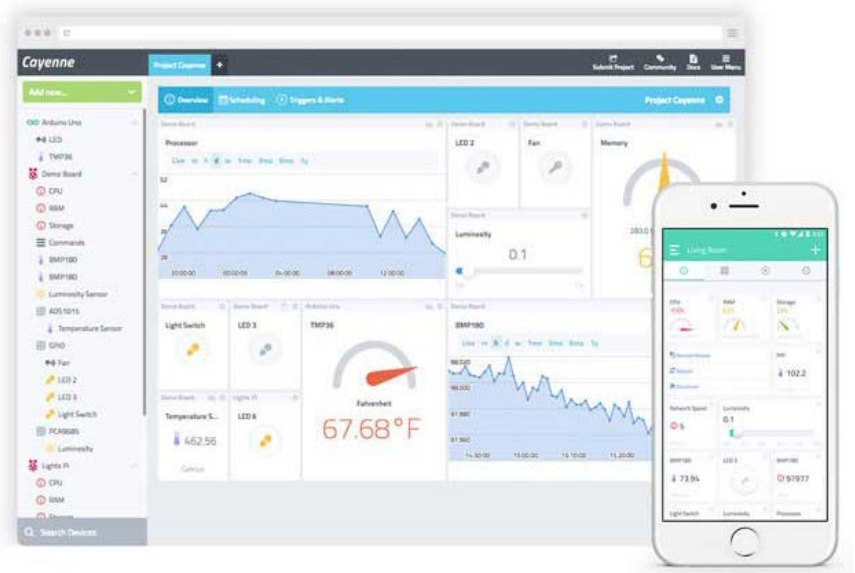


Ilustración 42 Ejemplo Dashboard de Cayenne
Fuente: https://programarfacil.com/blog/arduino-blog/cayenne-mydevices-arduino-sensores-iot/

La forma de trabajar en ambas plataformas es muy similar, siendo elegida la plataforma Thinger.io por ser una plataforma emergente y resultarnos más fácil de utilizar tras haber realizado un curso sobre ella. En la ilustración 43 vemos el logo de esta plataforma.

Ilustración 43 Logo de la plataforma elegida Thinger.io
Fuente: https://thinger.io/

2.9. Sensores utilizados en el sistema

A continuación, vamos a ver cada uno de los sensores utilizados en este proyecto.

2.9.1. Sensor de temperatura DS18B20

Nos va a permitir medir la temperatura del agua del acuario.

El sensor DS18B20 (20) proporciona una resolución desde 9 bits a 12 bits. El DS18B20, que vemos en la ilustración 44, se comunica a través de un Bus 1-Wire que por definición requiere solo una línea de datos (y tierra) para la comunicación con un microprocesador central. No necesidad de una fuente de alimentación externa. Cada DS18B20 tiene un código de serie único de 64 bits, que permite que múltiples DS18B20s funcionen en el mismo cable. Rango de temperatura -55 ° C a + 125 ° C (-67 ° F a + 257 ° F). Precisión en el rango de -10 ° C a + 85 ° C: ± 0,5 ° C.

Ilustración 44 Sensor de temperatura DS18B20
Fuente: *Propiedad del autor*

2.9.2. Sensor analógico de pH de DFROBOT

Con este sensor, que se muestra en la ilustración 45, vamos a medir el pH (21). Se compone de una sonda y una placa controladora que proporciona un valor analógico proporcional a la medida, tal y como se puede apreciar en la tabla 3.

Valor pH	Voltaje (mV)	Valor pH	Voltaje (mV)
0.00	414.12	14.00	- 414.12
1.00	354.96	13.00	- 354.96
2.00	295.80	12.00	- 295.80
3.00	236.64	11.00	- 236.64
4.00	177.48	10.00	- 177.48
5.00	118.32	9.00	- 118.32
6.00	59.16	8.00	- 59.16
7.00	0.00	7.00	0.00

Tabla 3 Relación mV y pH del sensor pH
Fuente: *https://www.dfrobot.com/wiki/index.php/PH_meter(SKU:_SEN0161)*

El kit viene con una muestra con un pH de 4.0 para calibrar la sonda. Las principales características de la sonda son:

Alimentación: 5 V
Rango de medición: 0 – 14 pH
Temperatura de medición: 0 – 60ºC
Precisión: ± 0,1 pH
Tiempo de respuesta: ≤ 1 min
Led Verde: Alimentación
Led Rojo: Límite de pH

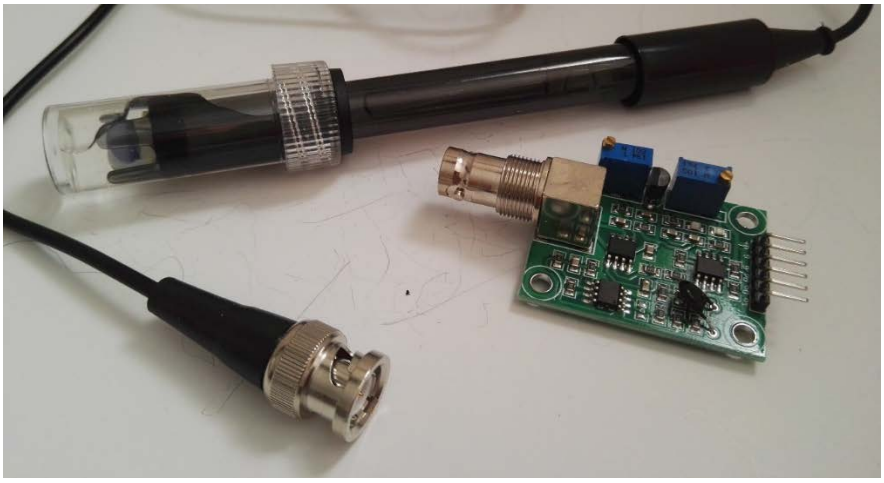


Ilustración 45 Sensor analógico de pH
Fuente: *Propiedad del autor*

2.9.3. Sensor nivel de líquidos horizontal Cebek C-7236

Nos va a permitir controlar el nivel del agua en el acuario. Para ello vamos a utilizar sensor tipo flotador de nivel como el que se muestra en la ilustración 46. Son sensores de tipo todo o nada.

Los sensores de nivel de líquidos C-7235 (22) están fabricados con polipropileno. Cuando el flotador magnético pivota al nivel adecuado, el sensor abrirá o cerrará sus contactos respecto a la posición de montaje. La sujeción al acuario se realiza mediante rosca y disponen de 50 cm. de cable para la conexión.

En este proyecto vamos a utilizar dos sensores de nivel, uno para el nivel máximo y otro para el nivel mínimo de agua del acuario.

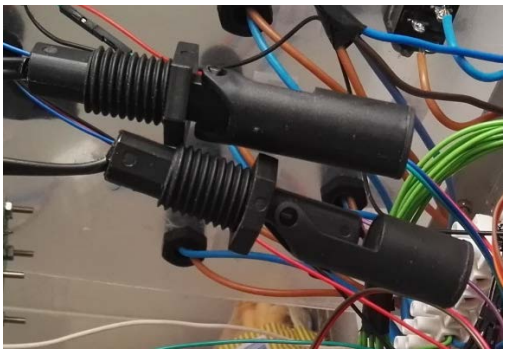


Ilustración 46 Sensor de nivel C-7236
Fuente: *Propiedad del autor*

2.10. Actuadores y otros dispositivos utilizados en el sistema

2.10.1. Módulo de relés de TOOGOO 5V - 8 canales

Un relé es un dispositivo electromecánico que va a permitir a nuestra placa de Arduino Mega controlar cargas de 220 v e intensidades de hasta 10 A. Dispone de luces rojas indicadoras de estado de funcionamiento. Ampliamente utilizado para todo el control de MCU, sector industrial, control del PLC, control casero elegante. En la ilustración 10 podemos ver el módulo utilizado en este proyecto.

Voltaje de trabajo: 5V

Nº de canales: 8

Ilustración 47 Módulo de relés 5V 8 canales
Fuente: *Propiedad del autor*

2.10.2. Calentador

Según pasa el tiempo, la temperatura del agua del acuario va descendiendo, lo cual es perjudicial para los peces y las plantas, puesto que necesitamos mantener una temperatura constante. Para evitarlo vamos a disponer de un calentador. En el mercado existen infinita variedad de opciones, pero no todas son válidas. Para una correcta elección el principal parámetro a tener en cuenta es el de la potencia del calentador y esta debe de ser, por regla general, de 1 watio por cada litro de agua del acuario (6).

La tabla 4 relaciona la diferencia térmica en ºC, con el volumen del acuario en litros y la potencia del calentador requerida en watios.

Potencia del calentador para acuarios								
	25 l	50 l	75 l	100 l	150 l	200 l	250 l	300 l
-5º C	50 W	50 W	50 W	100 W	100 W	150 W	200 W	300 W
-10º C	50 W	50 W	100 W	100 W	150 W	200 W	300 W	300 W
-15º C	100 W	100 W	150 W	200 W	300 W	2x200W	2x300W	2x300W

Tabla 4 Potencia del calentador
Fuente: *http://www.deudei.es/acuarios/calcular-potencia-calentador-acuario/*

Como el acuario que vamos a utilizar es de 60 litros, la potencia recomendada sería la indicada en la columna de 75 l. y como la diferencia térmica en invierno puede ser de hasta 10 ºC, la potencia recomendada para el calentador es de 100 w.

Otro factor importante a la hora de escoger el calentador es que disponga de termostato, función que vamos a utilizar como seguridad añadida, para evitar que la temperatura del agua supere el valor de consigna del termostato.

En este caso vamos a utilizar el calentador que ya tenemos, que es de 220 V – 150 W, y que podemos ver en la ilustración 48.

Ilustración 48 Calentador 220 V – 150 W con termostato
Fuente: *Propiedad del autor*

2.10.3. Bombas de agua

Para un correcto cuidado del acuario una de las partes más importantes es que el agua esté limpia y que conserve las bacterias beneficiosas para la salud de los peces. Para ello es necesario un correcto filtrado del agua. Existen diferentes tipos y modelos. Pueden ser internos o externos y con filtración biológica, química o mecánica.

En nuestro proyecto vamos a utilizar una bomba para hacer circular el agua, haciéndola pasar por un filtro natural formado por dos esponjas y entre ellas una capa de carbón activo, que vemos en la ilustración 49.

Ilustración 49 Esponjas y carbón activo para crear un filtro natural
Fuente: *Propiedad del autor*

Un buen sistema de filtración requiere que sea capaz de filtrar el agua total del acuario de 3 a 5 veces en una hora (6). Como nuestro acuario es de 60 l la bomba debe mover entre 180 – 300 l/h. La bomba a utilizar es una que ya disponemos, modelo TURBO-JET mini AS250, de 3,2 watios y un caudal de 250 l/h. Esta bomba la vemos en la ilustración 50.

Ilustración 50 Bomba de agua 3,2 W – 250 l/h
Fuente: *Propiedad del autor*

Además de la bomba de agua anterior, necesitamos otras dos bombas para usar en la automatización del cambio de agua, una para vaciar parcialmente de agua el acuario y otra para rellenar. En este caso ya disponemos de dos bombas, ver ilustración 51, modelo AT-1020 de 3 watios con un caudal máximo de 200 l/h y Hmax 0,5m.

Ilustración 51 Bomba de agua 3 W – 200 l/h
Fuente: *Propiedad del autor*

2.10.4. Iluminación

La iluminación del acuario es otro de los factores principales para la salud de los peces y de las plantas. Para una correcta evaluación de las necesidades lumínicas para nuestro acuario es necesario tener en cuenta varios factores. Por un lado, el tipo de iluminación: fluorescente o led. En este proyecto nos vamos a decantar por el uso de led puesto que, aunque es más caro, la eficiencia es un factor importante porque la iluminación va estar encendida durante muchas horas (de nueve a doce horas). Además, el calor que generan los tubos fluorescentes es otro punto en su contra. Otro de los factores a tener en cuenta es la profundidad del acuario porque la luminosidad va disminuyendo proporcionalmente al cuadrado de la distancia entre el foco y el objeto. A modo orientativo, y para el uso de led se necesitan 20 lúmenes por litro para una iluminación media y 35 lúmenes para una iluminación alta. Para los acuarios no se utilizan luces de menos de 5.500ºK dado que fomentan el crecimiento de algas. Las lámparas entre 5.500 y 6.500ºK son las más adecuadas.

En nuestro proyecto vamos a necesitar una iluminación media (20 lúmenes por litro) por lo que necesitamos 1200 lúmenes aproximadamente. Programaremos la luz en dos ciclos de cinco horas con un descanso de dos horas, esto no perjudica a las plantas y reduce la tasa de crecimiento de las algas. Para ello vamos a utilizar la barra de led que ya disponíamos y que se muestra en la ilustración 52.

El proyecto está preparado para añadir posteriormente una segunda iluminación.

Ilustración 52 Led 38 cm luz blanca
Fuente: *Propiedad del autor*

2.10.5. Aireador HiPumps YDQB4105

Su función es la de aportar oxígeno al agua, para ello sueltan burbujas en el fondo de la pecera a través de una piedra difusora que se instala debajo del sustrato. Cuanto más pequeñas y más numerosas son las burbujas más efectivas resultan. En la ilustración 53 vemos el aireador utilizado.

Ilustración 53 Airead0r HiPumps YDQB4105
Fuente: *Propiedad del autor*

2.10.6. Servomotor MG90S

El servomotor es un motor que se acopla a una caja de engranajes lo que le permite que se puede posicionar dentro de un rango de operación y permanecer en esa posición. Según su rango de rotación, se pueden clasificar en:

- Rango de giro limitado: Permiten una rotación de 180º. Son los más comunes.
- De rotación completa: Permiten una rotación de 360º, es decir, una vuelta completa.

Los servomotores poseen tres cables, a diferencia de los motores comunes que sólo tienen dos. Este tercer cable es necesario para la señal de control modulada (PWM).

Para nuestra aplicación, que es para la alimentación de los peces, un servomotor de giro limitado es suficiente. El modelo elegido es el Micro servo MG90S que podemos ver en la ilustración 54, así como el comedor una vez montado. El servo Tower Pro MG90S de JSUMO (23) es un servo miniatura de gran calidad y diminutas dimensiones, además es bastante económico.

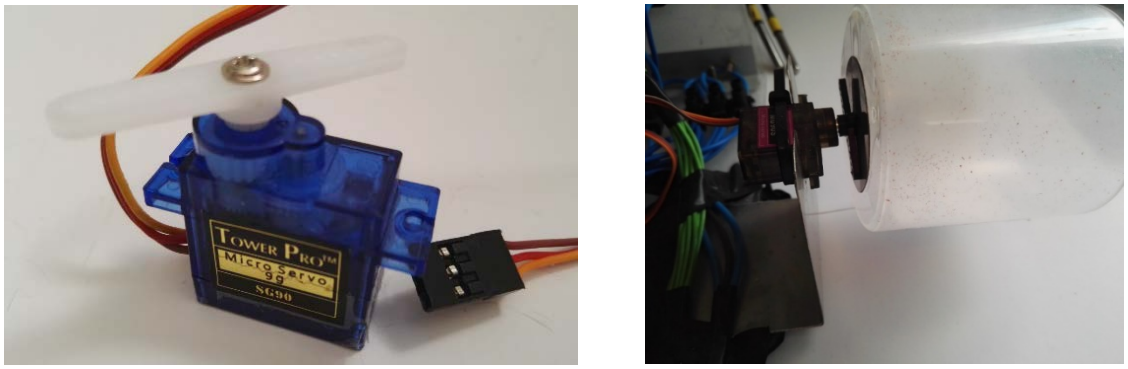


Ilustración 54 Micro Servo MG90S y comedor
Fuente: *Propiedad del autor*

2.10.7. Display LCD204

Se trata de un módulo LCD de 4 líneas y 20 columnas con retroiluminación, como podemos ver en la ilustración 55, que nos permite mostrar información en forma gráfica.

Ilustración 55 Display LCD204
Fuente: http://www.superrobotica.com/Images/S310118big.JPG

Dispone de doble interfaz que permite controlarlo desde un puerto serie o mediante el bus I2C. Para este segundo caso, que es como lo vamos a utilizar, necesitamos un módulo adaptador LC2 a I2C como el que vemos en la ilustración 56.

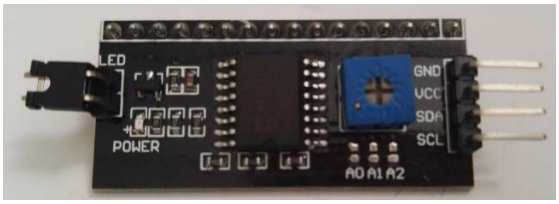


Ilustración 56 Módulo adaptador LCM 1602 IIC
Fuente: *Propiedad del autor*

3. Desarrollo del proyecto

Una vez realizado el análisis de los componentes más relevantes y el estudio de mercado de los diferentes elementos, vamos a desarrollar nuestro proyecto con las conclusiones obtenidas y adecuándolo a las necesidades, tanto tecnológicas como económicas.

En un acuario es muy importante el control de todos los parámetros del agua para conseguir un ecosistema donde las plantas se desarrollan, los peces estén saludables sin enfermedades, no existan algas y sea seguro para el usuario. Sin embargo, es necesario tener en cuenta el tipo de acuario a instalar, pues en función del tipo de peces y plantas los parámetros de bienestar varían y son los que nos van a determinar los valores correctos de temperatura, pH o dureza del agua. Hay que tener en cuenta que el acuario es un espacio vivo, lleno de vida, en el que las propiedades químicas del agua cambian por la actividad de sus ocupantes y en el que para mantener el equilibrio químico hay que realizar controles periódicos.

3.1. Requisitos funcionales del proyecto

El sistema que vamos a diseñar debe de cumplir los siguientes requisitos.

1. El sistema una vez conectado a la red eléctrica debe de comenzar a funcionar de manera autónoma con unos parámetros iniciales que son:
   - Temperatura: 23ºC
   - Histéresis Temperatura: 1,5ºC
   - Consigna pH: 7,0
   - Histéresis pH: 0,45

   Además, se debe mostrar en el display lcd el valor del pH, de las temperaturas y el modo de funcionamiento.
2. En modo manual se debe de poder encender/apagar cada uno de los actuadores.
3. En modo automático se debe de conectar a una plataforma IoT. En la plataforma se debe de visualizar la temperatura y el pH.
4. Medir la temperatura con una precisión de ± 0,5ºC
5. Medir el pH con una precisión de ± 0,2
6. Permitir modificar la consigna de temperatura, de pH y de sus histéresis desde la plataforma IoT.
7. Controlar el nivel de agua en el acuario, mediante la fijación de dos niveles, nivel máximo y nivel mínimo. El nivel de llenado del acuario se considera correcto si el nivel del agua se encuentra entre estos dos niveles.
8. Permitir ajustar el horario de funcionamiento de la iluminación y de la aireación.
9. Alimentar automáticamente a los peces dos veces al día.
10. Automatizar el proceso de cambio parcial de agua mensual.
11. Enviar alertas de Temperatura, de pH y de nivel por correo electrónico.
12. Filtrado continuo del agua.
13. Acceso a la información a través de cualquier medio externo que disponga de acceso a internet.

14. Aplicación para smartphone.
15. Visualización local de la temperatura y del pH.
16. Disponer en la plataforma de gráficas de temperatura y de pH.

3.2. Modos de funcionamiento

El sistema dispondrá de dos formas de funcionamiento:

- **Modo manual:** Permite al usuario controlar de forma local la activación/desactivación de los distintos componentes del sistema. No existe ningún control automático.
- **Modo automático:** El sistema actúa en función de la información recibida de los sensores o en consecuencia al horario establecido y realizará las acciones programadas. Se dispondrá de un sistema de monitorización, configuración y control a través de la plataforma Thinger.io, que dispondrá de las funcionalidades necesarias para el control del acuario. También permite la activación/desactivación manual, desde la plataforma, de los distintos componentes del sistema.

3.3. Desarrollo del hardware del sistema

3.3.1. Conexión del shield ethernet Arduino

Para el control del shield Ethernet Arduino se necesita la librería Ethernet.h que está integrada en el IDE de Arduino. La biblioteca permite que un Arduino se conecte a internet.

El shield se comunica con el microcontrolador por el bus SPI, por lo tanto, para usarlo siempre debemos incluir la librería “SPI.h”.

En la tabla 5 podemos ver los pines que utiliza Arduino para el protocolo SPI.

MOSI	MISO	SCK	SS
51	50	52	10

Tabla 5 Pines para conectar el Arduino Ethernet Shield R3
Fuente: Elaboración propia

El pin SS del hardware 53, no se utiliza para seleccionar el chip del controlador Ethernet, pero debe de mantenerse como salida. En la ilustración 57 vemos la ubicación de estos pines.

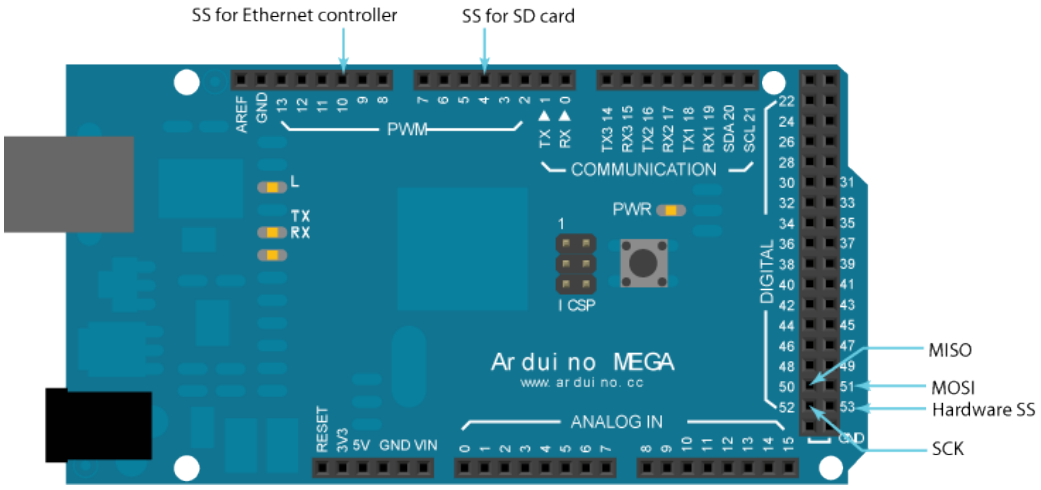


Ilustración 57 Situación de los pines SPI en el Arduino Mega 2560
Fuente: https://www.arduino.cc/en/Reference/Ethernet

La tarjeta Arduino Mega 2560 y el shield Arduino Ethernet se comunica a través del conector ICSP que aparece resaltado en la ilustración 58.

Ilustración 58 ICSP
Fuente: hhttp://arduinomega.blogspot.com/2011/06/reviving-dead-arduino-board-good-as-new.html

La conexión del shield ethernet Arduino al Arduino Mega 2560 es inmediata. En la ilustración 59 podemos ver las dos placas una vez montadas.

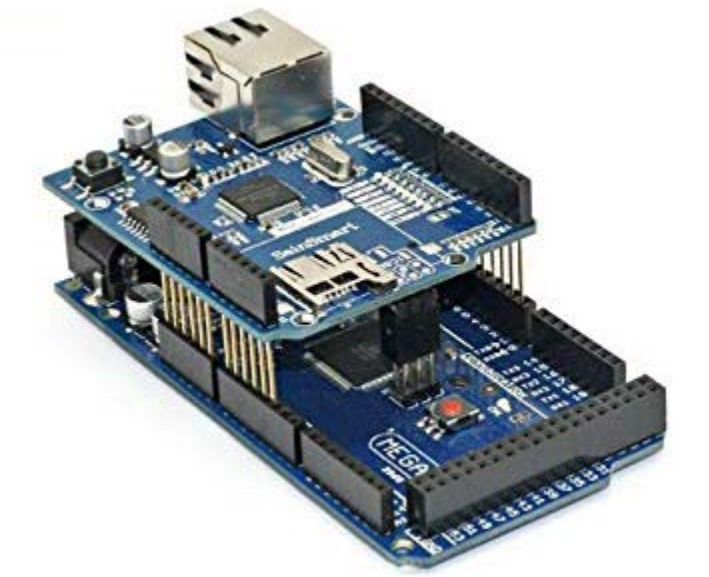


Ilustración 59 Arduino Mega 2560 con el shield Arduino Ethernet
Fuente: https://www.amazon.es/SainSmart-Arduino-Ethernet-Shield-Duemilanove/dp/B00I1W25KM

3.3.2. Conexión del sensor de temperatura DS18B20

El sensor digital de temperatura DS18B20 utiliza el protocolo 1-Wire para comunicarse. Necesitamos las librerías "OneWire.h" y "DallasTemperature.h" y un pin del Arduino Mega por el que vamos a comunicar los sensores. Vamos a leer la temperatura de cada sensor a través de la dirección del dispositivo que tiene 64 bits y viene establecida de fábrica. Para detectar la dirección del dispositivo conectamos el sensor a un pin del Arduino Mega como se ve en la ilustración 60 y se ejecuta el código que se muestra en el apartado 8.1. "Anexo 1. Obtener dirección del sensor DS18B20".

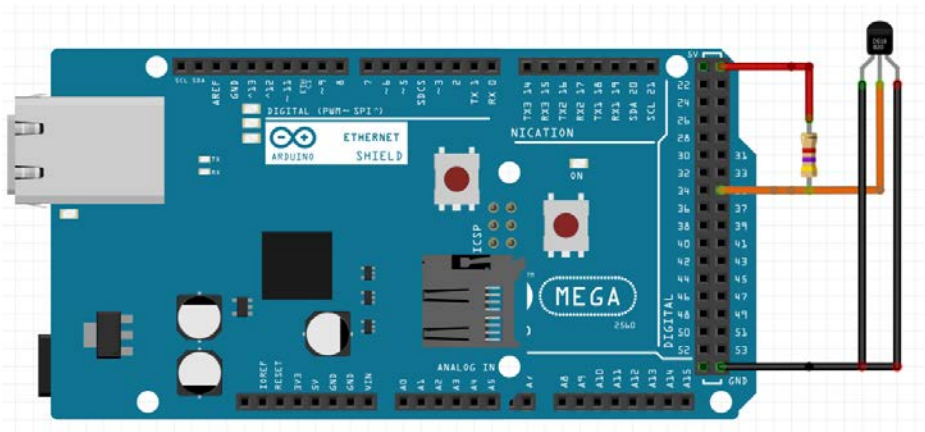


Ilustración *60* Conexión DS18B20
Fuente: Elaboración propia

En este proyecto vamos a utilizar dos sensores de temperatura, uno para medir la temperatura del agua y otro para la del sustrato. En la ilustración 61 vemos como conectar los dos sensores DS18B20 al pin 5 de la placa Arduino Mega.

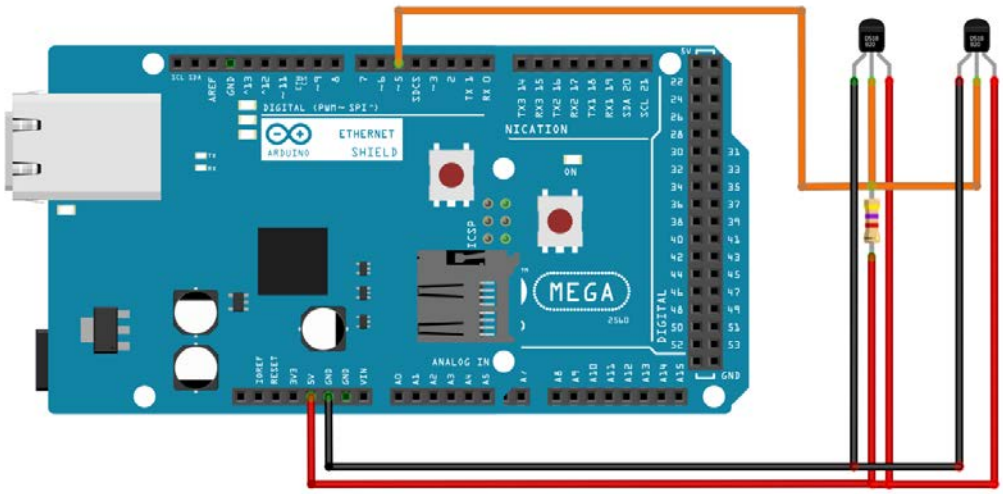


Ilustración *61* Esquema conexión de dos sensores DS18B20 en un Arduino Mega
Fuente: *Elaboración propia*

Como se ve en el esquema, es necesario conectar una resistencia Pull-Up de 4,7KΩ.

Una vez instaladas las librerías y realizado el montaje anterior ya podemos realizar las lecturas de temperaturas.

3.3.3. Conexión del sensor analógico de pH

Para conectar el sensor de pH al Arduino Mega es necesario una entrada analógica, en nuestro proyecto la A15, alimentación y dos GND pero que pueden ser el mismo. La sonda se conecta a la placa controladora a través de un conector BNC. La placa controladora dispone de 6 pines, como se ve en la ilustración 62:

To: Temperatura
Do: Señal límite de pH
Po: Valor de pH en V.
G: Masa del circuito analógico
G: Masa de alimentación
V+: Alimentación (5 V)

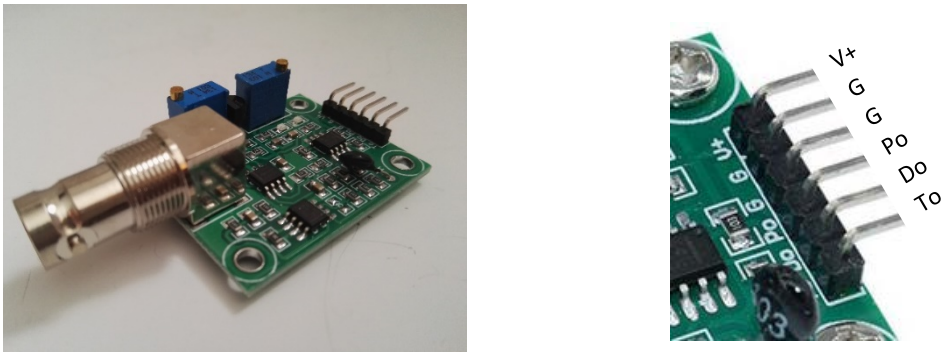


Ilustración *62* Placa del sensor analógico de pH
Fuente: *Propiedad del autor*

En este circuito solo necesitamos conectar la masa (G), la alimentación (V+) y la salida Po a una entrada analógica del Arduino Mega. En la ilustración 63 vemos como se conecta la sonda usando la entrada analógica A5 del Arduino Mega.

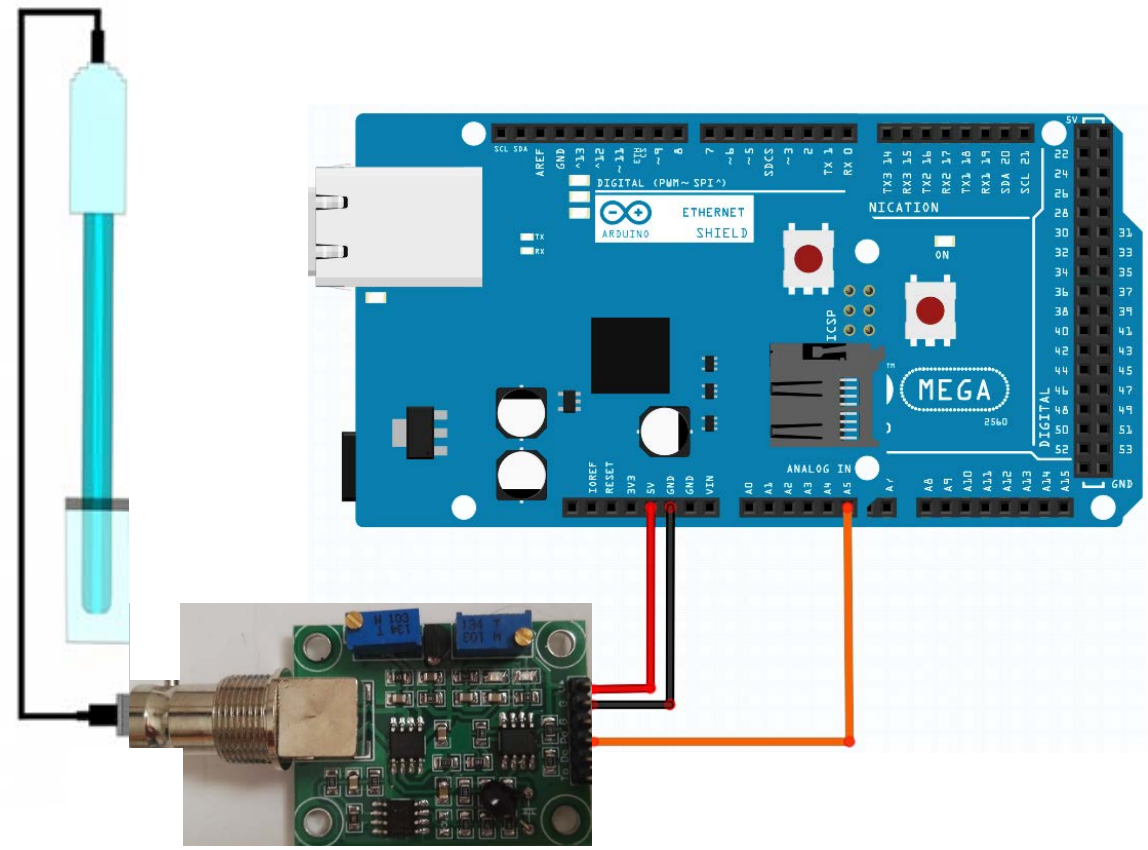


Ilustración *63* Esquema de conexión del Sensor analógico de pH
Fuente: *Elaboración propia*

En el apartado .2. “Anexo 2. Calibrar sensor de pH” se describen los pasos a seguir para el correcto ajuste de la sonda de pH.

3.3.4. Conexión del sensor nivel del agua

El funcionamiento del sensor de nivel es muy sencillo, cuando la boya se pone en posición horizontal, el circuito eléctrico se cierra.

La conexión del sensor de nivel es muy facil. Es igual que conectar un pulsador, se necesita conectar una resistencia a uno de los terminales. Esta sirve para mantener el estado en pull-down (a cero) si la resistencia va a masa como en nuestro caso o pull-up si la resistencia se conecta a la alimentación.

En nuestro proyecto el sensor de nivel de máximo se conecta al pin digital 16 y el de mínimo al pin digital 17. En la ilustración 64 se muestra el esquema de conexión.

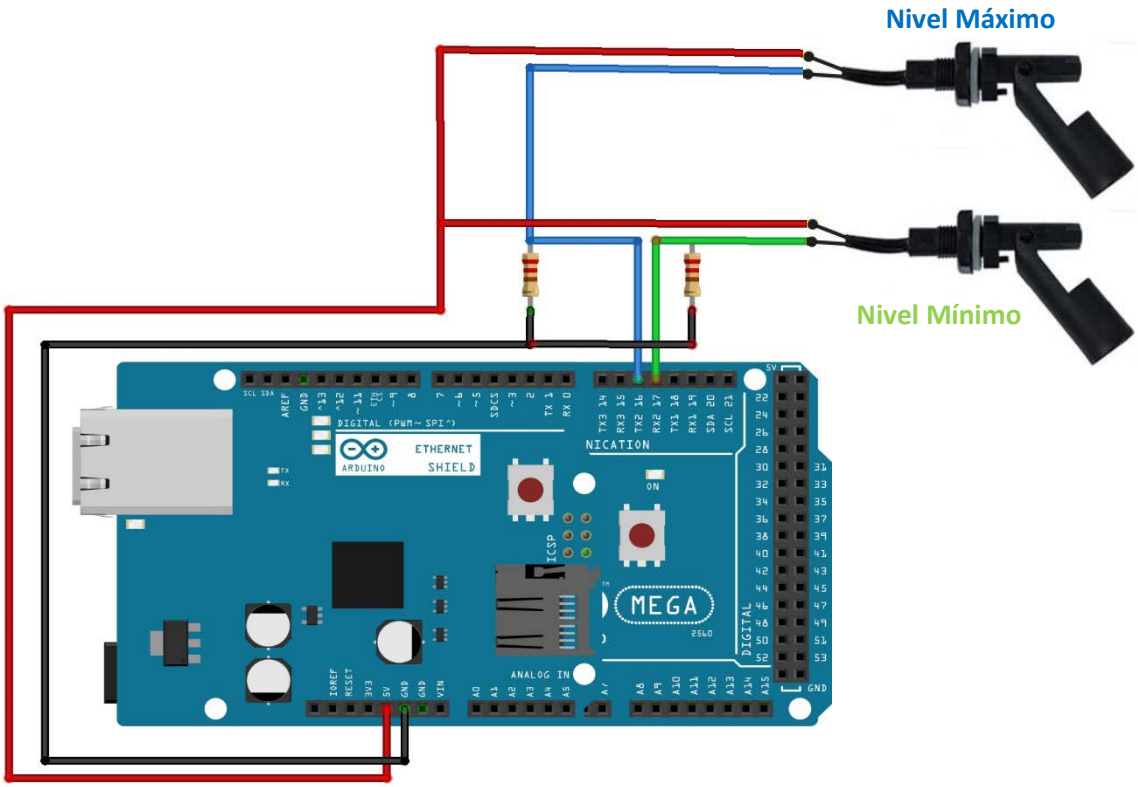


Ilustración *64* Esquema de conexión de los sensores de nivel
Fuente: *Elaboración propia*

3.3.5. Conexión del módulo de relés

El módulo que vamos a utilizar dispone de ocho entradas para controlar ocho relés. En la parte inferior derecha dispone de un jumper con tres pines que nos permite seleccionar si alimentar el módulo (los relés y los optoacopladores) de forma conjunta usando el puente entre los pines VCC y JD-VCC o usar una alimentación externa para el módulo, quitando el puente y conectándola a los pines Gnd y VCC. En este proyecto vamos a optar por la primera opción. Ver ilustración 65.

La conexión del módulo de relés al Arduino Mega se hace conectando los pines de alimentación, y luego conectando a cada una de las ocho entradas del módulo de relés un pin de salida digital del Arduino Mega, que serán los encargados de activar las bobinas de los relés correspondientes. En nuestro caso los pines del Arduino Mega utilizados son el 35, 37, 39, 41, 43, 45, 47 y 49.

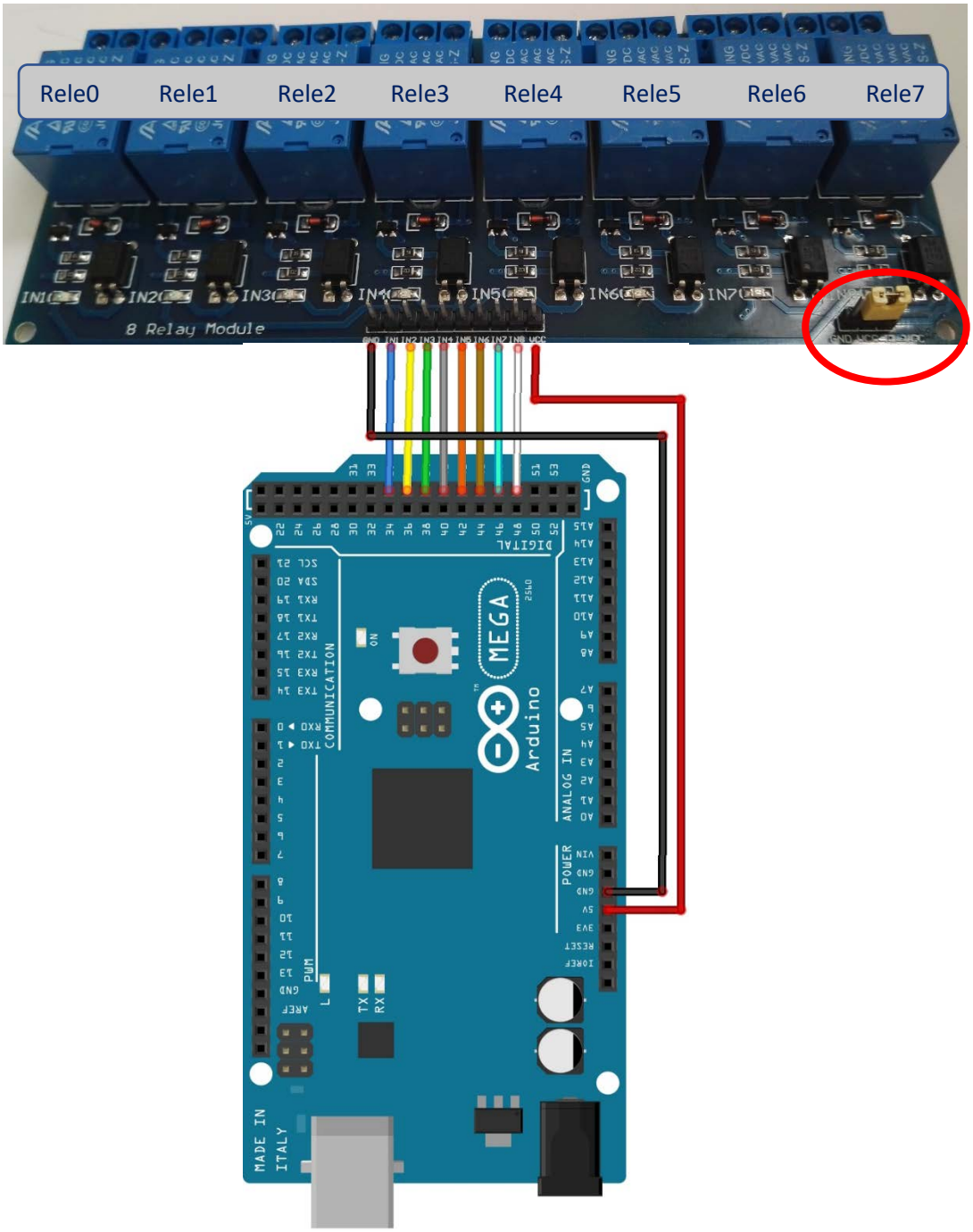


Ilustración *65* Esquema de conexión del módulo de relé
Fuente: *Elaboración propia*

3.3.6. Conexión del comedor

El servomotor funciona con una señal PWM, en nuestro caso vamos a utilizar el pin 3. Los cables en el conector están distribuidos de la siguiente forma: Rojo =Alimentación (+), Marrón = Alimentación (–) o tierra y Naranja= Señal PWM. En la ilustración 66 vemos el esquema de conexión del servomotor.

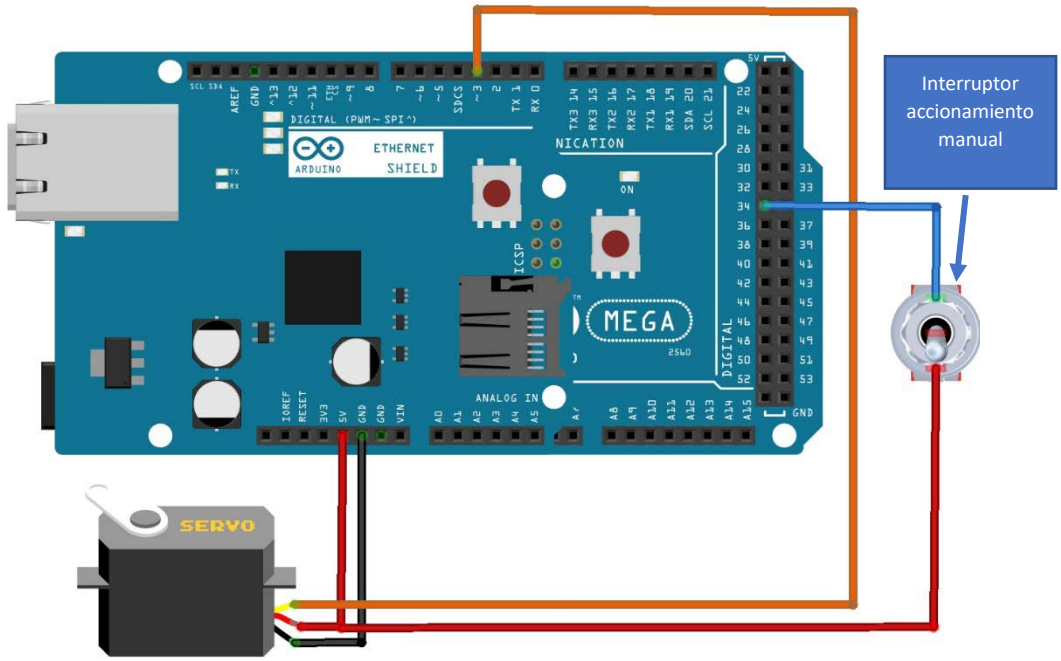


Ilustración 66 Esquema de conexión del servomotor
Fuente: *Elaboración propia*

3.3.7. Conexión del calentador, bombas de agua, iluminación y aireador

La relación de los pines de entrada, donde se conectan los pulsadores manuales, los pines de salida encargados de activar las bobinas de los relés, los actuadores y el relé al que se conecta, se muestra en la tabla 6.

Pin entrada	34	36	38	40	42	44	46	48
Pin salida	35	37	39	41	43	45	47	49
Relé	Rele0	Rele1	Rele2	Rele3	Rele4	Rele5	Rele6	Rele7
Actuador	No utilizado	Luz1	Luz2	Aireador	Calentador	Bomba ppal	Bomba llenar	Bomba vaciar

Tabla 6 Pines para conectar los pulsadores y los actuadores
Fuente: Elaboración propia

En la ilustración 67 vemos el esquema de conexión de los actuadores.

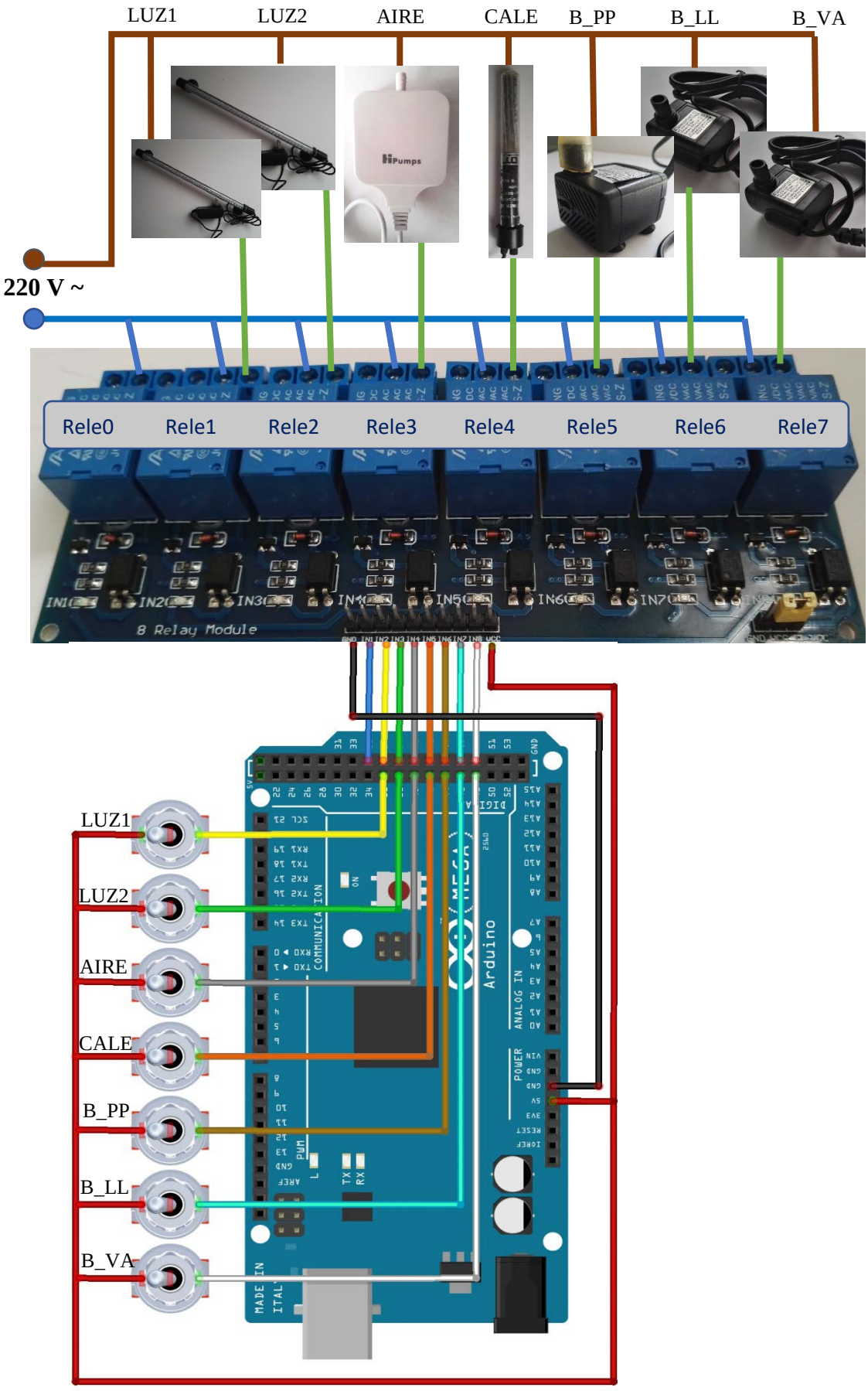


Ilustración *67* Esquema de conexión de los actuadores
Fuente: *Elaboración propia*

3.3.8. Conexión del display LCD204

Para conectar el display al Arduino Mega vamos a utilizar un controlador LCD I2C, que nos permite controlar el display a través del bus I2C utilizando solo dos pines del Arduino Mega. Se podría conectar directamente el display al Arduino Mega pero necesitaríamos usar varios pines de la placa Arduino con el consiguiente consumo de recursos

Para conectar el módulo adaptador LCD I2C al Arduino (24) solo necesitamos los pines SDA, SCL, GND y 5V. En la tabla 7 vemos los pines a utilizar en función del modelo de Arduino utilizado.

Adaptador LCD a I2C	Arduino Uno	Arduino Mega
GND	GND	GND
VCC	5 V	5 V
SDA	A4	20
SCL	A5	21

Tabla 7 Pines para conectar el LCD204
Fuente: Elaboración propia

En la placa Arduino Mega, los pines correspondientes son el 20 (SDA) y el 21 (SCL). Para usar el bus I^2C en Arduino tenemos la librería "Wire.h" que dispone de las funciones necesarias.

En algunas ocasiones el fabricante puede no facilitarnos la dirección del dispositivo, para determinarla podemos utilizar el sketch que se muestra en el apartado 8.3. "Anexo 3. Obtener dirección del sensor DS18B20".

En la ilustración 68 vemos el esquema de conexión del display LCD con el módulo adaptador LCM 1602 IIC al Arduino Mega 2560 con el shield Arduino Ethernet.

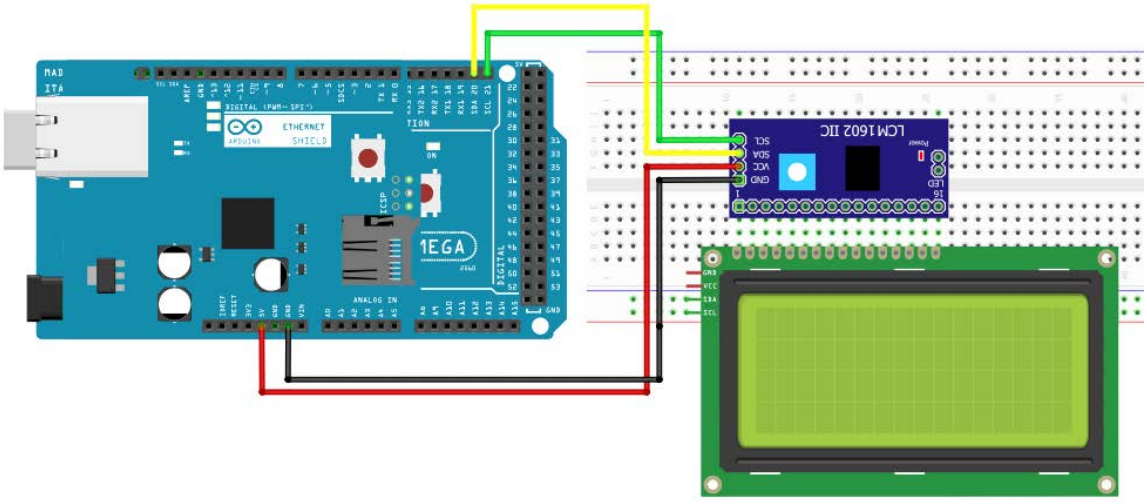


Ilustración 68 Esquema de conexión del Display LCD204 con módulo adaptador
Fuente: *Elaboración propia*

3.3.9. Esquema del montaje final del sistema

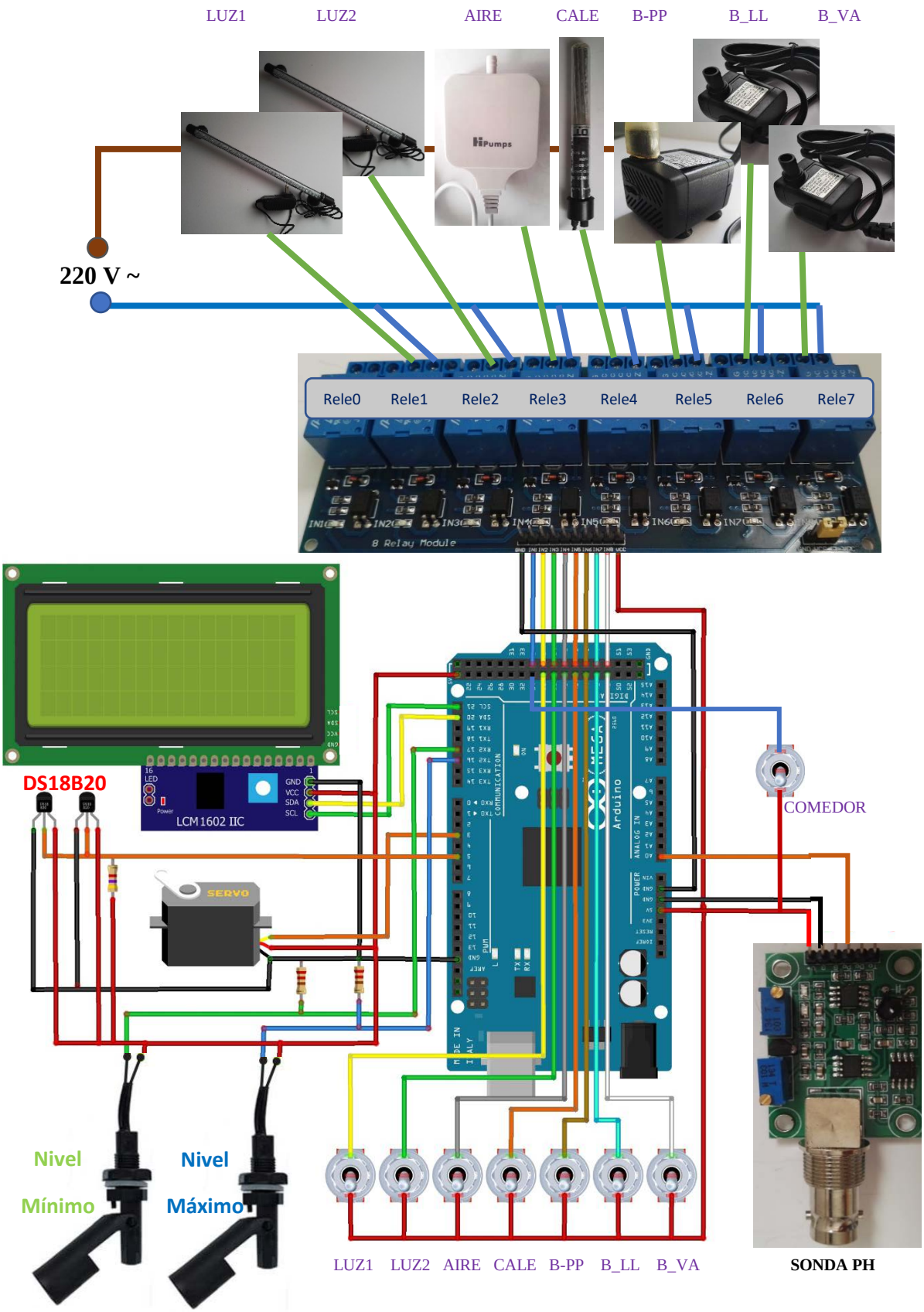


Ilustración *69* Esquema del montaje final
Fuente: *Elaboración propia*

En la tabla 8 se muestra a modo de resumen los pines que se usan del Arduino Mega y su funcionalidad.

Pin Arduino	Elemento conectado
3	Servomotor (Comedor)
4	Reservada para SD Card
5	Sensor de temperatura DS18B20
7	Led cambio de agua
10	SS. Controlador Ethernet
12	Interruptor modo funcionamiento Manual-Automático
16	Sensor horizontal de Nivel máximo
17	Sensor horizontal de Nivel mínimo
20	SDA. Para el protocolo I2C
21	SCL. Para el protocolo I2C
22	Led alarma nivel máximo
23	Led alarma nivel mínimo
24	Led alarma temperatura
25	Led alarma pH
34	Interruptor palanca para accionamiento comedor
35	Rele0. No utilizado
36	Interruptor palanca para accionamiento luz1
37	Rele1. Luz 1
38	Interruptor palanca para accionamiento luz2
39	Rele2. Luz 2
40	Interruptor palanca para accionamiento aireador

41	Rele3. Aireador
42	Interruptor palanca para accionamiento calentador
43	Rele4. Calentador
44	Interruptor palanca para accionamiento bomba ppal
45	Rele5. Bomba principal
46	Interruptor palanca para accionamiento bomba llenar
47	Rele6. Bomba de llenado
48	Interruptor palanca para accionamiento bomba vaciar
49	Rele7. Bomba de vaciado
50	MISO. Para el Protocolo SPI
51	MOSI. Para el Protocolo SPI
52	SCK. Para el Protocolo SPI
53	Hardware SS. Para el protocolo SPI. No se utiliza
A0	Sensor analizador de pH

Tabla 8 Resumen de los pines usados en el Arduino Mega 2560
Fuente: Elaboración propia

3.4. Desarrollo del software del sistema

Para la elaboración de este proyecto se han utilizado las siguientes herramientas software.

3.4.1. Arduino

Es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores.

Ilustración 70 Logo de Arduino
Fuente: https://www.arduino.cc/en/Trademark/CommunityLogo/

Cuando hablamos de "Arduino" deberíamos especificar el modelo concreto, ya que se han fabricado diferentes modelos de placas Arduino oficiales. A pesar de las varias placas que existen todas pertenecen a la misma familia (microcontroladores AVR marca Atmel), esto significa que comparten la mayoría de sus características de software, como arquitectura, librerías y documentación.

El software Arduino (IDE) se puede ejecutar en Macitosh, Windows y Linux y el lenguaje se puede expandir a través de bibliotecas de C++. Para realizar un programa en Arduino lo primero que tenemos que hacer es descargar de la página web de Arduino la última versión del entorno de desarrollo (IDE), que es la herramienta para desarrollar las aplicaciones. El IDE de Arduino dispone de una interfaz sencilla en la que se distinguen cinco zonas, ver ilustración 71:

- **Menú**. Es una barra situada en la parte superior donde se encuentran las diferentes funciones.
- **Menú de acceso rápido**. Es la barra situado debajo del menú que dispone de varios botones con las funciones más utilizadas
- **Editor de texto**. Donde se edita el código.
- **Consola**. Muestra la información detallada de la actividad del IDE.
- **Panel de mensajes**. Panel donde se muestran los mensajes de error o de compilación.

El código se estructura en dos partes principales:

- **Setup**. Es la parte del código que se ejecuta una sola vez al inicio del programa
- **Loop**. Es el código que se ejecuta en bucle.

Una vez instalado el programa, lo siguiente antes de empezar a programar es configurar la comunicación entre la placa Arduino a utilizar y el PC.

Ilustración 71 Zonas del IDE de Arduino
Fuente: Captura de pantalla

En el apartado 8.4 “Anexo 4. Programación en Arduino” podemos ver la información de forma más detallada.

En el apartado 8.7 “Anexo 7. Código fuente del Arduino” podemos ver el código de programación realizado para este proyecto.

3.4.2. Thinger.io

Es una plataforma para trabajar con IoT. La Cloud Console nos va permitir controlar, configurar y administrar los dispositivos, así como almacenar y visualizar la información en la nube.

Ilustración 72 Logo de Thinger.io
Fuente: https://thinger.io/

El primer paso es instalar la aplicación desde su página web. Una vez que se inicie la sesión accedemos a la página principal donde se muestra información básica de nuestra cuenta. En el lado izquierdo se encuentra el menú principal, que contiene todas las funciones de la plataforma que se necesitan para crear el proyecto. Ver ilustración 73.

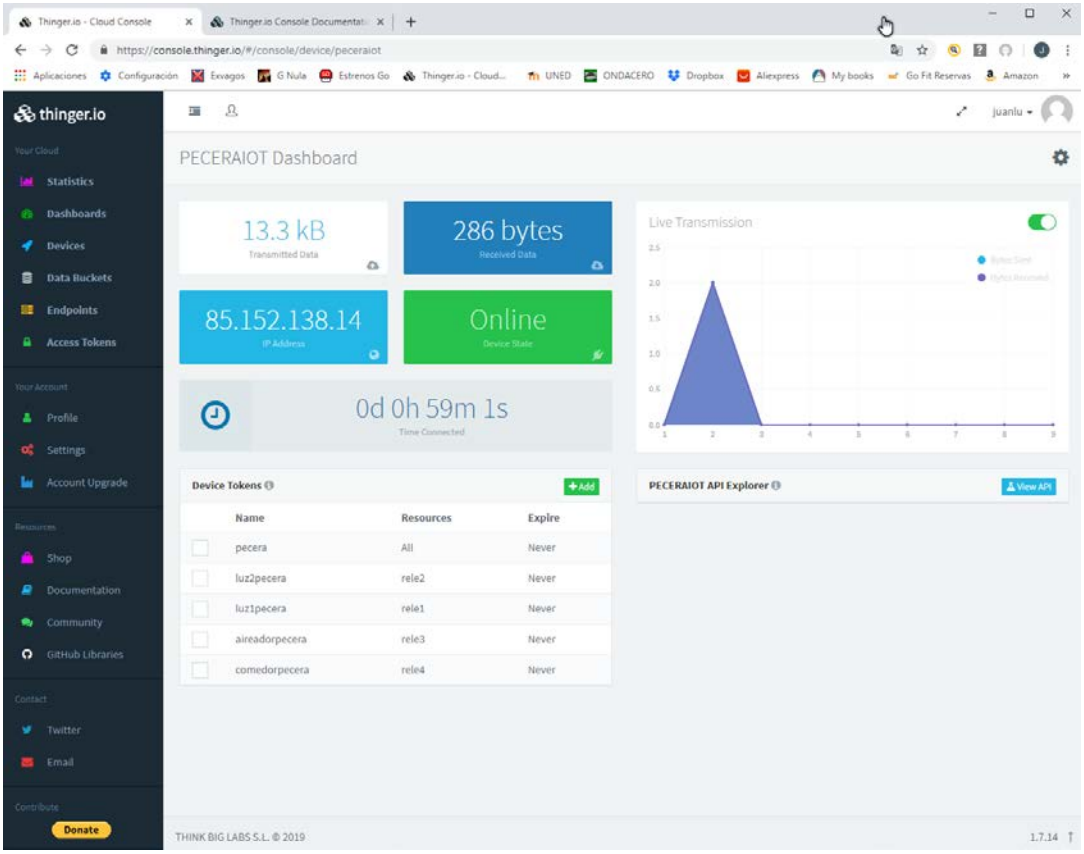


Ilustración 73 Pantalla principal de Thinger.io
Fuente: Captura de pantalla

3.4.2.1 Menú Devices

Para iniciar un proyecto de IoT en Thinger.io hay que acceder al menú Devices para crear un nuevo dispositivo, que le otorgará acceso para conectarlo a nuestra cuenta. Cualquier dispositivo en Thinger.io debe estar registrado para obtener acceso a la nube, ver ilustración 74. Cada uno tiene su propio identificador y credenciales y está relacionado con la cuenta de usuario.

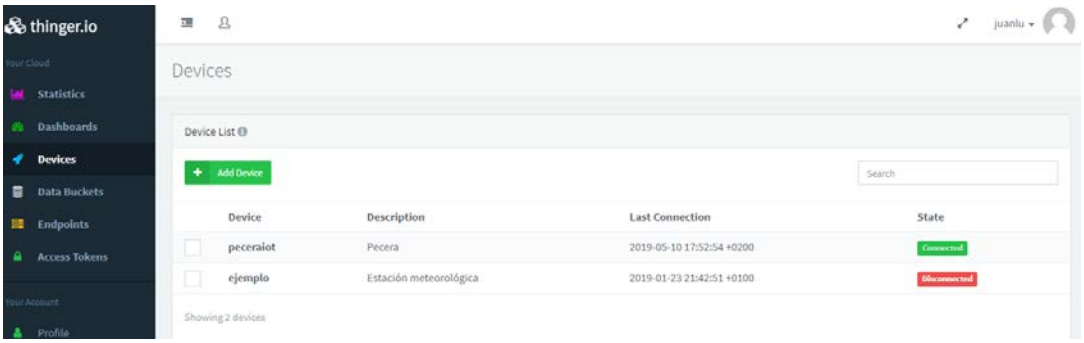


Ilustración 74 Pantalla principal de Thinger.io
Fuente: Captura de pantalla

3.4.2.2. Menú Data Buckets

Este menú nos permite configurar los parámetros para crear una base de datos que almacene los valores enviados por nuestro Arduino. En la ilustración 75, podemos ver parte de los datos almacenados en nuestro proyecto.

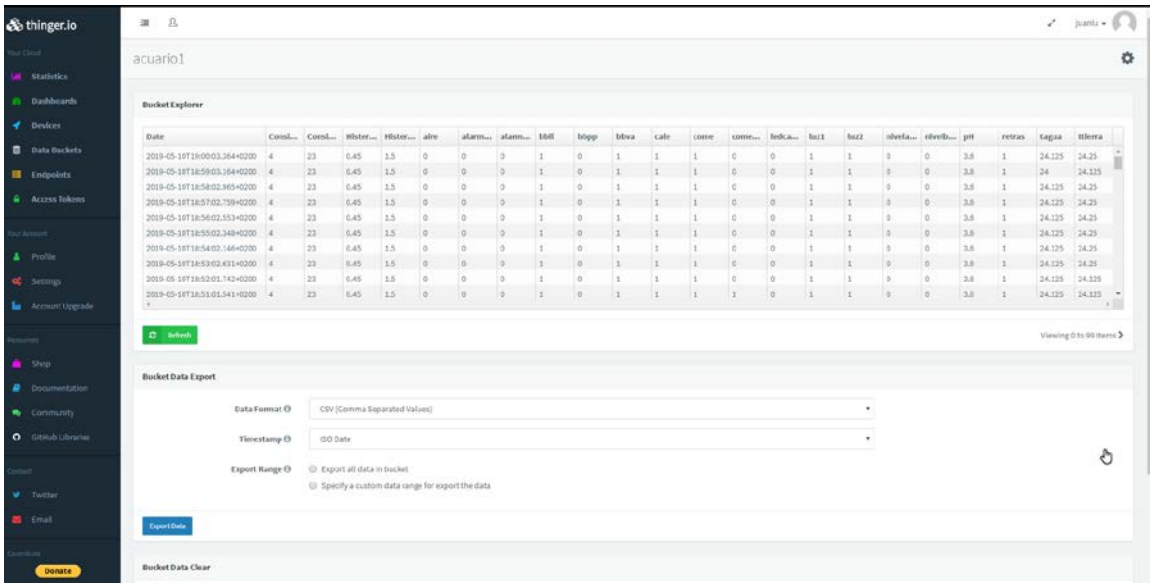


Ilustración 75 Datos del Data Bucket acuario 1
Fuente: Captura de pantalla

3.4.2.3. Menú Dashboard

Un dashboard es una interfaz gráfica de usuario que permite mostrar información en diferentes figuras y gráficos. Se puede configurar los dashboards con diferentes widgets, configurar su diseño, dimensión, color y fuentes de datos.

Los dashboards pueden mostrar información en tiempo real desde sus dispositivos o usar información histórica almacenada en grupos de datos que se actualiza a intervalos predefinidos. Los dashboards los vamos a utilizar no solo para mostrar datos, sino también para actuar en tiempo real sobre los dispositivos que hemos conectado.

En las ilustraciones 76, 77 y 78 se muestran los tres dashboards creados para este proyecto.

Ilustración 76 Dashboard dashpecera1
Fuente: Captura de pantalla

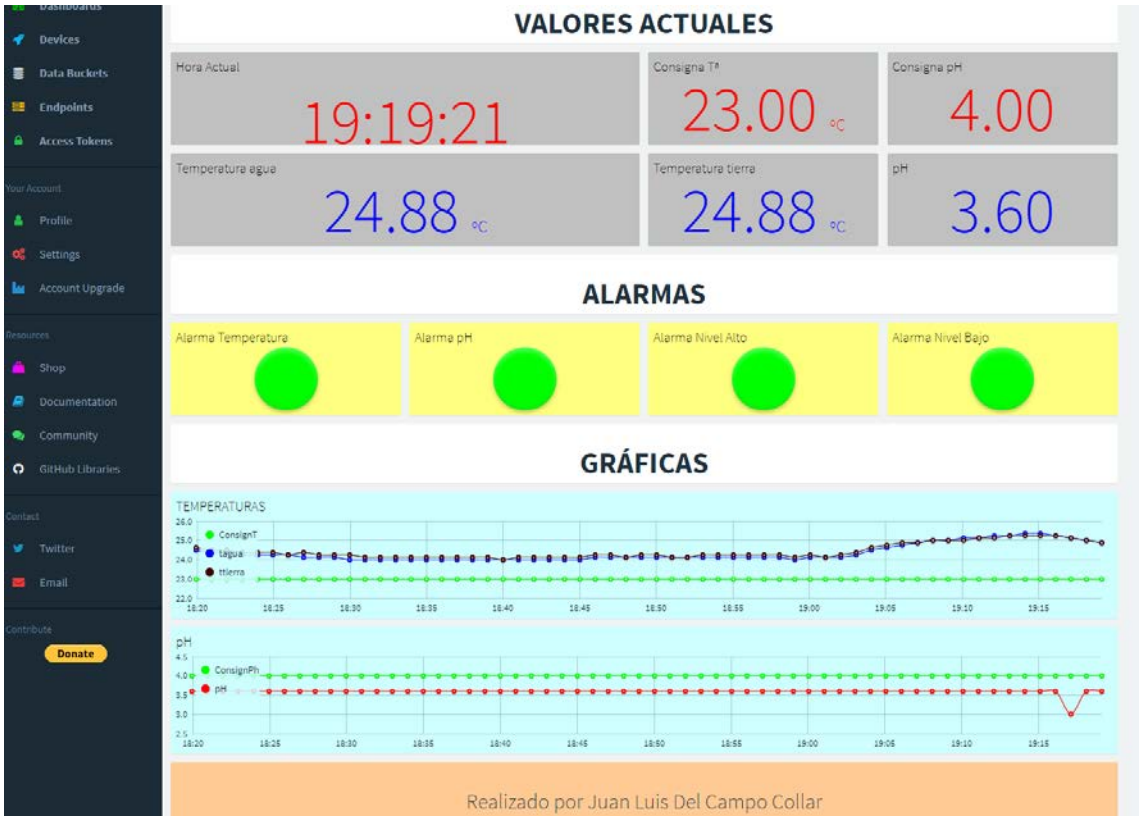


Ilustración 77 Dashboard dashpecera2
Fuente: Captura de pantalla

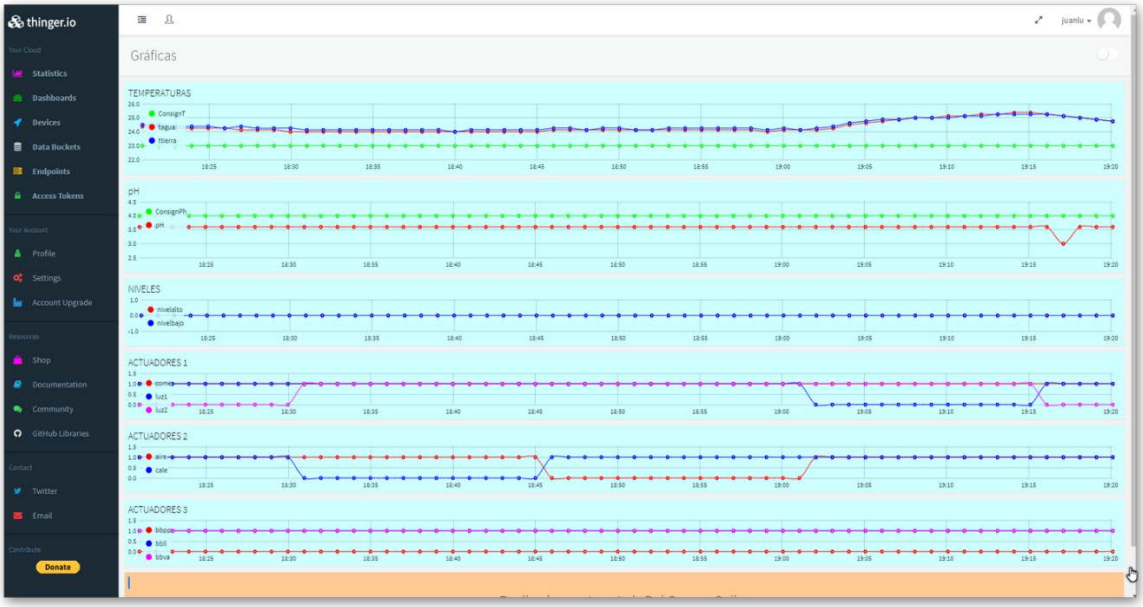


Ilustración 78 Dashboard dashpecera3
Fuente: Captura de pantalla

3.4.2.4. Menú Endpoints

Un endpoint, es el punto de entrada a un servicio, un proceso o cualquier otro destino. En Thinger.io, un endpoint puede definirse como un destino objetivo al que los dispositivos pueden llamar para realizar cualquier acción, como enviar un correo electrónico, enviar un SMS, llamar a una API REST, interactuar con IFTTT, llamar a un dispositivo desde otra cuenta, o llamar a cualquier otro punto final HTTP.

En nuestro proyecto hemos utilizado los endpoints para enviar por correo electrónico un aviso cuando se produce una alarma de temperatura, pH o de nivel. En la ilustración 79 vemos la configuración del endpoint EndpNipecera.

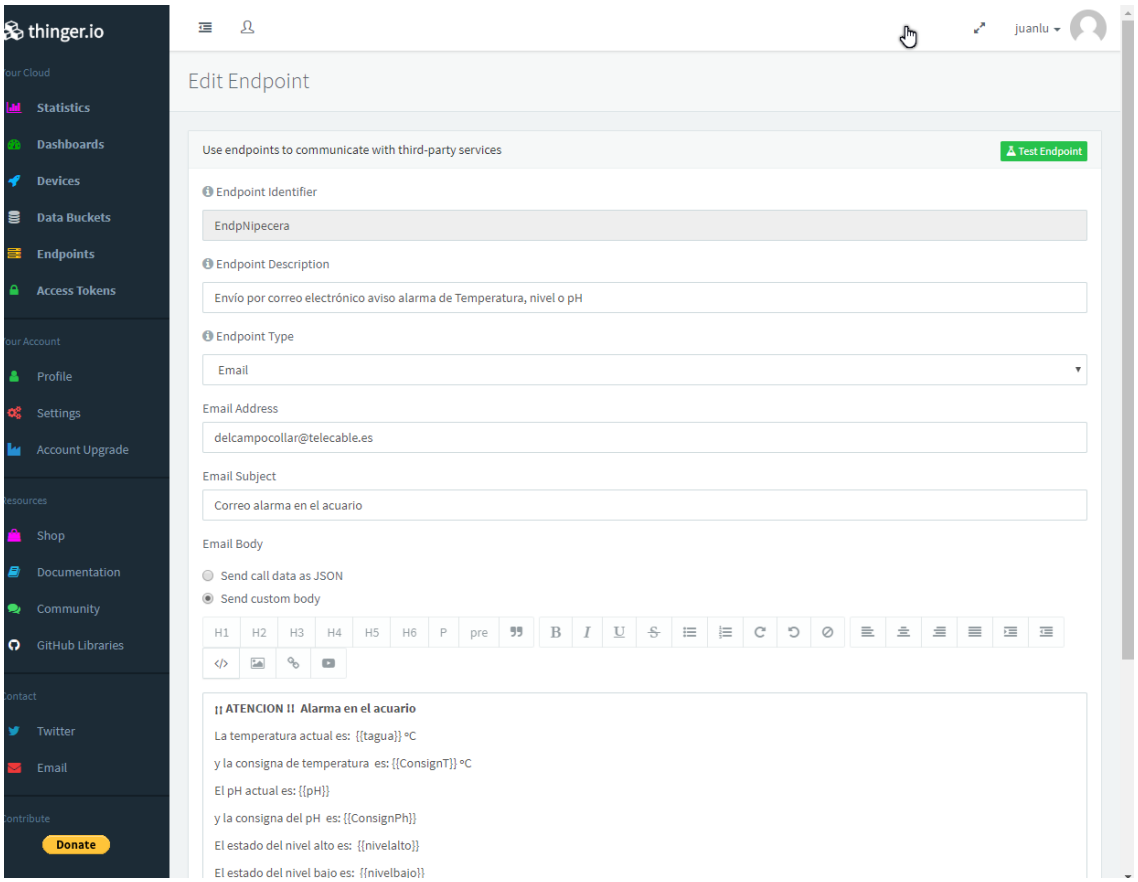


Ilustración 79 Endpoint EndpNipecera
Fuente: Captura de pantalla

En la ilustración 80 vemos una captura de pantalla del móvil donde se muestra el mensaje recibido por una alarma de temperatura.

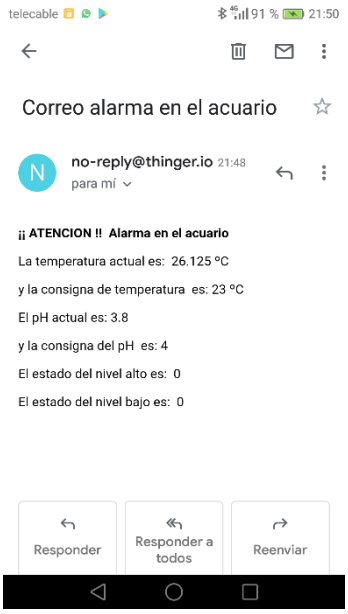


Ilustración 80 Correo por alarma de temperatura
Fuente: Captura de pantalla

3.4.2.5. Access Tokens

Se puede acceder a todos los servicios de Thinger.io a través de las llamadas a la API REST. Cada solicitud de API REST se debe de autenticarse para que surta efecto, por lo que en cada llamada hay que proporcionar el código de autorización. Los access tokens son la forma de proporcionar acceso a los servicios o aplicaciones que hemos creado a terceras personas, sin tener que compartir el nombre de usuario y la contraseña.

En la realización de este proyecto se han definido cuatro access tokens como se puede ver en la ilustración 81, que permiten acceder al dashpecera1, dashpecera2, dashpecera3 y al bucket acuario1.

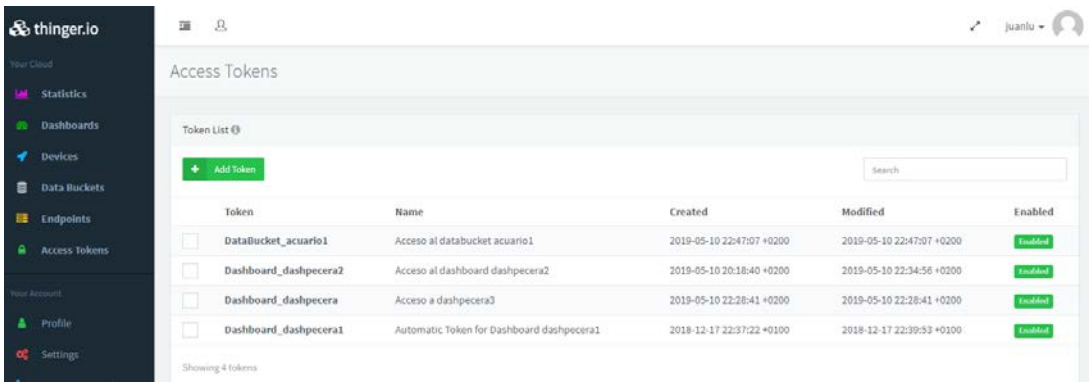


Ilustración 81 Correo por alarma de temperatura
Fuente: Captura de pantalla

3.4.2.6. Aplicación móvil

La aplicación móvil de la plataforma Thinger.io IoT está disponible en Google Play y App Store y está lista para descargarse. Esta aplicación nos va a permitir administrar los diferentes elementos de la plataforma en un Smartphone.

Lo primero es escanear el token QR del dispositivo en el ícono de QR, en la parte superior de la pantalla principal y apuntando con la cámara del móvil al QR. Ahora, la aplicación muestra los dispositivos escaneados en la pantalla principal. Seleccionamos el dispositivo y la información se nos muestra en la pantalla como se muestra en la ilustración 82.

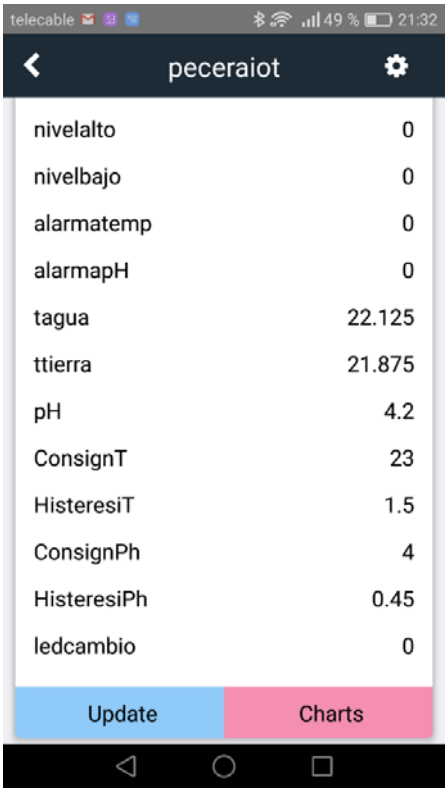


Ilustración 82 Información en la pantalla del móvil.
Fuente: Captura de pantalla del móvil

La aplicación móvil también nos va a permitir monitorizar los recursos en tiempo real con distintos tipos de gráficos como se muestra en la ilustración 83.

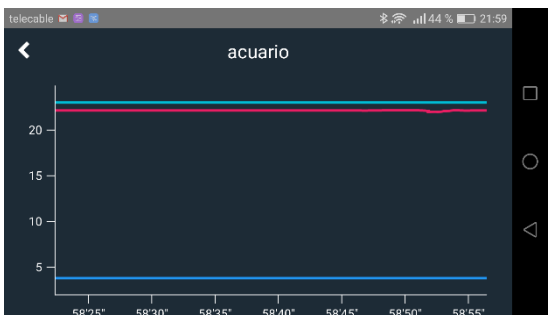


Ilustración 83 Gráfico de la aplicación móvil.
Fuente: Capturas de pantalla del móvil

3.4.2.7. Thinger.io y Arduino

El primer paso para comenzar a construir dispositivos thinger.io es instalar las bibliotecas requeridas en el IDE de Arduino para admitir la exposición de recursos del dispositivo como valores de sensores, luces, relés, etc. (25). Se requiere una versión de Arduino superior a la 1.6.3.

Para conectar el Arduino Mega 2560 a través del Arduino ethernet Shield hay que instalar la librería e incluir el siguiente código en la programación del Arduino

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

ThingerEthernet thing("username", "deviceId", "deviceCredential");

void setup() {
}

void loop() {
  thing.handle();
}
```

En general en el void setup() se debe de definir cualquier recurso del dispositivo, inicializar los dispositivos, configurar la dirección de entrada/salida de un pin digital, inicializar la velocidad del puerto serial, inicializar recursos, et.

El loop()es el lugar para llamar a la función thing.handle(), por lo que las bibliotecas de thinger pueden manejar la conexión con la plataforma. Este es el lugar también para llamar a sus puntos finales o transmitir datos en tiempo real a un WebSocket abierto. Ver ilustración 84

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

// initialize Thinger instance (type can change depending on your device)
ThingerEthernet thing("username", "deviceId", "deviceCredential");

void setup() {
    // initialize your sensors and pins

    // initialize wifi (see examples for your device)

    // add resources here, like sensors, lights, etc.
}

void loop() {
  // call always the thing handle in the loop and avoid any delay here
  thing.handle();
  // here you can call endpoints
  // and also you can stream resources
}
```

Ilustración 84 Logo de Thinger.io
Fuente: http://docs.thinger.io/arduino/#coding-sketch-overview

Para controlar o activar el dispositivo IoT, es necesario definir recursos de entrada y de salida. Un recurso de entrada es cualquier cosa que pueda proporcionar información a su dispositivo. Por ejemplo, puede ser un recurso para encender y apagar una luz o un relé, cambiar una posición de servo, ajustar un parámetro del dispositivo, etc. Los recursos de salida se deben usar en general cuando se necesita detectar o leer un valor del sensor, como la temperatura, la humedad, etc. La función de recursos de entrada/salida toma un parámetro de tipo pson que es un tipo variable que puede contener valores booleanos, números, flotantes, cadenas o incluso información estructurada como en un documento JSON. Para definir un recurso de entrada/salida, se usa el operador "<< " o ">>". En la ilustración 85 se muestran varios ejemplos de cómo definir un recurso de entrada/salida.

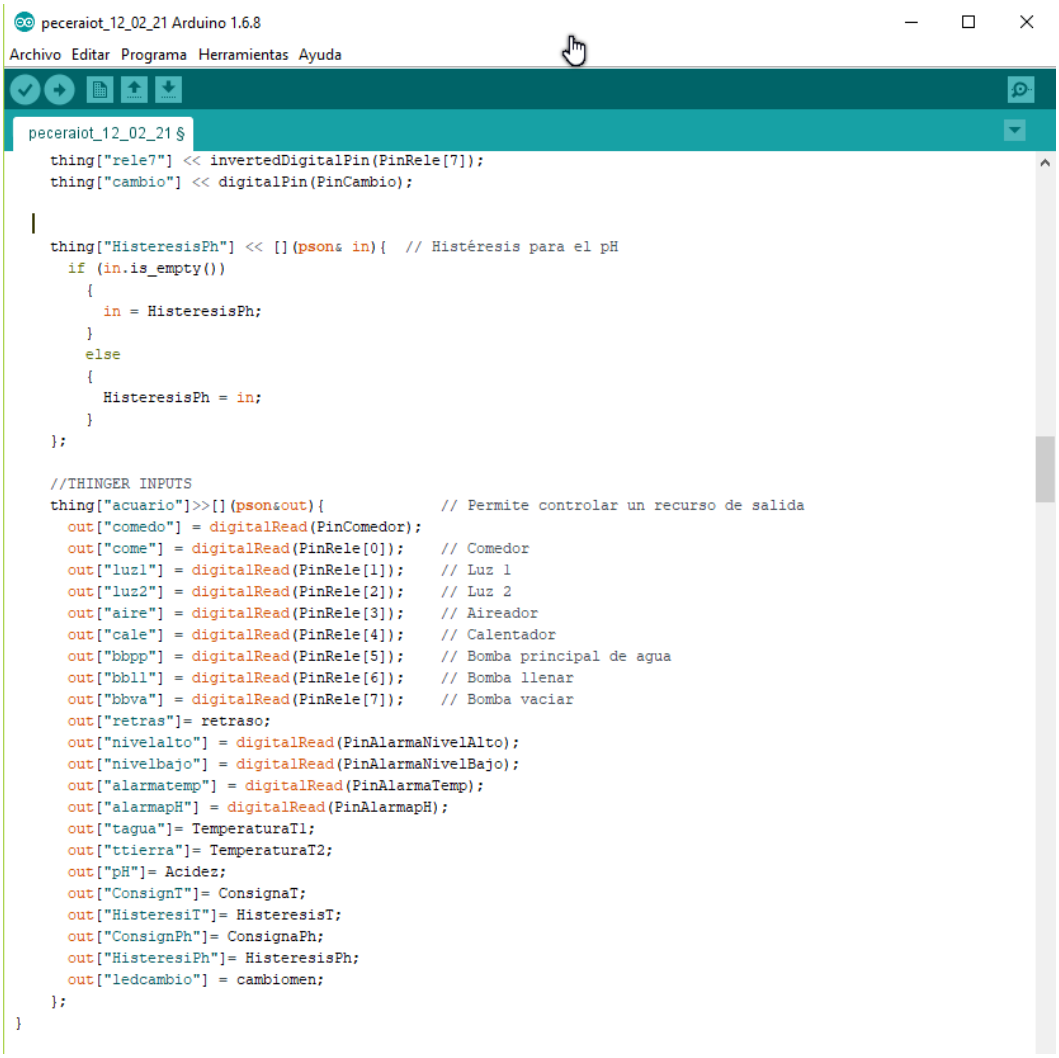


Ilustración 85 Recursos de entrada
Fuente: Captura de pantalla

En el apartado 8.5 "Anexo 5. Configuración plataforma Thinger.io" podemos ver la información de forma más detallada.

3.4.3. IFTTT

Es una plataforma que conecta diferentes aplicaciones web o servicios de internet. El significado de IFTTT es “If This Then That”. Nos va a permitir configurar los diferentes eventos que desencadenaran una reacción en nuestro proyecto. Su finalidad es que cuando suceda algo lance un proceso.

En IFTTT puedes relacionar dos plataformas y establecer una tarea para esa conexión, a esto se le llama crear «recetas», que es la base de esta herramienta (ver ilustración 86). Se pueden crear tantas combinaciones y órdenes como se quiera, lo que nos va a permitir automatizar tareas desde las más sencillas a las más complejas. Para lograr esta automatización se necesitan varias partes:

- **Triggers.** Acción que desencadena la reacción.
- **Ingredientes**. Sub-acción que permiten personalizar los requisitos para lanzar una reacción concreta.
- **Recetas**. Nombre que recibe la combinación acción-reacción establecida entre distintos sitios.
- **Canales**. Son los sitios que soporta la plataforma y que podemos interconectar.

Los pasos para utilizar IFTTT son:

- **Crear una cuenta**. Para ello hay que registrarse, de forma gratuita, en su página web y confirmar el correo que te envíen.
- **Activar canales**. Activar y conceder permiso a las plataformas que queremos acceder
- **Crear la receta**. Consiste en indicar una condición (trigger), que se produce en uno de los sitios aceptados, que haga que se ejecute una acción (action) en otro sitio.

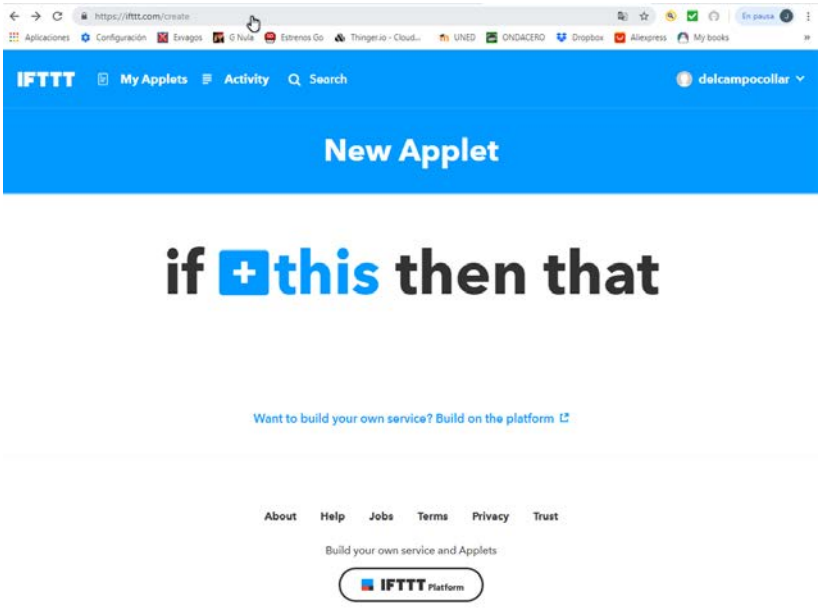


Ilustración 86 Pantalla nueva receta de IFTTT
Fuente: Captura de pantalla

En el apartado 8.6 “Anexo 6 Configuración plataforma IFTTT” podemos ver la información de forma más detallada.

3.4.4. Fritzing

Es una iniciativa de hardware de código abierto que hace que la electrónica sea accesible a cualquier persona. Ofrece una herramienta de software que permite a los usuarios documentar sus prototipos, compartirlos y diseñar y fabricar pcbs.

Ilustración 87 Logo de fritzing
Fuente: http://fritzing.org/home/

Para utilizarlo, lo primero que hay que hacer es descargar el software de su página e instalarlo. Una vez abierto Fritzing, nos aparece la pantalla principal que podemos ver en la ilustración 88, y en la que se distinguen cuatro partes que vamos a describir a continuación.

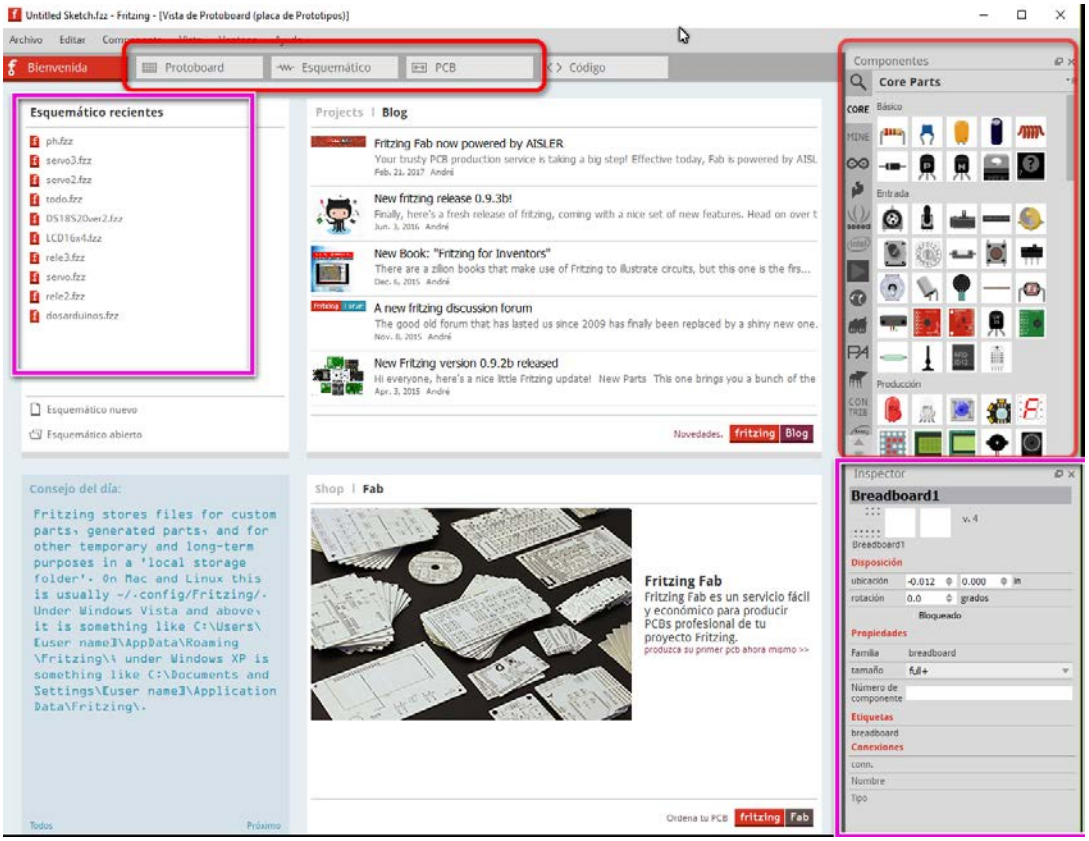


Ilustración 88 Zonas de la pantalla principal de IFTTT
Fuente: Captura de pantalla

- **Vistas disponibles**. Fritzing nos permite trabajar en nuestro proyecto en tres vistas diferentes, que son: Protoboard, Esquema y PCB.
- **Nuestros proyectos**. Acceso directo a los proyectos más recientes.
- **Lista de componentes**. Acceso a las distintas bibliotecas de componentes para montar nuestro circuito
- **Inspector de partes**. Nos permite acceder al detalle de cada componente que seleccionemos.

Para comenzar a trabajar seleccionamos uno de los tres tipos de vista.

3.4.4.1. Desarrollo de un proyecto en Vista Protoboard

Nos permite diseñar el circuito como si estiviesemos utilizando una placa protoboard. Lo único que tenemos que hacer es arrastrar los componentes hacia la protoboard. Para realizar el cableado hay que hacer click en los pines de los componentes y arrastrar hasta donde queramos conectarlo. Al hacer click derecho podemos cambiar el color del cable.

En la ilustración 89 vemos el diseño de la conexión del servomotor

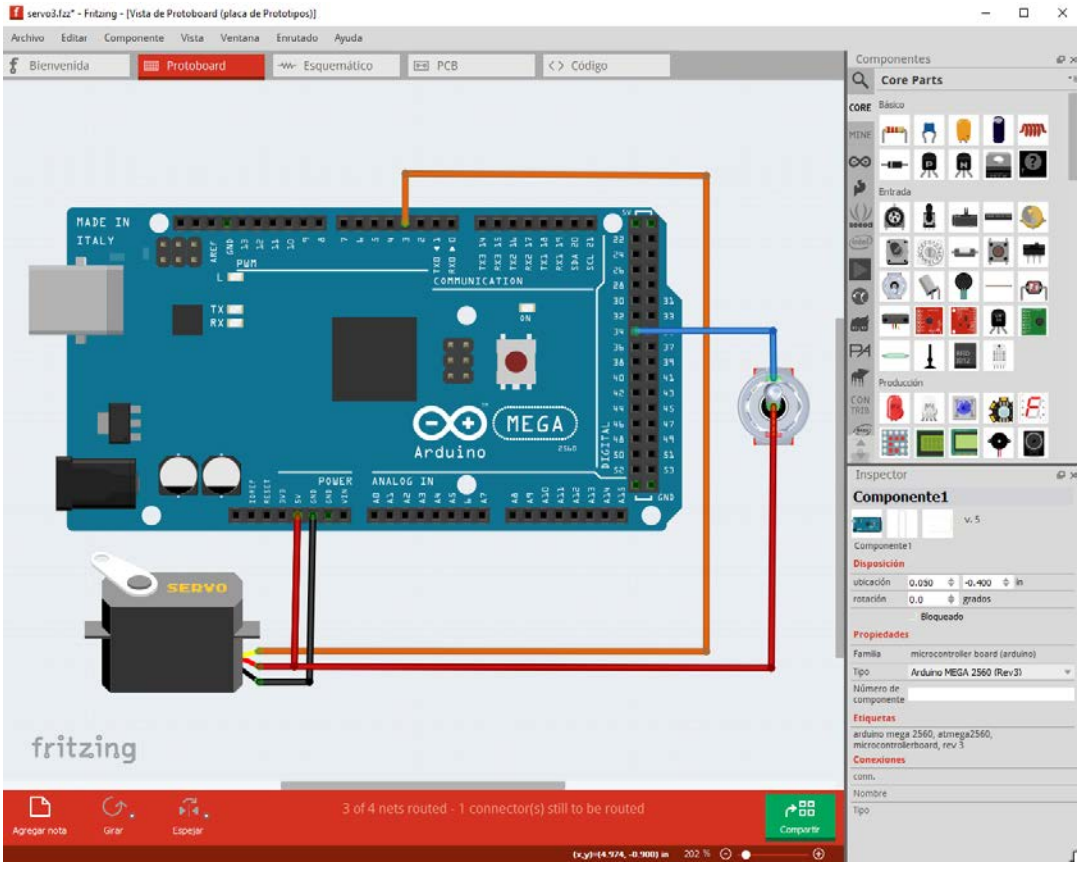


Ilustración 89 Vista protoboard. Ejemplo servomotor
Fuente: Captura de pantalla

3.4.4.2. Desarrollo de un proyecto en Vista esquema

En esta vista vamos a trabajar utilizando los símbolos de cada elemento. El modo de trabajar es igual que en la vista protoboard. En la ilustración 90 vemos el diseño de la conexión del servomotor

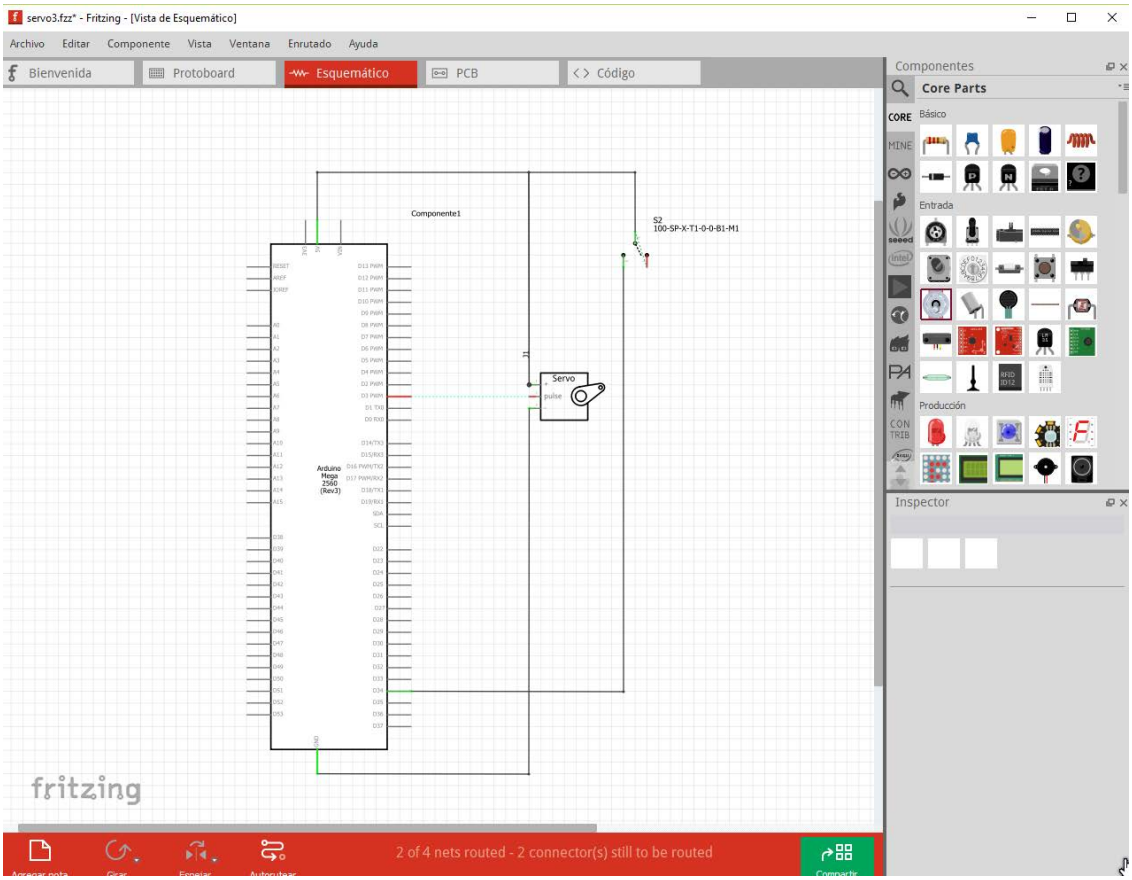


Ilustración 90 Vista esquema. Ejemplo servomotor
Fuente: Captura de pantalla

3.4.4.3. Desarrollo de un proyecto en Vista PCB

En esta vista vamos se muestra cómo queda nuestro circuito para imprimir en PCB.

En la parte inferior aparece un nuevo menú que nos permite ver las diferentes capas de nuestro circuito, autoroutear e incluso exportar para PCB.

Una vez terminado nuestro PCB se pueden exportar los archivos en PDF, SVG o Gerber para crear nuestro circuito físicamente. Existe la posibilidad de mandarlo fabricar con Fritzing y ahí mismo se hace un presupuesto del costo por impresión (26). El modo de trabajar es igual que en la vista protoboard.

En la ilustración 91 vemos el diseño de la conexión del servomotor

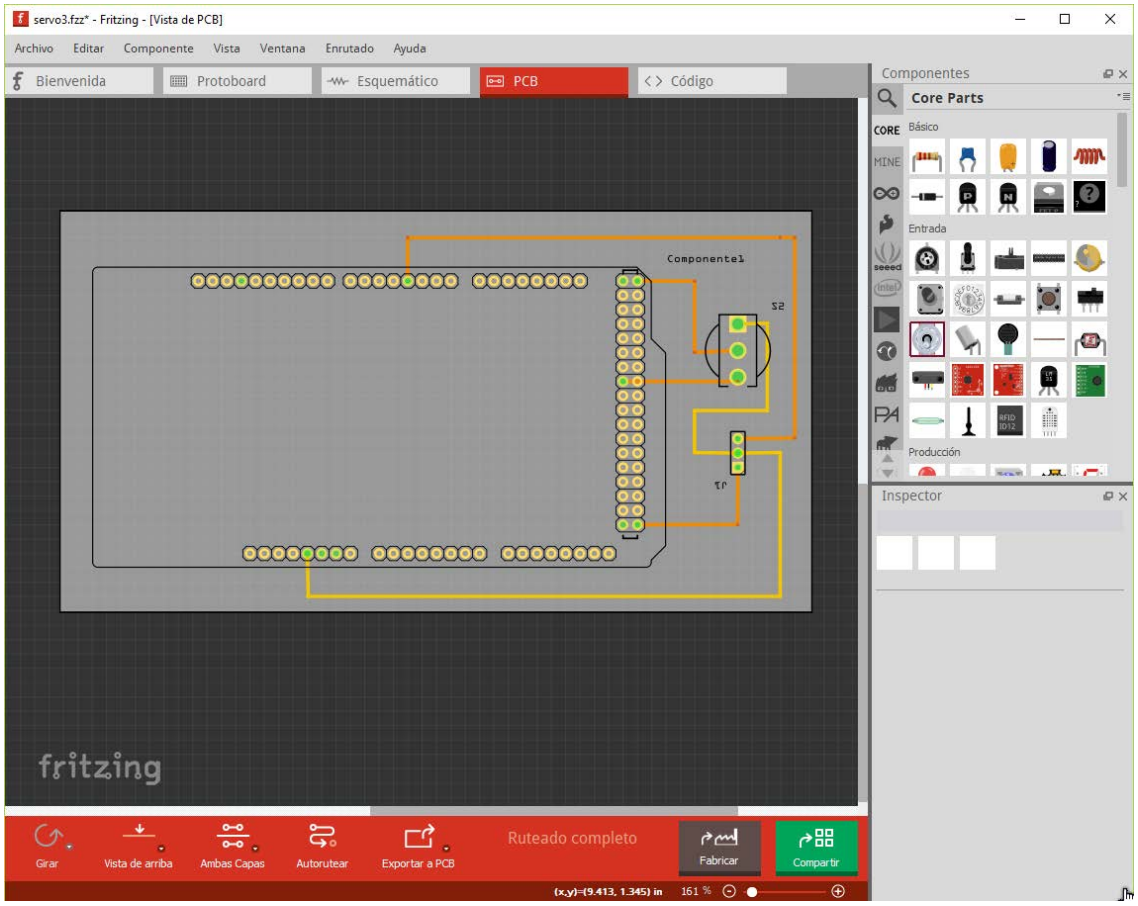


Ilustración 91 Vista PCB. Ejemplo servomotor
Fuente: Captura de pantalla

3.5. Prototipo diseñado

Para el desarrollo de este proyecto se ha diseñado el prototipo que se muestra en la ilustración 92. Tiene unas dimensiones de 220 x 293 x 325 mm.

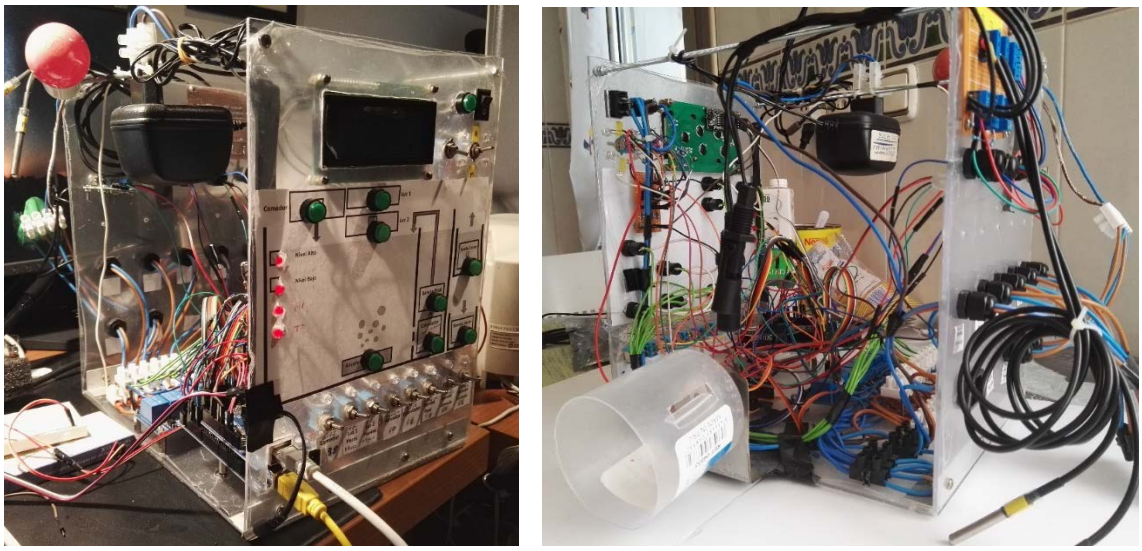


Ilustración 92 Vistas del prototipo deseñado
Fuente: Propiedad del autor

El prototipo ha sido diseñado para ir en un estante del mueble sobre el que está el acuario, como se ve en la ilustración 93.

Ilustración 93 Acuario instalado sobre el mueble
Fuente: Propiedad del autor

En la parte frontal del prototipo, ver ilustración 94, podemos distinguir tres partes:

- Una parte superior con un display lcd que nos muestra de forma permanente la temperatura y el pH, un interruptor de encendido y un interruptor de palanca para seleccionar el modo de funcionamiento, manual o automático.
- Una zona central con leds, indicadores de alarmas, y con pilotos verdes, indicadores del funcionamiento de los distintos dispositivos.
- Una zona inferior con interruptores de palanca para la activación/desactivación en modo manual de los distintos dispositivos y led indicador de activación.

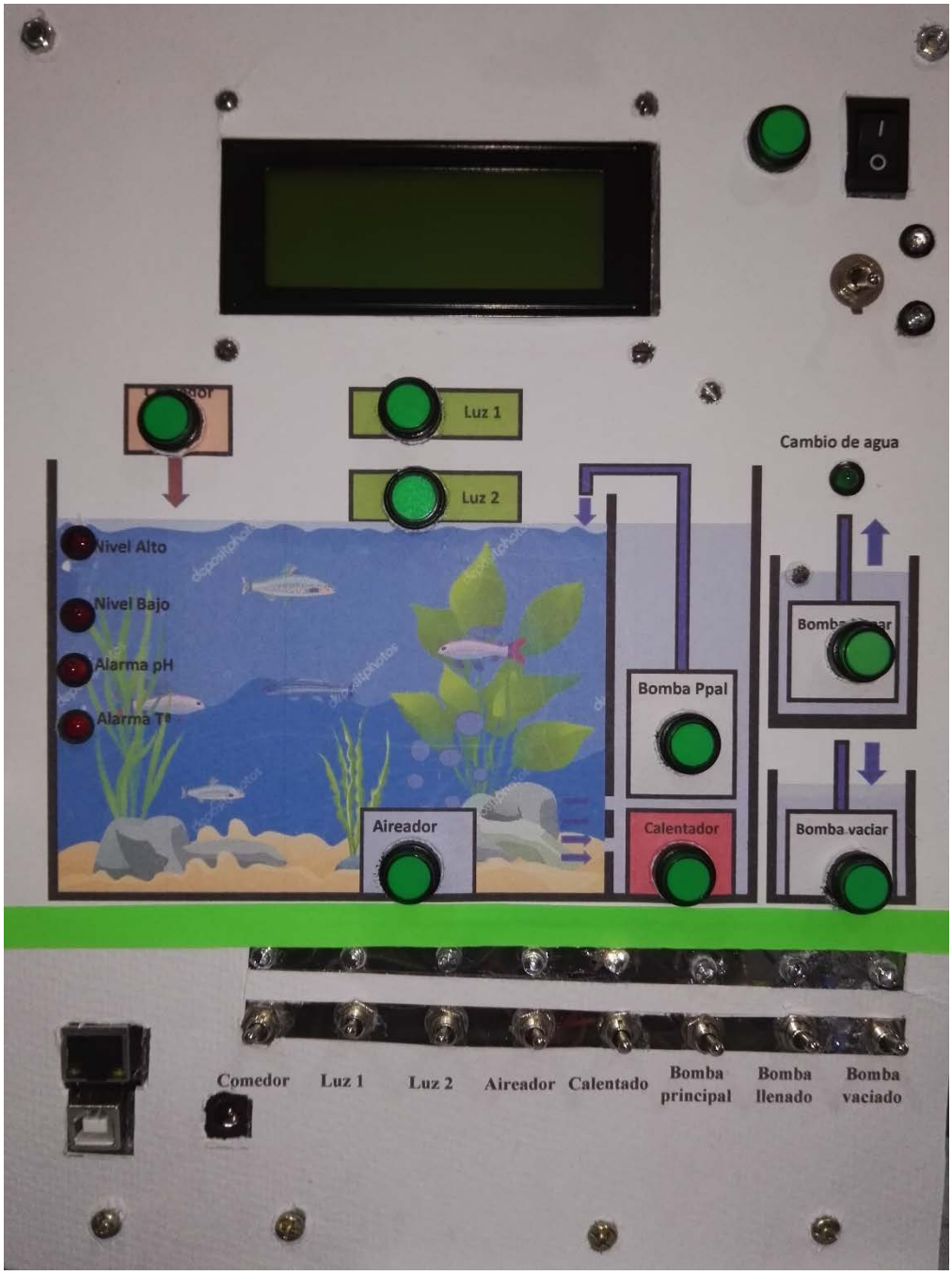


Ilustración 94 Vista frontal del prototipo
Fuente: Propiedad del autor

En la ilustración 95 vemos la parte posterior del dispositivo en donde distinguimos dos zonas:

- Una placa impresa en la parte superior donde se conectan los sensores de temperatura y niveles del agua y un conector BCN para la sonda de pH.
- La parte inferior donde se encuentran los pasamuros con los cables para la alimentación de los distintos dispositivos y el conector del cable de alimentación de red.

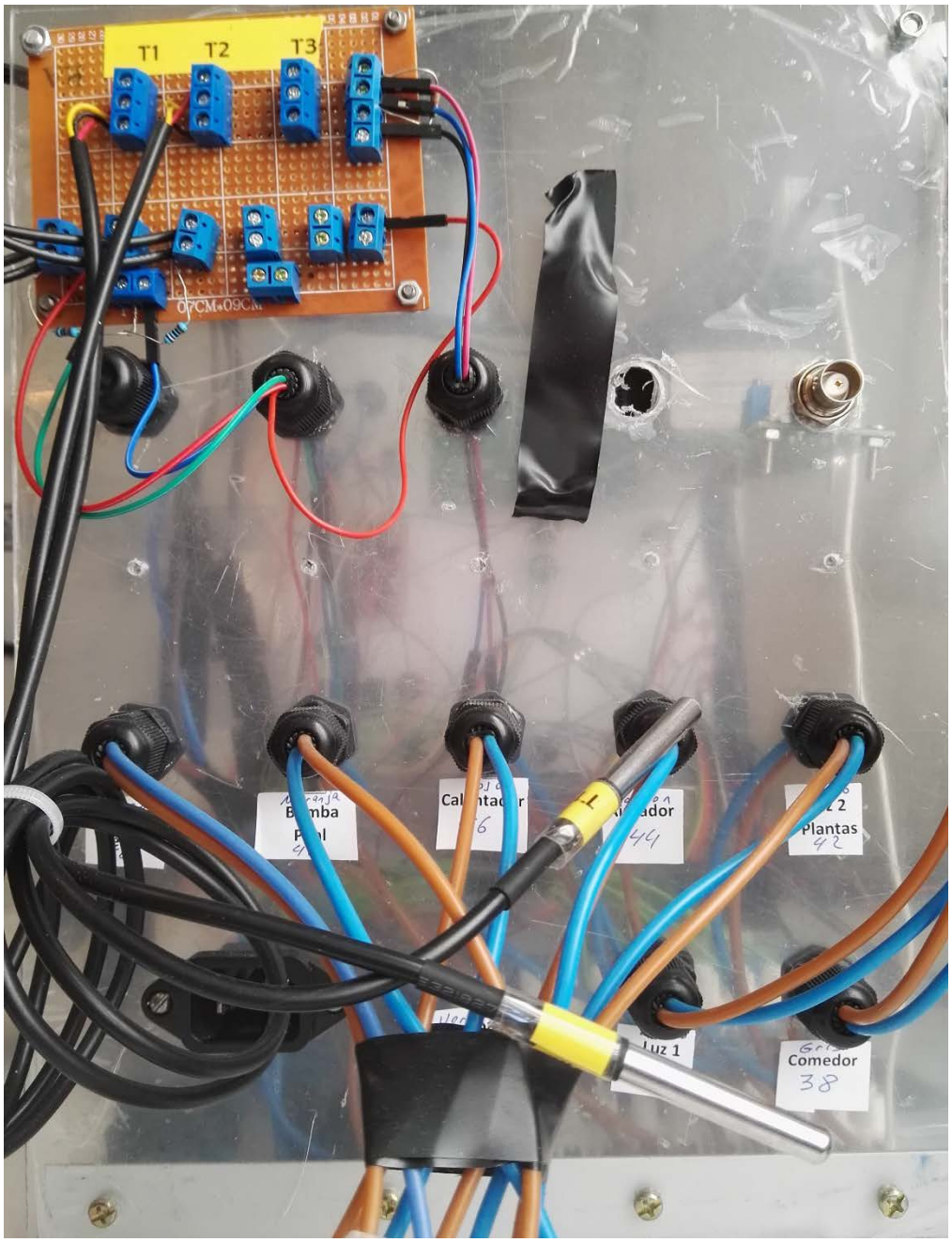


Ilustración 95 Vista posterior del prototipo
Fuente: Propiedad del autor

La disposición de los distintos componentes hardware, la placa Arduino Mega 2560, el shield Arduino Ethernet y el módulo de relés se puede apreciar en la ilustración 96.

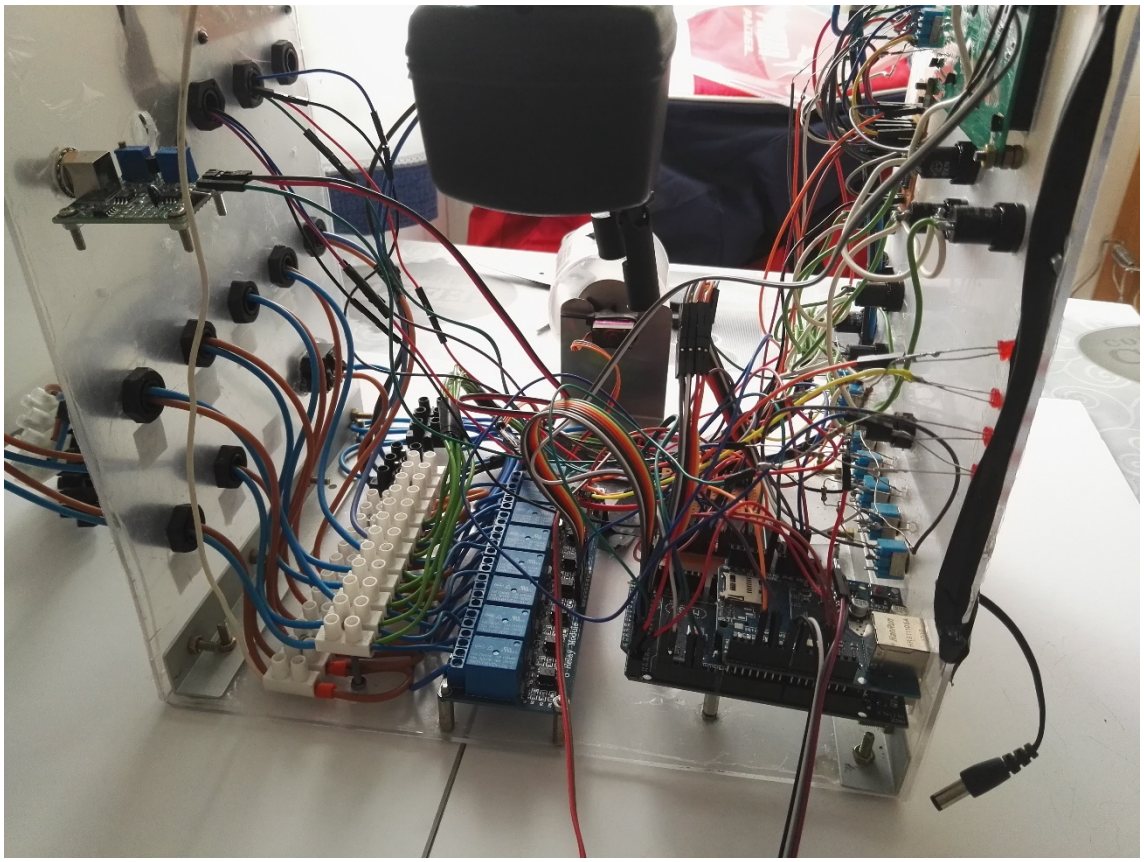


Ilustración 96 Vista interior del prototipo
Fuente: Propiedad del autor

4. Pruebas

A continuación, se muestra el resultado de las pruebas más relevantes que se han llevado a cabo durante la ejecución de este proyecto para comprobar su correcto funcionamiento y ver si se alcanzan los requisitos planteados al inicio del proyecto.

4.1. Prueba 1. Arranque del sistema

4.1.1. Descripción

Al conectar el sistema a la red eléctrica, este debe de arrancar de la manera adecuada al modo de funcionamiento seleccionado y aparecer en el display LCD el valor de la temperatura y del pH, así como el modo de funcionamiento y comprobando que hay una correcta comunicación con los sensores.

4.1.2. Procedimiento para la comprobación

Se acciona el interruptor ON-OFF de encendido.

Se calienta el sensor de temperatura del agua (T1).

Se introduce la sonda de pH en el líquido de prueba que tiene un pH de 4.0

4.2.3. Resultado obtenido

Se comprueba el correcto encendido, con la muestra en el display del valor de la temperatura, del pH y el modo de funcionamiento y el encendido del piloto de alimentación.

Se observa como al calentar el sensor de temperatura esta sube y al alcanzar el valor prefijado se activa el led de alarma de temperatura. Una vez que baja la temperatura la alarma desaparece.

Al introducir la sonda de pH en el líquido calibrado vemos como se indica el nuevo valor de pH y se activa el led de alarma de pH.

4.2. Prueba 2. Funcionamiento en modo manual

4.2.1. Descripción

Una vez seleccionado el modo manual se van seleccionando cada uno de los interruptores de los distintos accionamientos. Con la activación del interruptor se debe de activar/desactivar el actuador seleccionado, así como el piloto indicador correspondiente. Además, Se visualiza en la línea inferior del display lcd la última acción desarrollada.

4.2.2. Procedimiento para la comprobación

Se posiciona el interruptor de modo de funcionamiento en la posición modo manual.

Se activa/desactivan individualmente cada uno de los interruptores de los diferentes accionamientos.

4.2.3. Resultado obtenido

Se comprueba el correcto encendido/apagado de todos los actuadores y que en el display lcd se muestra la última acción.

4.3. Prueba 3. Comunicación entre el prototipo y la plataforma Thinger.io

4.3.1. Descripción

Una vez seleccionado el modo automático en el interruptor de modo de funcionamiento, se debe de realizar la conexión entre el prototipo y la plataforma Thinger.io y comenzar el funcionamiento automático.

4.3.2. Procedimiento para la comprobación

Se posiciona el interruptor de modo de funcionamiento en la posición modo automático.

Se accede a la plataforma Thinger.io y se accede a la pestaña devices.

Se accede a la pestaña Dashboards y se selecciona el dashboard dashpecera2.

Se accede a la pestaña Dashboards y se selecciona el dashboard dashpecera1.

4.3.3. Resultado obtenido

Se comprueba que en la ventana device de la plataforma thinger.io el dispositivo peceraiot aparece como conectado.

Se comprueba que en el dashboard dashpecera2 aparecen los valores de temperatura, de pH, las consignas prefijadas y las gráficas de temperaturas y de pH.

Se comprueba, tanto en el panel del prototipo como en el dashpecera1, que la bomba principal de agua está en funcionamiento.

4.4. Prueba 4. Modificar las consignas desde la plataforma Thinger.io

4.4.1. Descripción

Una vez seleccionado el modo automático en el interruptor de modo de funcionamiento, se debe poder cambiar las consignas de temperatura, pH y sus histéresis desde la plataforma Thinger.io

4.4.2. Procedimiento para la comprobación

Se posiciona el interruptor de modo de funcionamiento en la posición modo automático.

Se accede a la pestaña Dashboards, se selecciona el dashboard dashpecera1.

Se modifican los distintos valores de temperatura, pH, sus histéresis y tiempo para el envío de datos, actuando sobre la barra deslizante correspondiente.

4.4.3. Resultado obtenido

Se comprueba que en el dashboard dashpecera2 aparecen los valores de las consignas prefijadas y como estas se modifican según se actúa sobre las barras deslizantes. En las distintas gráficas podemos ver como se reflejan estas modificaciones.

4.5. Prueba 5. Control del nivel de agua del acuario

4.5.1. Descripción

El prototipo dispone de dos detectores de nivel, de mínimo y de máximo. Estos se deben activar cuando el nivel del agua esté por debajo del nivel mínimo o supere el nivel máximo.

4.5.2. Procedimiento para la comprobación

Se activa manualmente cada uno de los detectores de nivel.

4.5.3. Resultado obtenido

Se comprueba que en el panel frontal del prototipo se activa el led correspondiente de alarma de nivel y que en el display lcd sale el mensaje de alarma.

Se comprueba que en los dashboards dashpecera1 y dashpecera2 se activa la alarma de nivel correspondiente y que en el dashpecera1se activa el widget del nivel. Hay ocasiones que la actualización no es inmediata debido al retardo del envío de datos a la plataforma que tiene la cuenta gratuita que utilizamos.

4.6. Prueba 6. Automatización horaria de la iluminación, aireación y comedor del acuario

4.6.1. Descripción

A través de la plataforma IFTTT se programa el horario de encendido y apagado de la iluminación, aireación y del comedor. Para la comprobación de su correcto funcionamiento se cambia la programación para que estos dispositivos se activen cíclicamente durante quince minutos y solo uno de los dispositivos de cada vez.

4.6.2. Procedimiento para la comprobación

Se cambia la programación de las recetas en la plataforma IFTTT.

4.6.3. Resultado obtenido

Se comprueba que en el panel frontal del prototipo se activa el piloto correspondiente de cada dispositivo de forma cíclica.

Se comprueba que en el dashboard dashpecera1 en la gráfica actuadores se reflejan los cambios de estado de cada uno de los dispositivos, ver ilustración 97. Hay ocasiones que el encendido/apagado no es inmediato debido al retardo del envío de datos a la plataforma que tiene la cuenta gratuita que utilizamos.

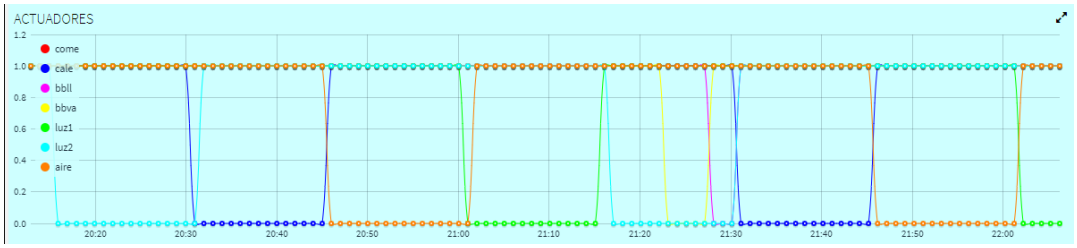


Ilustración 97 Gráfica de los actuadores durante la realización de la prueba 6
Fuente: Captura de pantalla del móvil

4.7. Prueba 7. Cambio parcial de agua del acuario

4.7.1. Descripción

Con el paso del tiempo el nivel del agua del acuario disminuye por la evaporación. Además, para la salud de los peces es recomendable un cambio parcial del agua mensual.

En la realización de este proyecto no disponemos de los dos depósitos adicionales necesarios para el cambio automático de agua. Sin embargo, se ha diseñado un control para realizar este cambio, por si se añaden en un futuro.

El modo de funcionamiento de este control se describe a continuación. Una vez que se active el control, se pone en marcha la bomba de vaciar hasta que el nivel del agua active el sensor de nivel mínimo. A continuación, se para la bomba de vaciar y se pone en marcha la bomba de llenar hasta que el nivel del agua alcance el nivel máximo, momento en que se para y se da por finalizado el cambio de agua. Si al activar el control el nivel mínimo de agua está activado, se omite el primer paso y se activa directamente la bomba de llenar.

4.7.2. Procedimiento para la comprobación

Se accede a la pestaña Dashboards y se selecciona el dashboard dashpecera1.

En el apartado interruptores se activa el control “Cambio de agua” y a continuación lo desactivamos. Esto es necesario porque la plataforma Thinger.io no tiene definido un widget tipo pulsador.

4.7.3. Resultado obtenido

Se comprueba que en el panel frontal del prototipo se activa el led correspondiente al cambio de agua, mientras dura todo el proceso.

La bomba de vaciar se enciende y permanece en ese estado hasta que activamos el sensor de nivel mínimo de agua.

Al activar manualmente el nivel mínimo de agua la bomba de vaciar se para y arranca la bomba de llenar.

La bomba de llenar permanece en marcha hasta que se activa manualmente el nivel de máximo, momento en el que para la bomba de llenar y se apaga el led de cambio de agua.

4.8. Prueba 8. Envío de alertas por correo electrónico

4.8.1. Descripción

Cuando la plataforma Thinger.io recibe una alarma por temperatura del agua alta o baja, nivel de pH alto o bajo, o por nivel máximo o mínimo de agua, envía un correo electrónico al móvil. Se utiliza el mismo correo para todas las alarmas.

4.8.2. Procedimiento para la comprobación

Se activa manualmente cada uno de los detectores de nivel.

Se calienta y enfría manualmente el sensor de temperatura.

Se introduce la sonda de pH en líquidos con distinto pH.

4.8.3. Resultado obtenido

Se comprueba que en el móvil se recibe el correo electrónico con el aviso de alarma. En la ilustración 98 podemos ver tres ejemplos con el mensaje recibido.

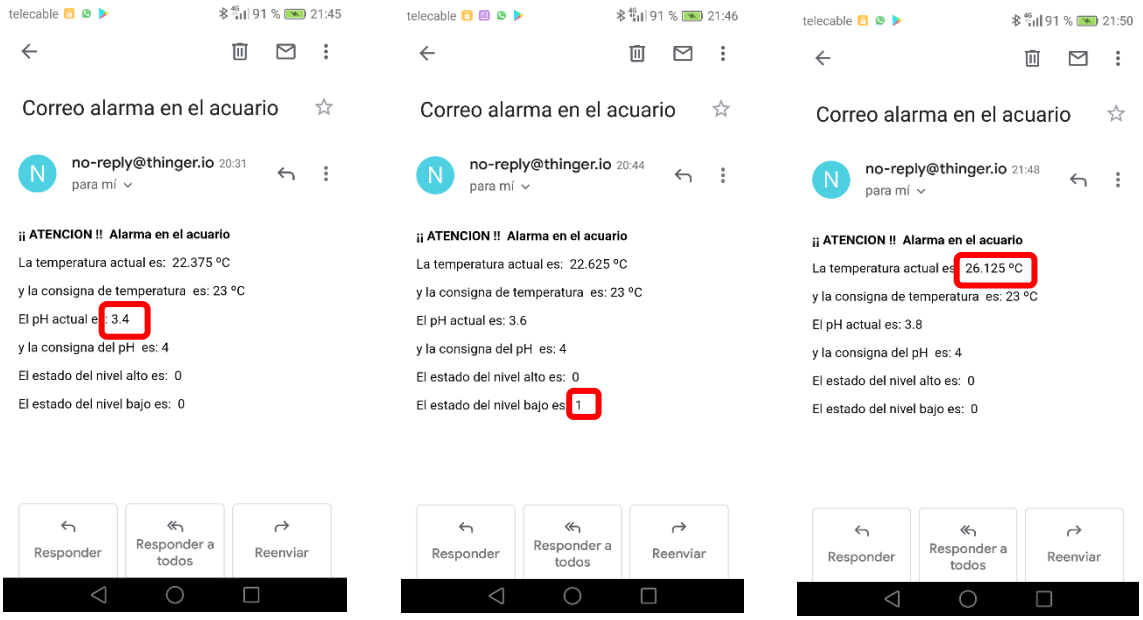


Ilustración 98 Alerta en el correo electrónico por pH bajo, nivel bajo y temperatura alta
Fuente: Captura de pantalla del móvil

4.9. Prueba 9. Acceso a la información desde un dispositivo externo

4.9.1. Descripción

La plataforma Thinger.io nos permite crear autorizaciones de acceso a los recursos de nuestra cuenta que van a permitir interactuar con nuestros recursos desde un sistema externo.

4.9.2. Procedimiento para la comprobación

Generar la api rest completa para acceder al dashboard dashpecera3.

4.9.3. Resultado obtenido

Se comprueba que introduciendo la URL en el navegador se accede al dashpecera3.

4.10. Prueba 10. Aplicación para smartphone

4.10.1. Descripción

La plataforma Thinger.io nos permite crear una aplicación que nos va a permitir administrar diferentes recursos de nuestro dispositivo desde un smartphone.

4.10.2. Procedimiento para la comprobación

Descargar la aplicación desde Google Play o App Store.

Escanear el toquen QR del dispositivo a controlar.

4.10.3. Resultado obtenido

Se comprueba que en el móvil se visualizan los datos correctamente y que permite cambiar los distintos parámetros. Ver ilustración 99.

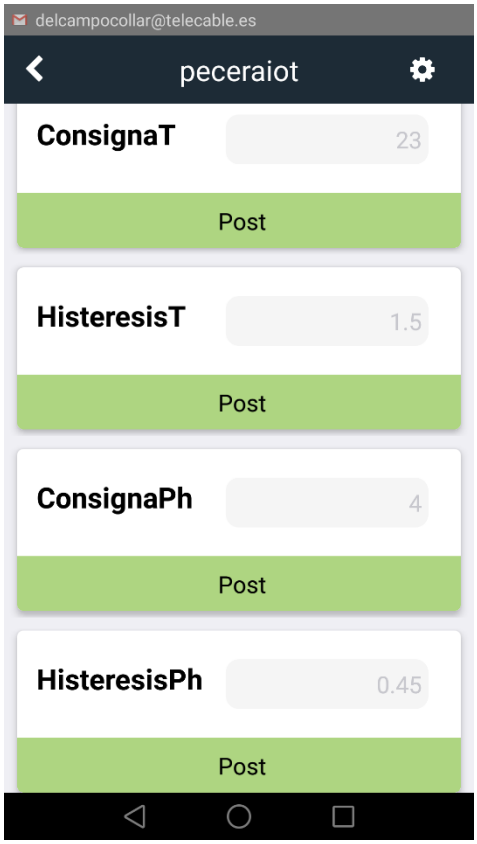


Ilustración 99 Pantalla para cambiar los parámetros desde el móvil
Fuente: Captura de pantalla del móvil

4.11. Cuadro resumen de las pruebas realizadas

En la tabla 9 se muestra un resumen de las pruebas que se han realizado para verificar el correcto funcionamiento del proyecto.

Prueba	Descripción	Requisito	Resultado
1	Arranque del sistema	1, 4 y 5	OK
2	Funcionamiento en modo manual	2 y 15	OK
3	Comunicación entre el prototipo y la plataforma	3, 12 y 16	OK
4	Modificar las consignas desde la plataforma Thinger	6	OK
5	Control del nivel de agua del acuario	7	OK
6	Automatización horaria de la iluminación, aireación, comedor.	8 y 9	OK
7	Cambio parcial de agua del acuario	10	OK
8	Envío de alertas por correo electrónico	11	OK
9	Acceso a la información desde un dispositivo externo	13	OK
10	Aplicación para smartphone	14	OK

Tabla 9 Cuadro resumen de las pruebas realizadas
Fuente: *Elaboración propia*

5. Planificación y presupuesto

5.1. Planificación

En la tabla 10 se muestra la duración de las diferentes tareas realizadas para la elaboración de este proyecto, así como su duración y fechas de inicio y de finalización.

Tarea	Duración (días)	Fecha inicio	Fecha fin
Estudios previos	7	14-01-2019	22-01-2019
Análisis y selección de soluciones	8	23-01-2019	01-02-2019
Diseño hardware	15	04-02-2019	22-02-2019
Diseño software	20	25-02-2019	22-03-2019
Construcción del prototipo	28	18-02-2019	27-03-2019
Realización de pruebas	7	28-03-2019	05-04-2019
Realización de la documentación	55	28-01-2019	12-04-2019
TOTAL		14-01-2019	12-04-2019

Tabla 10 Planificación
Fuente: *Elaboración propia*

La tabla 11 muestra el diagrama de Gantt de las tareas realizadas, que nos permite ver de una forma más visual el periodo de desarrollo del proyecto. Nos muestra de forma gráfica los tiempos empleados para la realización de cada una de las tareas desarrolladas.

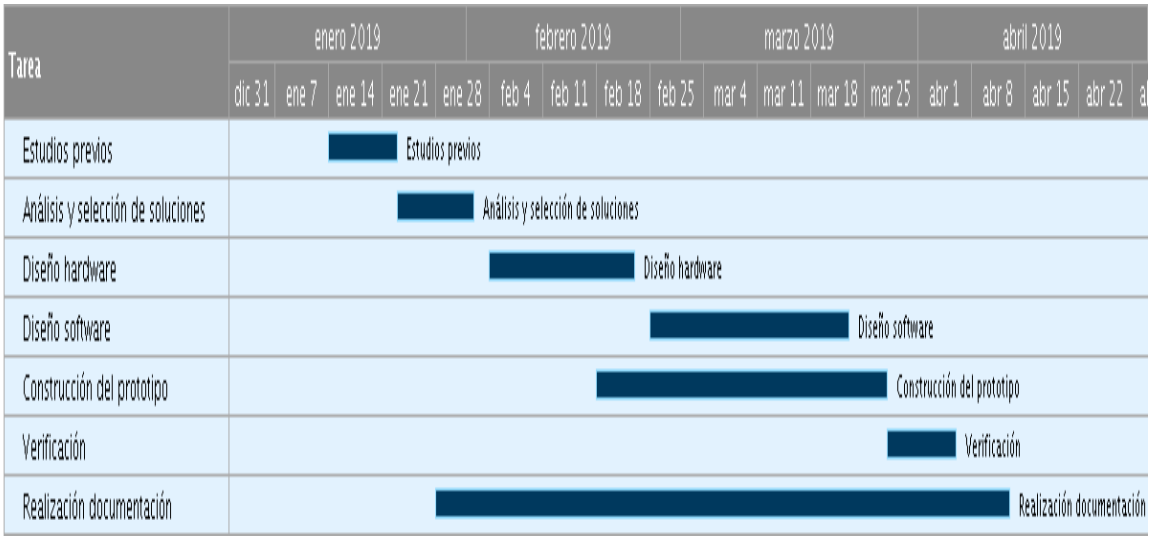


Tabla 11 Planificación. Diagrama de Gantt
Fuente: *Elaboración propia*

5.2. Presupuesto

5.2.1. Presupuestos parciales.

5.2.1.1. Recursos Hardware

En la tabla 12 se presenta el presupuesto parcial de los recursos hardware utilizados.

Recurso	Ud	Precio/ud	Importe
Sensor de temperatura DS18B20	2	2,80 €	5,60 €
Sensor de pH de DFROBOT + Sonda BNC	1	22,99 €	22,99 €
Sensor de nivel horizontal Cebek C-7236	2	1,16 €	2,32 €
Módulo de relés TOOGOO 5V 8 canales	1	3,57 €	3,57 €
Calentador 220V – 150 W	1	14,99 €	14,99 €
Bomba de agua AS250 3,2 W	1	3,39 €	3,39 €
Bomba de agua G205M 220V 3 W	2	2,98 €	5,96 €
Fluorescente led blanco 220V, 38 cm. x 2 cm	1	11,43 €	11,43 €
Aireador HiPumps YDQB4105	1	11,12 €	11,12 €
Servomotor JSUMO MG90S	1	1,72 €	1,72 €
Display LCD 2004 + Módulo LC2 a I2C	1	3,86 €	3,86 €
LCD 1602 Converter Board IIC/I2C/TWI/SPI	1	0,76 €	0,76 €
Arduino MEGA 2560 + Cable USB	1	8,21 €	8,21 €
Arduino Ethernet Shield W5100	1	4,69 €	4,69 €
Router Technicolor TC7210	1	30,00 €	30,00 €
Resistencias	16	0,05 €	0,80 €
Leds	15	0,05 €	0,75 €
Interruptores de palanca SPDT, 3 pines	9	0,25 €	2,25 €
Interruptor ON-OFF, 3 pines 250V - 3A	1	0,12 €	0,12 €
Piloto señalización NXD-212, 220V-verdes	9	0,23 €	2,07 €
Pasamuros	10	0,10 €	1,00 €
Varios: Cables, tornillos, separadores, etc.	1	20,00 €	20,00 €
Ordenador portátil	1	171,37 €	171,37 €
SUBTOTAL			**328,97 €**

Tabla 12 Presupuesto parcial. Recursos Hardware
Fuente: *Elaboración propia*

La amortización del ordenador se calcula de acuerdo con las tablas de amortización para un periodo de 2 años, con un precio de compra de 402,74 € y un valor neto residual de 60,00 €.

Precio portátil: 402,74 €

Valor residual: - 60,00 €

342,74 €

Amortización anual: 342,74 € ÷ 2 años = 171,37 €/año

5.2.1.2. Recursos Software

El coste en cuanto al software utilizado es nulo, al venir incluido en el PC o por ser de acceso libre.

Recurso	Unidades	Importe
Windows 10	1 unidad	Incluido en PC
Arduino IDE	1 unidad	Software libre
Thinger.io	1 unidad	Software libre
IFTTT	1 unidad	Software libre
SUBTOTAL		**0,00 €**

Tabla 13 Presupuesto parcial. Recursos Software
Fuente: *Elaboración propia*

5.2.1.3. Recursos Humanos

Para la realización de este proyecto solo ha sido necesario la participación de una persona.

Capítulo	Coste/hora	Horas	Importe
Estudios previos	30,00 €	56	1.680,00 €
Análisis y selección de soluciones	30,00 €	60	1.800,00 €
Diseño hardware	20,00 €	90	1.800,00 €
Diseño software	25,00 €	120	3.000,00 €
Construcción del prototipo	20,00 €	46	920,00 €
Realización de pruebas	20,00 €	49	980,00 €
Realización de la documentación	20,00 €	99	1.980,00 €
TOTAL			**12.160,00 €**

Tabla 14 Presupuesto parcial. Recursos Humanos
Fuente: *Elaboración propia*

5.2.2. Presupuesto final

En la tabla 15 se muestra el presupuesto final.

Capítulo	Importe total
Recursos Hardware	328,97 €
Recursos Software	0,00 €
Recursos humanos	12.160,00 €
SUBTOTAL	12.488,97 €
BENEFICIO INDUSTRIAL (10%)	1.248,90 €
GASTOS GENERALES (15%)	1.873,35 €
TOTAL	**15.611,22 €**
IVA (21 %)	**3.278,36 €**
TOTAL, IVA INCLUIDO	**18.889,58 €**

Tabla 15 Presupuesto final
Fuente: *Elaboración propia*

El presupuesto asciende a la cantidad de **dieciocho mil ochocientos ochenta y nueve euros con cincuenta y ocho céntimos**.

6. Conclusiones y posibles mejoras

En este apartado se van a comentar las conclusiones obtenidas y las posibles mejoras a realizar tomando como base este proyecto.

Con este proyecto se ha conseguido alcanzar los objetivos planteados, integrar un sistema IoT para el control de los parámetros de funcionamiento de un acuario. El sistema diseñado está dotado de inteligencia y funcionalidades que le otorgan un valor añadido, que permite el control automático y la monitorización de los principales parámetros del acuario, como son la temperatura, el pH o el nivel de agua, tanto a nivel de valores como a través de gráficos, la programación horaria de diversas funcionalidades, la supervisión a través de un smartphone o el envío alertas a través del correo electrónico.

Todo esto permite que las personas que tengan un acuario dispongan de más tiempo libre al no tener que estar pendiente de su cuidado y sobre todo cuando se marcha de vacaciones, mejorando así su calidad de vida.

Las conclusiones obtenidas en la realización de este proyecto son varias:

- La realización de este proyecto tiene cabida actualmente para el control de acuarios instalados en hogares, puesto que la gran mayoría carecen de un control electrónico, y aunque existen algunos desarrollos comerciales estos resultan demasiado caros, por lo que con este proyecto obtenemos unas prestaciones similares a las que ofrecen los productos comerciales, pero a un coste mucho más económico y con una gran versatilidad.
- A pesar de las enormes posibilidades de la plataforma Arduino, la gran cantidad de información existente, la facilidad para la realización de proyectos y el bajo coste de sus componentes, creemos que esta no es una opción adecuada para un uso profesional, tanto por la fragilidad del sistema para trabajar en un entorno industrial como por la falta total de seguridad que ofrece el sistema, considerando que su utilidad queda reducida a entornos de experimentación, educativos o uso particular.
- Tal y como comentamos en el proyecto, creemos que la comunicación vía ethernet no es la más adecuada, principalmente por las limitaciones a la hora de situar la pecera, puesto que necesita un punto de conexión próximo, problema que se ve resuelto con la utilización de una red WiFi. Además, la utilización de la red WiFi no añade ninguna complejidad a la programación pues esta sería igual y solo hace falta cambiar en el programa de Arduino la librería de ethernet por la librería wifi y añadir dos líneas al programa para introducir el nombre de la red WiFi y la contraseña.
- La plataforma de IoT utilizada, Thinger.io, tiene un gran potencial, sin embargo, todavía está en fase de desarrollo y se detectan varias carencias, como, por ejemplo, faltan widgets básicos (pulsadores) y faltan herramientas (no existe la posibilidad de programar eventos temporales, no permite la programación de triggers tipo if … then, biblioteca para conexión con NodeRed, etc.). Cabe destacar que están en vías de resolver todas estas carencias, con lo que una vez resueltas, va a ser una plataforma muy completa.

- A nivel personal, la experiencia obtenida al desarrollar este proyecto ha sido muy satisfactoria al permitirme obtener conocimientos tanto sobre IoT como de Arduino.

Como posibles mejoras que se han ido detectando durante la realización del proyecto, y que no ha sido posible realizar por falta de tiempo, podemos destacar:

- Utilizar el shield Arduino WiFi en lugar del shield Arduino ethernet para una conexión WiFi.
- Diseñar una PCB a la que se conecten todos los componentes y que se acople directamente a la placa Arduino Mega 2560.
- Mejora del proceso de renovación parcial del agua del acuario. Añadir dos depósitos, con niveles, calentador y con electroválvulas para el control de llenado y de vaciado.
- Instalar una cámara web para ver el acuario online.
- Utilización del análisis de datos para obtener información y utilizarla para depurar el funcionamiento del sistema (horas de luz necesarias, cambios de agua, etc.).
- Realizar un estudio de seguridad del entorno.

7. Referencias

1. **Peño, Paz.** Cultura fotográfica. [En línea] 21 de 09 de 2016. [Citado el: 03 de 03 de 2019.] https://culturafotografica.es/acuarios-espectaculares/.

2. **CurioSfera.** *CurioSfera.* [En línea] [Citado el: 02 de 03 de 2019.] https://www.curiosfera.com/historia-del-acuario/.

3. **ProFiLux GLH Iberia. [En línea] 2019. [Citado el: 03 de 03 de 2019.] http://www.profilux.es/.**

4. Aquatronica. [En línea] 2014. [Citado el: 5 de 3 de 2019.] https://www.aquatronica.com/pc_index.php?L=ita&PAGE=home&OTP=.

5. Neptune Systems. [En línea] 2018. [Citado el: 5 de 3 de 2019.] https://www.neptunesystems.com/.

6. Tarraco Goldfish. [En línea] 2017. [Citado el: 3 de 3 de 2019.] http://www.tarracogoldfish.com/guia-acuario-de-agua-dulce-tropical/.

7. Tropiacuarium Bilbao. [En línea] [Citado el: 3 de 3 de 2019.] http://www.tropiacuarium.com/lo-mas-importante-la-calidad-del-agua-lo-mas-importante-para-cuidar-nuestros-acuarios-conocer-la-calidad-del-agua-dependiendo-del-3.html.

8. Arduino. [En línea] 2019. [Citado el: 5 de 3 de 2019.] https://www.arduino.cc/.

9. Isaac. comohacer.eu. [En línea] 2017. [Citado el: 11 de 3 de 2019.] https://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/.

10. Triana, Ingenio. Ingenio Triana. [En línea] 25 de 1 de 2017. [Citado el: 12 de 3 de 2019.] http://ingenio-triana.blogspot.com/2017/01/comunicacion-con-arduino-usb-serie-i2c.html.

11. Wikipedia. [En línea] 7 de 3 de 2013. [Citado el: 12 de 3 de 2019.] https://es.wikipedia.org/wiki/6LoWPAN.

12. Universidad Internacional de Valencia. [En línea] 2018. [Citado el: 3 de 4 de 2019.] https://www.universidadviu.es/que-es-gsm-y-como-funciona/.

13. Sistemas. [En línea] [Citado el: 3 de 4 de 2019.] https://sistemas.com/gprs.php.

14. ValorTop. [En línea] 15 de 9 de 2015. [Citado el: 3 de 4 de 2019.] http://www.valortop.com/blog/3g-4g-definicion-diferencias.

15. Arduino. [En línea] [Citado el: 14 de 3 de 2019.] https://www.arduino.cc/en/Main/ArduinoEthernetShieldV1.

16. Singh, Santosh. InternetofThingsWiki. [En línea] 17 de 3 de 2019. [Citado el: 3 de 4 de 2019.] https://internetofthingswiki.com/top-20-iot-platforms/634/.

17. Del Valle, Luis. Programarfacil. [En línea] 2018. [Citado el: 5 de 4 de 2019.] https://programarfacil.com/podcast/proyectos-iot-con-arduino/.

18. Thinger.io. [En línea] 2018. [Citado el: 5 de 4 de 2019.] https://thinger.io/.

19. Del Valle, Luis. Programarfacil. [En línea] 2018. [Citado el: 5 de 4 de 2019.] https://programarfacil.com/blog/arduino-blog/cayenne-mydevices-arduino-sensores-iot/.

20. Llamas, Luis. Luis Llamas. [En línea] 27 de 6 de 2016. [Citado el: 3 de 3 de 2019.] https://www.luisllamas.es/temperatura-liquidos-arduino-ds18b20/.

21. DFROBOT. [En línea] 2008. [Citado el: 4 de 3 de 2019.] https://www.dfrobot.com/product-1025.html.

22. Cetronic. [En línea] [Citado el: 4 de 3 de 2019.] https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=150272071&cPath=787.

23. JSUMO. [En línea] 2012. [Citado el: 5 de 3 de 2019.] https://www.jsumo.com/mg90s-micro-servo-motor.

24. Naylamp Mechatronics. [En línea] [Citado el: 5 de 3 de 2019.] https://naylampmechatronics.com/blog/35_Tutorial--LCD-con-I2C-controla-un-LCD-con-so.html.

25. Thinger.io. [En línea] [Citado el: 10 de 05 de 2019.] http://docs.thinger.io/arduino/.

26. Viccarre. HardwareHackingMX. [En línea] 01 de 04 de 14. [Citado el: 10 de 05 de 2019.] https://hardwarehackingmx.wordpress.com/2014/01/04/fritzing-que-es-y-como-se-usa/.

27. Mejor Antivirus. [En línea] 2018. [Citado el: 5 de 4 de 2019.] http://www.mejor-antivirus.es/analisis/el-malware-contra-los-dispositivos-iot-se-ha-duplicado.html.

28. Kaspersky Lab. [En línea] 23 de 6 de 2017. [Citado el: 5 de 4 de 2019.] https://www.kaspersky.es/about/press-releases/2017_malware-against-iot-devices-has-doubled-2017.

29. Arteaga, Sandra. Computer Hoy. [En línea] 16 de 4 de 2018. [Citado el: 5 de 4 de 2019.] https://computerhoy.com/noticias/software/roban-datos-casino-traves-del-termometro-del-acuario-79195.

30. Martín Gutiérrez, Sergio y Castro Gil, Manuel A. Vulnerabilidades hardware. *Curso Experto en Internet de las Cosas con Arduino.* s.l. : Fundación UNED, 2018, pág. 7.

31. FIRMA-e. [En línea] 14 de 10 de 2014. [Citado el: 5 de 4 de 2019.] https://www.firma-e.com/blog/pilares-de-la-seguridad-de-la-informacion-confidencialidad-integridad-y-disponibilidad/.

32. Universidad Internacional de Valencia. [En línea] 18 de 4 de 2018. [Citado el: 5 de 4 de 2019.] https://www.universidadviu.es/las-4-claves-la-seguridad-la-informacion/.

33. Morgan, Jecinta. [En línea] 6 de 3 de 2018. [Citado el: 5 de 4 de 2019.] http://www.differencebetween.net/language/words-language/difference-between-safety-and-security/.

34. incibe_. [En línea] 20 de 3 de 2017. [Citado el: 5 de 4 de 2019.] https://www.incibe.es/protege-tu-empresa/blog/amenaza-vs-vulnerabilidad-sabes-se-diferencian.

35. Martín Gutiérrez, Sergio y Castro Gil, Manuel A. Introducción a la seguridad. *Curso Experto en Internet de las Cosas con Arduino.* s.l. : Fundación UNED, 2018, pág. 7.

36. —. Vulnerabilidades software. *Curso Experto en Internet de las Cosas con Arduino.* s.l. : Fundación UNED, 2018, pág. 8.

37. G. Soto, Marvin. Medium work. [En línea] 27 de 6 de 2016. [Citado el: 5 de 4 de 2019.] https://medium.com/@marvin.soto/qu%C3%A9-es-el-envenenamiento-arp-o-ataque-arp-spoofing-y-c%C3%B3mo-funciona-7f1e174850f2.

8. Anexos

8.1. Anexo 1. Obtener dirección del sensor DS18B20

```
#include <OneWire.h>

OneWire ourWire(5);                //Se establece el pin 5  como bus OneWire

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte addr[8];
  Serial.println("Obteniendo direcciones:");
  while (ourWire.search(addr))
    {
            Serial.print("Address = ");
            for( int i = 0; i < 8; i++)
            {
               Serial.print(" 0x");
               Serial.print(addr[i], HEX);
            }
            Serial.println();
    }
 Serial.println();
 ourWire.reset_search();
 delay(8000);
}
```

8.2. Anexo 2. Calibrar sensor de pH

La placa del sensor de pH necesita de una alimentación de 5V, dispone de un led verde indicador de alimentación, un led rojo indicador límite de pH y de seis pines como podemos ver en la ilustración 100.

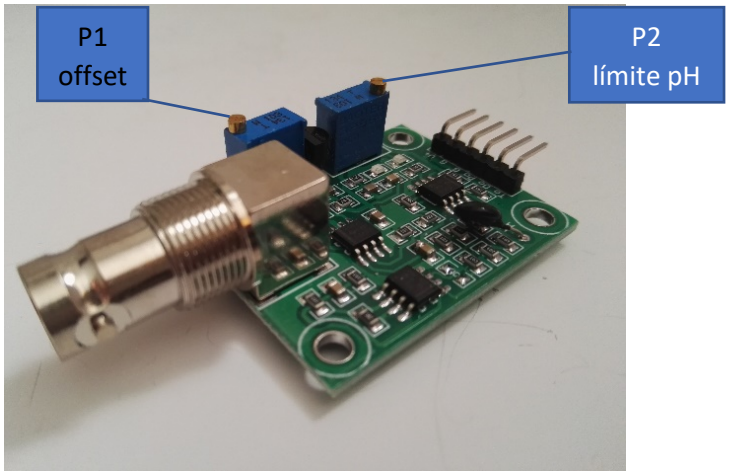


Ilustración 100 Placa sensor pH
Fuente: Elaboración propia

Podemos observar que la placa dispone de dos potenciómetros, el más cercano al conector BNC (P1) sirve para regular el offset y con el otro potenciómetro (P2) regulamos el límite de pH.

1. **Offset:** El rango de medida de la sonda varía entre valores positivos y valores negativos, representando el valor 0 un pH de 7.0. Para poder utilizar con Arduino este circuito añade un valor de offset al valor medido por la sonda, para que solo lleguen valores positivos de tensión. Para ajustar el offset tenemos que forzar un pH 7.0. Para ello desconectamos la sonda del circuito y cortocircuitamos la parte interna del conector BNC con la parte externa. Con un multímetro medimos el valor del pin Po y ajustamos el potenciómetro P1 para que el valor sea de 2,5 V.

2. **Límite de pH:** Con el potenciómetro P2 establecemos un valor límite del circuito sensor de pH que hace que el led rojo se encienda y que la señal digital del pin Do se ponga en ON.

8.3. Anexo 3. Obtener dirección del LCD

```
/*--------------------------------------------------
  Programa para buscar dispositivos I2C conectados.
  Manda las direcciones que encuentra en el puerto serie
    Encontrado en:
      http://playground.arduino.cc/Main/I2cScanner
--------------------------------------------------
*/

#include <Wire.h>

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
}

void loop()
{
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    // The i2c_scanner uses the return value of
    // the Write.endTransmisstion to see if
    // a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();
    if (error == 0)
    {
      Serial.print("I2C device found at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Unknow error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
```

8.4. Anexo 4. Programación en Arduino

8.4.1. Instalación de Arduino y configuración para placas Arduino Mega

Los pasos para instalar Arduino en Windows son:

8.4.1.1. Descargar el entorno de desarrollo de Arduino

Para poder programar la tarjeta de Arduino Mega es necesario tener instalado el entorno de desarrollo de Arduino, el cual se puede descargar desde:

https://www.arduino.cc/en/Main/Software

Este enlace nos permite acceder a la página web del desarrollador y allí seleccionamos la versión más adecuada a nuestro sistema operativo. Pinchamos en "Windows installer, for Windows XP and up". En las ilustraciones 101 y 102 se muestran las capturas de pantalla obtenidas durante la realización de este proceso.

Ilustración 101 Pagina descarga de Arduino
Fuente: https://www.arduino.cc/en/Main/Software

En la nueva ventana hacemos clic en “JUST DOWNLOAD” y comienza la descarga.

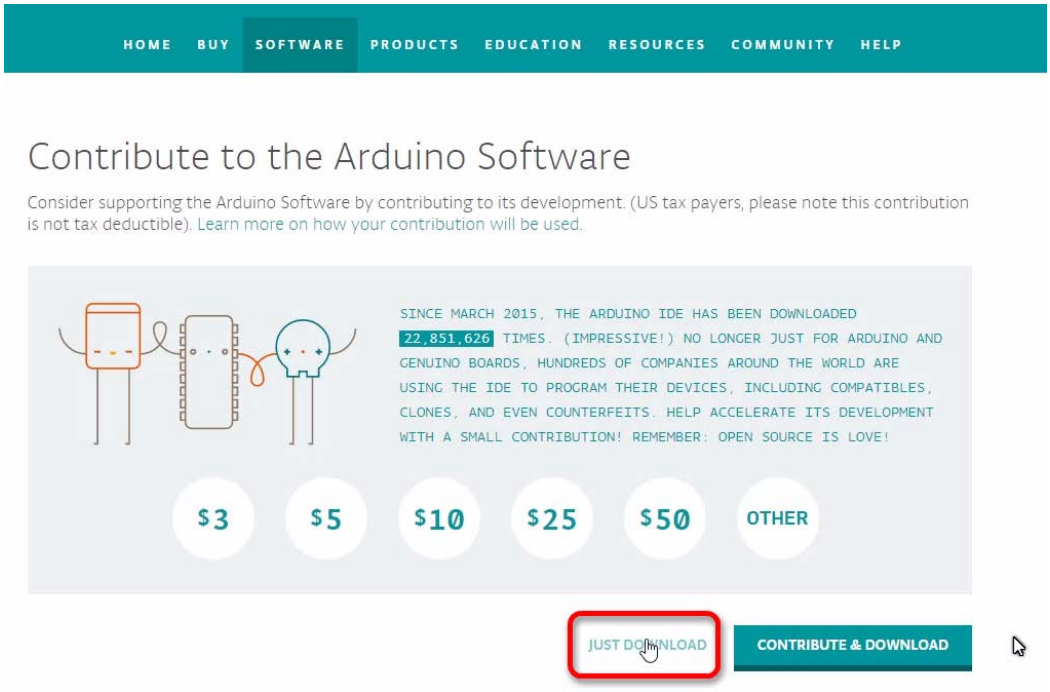
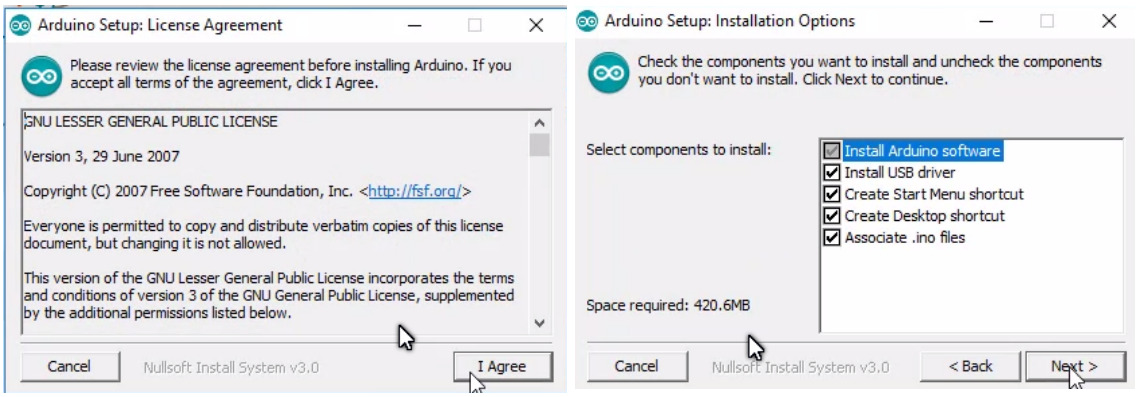


Ilustración 102 Pagina donativo de Arduino
Fuente: https://www.arduino.cc/en/Main/Donate

8.4.1.2. Instalamos el software de desarrollo de Arduino

Una vez descargado el software procedemos a su instalación. Para ello ejecutamos el fichero de instalación y aceptamos todas las opciones de instalación que nos va ofreciendo. En la ilustración 103 vemos cuatros capturas de pantalla realizadas durante el proceso de instalación.



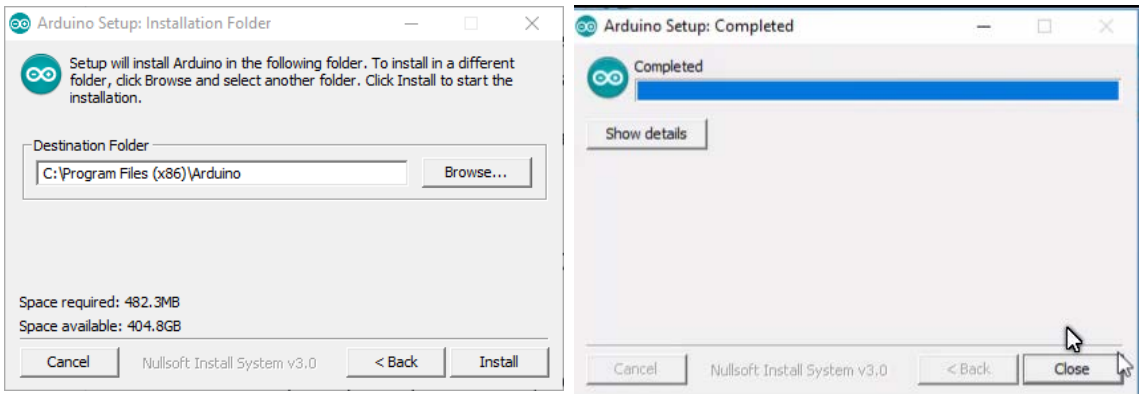


Ilustración 103 Distintas capturas durante la instalación de Arduino
Fuente: Captura de pantalla

8.4.1.3. Instalamos la librería de Thinger.io

Conectamos la tarjeta Arduino Mega 2560 con un cable USB al PC. Observamos que el led de encendido se ilumina, como se aprecia en la ilustración 104.

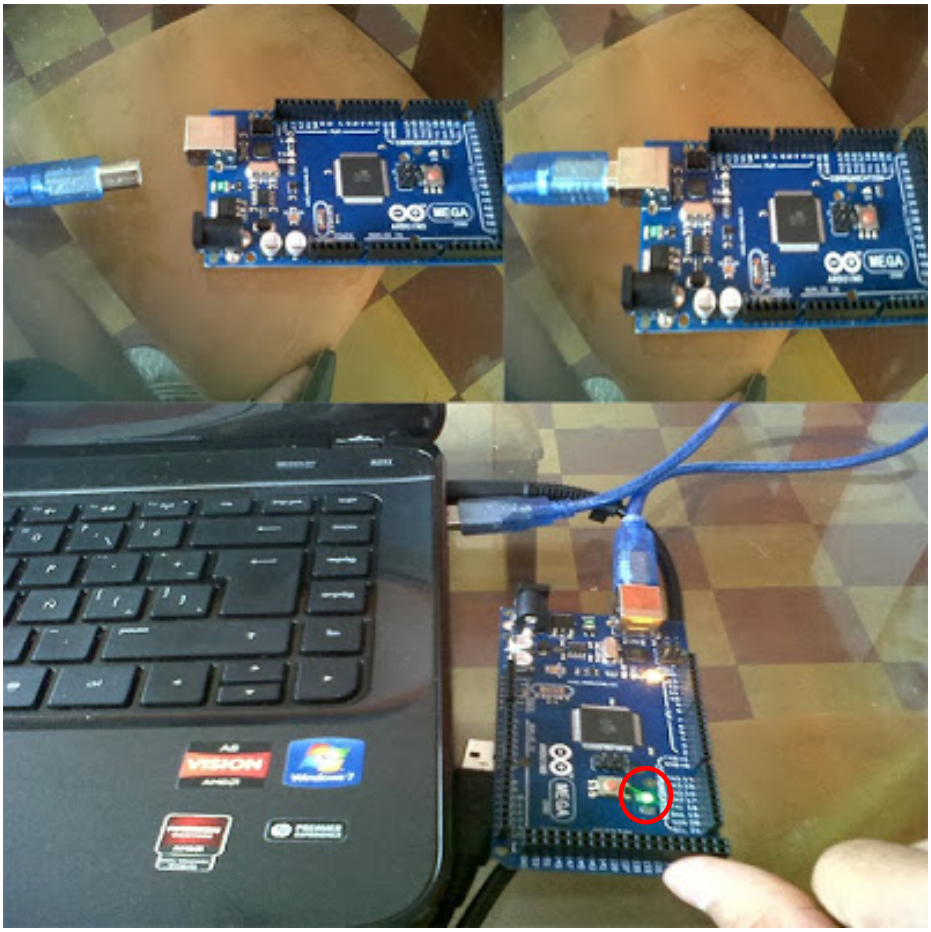


Ilustración 104 Distintas capturas de la conexión de la tarjeta Arduino Mega al PC
Fuente: http://www.blogginred.com/2014/01/conectar-configurar-arduino-mega-2560.html#.XKjgh5gzZhE

Ahora desde el escritorio hacemos doble clic en el acceso directo de la aplicación Arduino que se muestra en la ilustración 105 y se abre la aplicación apareciendo la ventana principal del IDE que se muestra en la ilustración 106. El IDE oficial de Arduino posee una interfaz muy sencilla e intuitiva. Las siglas IDE significan entorno de desarrollo integrado, y puede definirse como la herramienta que nos permite desarrollar nuestras aplicaciones de una manera cómoda.

Ilustración 105 Icono de acceso directo de la aplicación Arduino
Fuente: Captura de pantalla

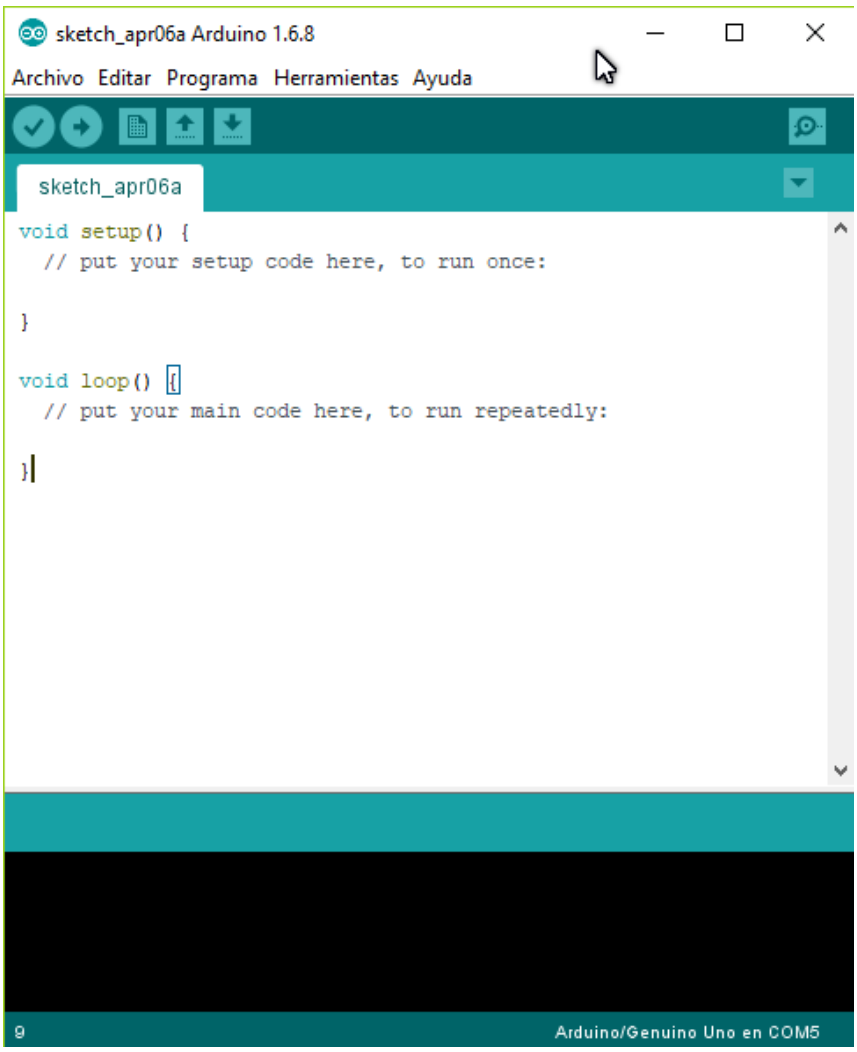


Ilustración 106 Ventana principal de la aplicación Arduino
Fuente: Captura de pantalla

Accedemos al gestor de librerías. Para ello hay que dirigirse a Programa > Incluir Librería > Gestionar Librerías. Ver ilustración 107.

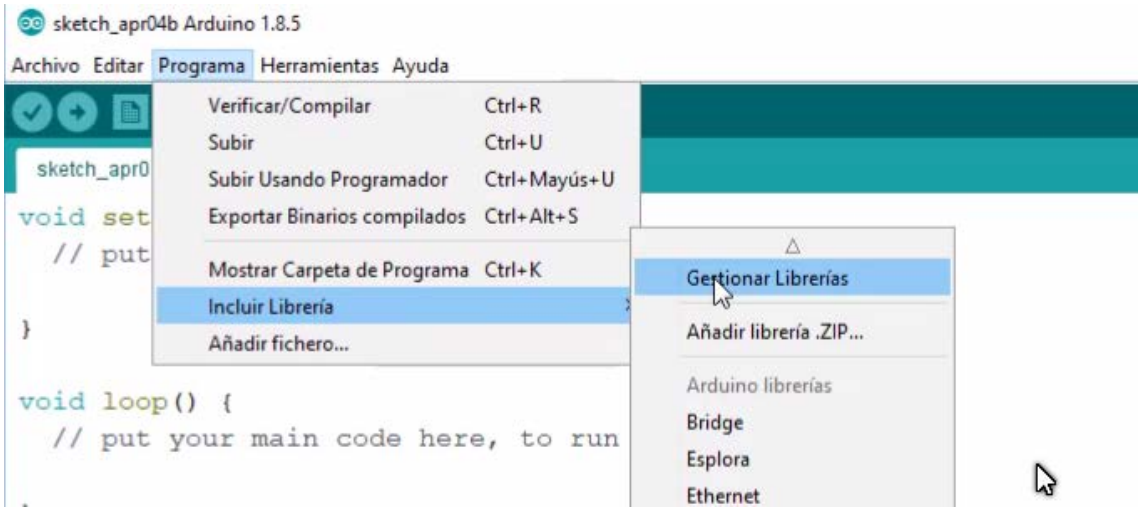


Ilustración 107 Gestionar librerías en Arduino
Fuente: Captura de pantalla

Escribimos thinger.io en el campo "Filtre su búsqueda" como vemos en la ilustración 108, e instalamos la librería thinger.io.

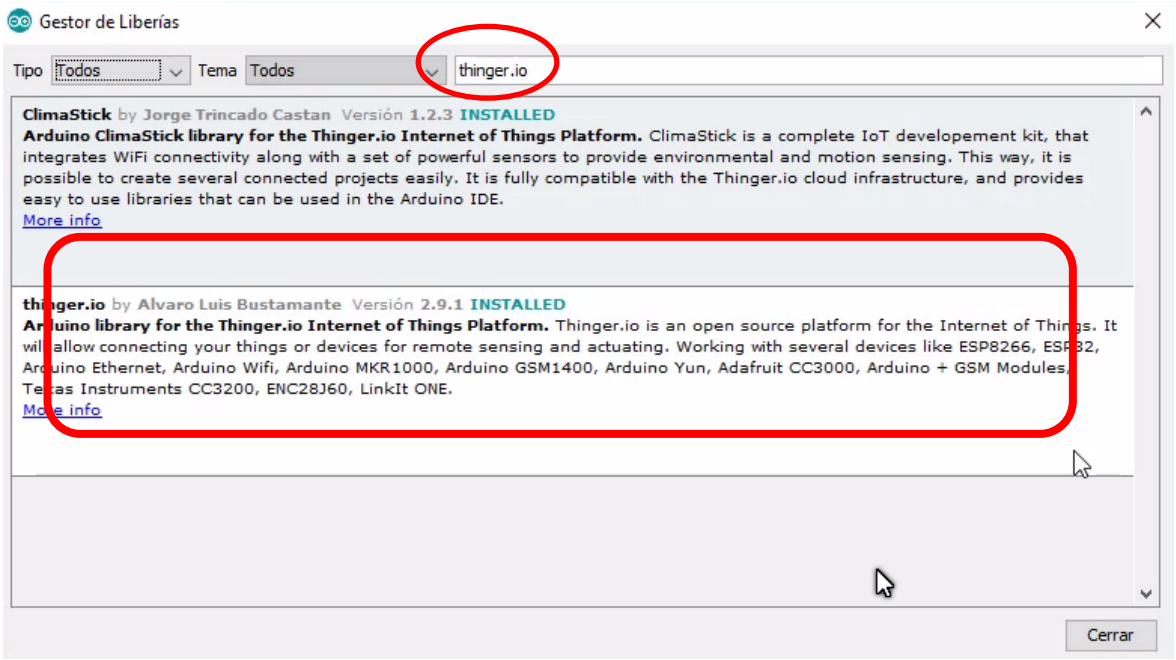


Ilustración 108 Instalar librería thinger.io en Arduino
Fuente: Captura de pantalla

8.4.1.4. Configuración del entorno de Arduino

Para finalizar necesitamos escoger el modelo de placa conectada, que en nuestro caso es Arduino Mega 2560, seleccionar el puerto COM donde se conecta la tarjeta y configurar la velocidad de conexión.

Para especificar el modelo de placa lo seleccionamos en Herramientas > Placa > Arduino Genuino Mega or Mega 2560 como se muestra en la ilustración 109.

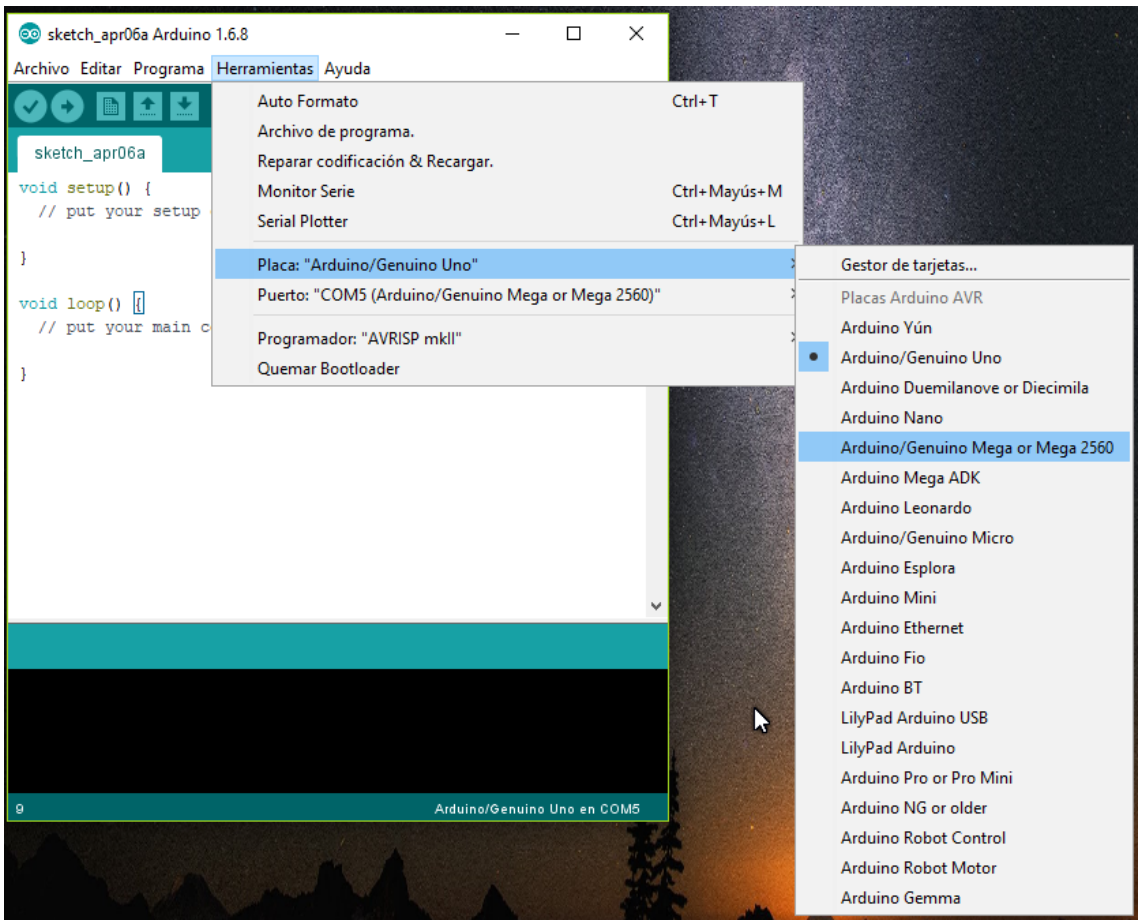


Ilustración 109 Seleccionar modelo de placa
Fuente: Captura de pantalla

Para especificar el puerto COM donde tenemos conectada nuestra placa lo seleccionamos en Herramientas > Puerto, tal y como vemos en la ilustración 110.

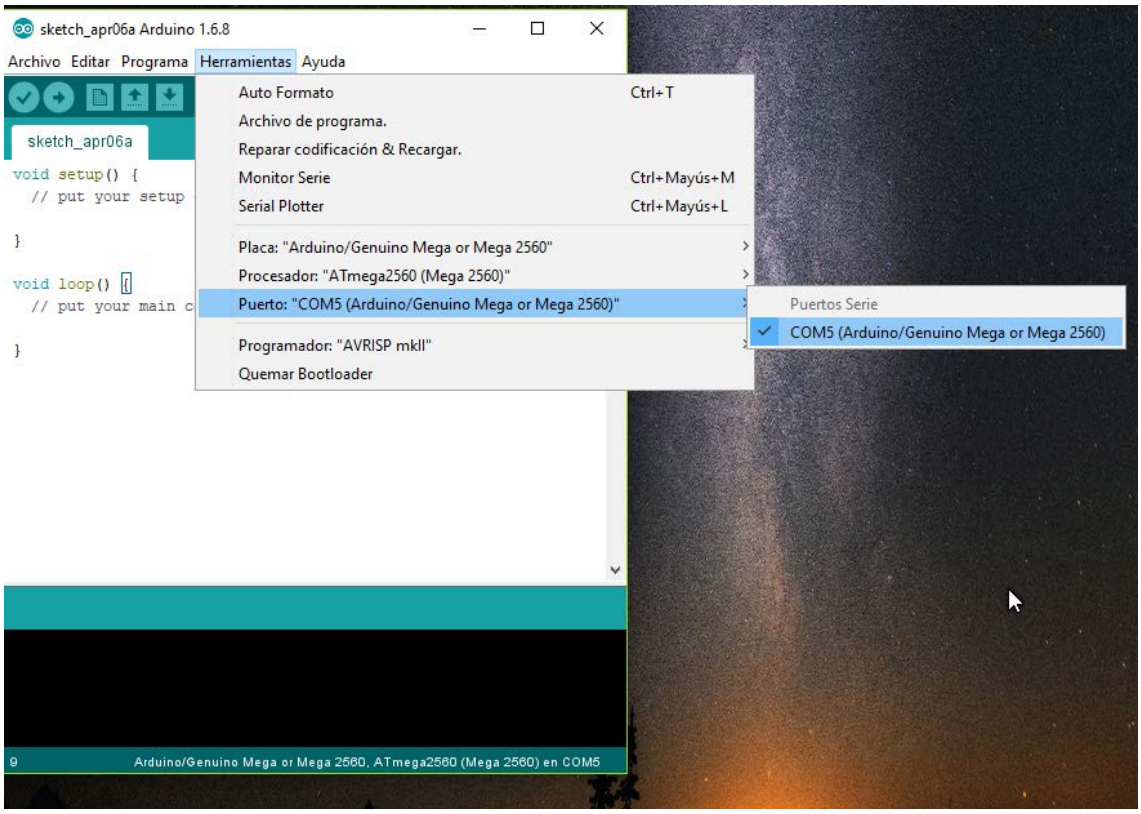


Ilustración 110 Seleccionar puerto COM
Fuente: Captura de pantalla

Para comprobar que los camios que hemos hecho han sido aplicados basta con mirar en la parte inferior del IDE de Arduino, que nos muestra el modelo de placa y el puerto COM seleccionados, como podemos ver en la ilustración 111.

Ilustración 111 Parámetros de configuración del Arduino seleccionados
Fuente: Captura de pantalla

Para seleccionar la velocidad de comunicación entre el Arduino y el ordenador vamos a abrir el monitor serie.

Para abrir el monitor serie hay que hacer clic en la lupa situada en la parte superior derecha del Arduino, como se señala en la ilustración 112, y seleccionamos una velocidad de bit de 115200 baudios.

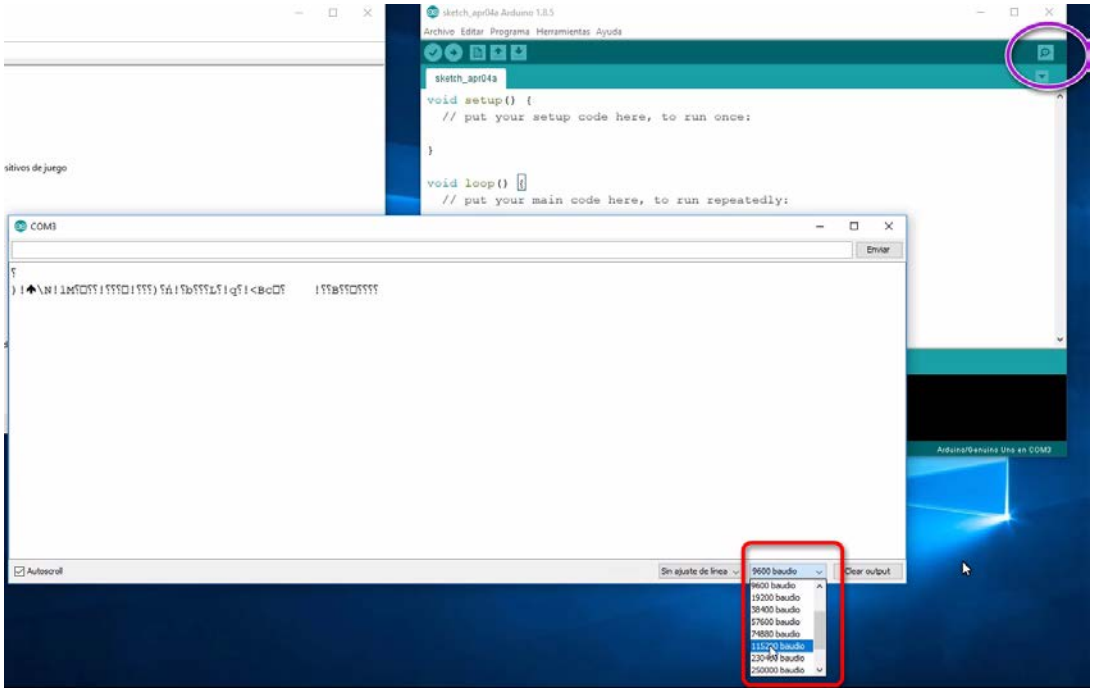


Ilustración 112 Seleccionar puerto COM
Fuente: Captura de pantalla

Para finalizar hacemos una verificación de que todo está correcto. Vamos a pinchar en el botón "Subir" para cargar un programa vacío en el sistema, como se muestra en la ilustración 113. Si ocurre algún problema aparecerá un mensaje de error en la consola inferior. Si el proceso alcanza el 100% significa que todo está correcto y ya se puede empezar a desarrollar programas.

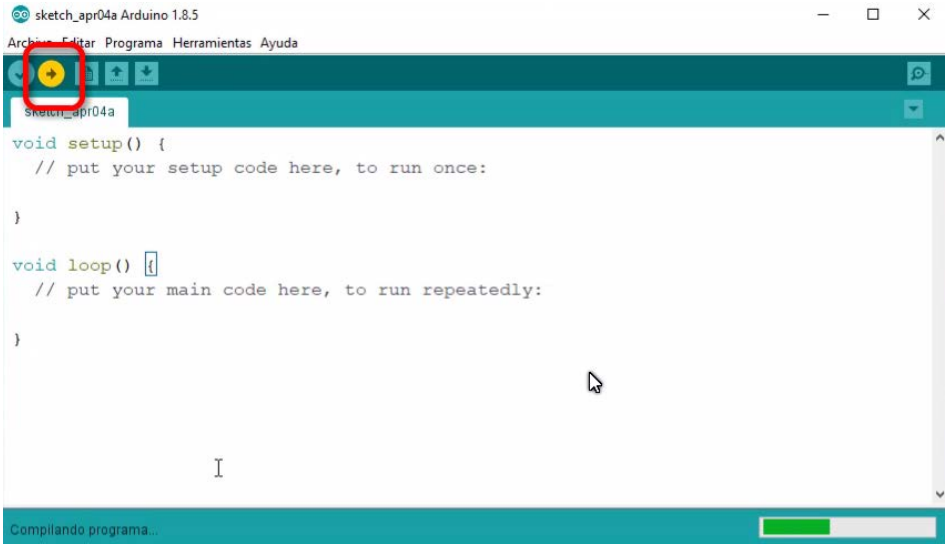


Ilustración 113 Compilando un programa en Arduino
Fuente: Captura de pantalla

8.4.2. Introducción a la programación de Arduino

La programación da la placa Arduino Mega 2560 se realiza en lenguaje C++. La estructura de un programa en Arduino se compone de al menos dos partes fundamentales. La función setup() que se ejecuta solo una vez al inicio del programa y es la encargada de recoger la configuración y la función loop() que es un bucle que contiene un código que se ejecuta permanentemente. Estas dos rutinas que se muestran en la ilustración 114, deben de aparecer siempre. Se pueden crear todas las rutinas adicionales que se quiera tanto antes o después de setup y loop.

El entorno IDE de Arduino dispone de una serie de botones, destacados en la ilustración 85, que sirven para un manejo rápido del código.

- Verificar: Comprueba el código en busca de errores.
- Subir: Compila el código y lo transfiere a la placa de Arduino
- Nuevo: Crea un nuevo sketch.
- Abrir: Presenta una ventana con los programas sketch instalados. Un clic sobre ellos lo abre en la ventana actual.
- Salvar: Guarda el programa sketch actual.
- Monitor Serie: Abre la ventana de monitorización serie.

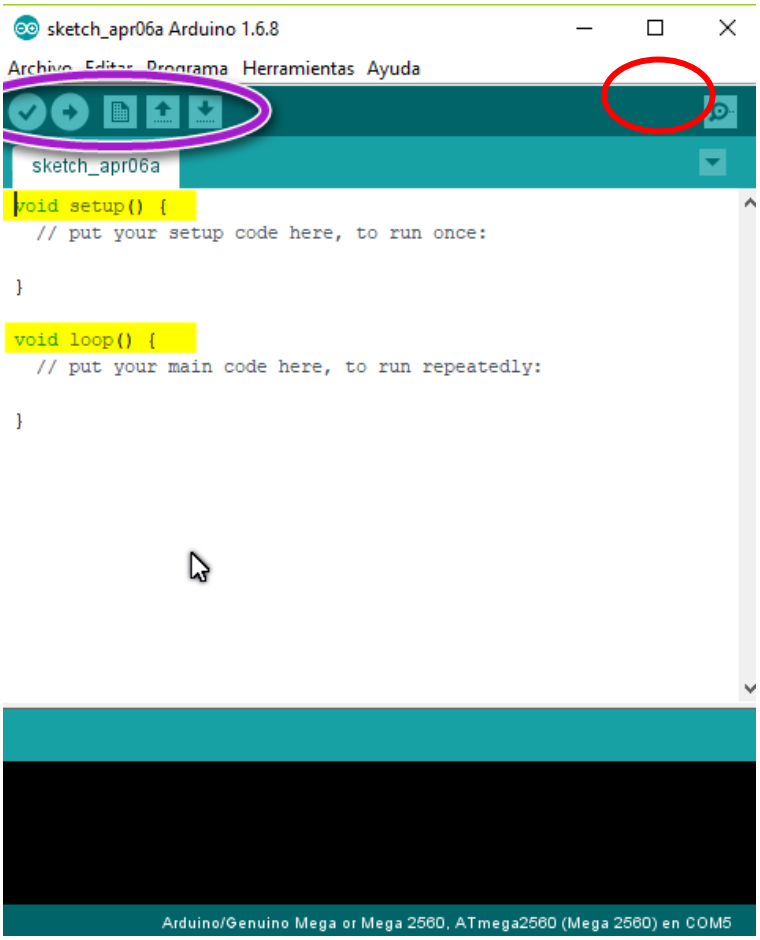


Ilustración 114 Entorno IDE de Arduino
Fuente: Captura de pantalla

8.4.3. Funciones básicas de Arduino

8.4.3.1. Control de entrada y salida

- **pinMode(<pin>, INPUT/OUTPUT).** Configura un pin como entrada o salida.

8.4.3.2. Programación de entradas/salidas digitales

- **digitalRead(<pin>);** Devuelve un valor 0 cuando la entrada es = GND o similar y un valor de 1cuando la entrada tiene voltaje = Vcc (3,3V - 5 V).
- **digitalWrite(<pin>, LOW/HIGH);** HIGH = 1 = encender. LOW = 0 = apagar

8.4.3.3. Programación de entradas/salidas analógicas

El microcontrolador digital no puede generar una señal analógica pero sí leerla. Para simular una salida analógica se utiliza la técnica de modulación por ancho de pulsos PWM, que consiste en enviar pulsos de voltaje fijo, pero con diferente duración.

- **analogRead(Ax);** Lectura analógica del pin x. Debe de tratarse de un puerto analógico (ADC). Devuelve un valor entre 0 y 1024.
- **analogWrite(<pin>, <valor>);** Puede emplearse en cualquier puerto. El valor del número tiene que estar entre 0 y 1024.

8.4.3.4. Programación del puerto serie

La forma de acceder al puerto serie es, como ya hemos visto, a través de la lupa que se encuentra en la parte superior derecha del IDE de Arduino. Nos va a servir de interfaz con el programa, al conectarse al PC, y para depurar el código y reprogramar la placa.

Hay que asegurarse que deben de coincidir los valores de velocidad de transferencia indicada en el código con el seleccionado en la interfaz del puerto serie, como se muestra en la ilustración 115.

Ilustración 115 Comprobación de la velocidad de transferencia seleccionada
Fuente: Captura de pantalla

- **Serial.begin(<baudrate>);** baudrate es la velocidad de transferencia. A mayor velocidad peor calidad de señal.
- **Serial.println("texto"); or Serial.println(<variable>);** Nos permite incluir una línea nueva de texto o con el valor de una variable en la consola.
- **Serial.print("texto");** Para que la impresión se haga en la misma línea.
- **Serial.available();** Devuelve un 1 si hay un mensaje en el registro de recepción, en caso contrario devuelve un 0.
- **Serial.read();** Devuelve la lectura del código ASCII recibido por el puerto serie.

8.4.3.5. Uso de retardos

- **delay(milisegundos);** Hace que la ejecución quede detenida durante el tiempo especificada. Durante este tiempo el procesador no hace nada.
- **delaymicroseconds(microsegundos);** Permite pausas de mayor precisión.

Las instrucciones anteriores son bloqueantes. Durante el tiempo de retardo el microprocesador no hace nada, y esto no es bueno.

- **millis();** Devuelve el tiempo de ejecución en milisegundos. Su valor se reinicia al alcanzar un valor determinado.

8.4.4. Uso de librerías I2C en Arduino

Generalmente cuando trabajamos con sensores es necesario importar una librería donde va a estar todo el código que necesitamos para usar el sensor y que ha sido implementado previamente por algún desarrollador. Vamos a descargar varios ficheros entre ellos, un fichero .H donde se encuentran las definiciones de las variables y de las rutinas que se pueden utilizar y un fichero .CPP donde se encuentra todo el código que debe de ejecutarse para trabajar con el sensor, para el protocolo de comunicación, etc.

Para integrar un sensor hay que seguir los siguientes pasos:

1. **Importar la librería.**

Para ello seguiremos el mismo procedimiento que el indicado en el punto 4.2.1.3 donde instalamos la librería de Thinger.io. Después de la instalación de la librería podemos encontrar los ficheros .H y .CPP accediendo a la nueva carpeta creada en el dentro del directorio donde esté instalado Arduino. Es interesante acceder al código de las rutinas para ver todas las capacidades de la librería. Como ejemplo vamos a instalar la librería LiquidCrystal_I2C.h que vamos a utilizar para el control de la pantalla de cristal líquido.

2. **Cargar la librería en nuestro código**

```
#include <LiquidCrystal_I2C.h>
```

3. **Instanciar el objeto del sensor**

```
LiquidCrystal_I2C lcd(0x3F,20,4);  (Declaración de la variable lcd de tipo LiquidCrystal_I2C)
```

4. **Dar de alta la comunicación**

```
lcd.init();
```

5. **Ejecutar una escritura**

```
lcd.print(Mensaje[0]);        (Muestra el mensaje 0 en la pantalla)
```

8.5. Anexo 5. Configuración plataforma Thinger.io

8.5.1. Iniciar Thinger.io

Lo primero que tenemos que hacer es crear una cuenta en la plataforma. Para ello accedemos a su página web https://thinger.io/, que vemos en la ilustración 116. Para el proceso de registro hacemos clic en “Sign up Now”.

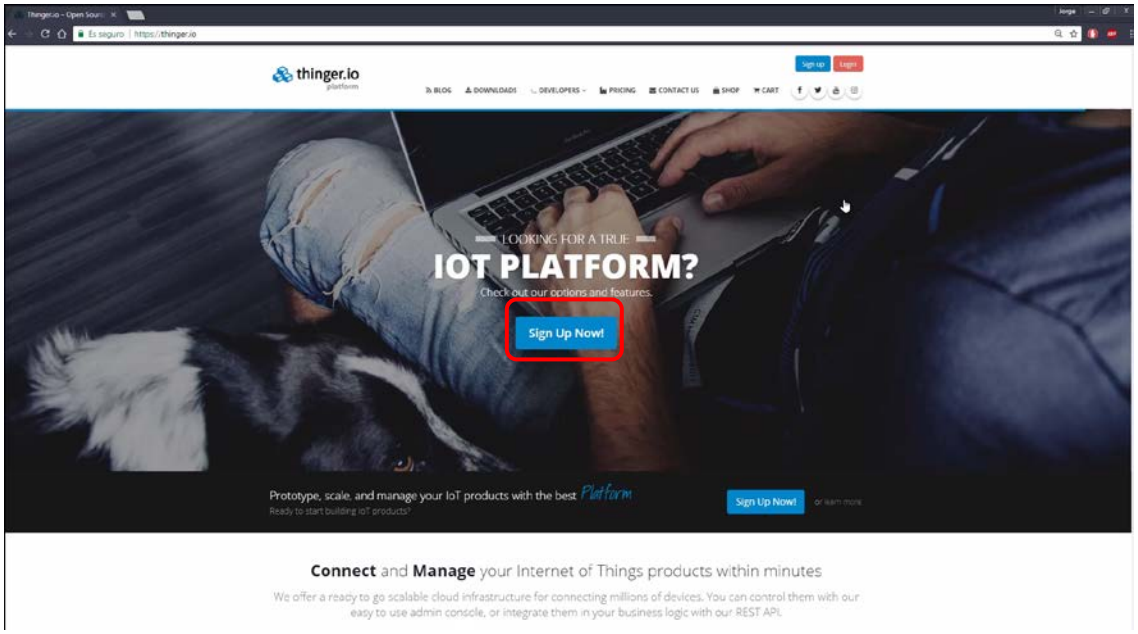


Ilustración 116 Pantalla principal de Thinger.io
Fuente: Captura de pantalla

Nos aparece la pantalla que se muestra en la ilustración 117. Una vez cubierto todo hacemos clic en Sign up.

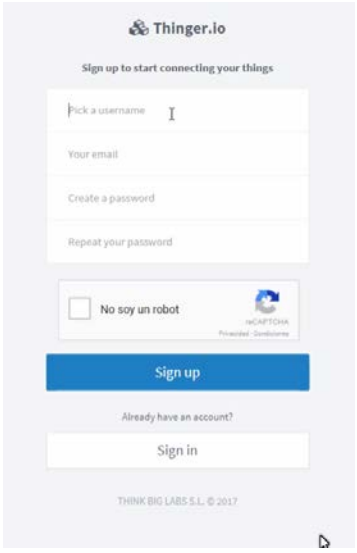


Ilustración 117 Pantalla de registro de Thinger.io
Fuente: Captura de pantalla

Ahora solo queda acceder al correo electrónico y confirmar el email.

Una vez creada la cuenta solo queda logearnos para entrar en la plataforma, para ello introducimos los datos de la cuenta creada y hacemos clic en “Log in” en la pantalla que aparece en la ilustración 118.

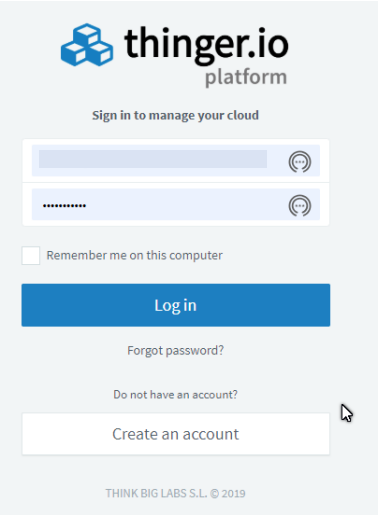


Ilustración 118 Pantalla de acceso de Thinger.io
Fuente: Captura de pantalla

Y accedemos al workspace de la plataforma, que vemos en la ilustración 119. Es una página con los contadores y estadísticas de uso. Nos muestra de una forma visual los recursos disponibles, los definidos, su ubicación y estadísticas de la transmisión de datos.

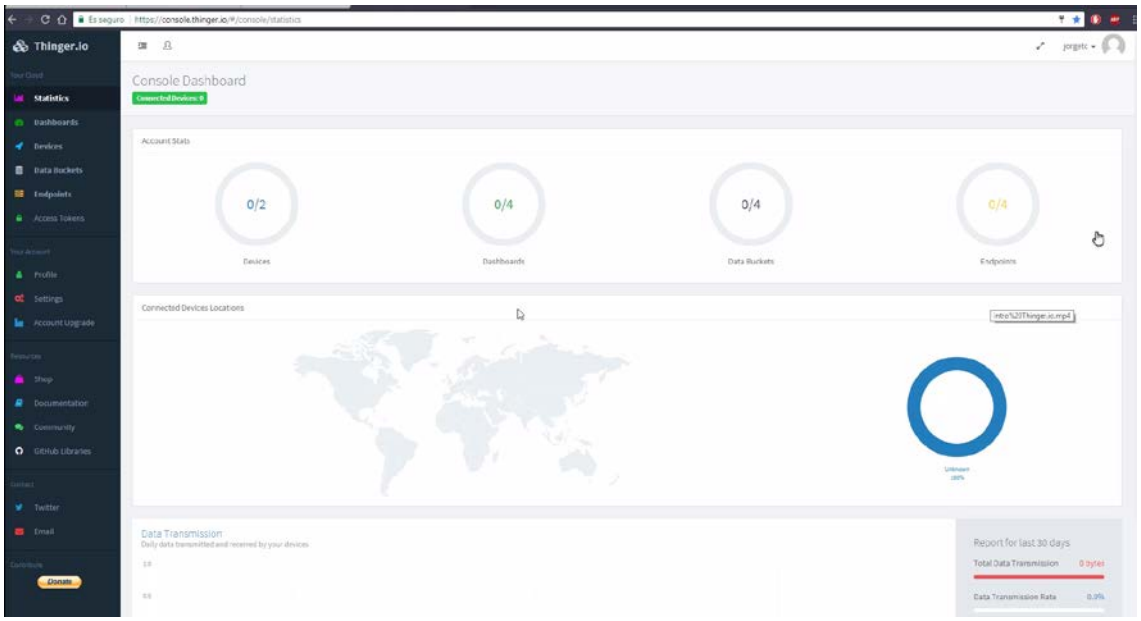


Ilustración 119 Workspace de Thinger.io
Fuente: Captura de pantalla

En el lateral izquierdo encontramos el menu. Es de destacar el acceso a la comunidad de usuarios.Ver la ilustración 120.

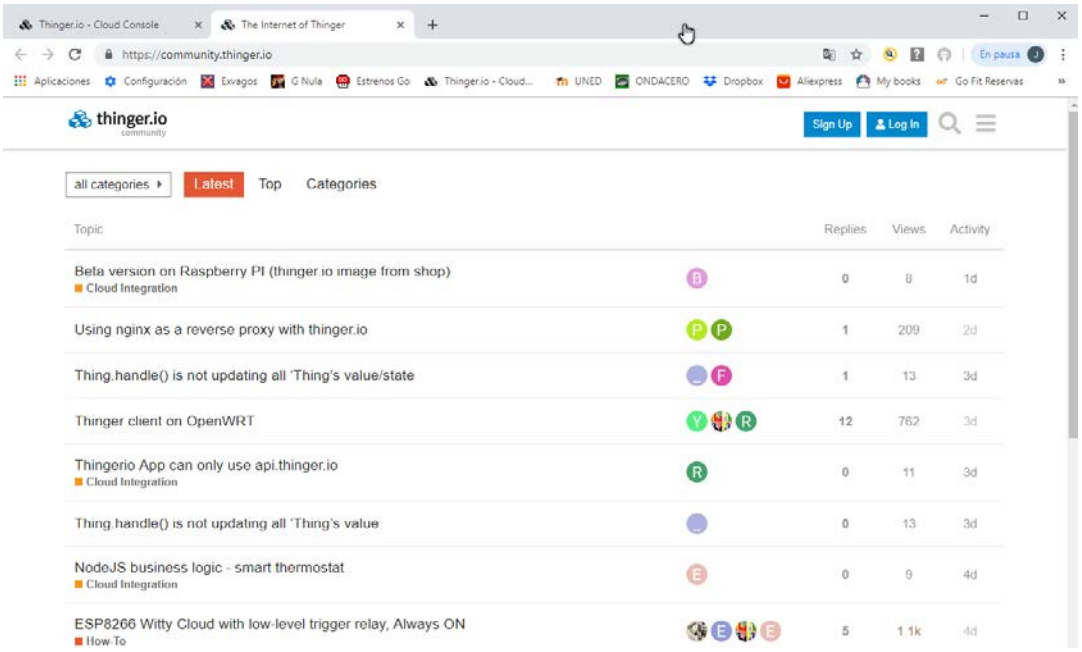


Ilustración 120 Comunidad de usuarios de Thinger.io
Fuente: Captura de pantalla

y a la documentación (ilustración 121),

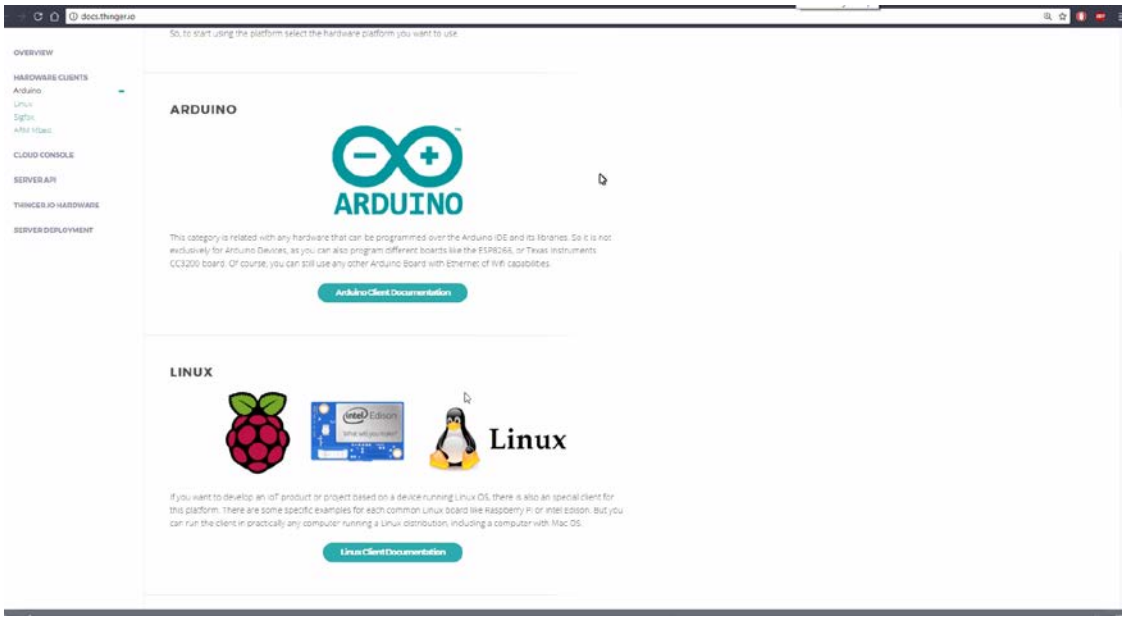


Ilustración 121 Documentación de Thinger.io
Fuente: Captura de pantalla

con un apartado específico a Arduino que vemos en la ilustración 122.

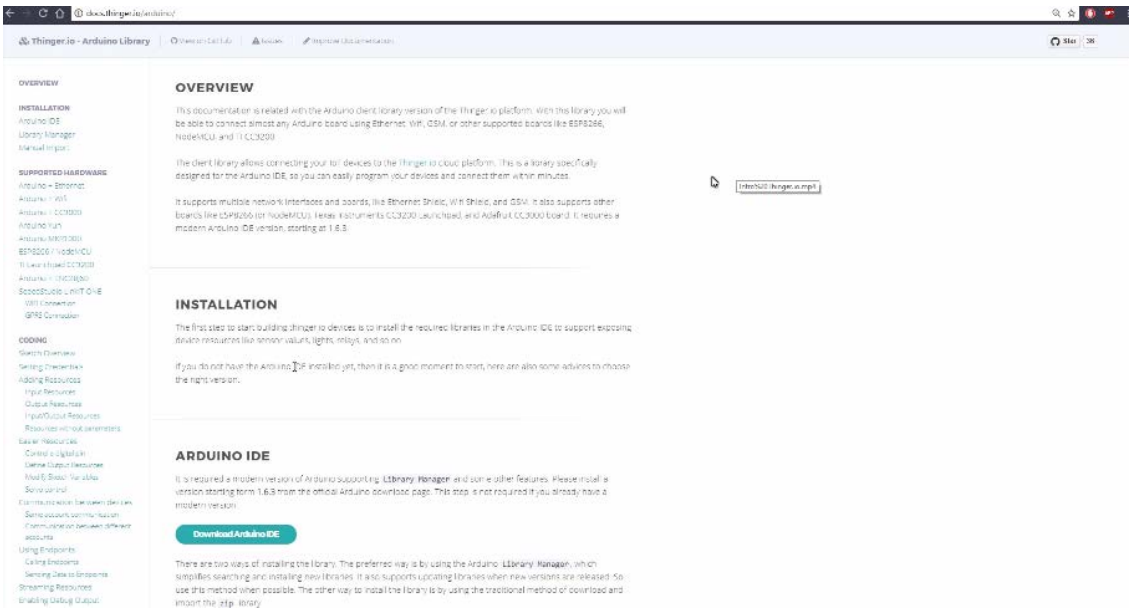


Ilustración 122 Documentación sobre Arduino en Thinger.io
Fuente: Captura de pantalla

8.5.2. Dar de alta un dispositivo IoT

Accedemos a la pestaña Devices y pinchamos en el botón Add Device. Ver ilustración 123.

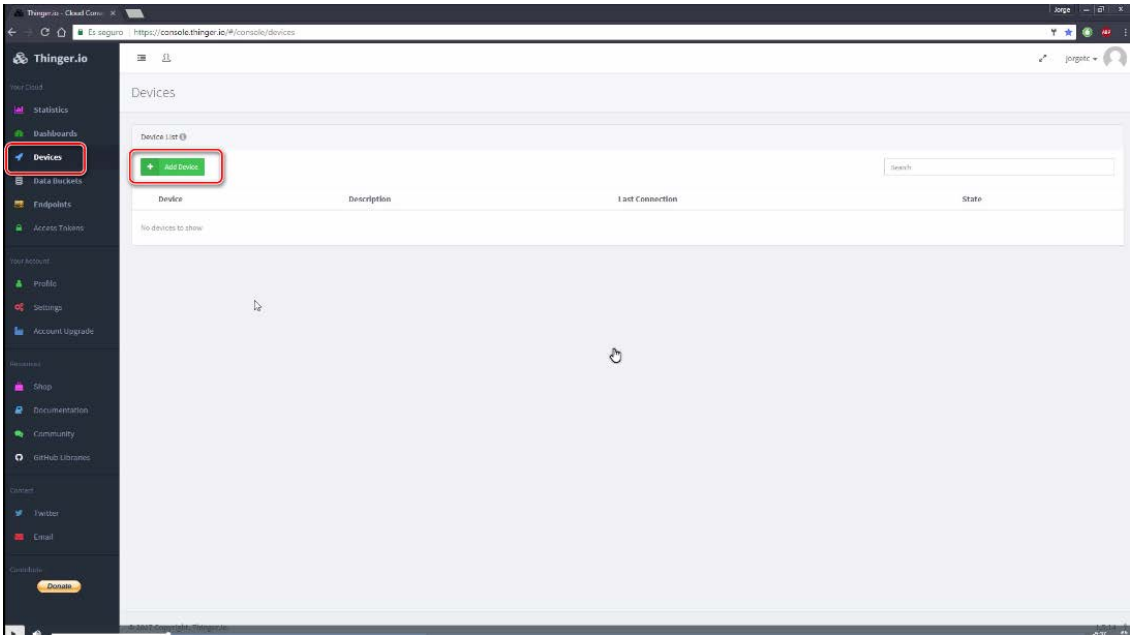


Ilustración 123 Dar de alta un dispositivo en Thinger.io
Fuente: Captura de pantalla

Nos aparece un nuevo formulario. El device credencial podemos introducir la clave que queramos o generarla aleatoriamente con el botón Generate Random Credential. Cumplimentamos todos los campos y hacemos clic en Add Device en la pantalla que vemos en la ilustración 124.

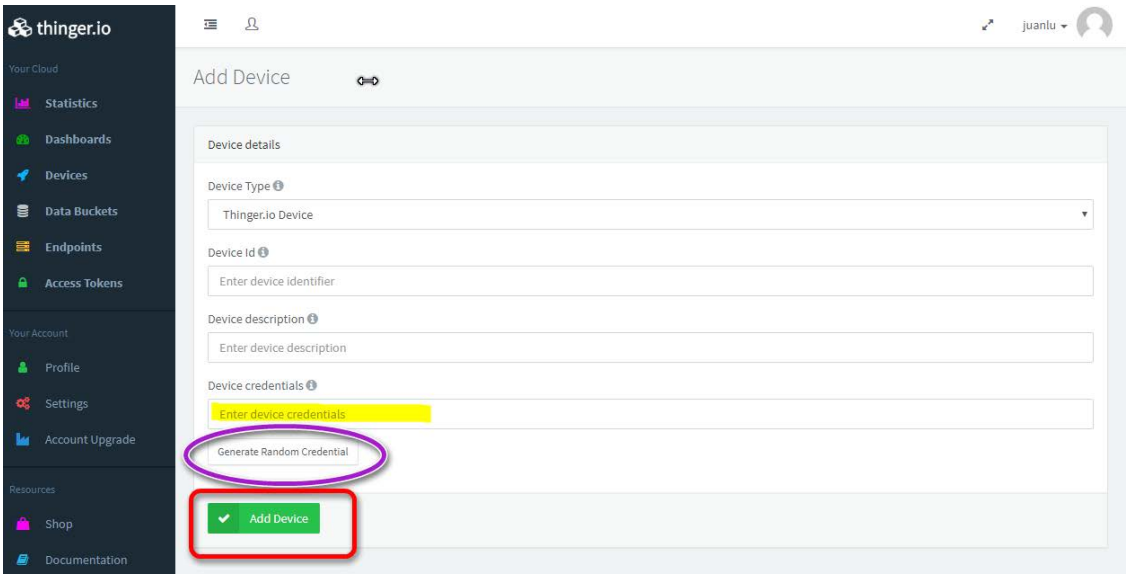


Ilustración 124 Campos para añadir un dispositivo en Thinger.io
Fuente: Captura de pantalla

Ya tenemos el primer dispositivo. Si vamos al interfaz anterior aparece ya el dispositivo, no obstante, vemos que está desconectado como se muestra en la ilustración 125.

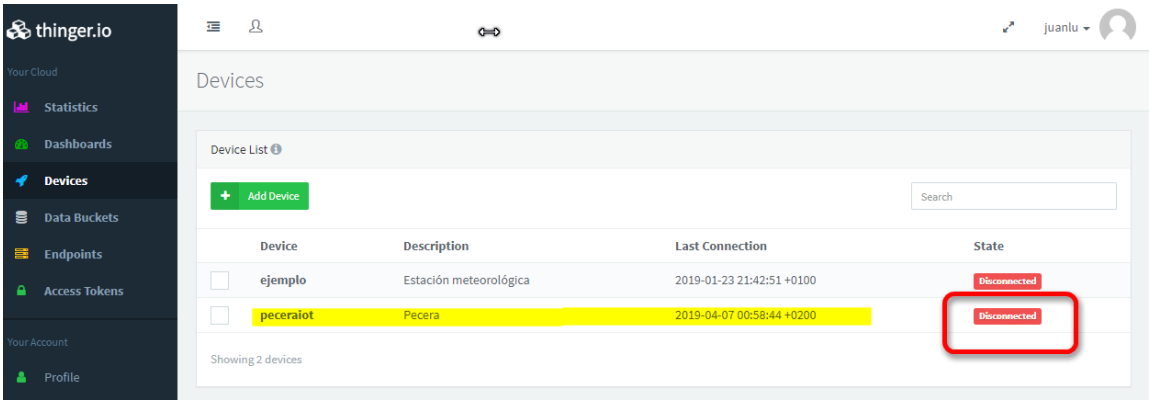


Ilustración 125 Añadir un dispositivo en Thinger.io
Fuente: Captura de pantalla

El siguiente paso es ir a Arduino y reprogramar la placa de Arduino Mega para establecer la comunicación entre ella y Thinger.io.

8.5.3. Conectar la placa de Arduino Mega con Thinger.io

Es necesario tener cargada la librería Thinger.io. Abrimos el IDE de Arduino. Para ver el funcionamiento la forma más fácil es a partir de un ejemplo, para ello vamos a Archivo > Ejemplos > thinger.io > Arduino > ArduinoEthernet, para cargar el ejemplo ArduinoEthernet. Ver ilustración 126.

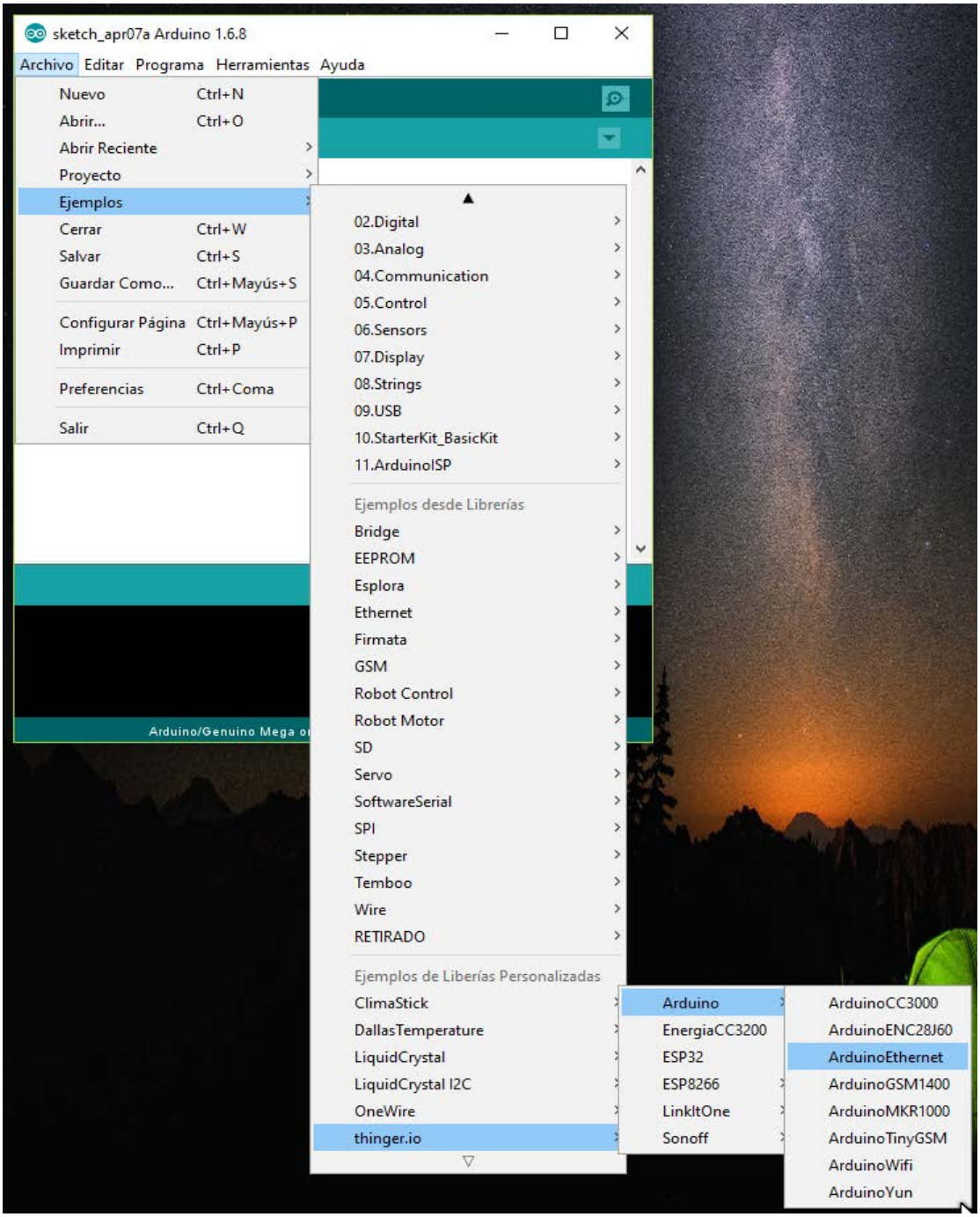


Ilustración 126 Cargar el archivo ejemplo ArduinoEthernet
Fuente: Captura de pantalla

Para que el programa sea operativo hay que rellenar los parámetros marcados en la ilustración 127, antes de poder subirlo a la placa Arduino.

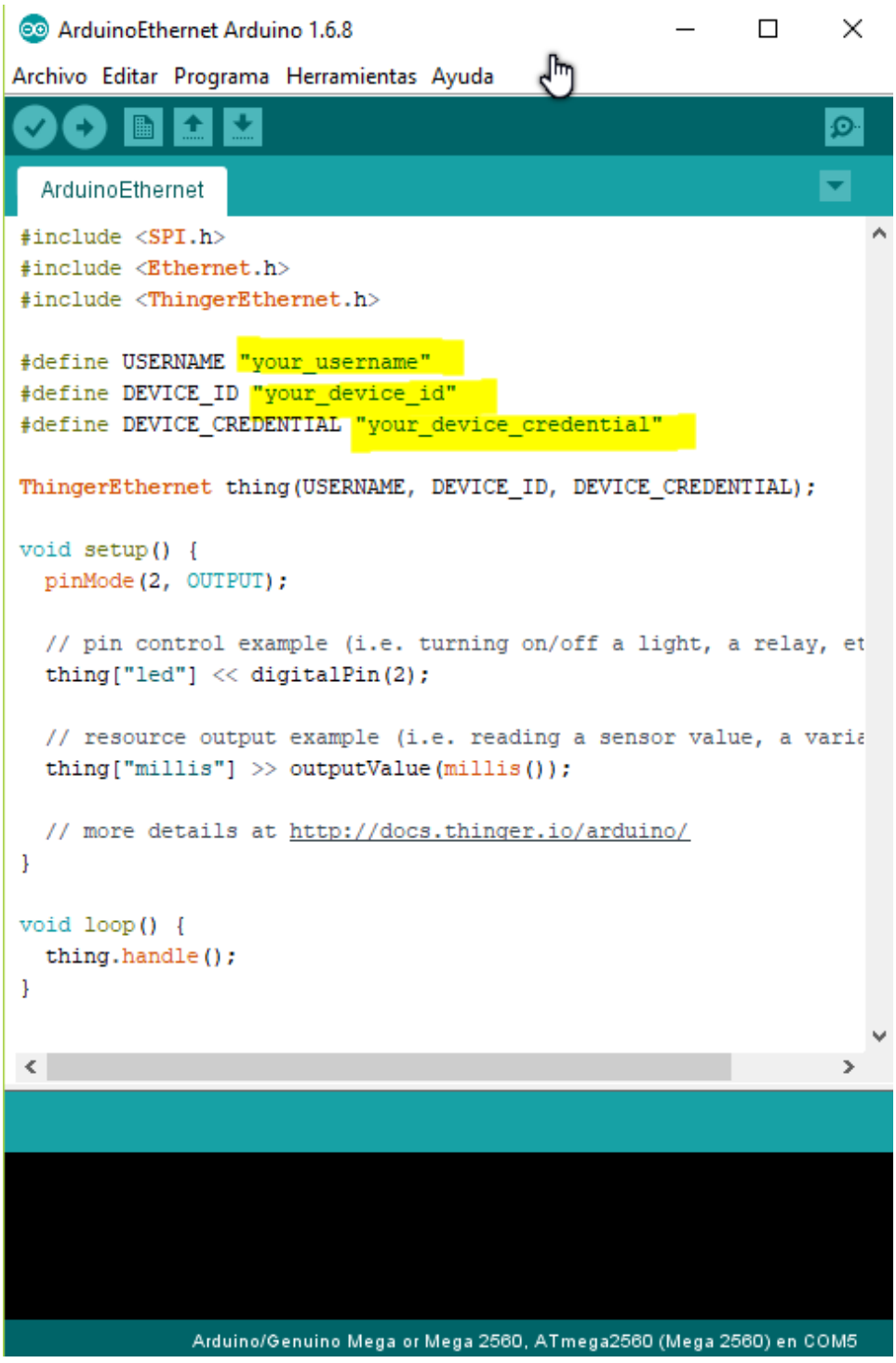


Ilustración 127 Código del Programa ejemplo ArduinoEthernet
Fuente: Captura de pantalla

Estos datos los obtenemos de la plataforma Thinger.io, accediendo al menú Devices. En la ilustración 128 se muestra donde se obtienen los datos.

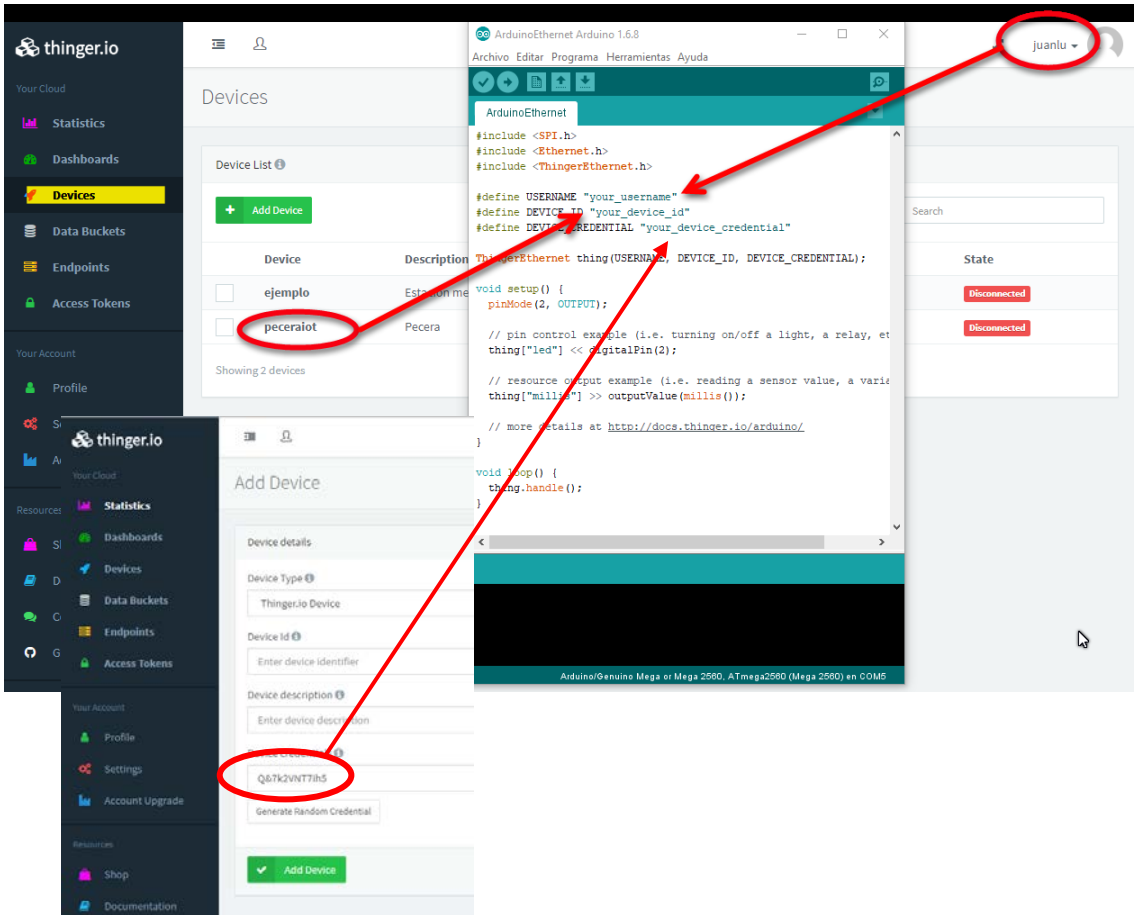


Ilustración 128 Datos para la conexión ethernet
Fuente: Captura de pantalla

En el ejemplo ArduinoEthernet que hemos cargado ya podemos ver alguno de los comandos básico. Un recurso de entrada nos va a permitir y apagar un led conectado en el pin 2 de la placa Arduino Mega y un recurso de salida que va a enviar a la plataforma de Thinger.io el valor del contador del programa. Estos recurso están dentro de la función setup(). Antes del setup() vemos la instrucción que se encarga de la conexión ethernet.

ThingerEthernet thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

En la función loop() vemos la instrucción encargada de la transferencia de datos

thing.handle();

Todo esto lo podemos ver en la ilustración 129

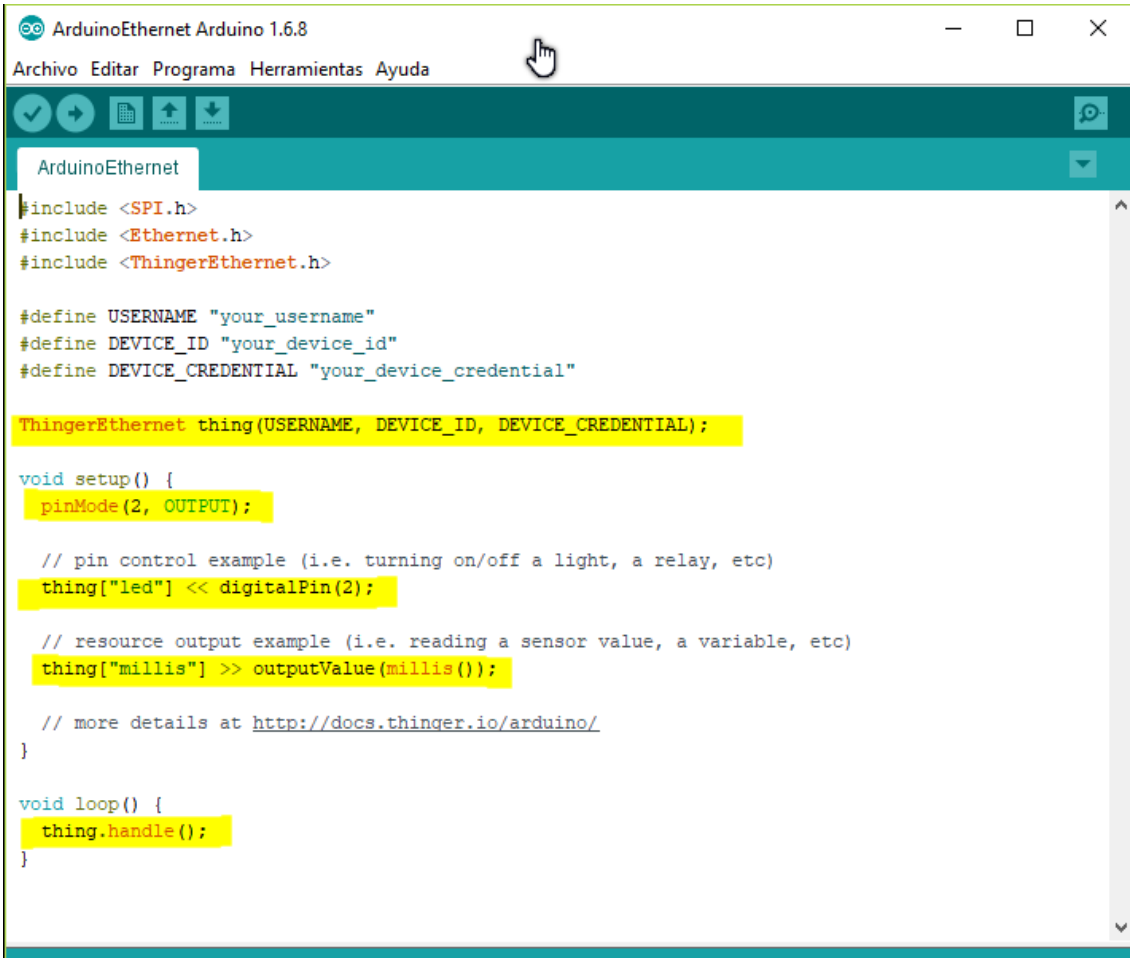


Ilustración 129 Comandos básico
Fuente: Captura de pantalla

8.5.4. Recursos con Thinger.io

Si volvemos a la API de Thinger.io, dentro del menú Devices, vemos que el dispositivo está conectado, como muestra la ilustración 130.

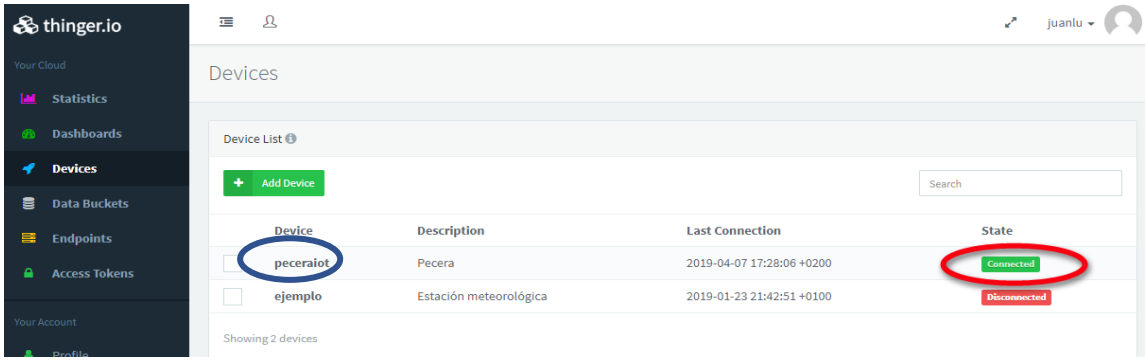


Ilustración 130 Dispositivo conectado
Fuente: Captura de pantalla

Si hacemos clic en el nombre del dispositivo nos aparece la ventana mostrada en la ilustración 131. Para acceder a la API del dispositivo hacemos clic en el botón derecho inferior View API.

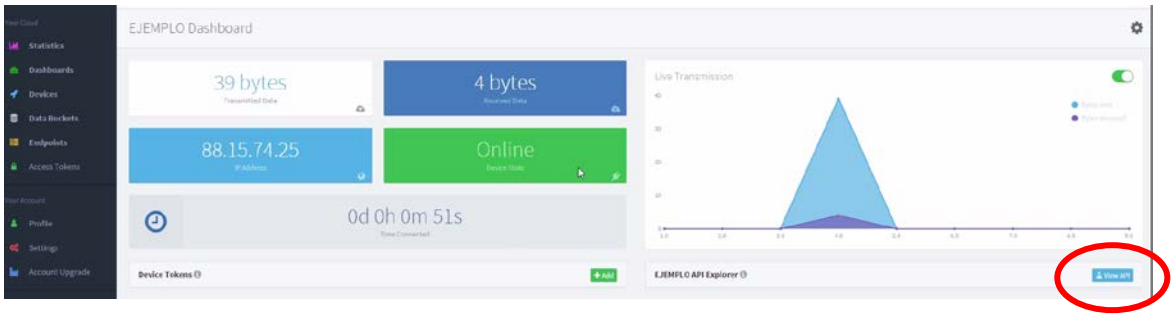


Ilustración 131 Dashboard de dispositivo
Fuente: Captura de pantalla

Y se abre una pantalla nueva que muestra los recursos, que coincide con los recursos creados en el código del programa Arduino, como comprobamos en la ilustración 132.

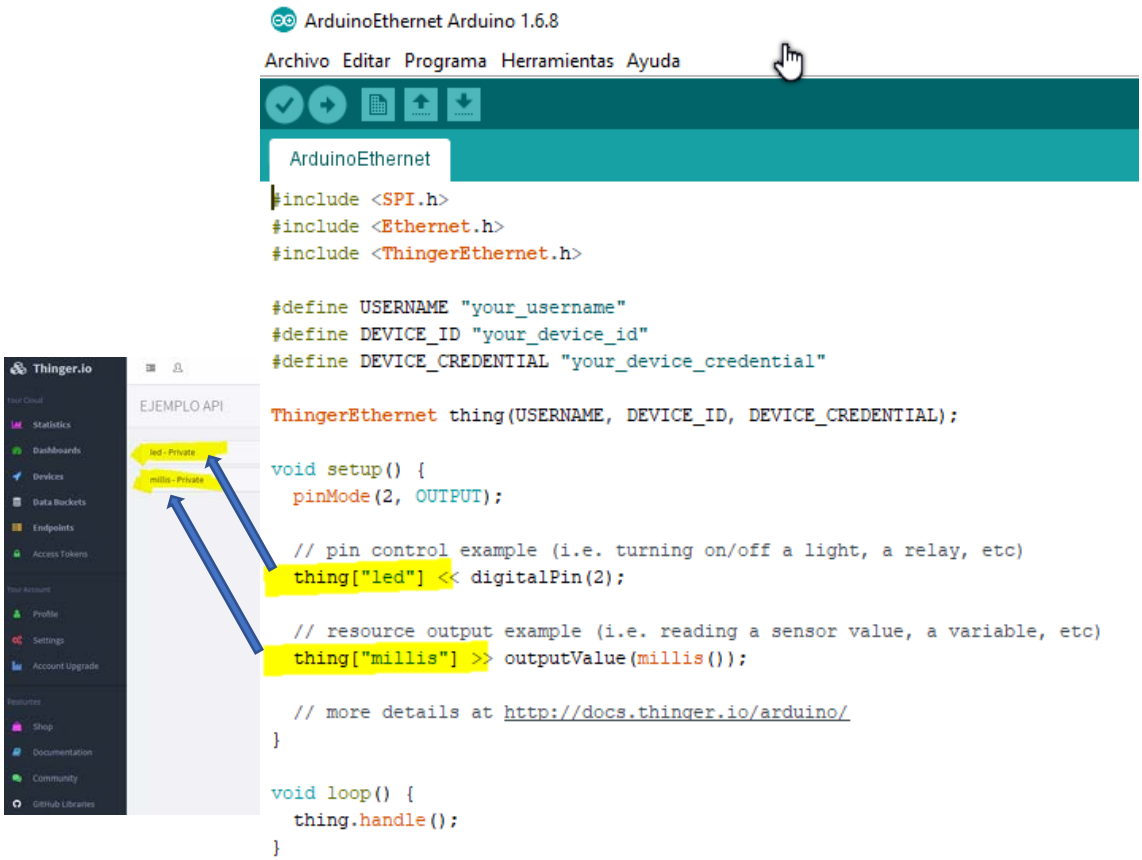


Ilustración 132 Coincidencia de recursos Arduino Thinger
Fuente: Captura de pantalla

Haciendo clic en cada uno de los recursos podemos abrirlos y explorarlos. Ver ilustración 133.

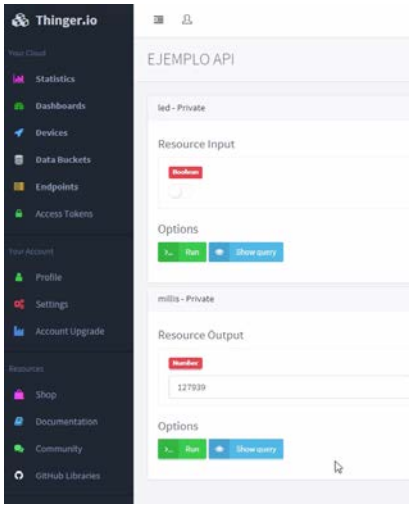


Ilustración 133 Pantalla recursos
Fuente: Captura de pantalla

En el caso del recurso “led” aparece un interruptor y cada vez que lo pulsemos se enciende y paga el led que está conectado en el pin 2 de la placa Arduino Mega. En el caso del recurso “millis” vemos como el valor del contador de programa se refresca cada vez que hacemos clic sobre el botón “Run”. Si hacemos clic sobre el botón “query” vemos sintácticamente como se realiza la petición. En la ilustración 134 vemos el código para desactivar el recurso rele0.

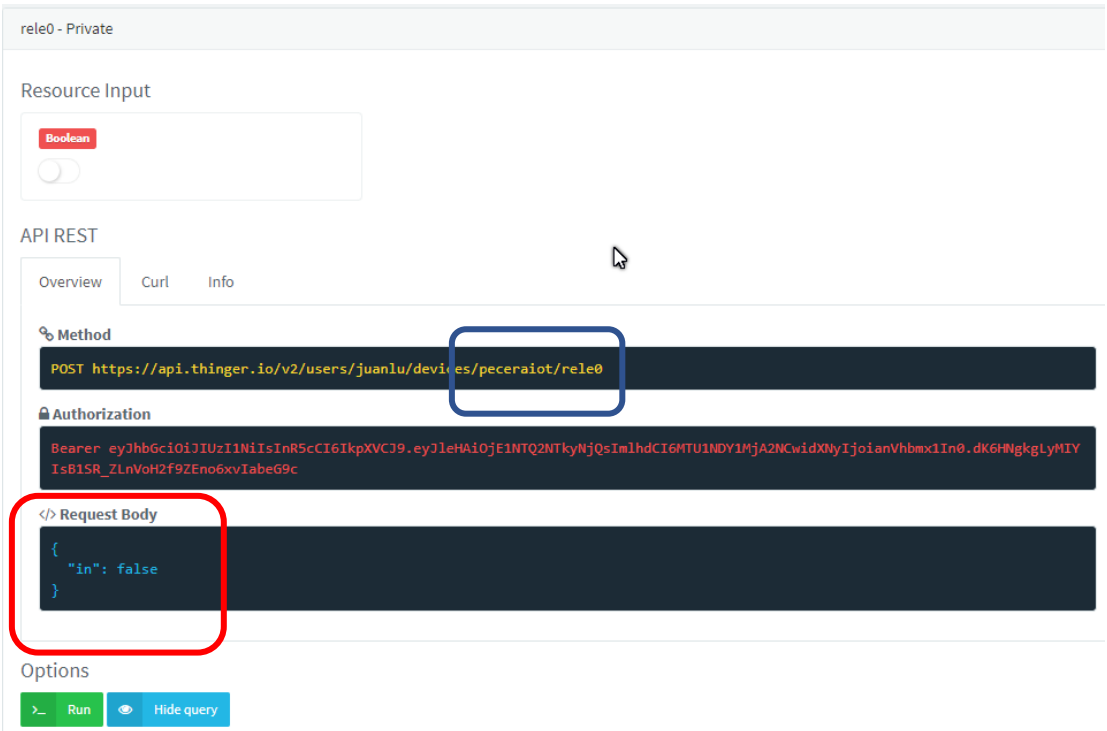


Ilustración 134 Código de petición de un recurso
Fuente: Captura de pantalla

8.5.5. Almacenamiento de datos con Thinger.io. Data Buckets

8.5.5.1. Crear un Data Bucket

Para crear un bucket accedemos a la pestaña Data Buckets y aparece la pantalla mostrada en la ilustración 135.

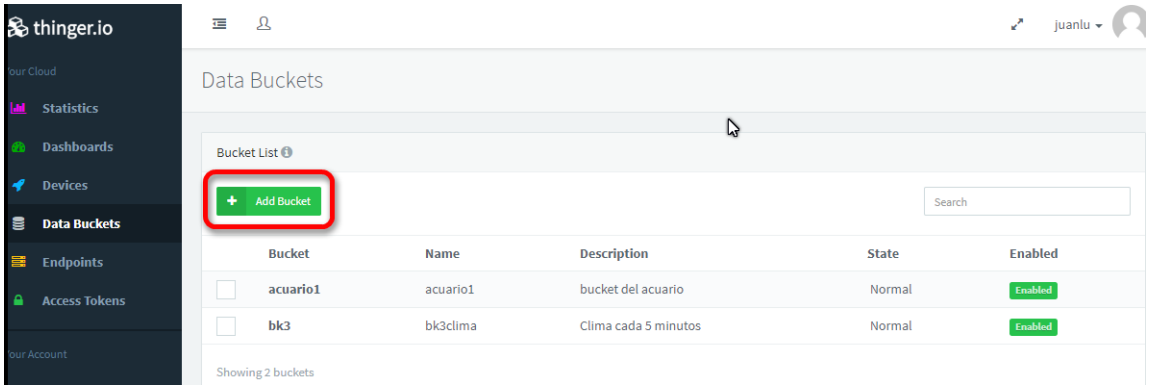


Ilustración 135 Añadir un bucket
Fuente: Captura de pantalla

Haciendo clic en Add Bucket nos aparece la pantalla mostrada en la ilustración 136 y vamos cumplimentando los datos que nos solicita. El botón "Enabled" sirve para que en un momento determinado podamos dejar de almacenar los datos de un dispositivo sin necesidad de desconectarlo o de cambiar su programación. A continuación, seleccionamos una fuente de datos, de las dos opciones disponibles.

- **From Device Resource:** Nos permite elegir el dispositivo y el recurso del que queremos almacenar la información.
- **From Write Call:** Nos permite almacenar datos que provengan de diferentes dispositivos y diferentes recursos, por ejemplo, para crear una red de dispositivos que vuelquen información sobre una misma base de datos.

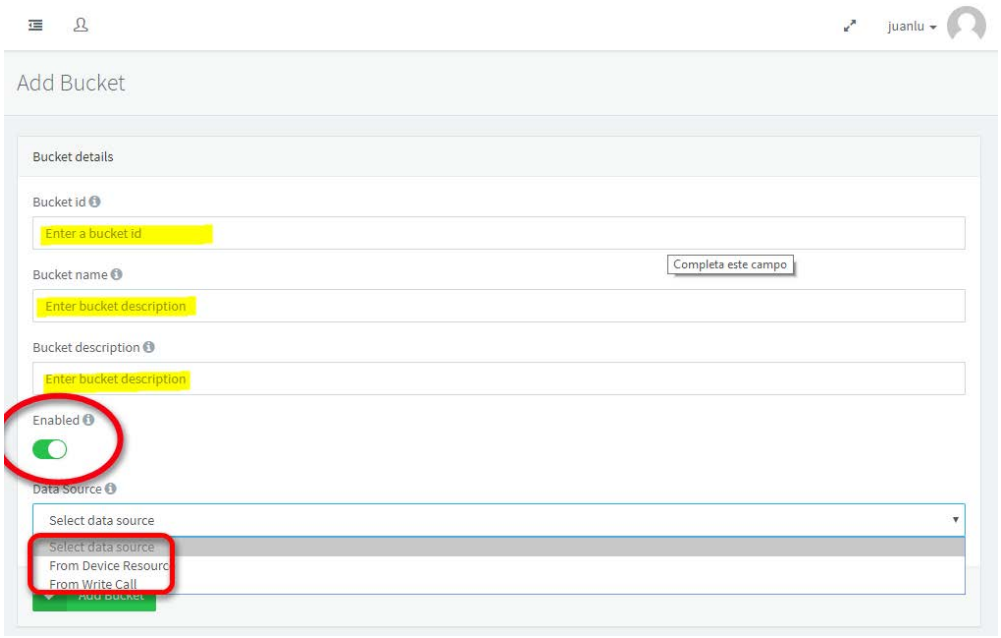


Ilustración 136 Select data source
Fuente: Captura de pantalla

Seleccionamos **From Device Resource**. A continuación, seleccionamos el dispositivo. Para ello es necesario que el dispositivo esté conectado. Ver ilustración 137.

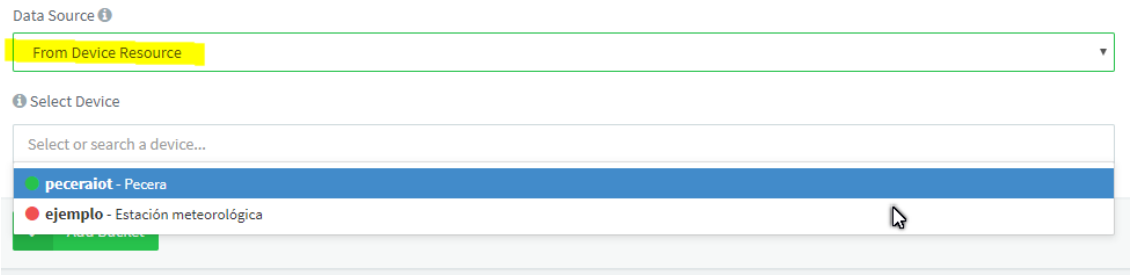


Ilustración 137 Seleccionar dispositivo
Fuente: Captura de pantalla

Seleccionamos el recurso y por último el modo de refresco. Ver ilustración 138.

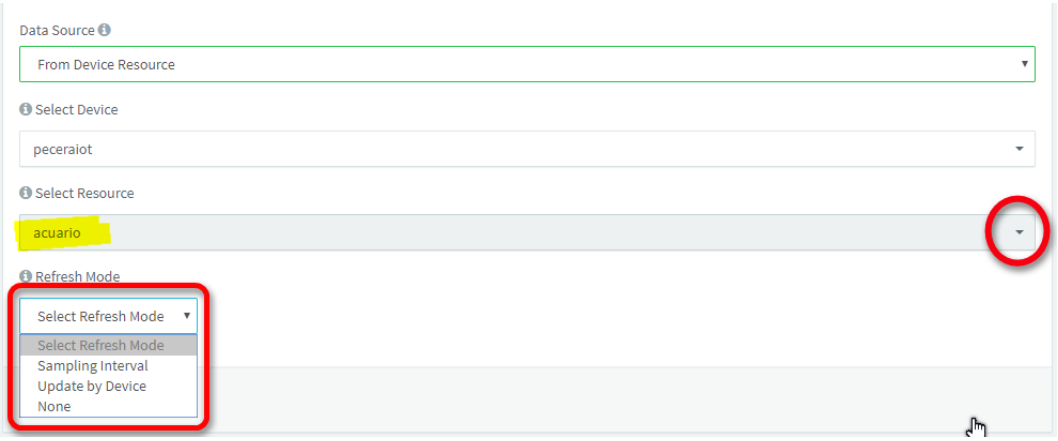


Ilustración 138 Select Refresh Mode
Fuente: Captura de pantalla

Para el modo de refresco tenemos tres opciones.

- **Sampling Interval:** Permite indicar un periodo de tiempo fijo para el envío de datos.
- **Update by Device:** El dispositivo es el que decide cuando se envían los datos al bucket.
- **None**: Sin tiempo de muestreo.

Para finalizar solo queda hacer clic en “Add Bucket”.

Veamos ahora como utilizar la funcionalidad **From Write Call**, que nos permite enviar datos desde diferentes dispositivos. Ver ilustración 139.

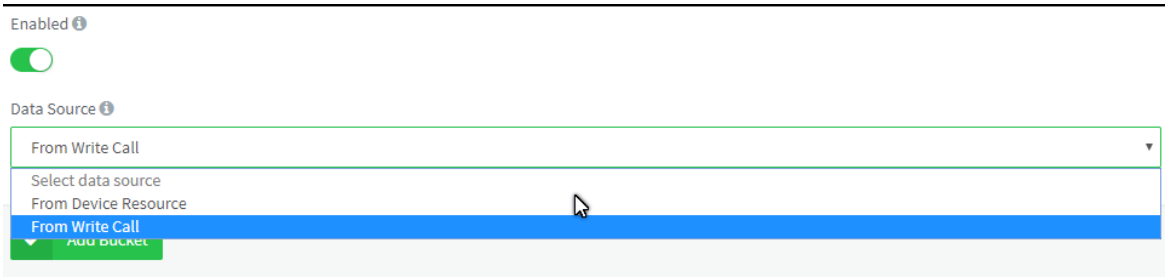


Ilustración 139 From Write Call
Fuente: Captura de pantalla

Para que esta opción funcione hay que implementar en el código del programa Arduino una función "stream" como vemos en la ilustración 140.

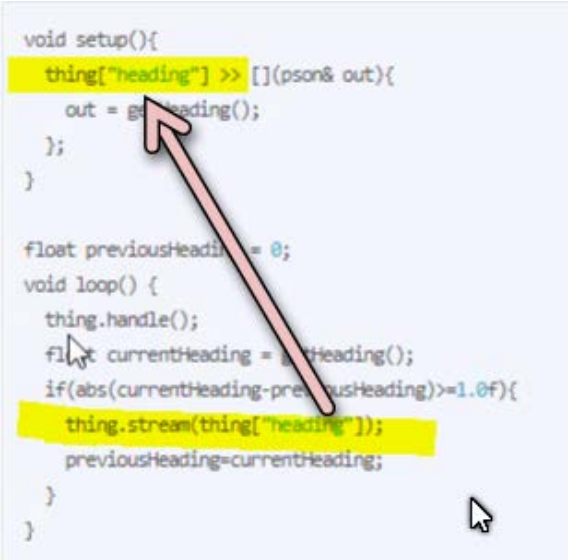


Ilustración 140 Función Thing.stream()
Fuente: Captura de pantalla

Es muy importante que exista una estructura de control que evite el continuo envío de datos a la plataforma.

8.5.5.2. Explorar un Data Bucket

Desde el menú principal seleccionamos la pestaña Data Buckets, y hacemos clic en el Data bucket que queremos explorar. Como vemos en la ilustración 141 nos aparece una pantalla que se divide en tres partes.

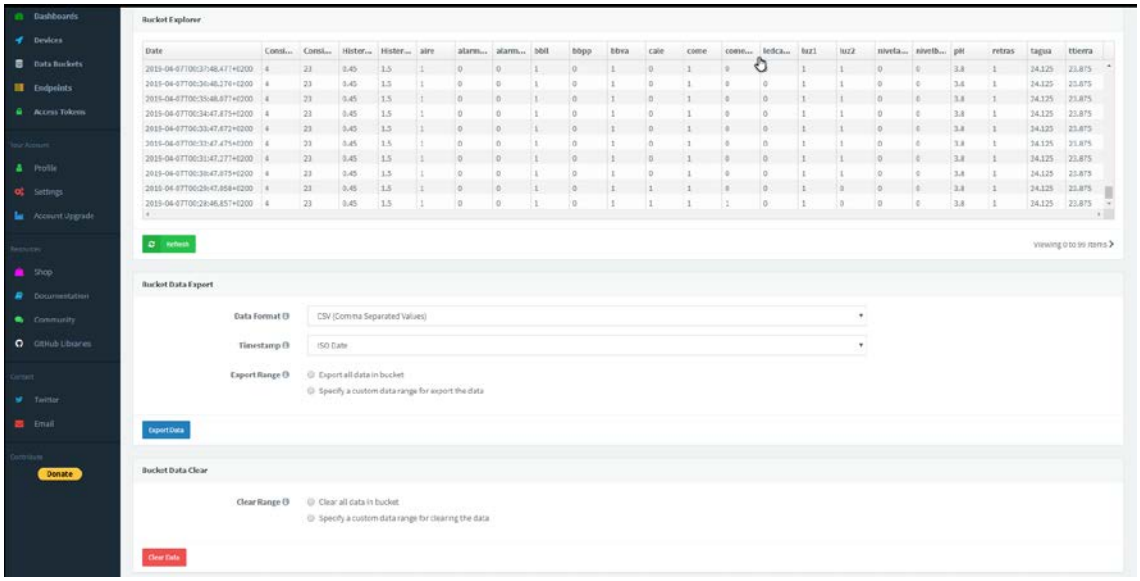


Ilustración 141 Pantalla de un Data Bucket
Fuente: Captura de pantalla

En la parte superior aparecen los últimos datos registrados. Ver ilustración 142. Haciendo clic en el botón “Refreís” se muestran los nuevos datos almacenados.

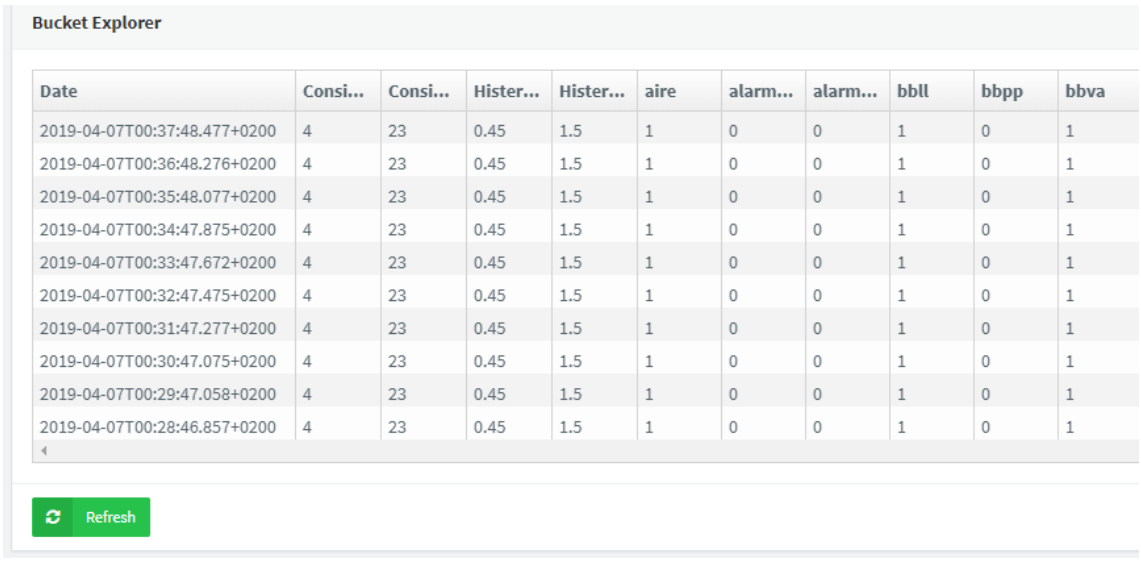


Ilustración 142 Datos registrados en un Data Bucket
Fuente: Captura de pantalla

En la pantalla intermedia nos aparecen unas opciones para configurar los datos para exportar. Podemos escoger el formato para exportar los datos, que puede ser CSV, ARFF o Json, como vemos en la ilustración 143.

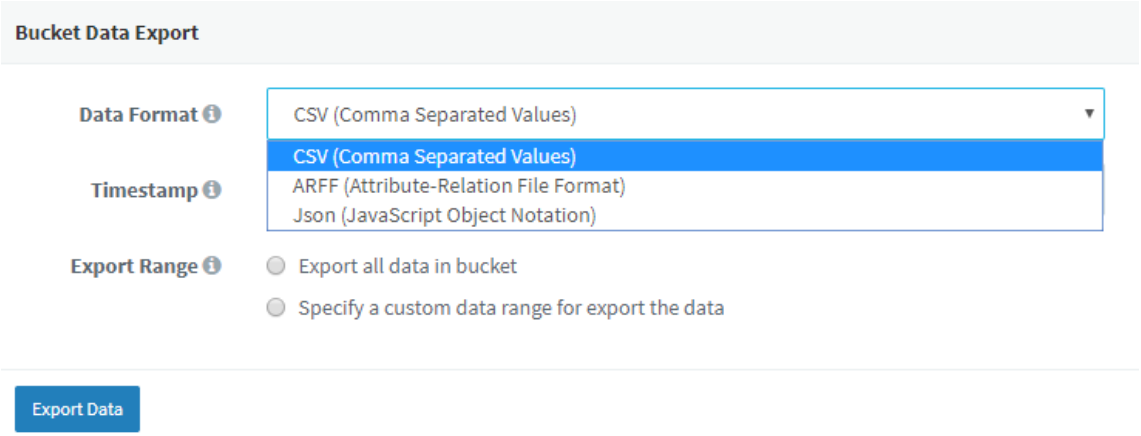


Ilustración 143 Tipos de formatos para exportar datos
Fuente: Captura de pantalla

También podemos seleccionar el formato cronológico, ilustración 144.

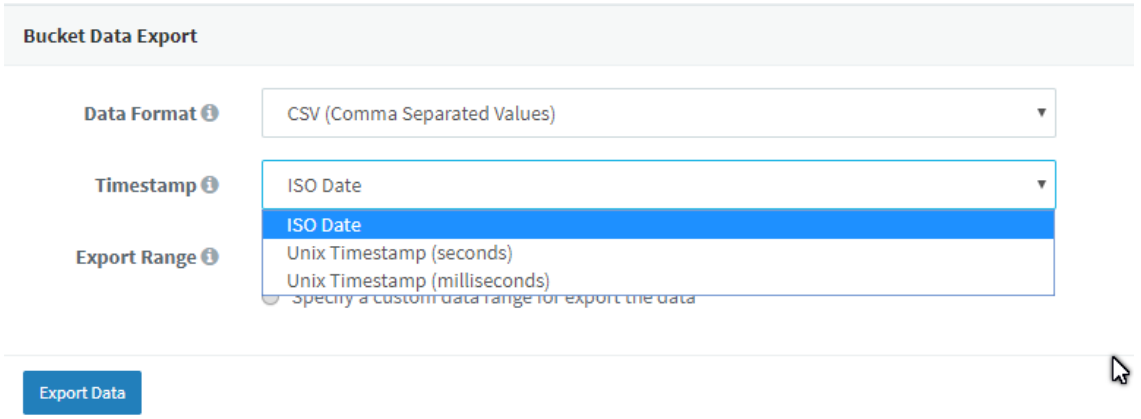


Ilustración 144 Formato cronológico para exportar datos
Fuente: Captura de pantalla

Y finalmente escogemos si exportar todos los datos o un rango. Ver ilustración 145.

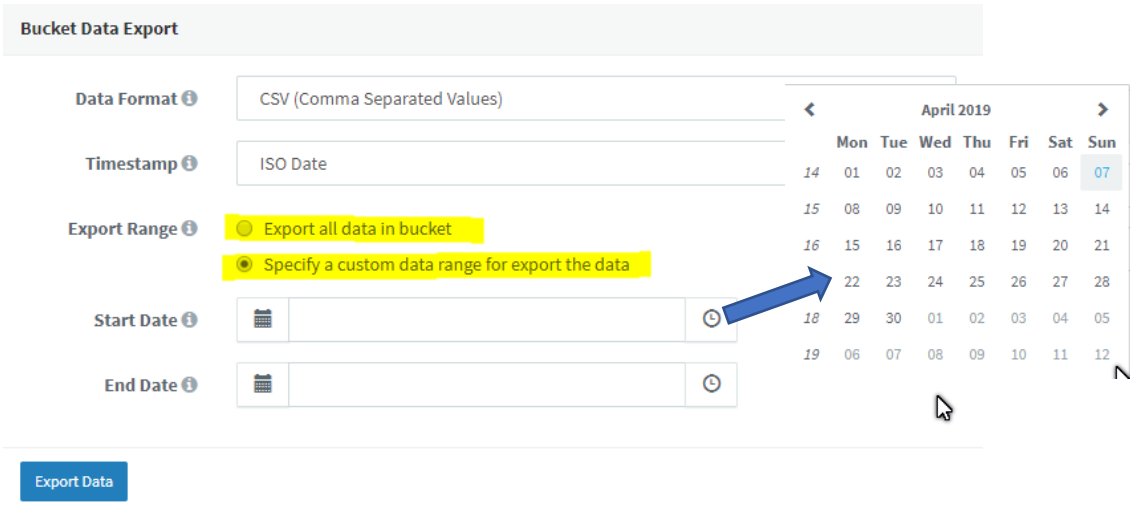


Ilustración 145 Selección de datos para exportar
Fuente: Captura de pantalla

En la parte inferior tenemos las opciones para eliminar datos. Podemos eliminar el Bucket o especificar un rango de datos para borrar. Ver ilustración 146.

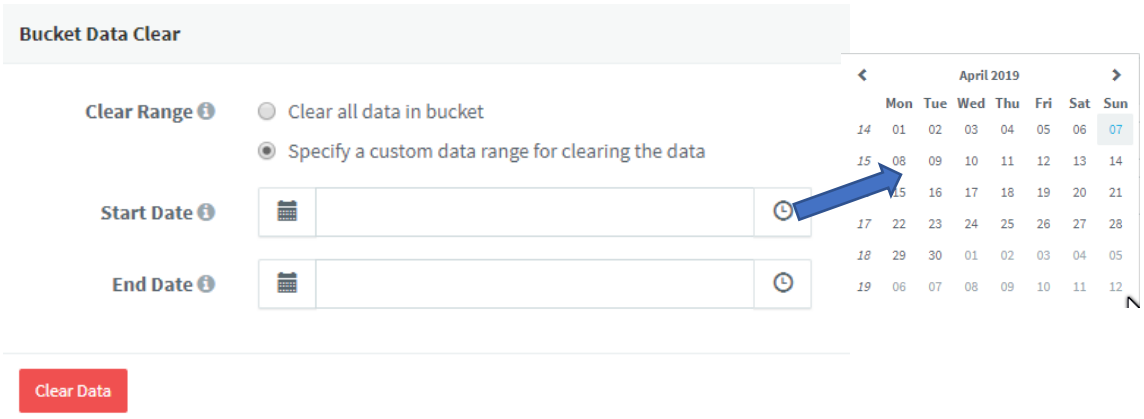


Ilustración 146 Selección de datos para eliminar
Fuente: Captura de pantalla

8.5.6. Visualización y análisis gráfico de datos. Dashboards

Para crear un Dashboard hay que hacer ir al menú Dashboard y hacer clic en el botón "Add Dashboard", como se muestra en la ilustración 147.

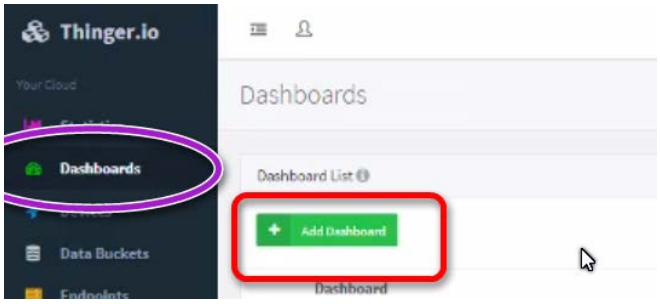


Ilustración 147 Añadir un Dashboard
Fuente: Captura de pantalla

A continuación, rellenamos los campos de la nueva pantalla y hacemos clic en el botón "Add Dashboard", como se muestra en la ilustración 148.

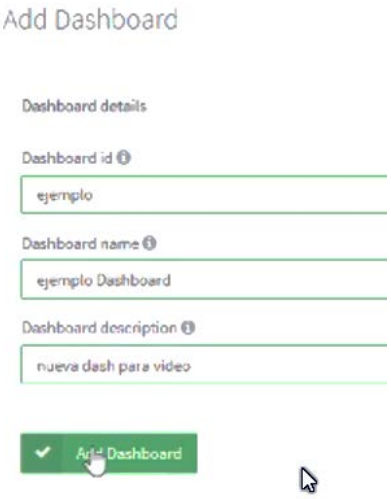


Ilustración 148 Campos para añadir un Dashboard
Fuente: Captura de pantalla

Con esto el dashboard está creado. Si ahora entramos en el dashboard que hemos creado, vemos que está vacío y nos sale el mensaje de la ilustración 149. Para añadir los widgets hay que hacer clic en el swich superior derecho que habilita la edición del dashboard.

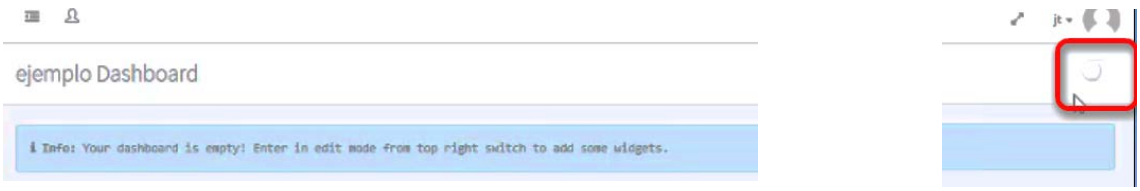


Ilustración 149 Habilitar un Dashboard
Fuente: Captura de pantalla

Una vez habilitada la edición, ya podemos pinchar en el botón que aparece “Add Widget” para añadir un nuevo widget (ver ilustración 150) o redimensionar y reordenar los widgets existentes según nuestras necesidades.

Ilustración 150 Añadir un widget
Fuente: Captura de pantalla

Nos aparece la ventana mostrada en la ilustración 151 donde podemos escoger entre los diferentes tipos de widgets. Como podemos observar hay dos categorías: de visualización y de control.

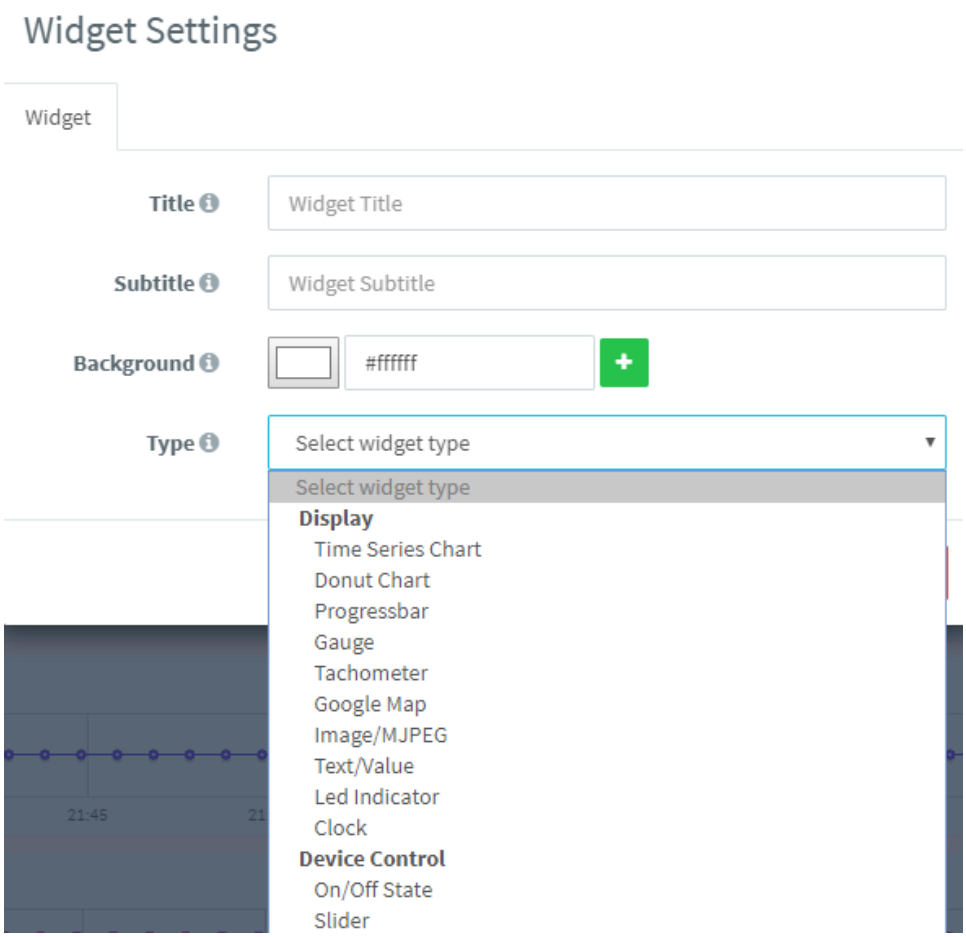


Ilustración 151 Tipos de widget
Fuente: Captura de pantalla

8.5.6.1. Añadir un gráfico al Dashboard

Continuando desde donde lo dejamos en el punto anterior, seleccionamos "Time series Chart" y nos aparece la ventana que vemos en la ilustración 152, Widget Settings, que dispone de tres pestañas. Lo primero que tenemos que hacer es elegir la fuente de los datos, que puede ser directamente desde un dispositivo o desde un bucket.

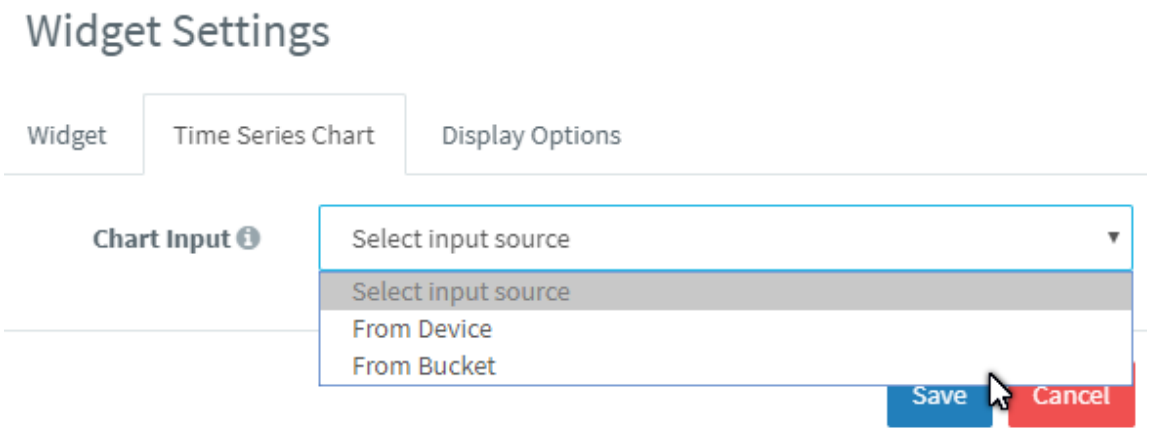


Ilustración 152 Seleccionar fuente de datos para un widget
Fuente: Captura de pantalla

- Si seleccionamos From Device, tendremos que seleccionar el dispositivo, el recurso y los campos que queremos representar, como vemos en la ilustración 153.

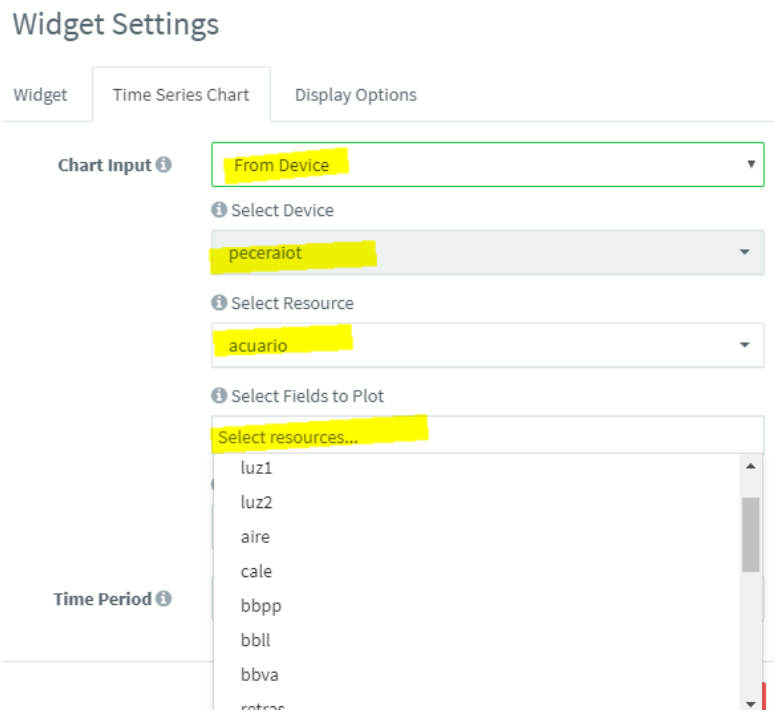


Ilustración 153 Seleccionar fuente de datos From Device para un widget
Fuente: Captura de pantalla

- Si seleccionamos From Bucket, tenemos que seleccionar el bucket y los campos que queremos representar, como vemos en la ilustración 154.

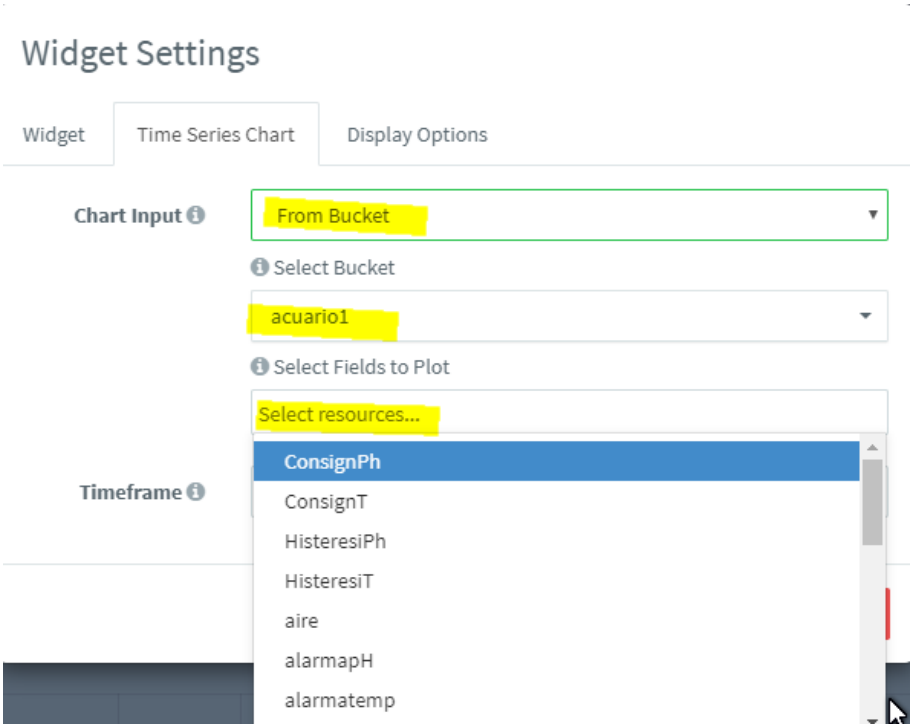


Ilustración 154 Seleccionar fuente de datos From Bucket para un widget
Fuente: Captura de pantalla

El siguiente paso es seleccionar el intervalo de tiempo con el que se van a muestrear los datos, y aparecen distintas opciones en función de la opción anteriormente elegida, como mostramos en la ilustración 155.

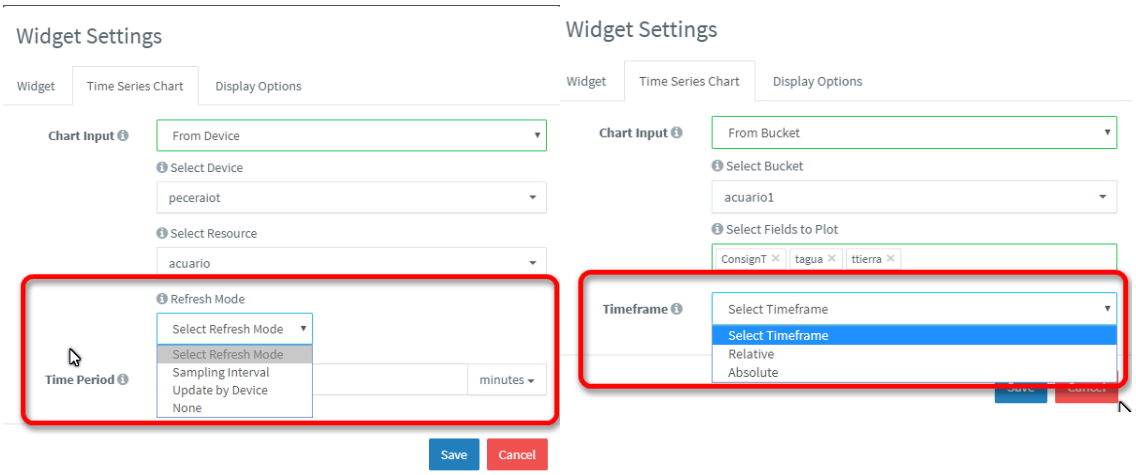


Ilustración 155 Seleccionar intervalo de tiempo para muestrear los datos
Fuente: Captura de pantalla

A continuación, seleccionamos el Time Period, que es el intervalo de tiempo que va a mostrar la gráfica. Ver ilustración 156.

Ilustración 156 Seleccionar intervalo de tiempo para mostrar en la gráfica
Fuente: Captura de pantalla

En la pestaña Display Options, podemos configurar distintas opciones de representación. Ver ilustración 157. La opción Multiple Axes adapta la escala de cada una de las líneas gráficas a sus valores. También permite escoger el color para cada una de las líneas gráficas.

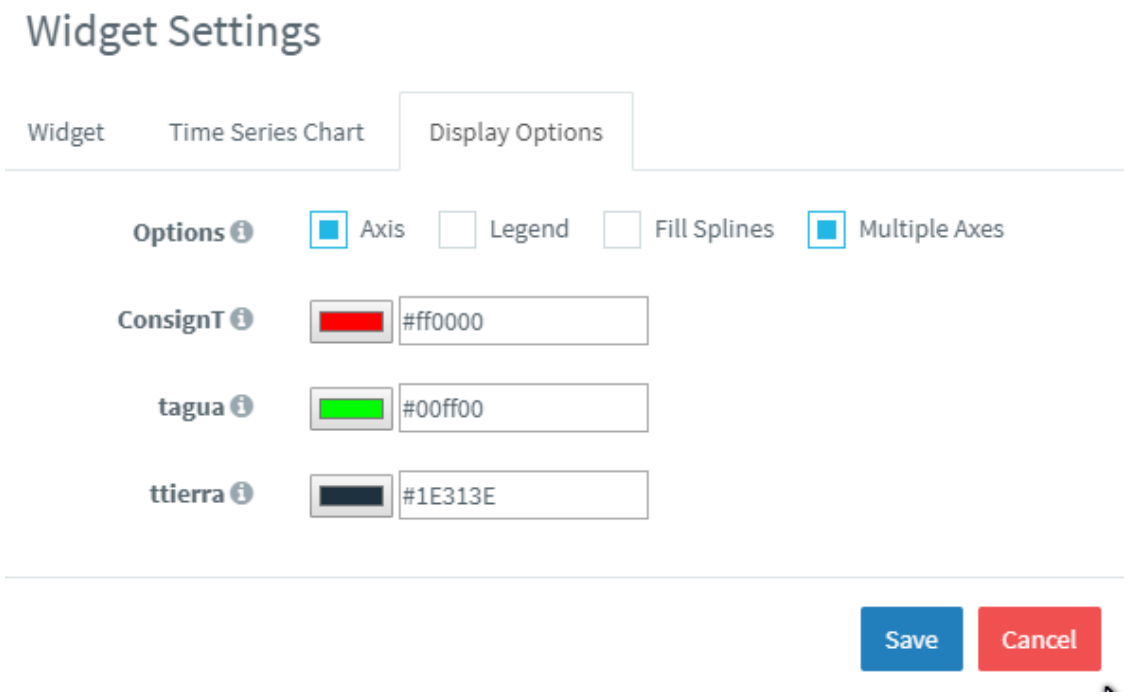


Ilustración 157 Configurar ajustes de representación de la gráfica
Fuente: Captura de pantalla

En la ilustración 158 podemos ver uno de los dashboard creados para este proyecto, el dashpecera2.

Ilustración 158 Dashpecera2. Datos pecera
Fuente: Captura de pantalla

8.5.7. Los Endpoints

Un servidor es un ordenador conectado permanentemente a internet que pone recursos a disposición de los clientes que lo solicitan. Existen diferentes tipos de servidores en función del tipo de comunicación entre el servidor y el cliente. Hay servidores que se diseñan para el intercambio de información con el cliente de forma automática, y la interfaz para solicitudes de otras máquinas recibe el nombre de endpoint. El protocolo REST es el más utilizado para este tipo de solicitudes y en la trama del mensaje se indica la dirección del servidor y el endpoint al que se desea acceder, seguido de atributos.

Anteriormente había que implementar todo el código de petición de un servicio a un servidor en el código de programa que estábamos desarrollando. Ahora, podemos utilizar el servidor de thinger.io como un intermediario. Con ello, nuestro dispositivo solo va a necesitar hacer una petición al servidor de thinger.io para que sea este el que solicite al servidor externo el servicio que hayamos demandado. Para que nuestro dispositivo haga la petición al servidor de thinger.io, solo se necesita definir un endpoint en el gestor de endpoints que tiene la plataforma thinger.io y ejecutar la instrucción de llamada al endpoint desde nuestro programa. Con esto lo que hacemos es disminuir las necesidades de recursos de nuestro equipo porque es el servidor de thinger.io el que se encarga de implementar todo el código para la creación y envío de la api rest al servidor externo. Ver ilustración 159.

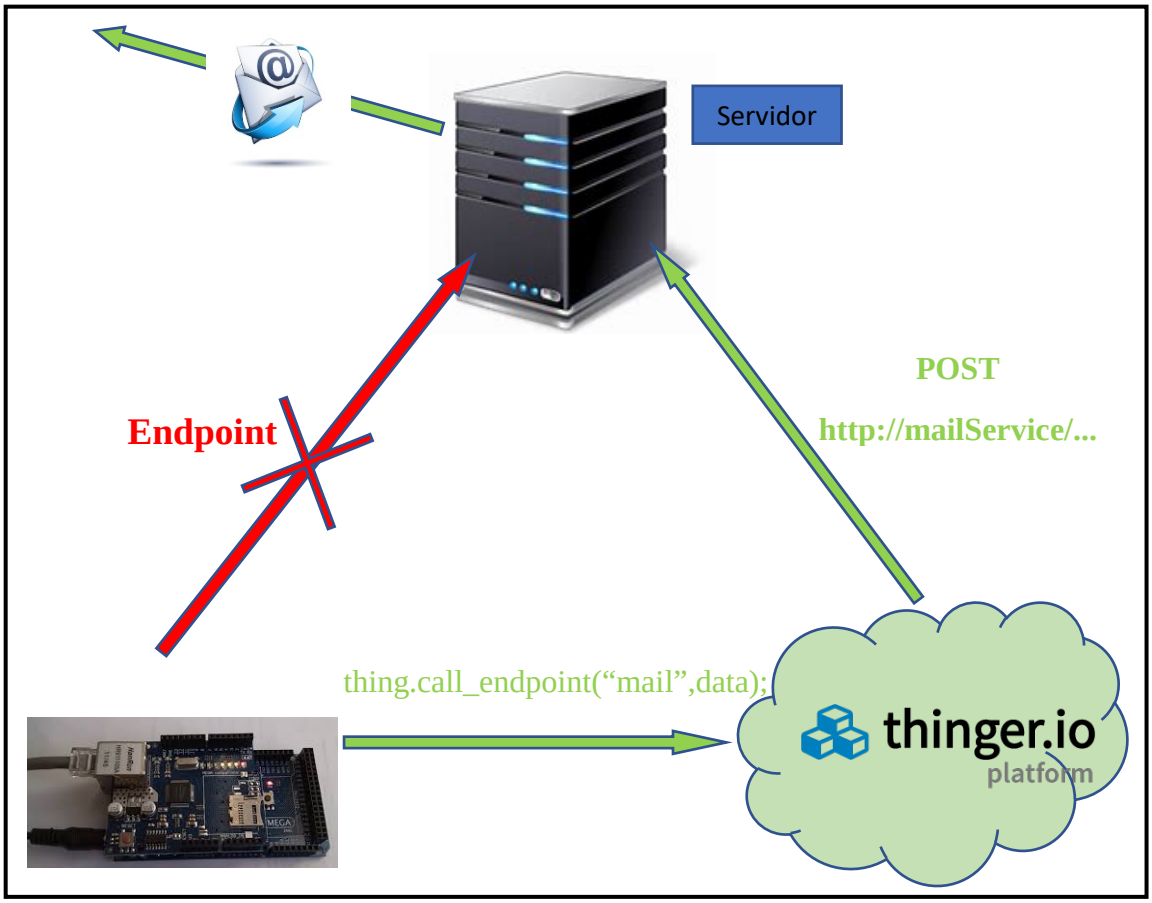


Ilustración 159 Solicitud de Endpoint para el envío de un mail
Fuente: Elaboración propia

Para crear un Endpoint en Thinger.io lo primero es hacer clic en la pestaña Endpoints y luego en Add Endpoint como se muestra en la ilustración 160.

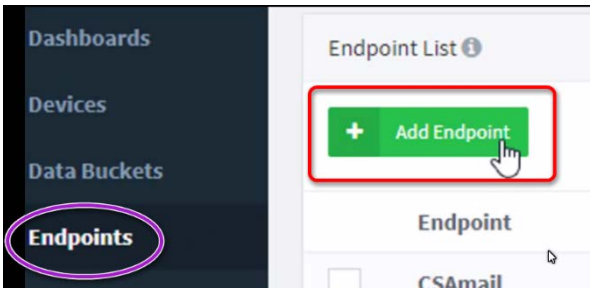


Ilustración 160 Añadir un Endpoint
Fuente: Captura de pantalla

A continuación, completamos el formulario que aparece en pantalla. Un campo importante es el Endpoint identifier, que va a ser el nombre con el que vamos a invocar al endpoint desde nuestro programa de Arduino. Añadimos una descripción y seleccionamos el tipo de endpoint entre los diferentes preestablecidos. Ver ilustración 161.

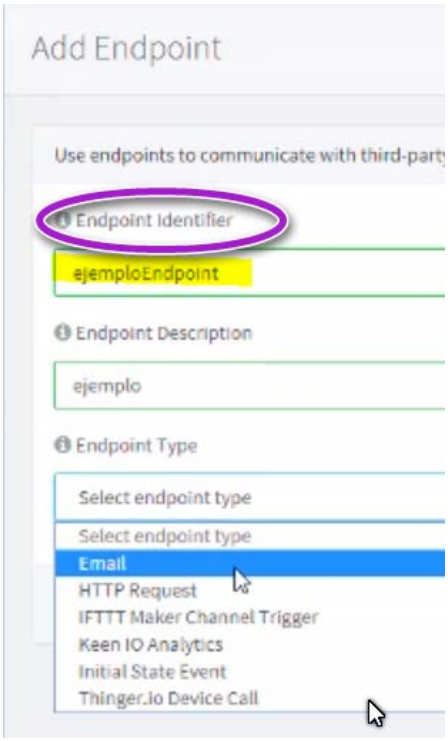


Ilustración 161 Formulario Endpoint
Fuente: Captura de pantalla

Si el servicio que necesitamos no aparece en el listado, seleccionando la opción HTTP Request podemos crear lo solicitud de cualquier otro tipo de servicio.

Como ejemplo de uso de un endpoint, vamos a elegir la opción Email. Cuando llamemos a este endpoint se va a enviar un correo electrónico a la dirección de correo indicada en el apartado Email Address. Un aspecto a destacar, es que en el correo electrónico podemos incluir los datos leídos por los sensores, para ello solo hace falta incluir el nombre del dato entre llaves dobles {{dato}}.

En la ilustración 162 vemos el endpoint creado en este proyecto para el envío de un correo a mi dirección de correo, cuando se produce una alarma de temperatura, pH o nivel.

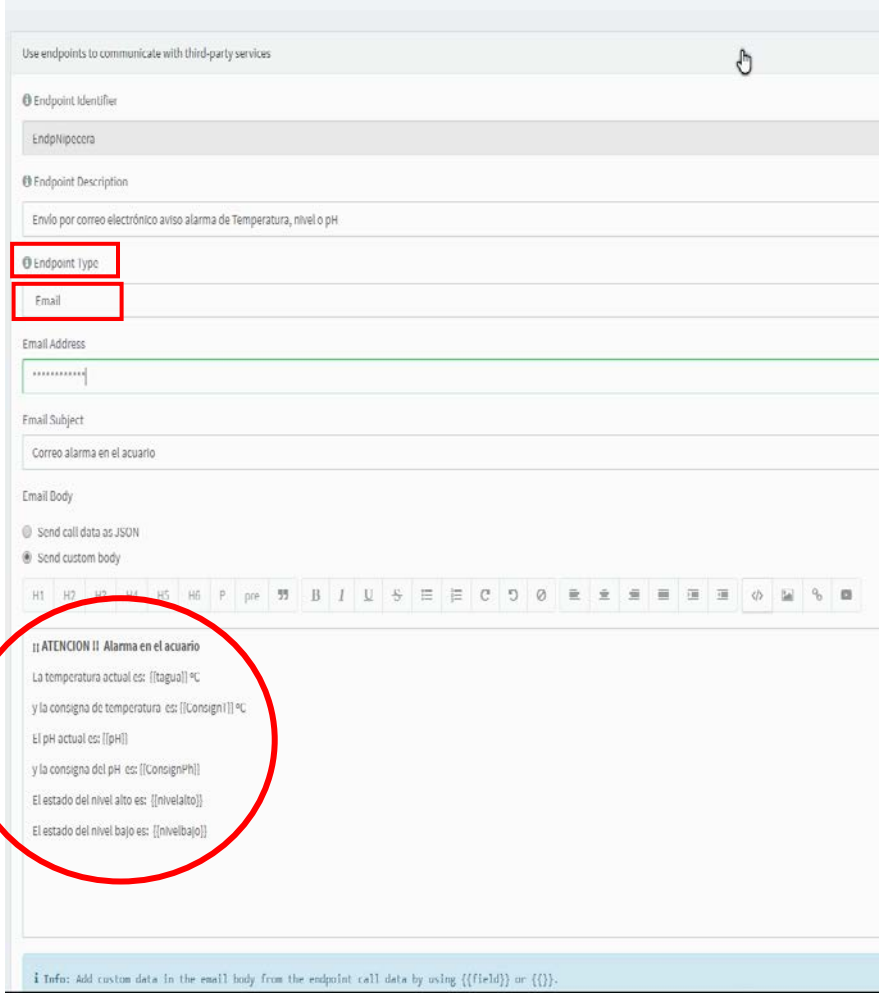


Ilustración 162 EndpNipecera. MSM de alarma
Fuente: Captura de pantalla

Solo falta modificar el código del programa Arduino para que llame al endpoint, como se muestra en la ilustración 163. Para ello es necesario que los identificadores de las variables tengan el mismo nombre que el que hemos metido entre corchetes dobles en el cuerpo del mensaje de correo. Para añadir la llamada al endpoint hay que comprobar que el nombre, entre comillas, se corresponde con el indicado en el Endpoint Identifier. A continuación del nombre indicamos el recurso de salida que queremos enviar que contiene los datos que queremos inclir en el correo. Un detalle importante es que hay que poner algún algoritmo de control que evite que la llamada al endpoind se ejecute constantemente.

La instrucción a utilizar es:

thing.call_endpoint(“Endpoint Identifier”, recurso);

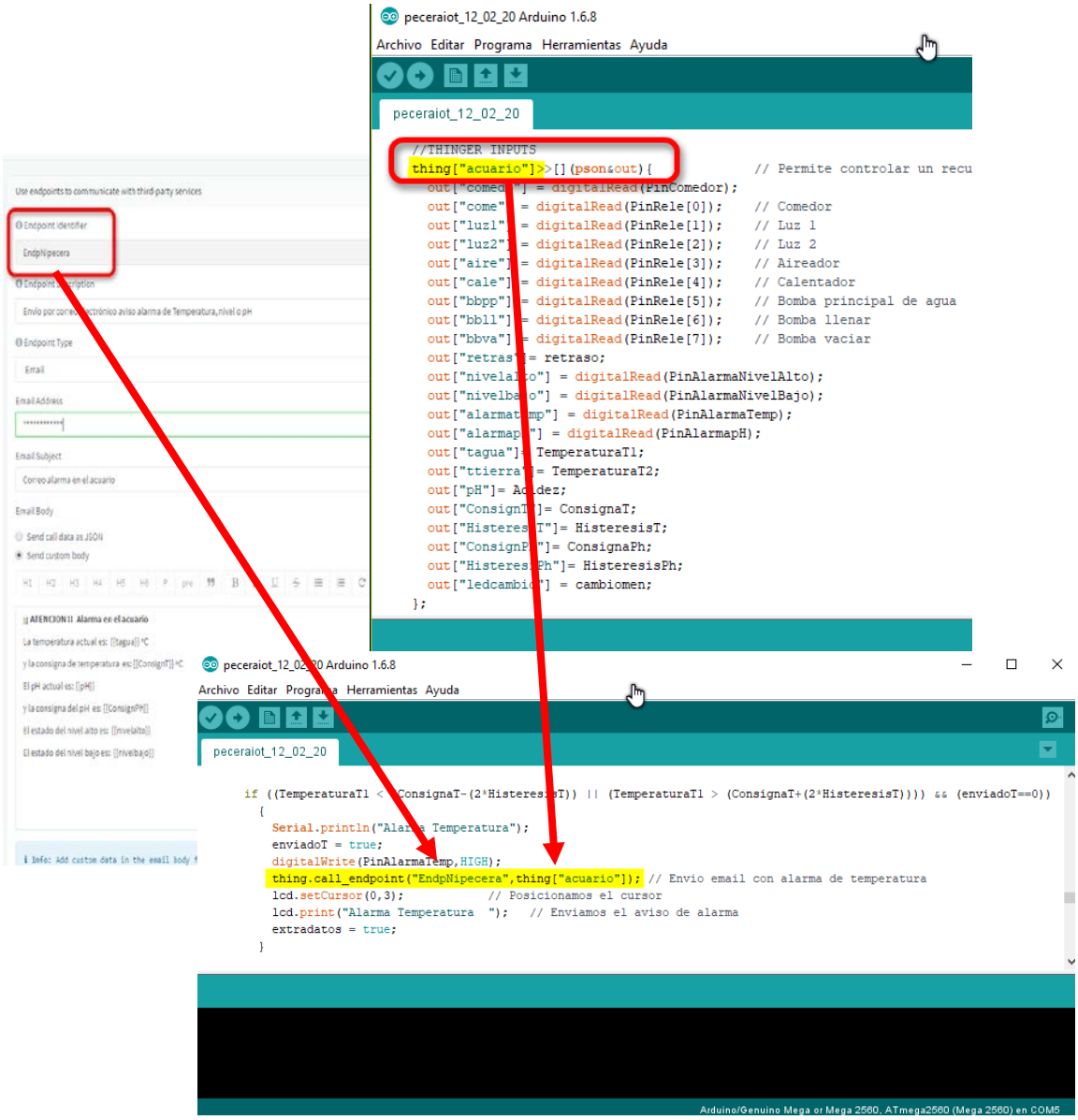


Ilustración 163 Llamada desde Arduino a un Endpoint.
Fuente: Captura de pantalla

8.5.8. Los Device Tokens.

8.5.8.1. Interactuar con los recursos a través de un device token

Con el uso de los devices tokens vamos a poder interactuar, sin usar la interfaz de thinger.io, con los recursos de nuestros dispositivos.

Para añadir un device token tenemos que, desde la pestaña de administración del dispositivo, hacer clic en el botón verde Add, que está en la parte inferior de la pantalla, como se muestra en la ilustración 164.

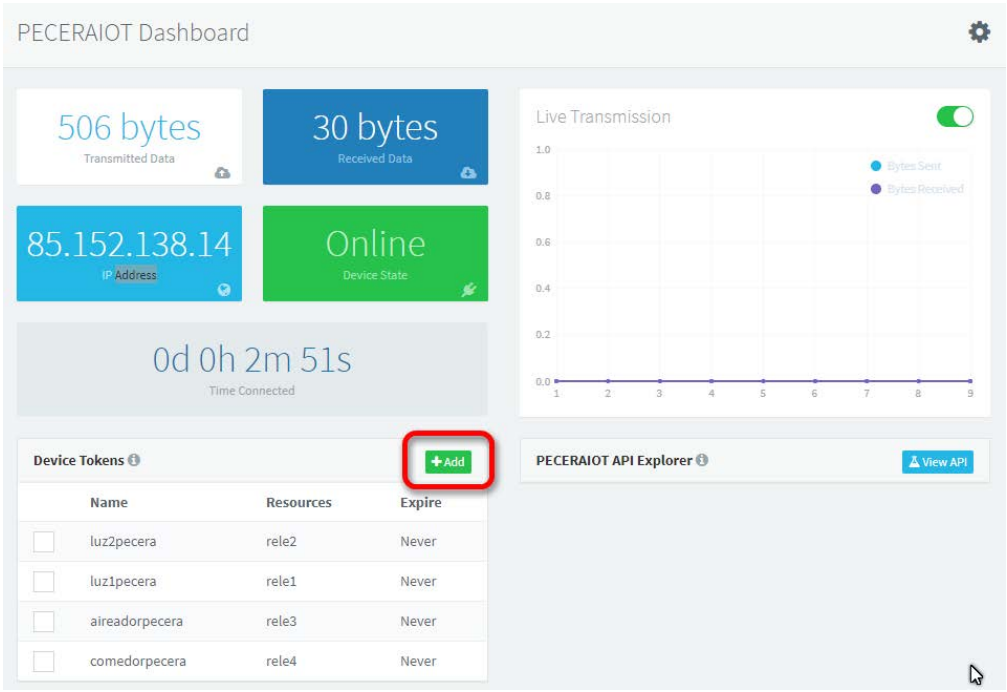


Ilustración 164 Añadir un Device Token.
Fuente: Captura de pantalla

Nos aparece una nueva ventana desde donde podemos asignar un nombre, elegir los recursos a los que queremos dar acceso y la duración del permiso. Ver Ilustración 165.

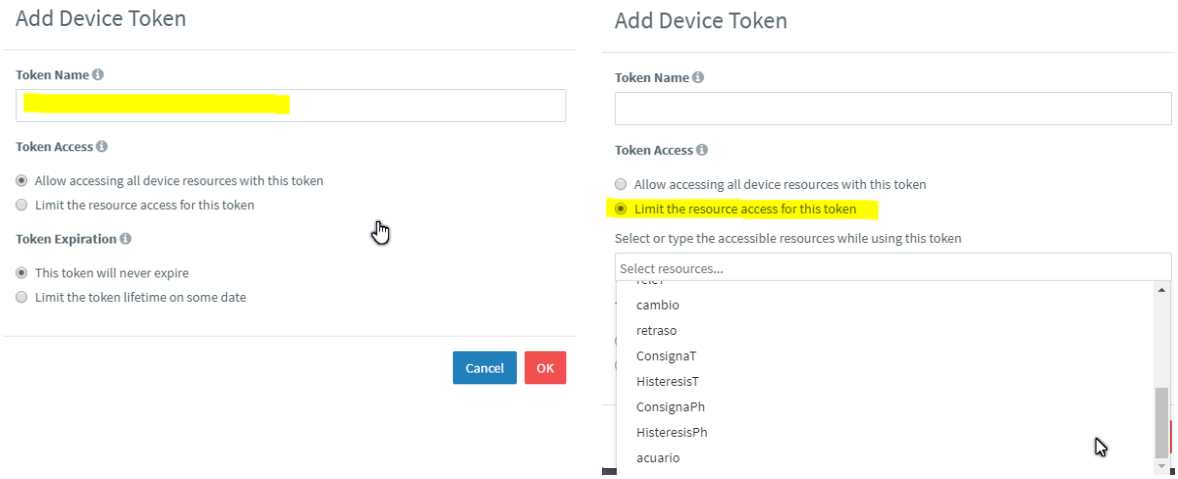


Ilustración 165 Configurar un Device Token.
Fuente: Captura de pantalla

Una vez cubiertos todos los campos hacemos clic en el botón “OK”, y vemos como en la parte inferior de la ventana aparece un nuevo apartado llamado Device Token que contiene una larga cadena de caracteres, como podemos apreciar en la ilustración 166, que son los que contienen toda la información necesaria para acceder al recurso.

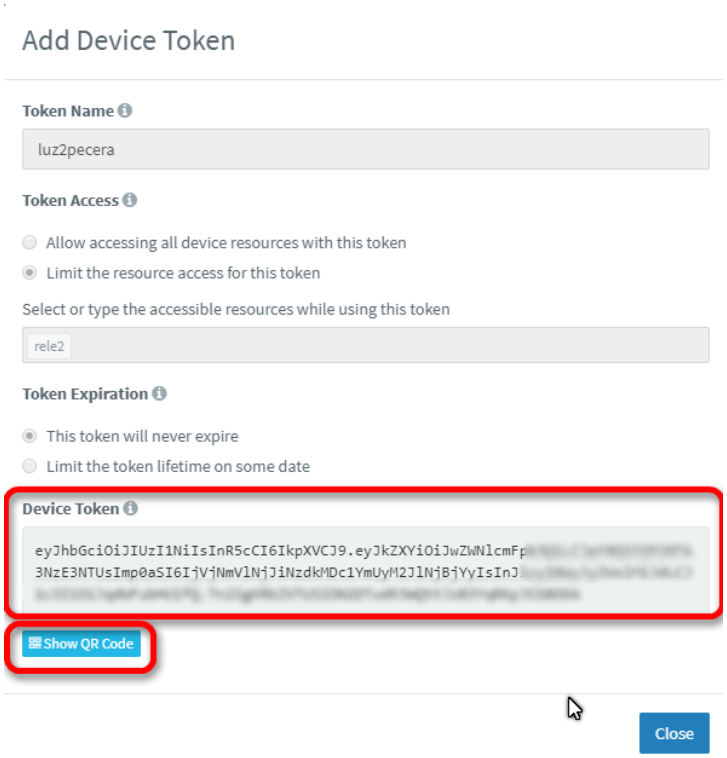


Ilustración 166 Device Token.
Fuente: Captura de pantalla

Si hacemos clic en el botón “Show QR Code” nos aparece el device token en formato QR, como podemos ver en la ilustración 167.

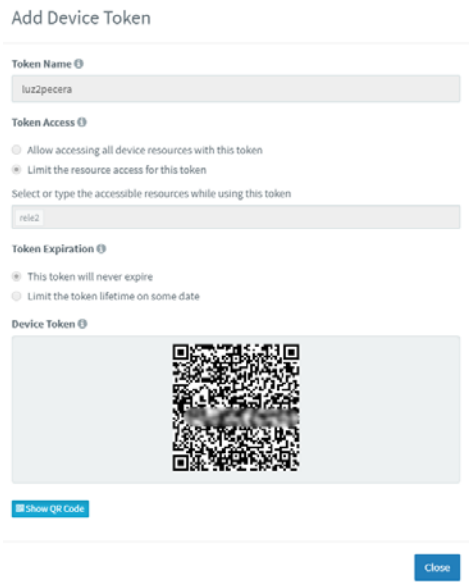


Ilustración 167 Código QR de un Device Token.
Fuente: Captura de pantalla

8.5.8.2. Aplicación móvil de Thinger.io

Esta aplicación nos va a permitir administrar diferentes recursos de nuestro dispositivo desde un smartphone. La aplicación se puede descargar desde Google Play y App Store.

Ilustración 168 Logos de Google Play y App Store.
Fuente: https://play.google.com/intl/en_us/badges/

Una vez instalada y en funcionamiento la aplicación, lo primero que necesitamos es escanear el toquen QR del dispositivo a controlar. Para ello pulsamos en el icono de la parte superior derecha de la pantalla principal y apuntamos con la cámara en el QR. Una vez escaneado, nos aparece el nombre del dispositivo en la pantalla principal. Ver ilustración 169.

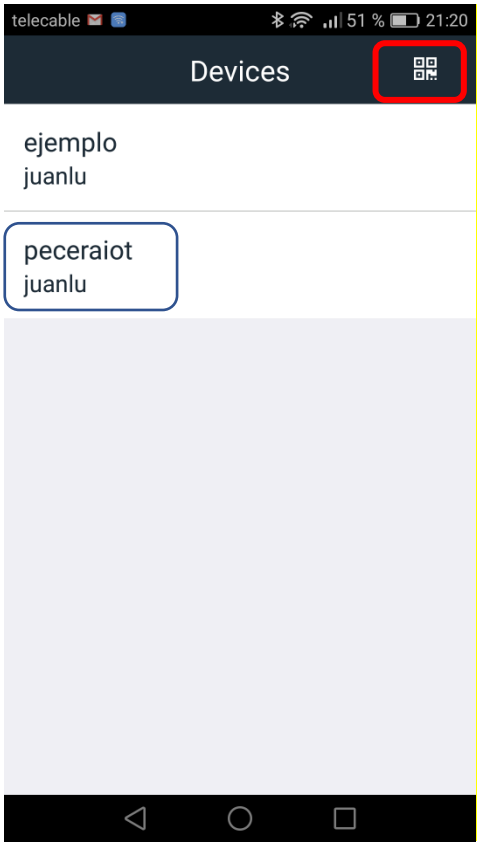


Ilustración 169 Página principal de la aplicación móvil.
Fuente: Captura de pantalla del móvil

Pulsamos en el nombre del dispositivo, “peceraiot”, y ya nos aparece una pantalla con toda la información disponible. Los recursos del dispositivo se dividen en diferentes secciones. En la ilustración 170 podemos ver distintas capturas de pantalla.

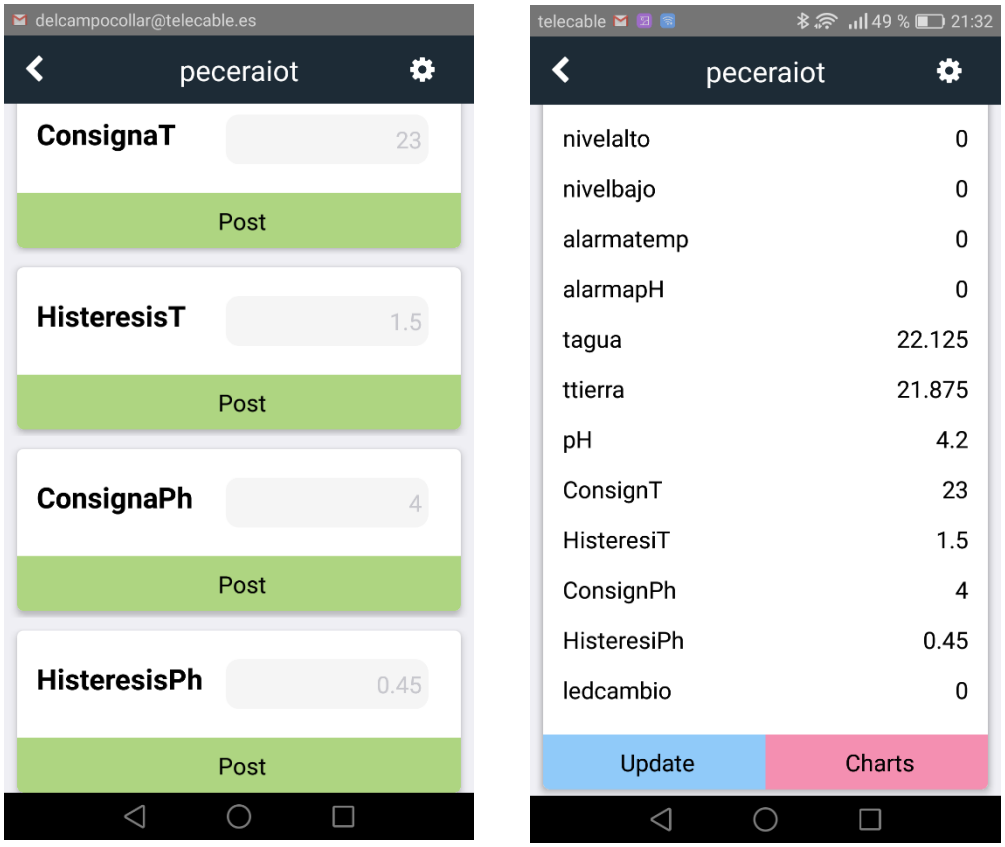


Ilustración 170 Distinta información en la pantalla del móvil.
Fuente: Capturas de pantalla del móvil

Como se ve, podemos editar y publicar los recursos de entrada y visualizar y actualizar los recursos de salida. Para actualizar un recurso solo es necesario pulsar el botón azul “Update” o arrastrar la pantalla hacia abajo para actualizarlos todos. Para publicar un recurso basta con cumplimentar el campo de entrada y pulsar el botón verde “Post”.

La aplicación móvil también nos permite monitorizar los recursos en tiempo real. Para ello nos permite escoger entre tres tipos de gráficos diferentes:

- Gráfico de líneas
- Gráfico circular
- Gráfico de barras

Para ver un gráfico de un recurso solo hay que hacer clic en su botón rosa “Chart”. Luego seleccionamos los elementos que queremos visualizar. Al presionar sobre cada elemento podemos seleccionarlo/deseleccionarlo. Ver ilustración 171.

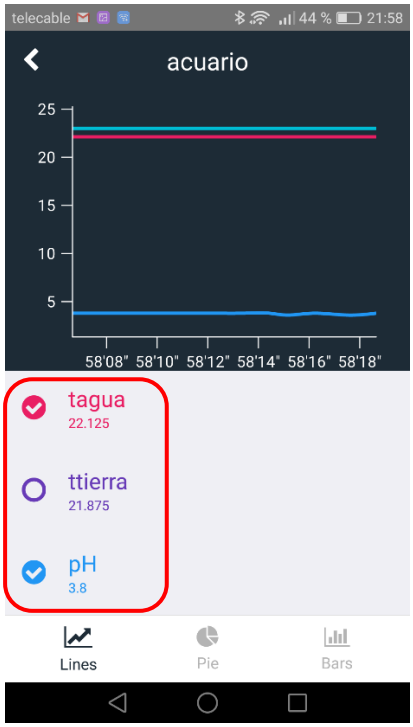


Ilustración 171 Pantalla configuración Gráfico de la aplicación móvil.
Fuente: Captura de pantalla del móvil

En la ilustración 172 vemos un gráfico de líneas y otro de barras con la consigna de temperatura (azul claro) y el valor de la temperatura (rojo) y del pH (azul oscuro).

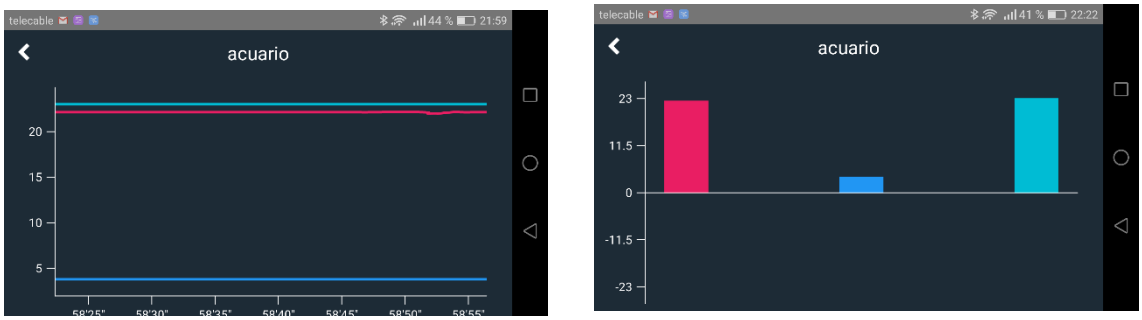


Ilustración 172 Gráficos de la aplicación móvil.
Fuente: Capturas de pantalla del móvil

Para editar las configuraciones del dispositivo, ilustración 173, basta con presionar la rueda dentada que aparece en la parte superior derecha de la pantalla.

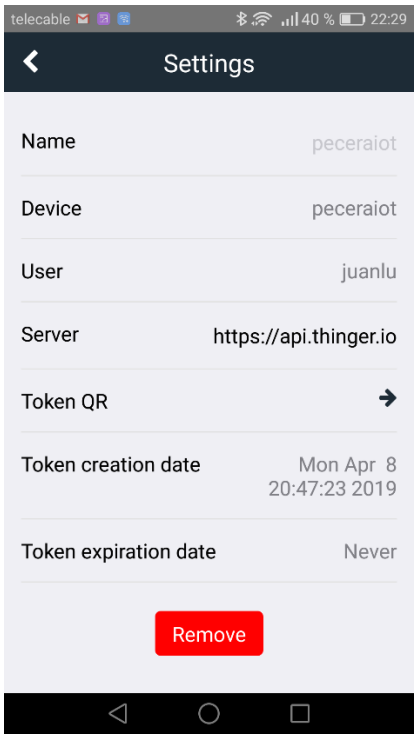


Ilustración 173 Configuración del dispositivo en la aplicación móvil.
Fuente: Captura de pantalla del móvil

8.5.8.3. Crear un endpoint para Interactuar con nuestros recursos

Con esta función vamos a poder pasar información a un recurso de un dispositivo. Para ello es necesario conocer la dirección del endpoint del recurso. Esta la podemos obtener accediendo a la api del dispositivo, ver ilustración 174, y examinando la petición del recurso.

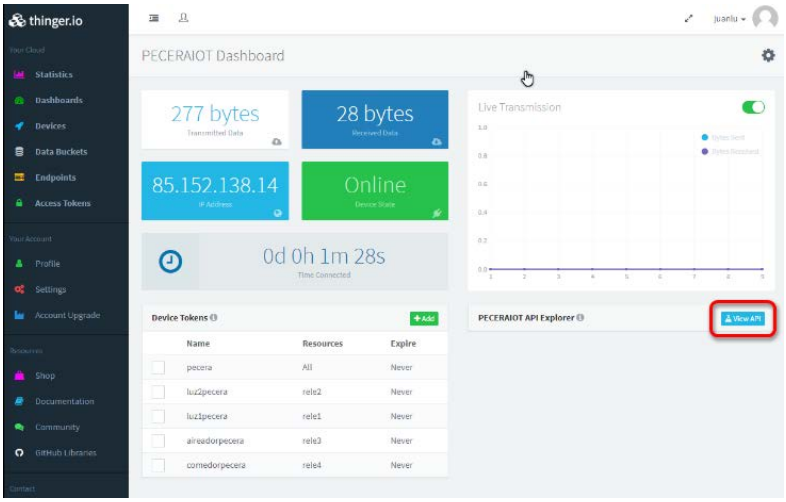


Ilustración 174 Acceso a la api del dispositivo.
Fuente: Captura de pantalla

Seleccionamos el recurso al que queremos acceder y hacemos clic en el botón “Show query” que vemos en la ilustración 175.

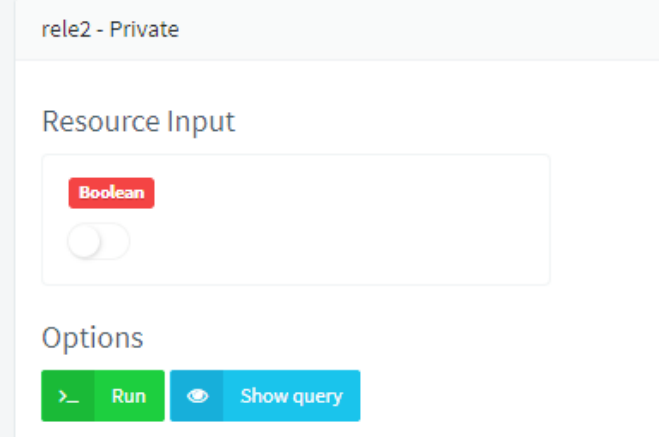


Ilustración 175 Acceso a la información de un recurso.
Fuente: Captura de pantalla

Y nos aparece la información del recurso, como se muestra en la ilustración 176.

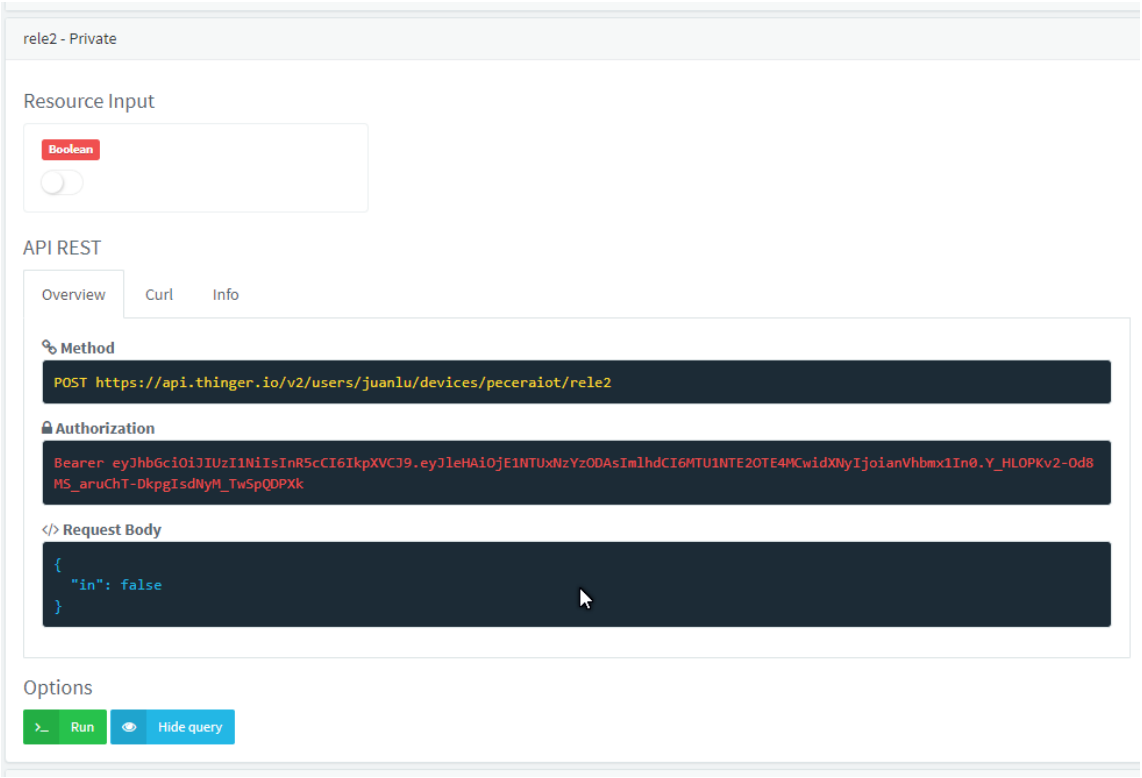


Ilustración 176 Información de un recurso.
Fuente: Captura de pantalla

En el primer apartado se muestra la dirección a la que accede la consola de thinger.io cada vez que se modifica el estado del dispositivo. Lo que hace la consola es enviar un pequeño fichero JSON a la dirección del dispositivo con una autorización que caduca al cabo de unos minutos.

Esta forma de trabajo la podemos utilizar en nuestro programa o desde un programa escrito en C. Para ver el código en C accedemos a la pestaña “Curl”. Vemos este código en la ilustración 177.

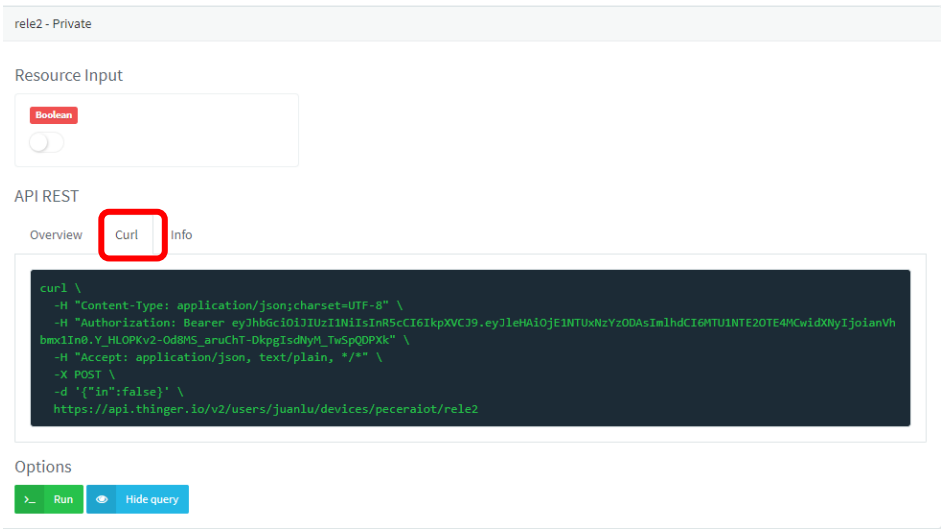


Ilustración 177 Código en C para el acceso a un recurso.
Fuente: Captura de pantalla

Para poder utilizar esta opción en nuestro programa es necesario crear un endpoint. En el menú Endpoint seleccionamos “Add Endpoint” y en el campo Endpoint Type escogemos “Thinger.io Devicel Call”. Ver ilustración 178.

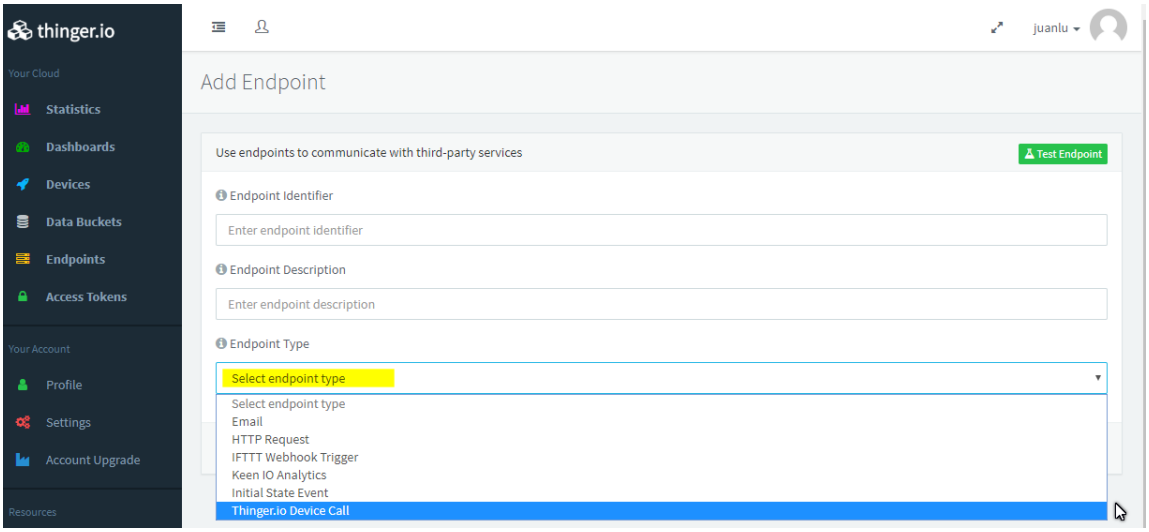


Ilustración 178 Crear endpoin tipo thinger.io Device call.
Fuente: Captura de pantalla

Cumplimentamos los campos con los datos de la información del recurso como se muestra en la ilustración 179. Una vez cumplimentada la información hacemos clic en “Add Endpoint”.

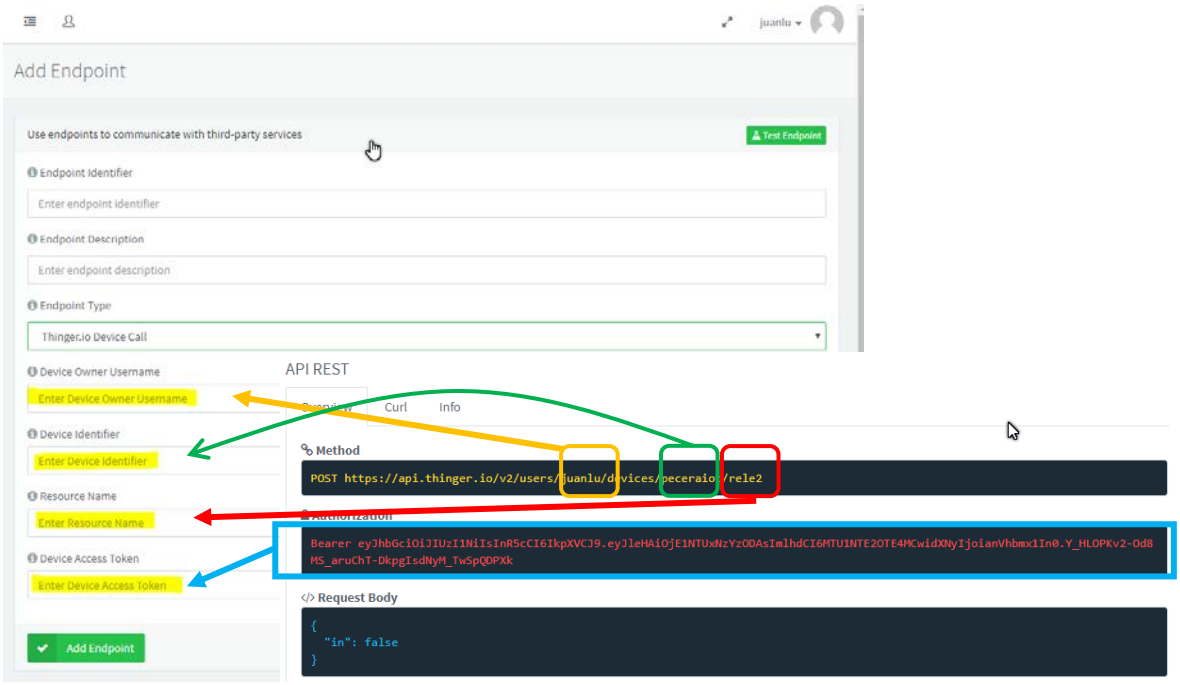


Ilustración 179 Cumplimentar endpoin tipo thinger.io Device call.
Fuente: Captura de pantalla

Una vez que hemos credo el endpoint solo hay que llamarlo como a cualquier otro endpoint y pasarle la información que deseemos, por ejemplo, un “1” o un “0” para que se active o desactive el relé.

thing.call_endpoint(“Endpoint Identifier”, dato);

8.5.9. Los Access Tokens.

Un access token es una función que nos permite crear autorizaciones de acceso a los recursos de nuestra cuenta. Nos van a permitir interactuar con los recursos de la plataforma thinger.io desde un sistema externo.

Para crear un nuevo Access Token hay que hacer clic en el botón “Add Token” del menú Access Token, como vemos en la ilustración 180.

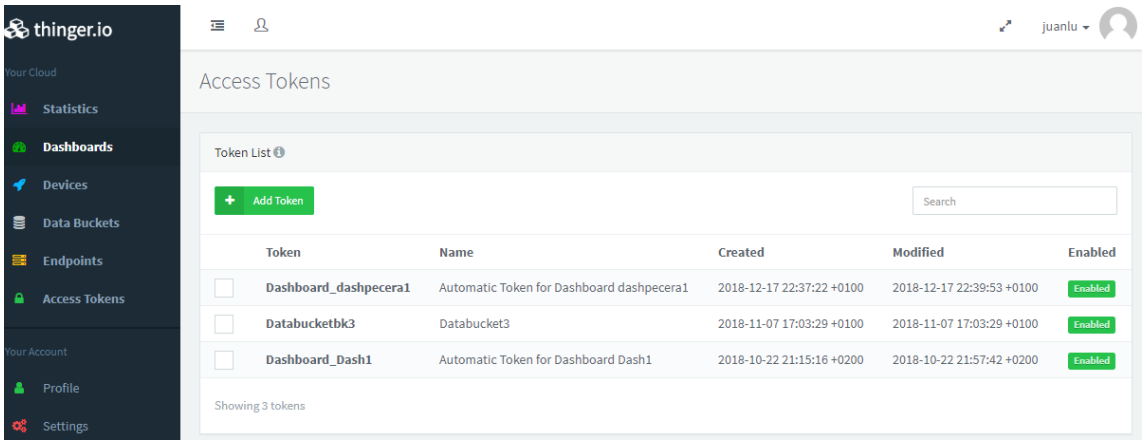


Ilustración 180 Añadir un nuevo Access Token
Fuente: Captura de pantalla

Nos aparece el formulario que se muestra en la ilustración 181 y que iremos cumplimentando. En el apartado Token Permissions añadimos uno a uno los recursos de nuestra cuenta a los que vamos a permitir el acceso a través de este token. Para añadir los recursos hacemos clic en el botón “+Add” y se abre un cuadro de dialogo para escoger el tipo de permiso.

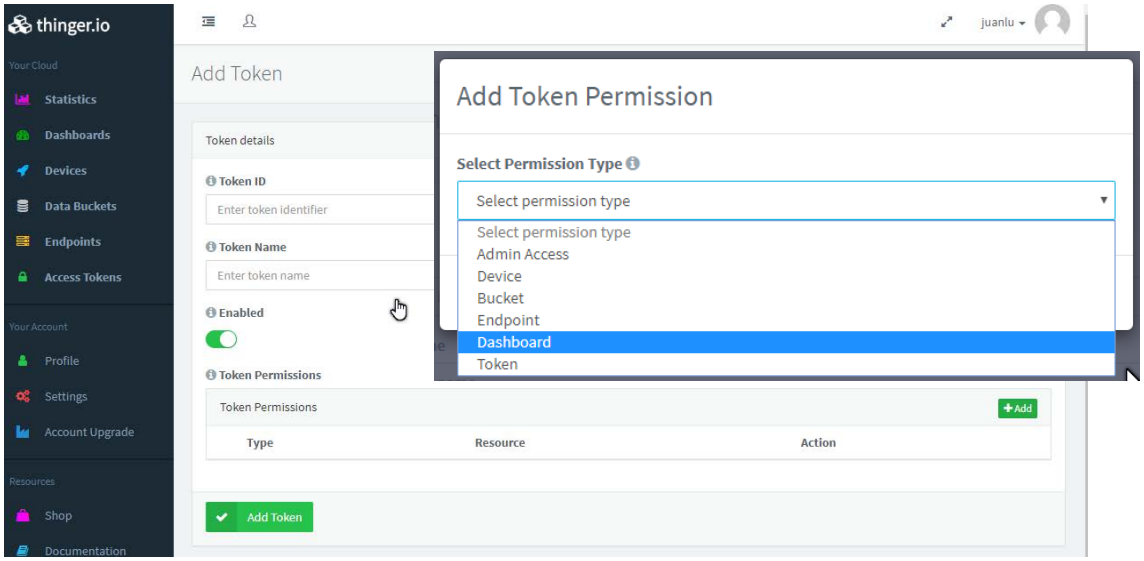


Ilustración 181 Seleccionar tipo de permiso para un nuevo Access Token
Fuente: Captura de pantalla

La opción “Admin Acces” da acceso a todas las funciones de la cuenta. Como ejemplo de uso vamos a seleccionar la opción Dashboard. Vemos que se abre una pestaña nueva como la de la ilustración 182 donde podemos escoger todos los Dashboards o seleccionar solo uno.

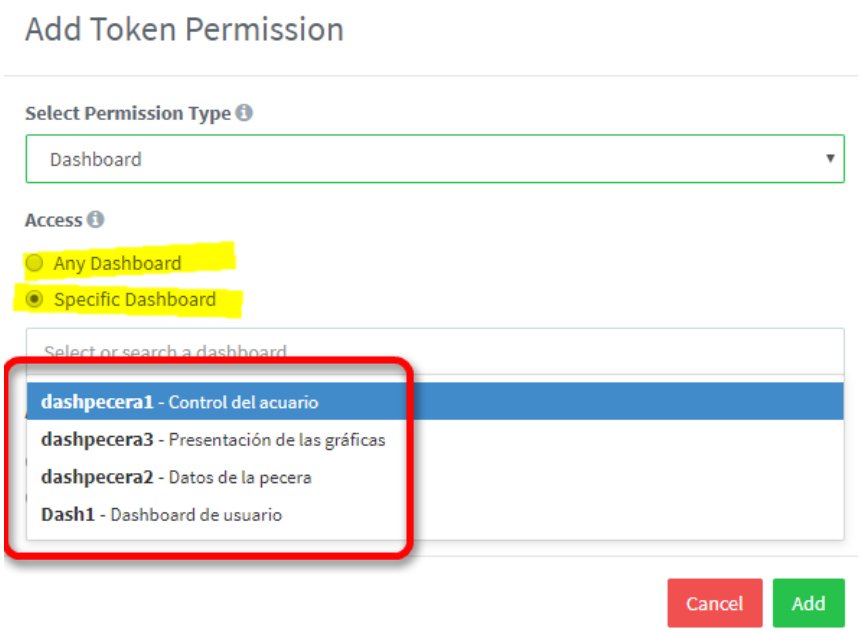


Ilustración 182 Seleccionar Dashboard para un nuevo Access Token
Fuente: Captura de pantalla

A continuación, escogemos las operaciones, ver ilustración 183, que vamos a permitir poder hacer con el token creado.

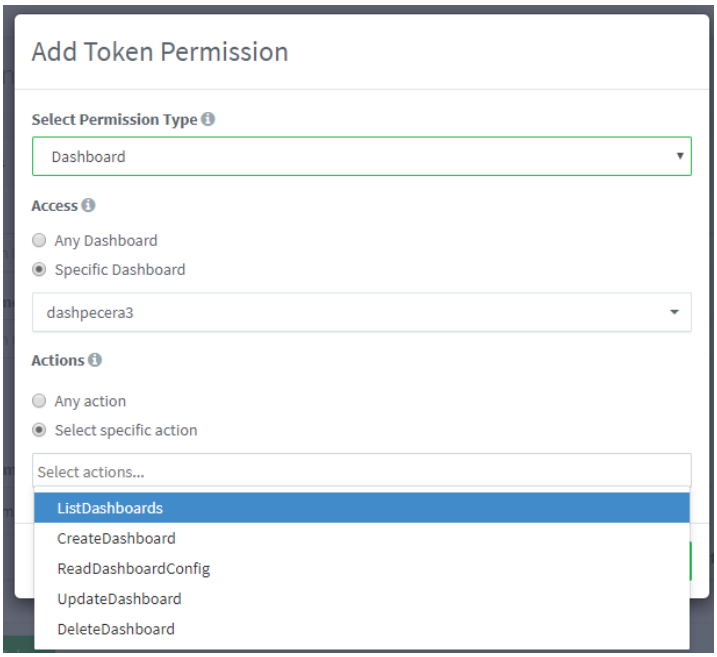


Ilustración 183 Seleccionar permisos para un nuevo Access Token
Fuente: Captura de pantalla

Para finalizar hacemos clic en “Add Token” y vemos que aparece un nuevo apartado llamado “Access Token” con una cadena de caracteres que contienen la autorización de acceso al recurso. Ver ilustración 184.

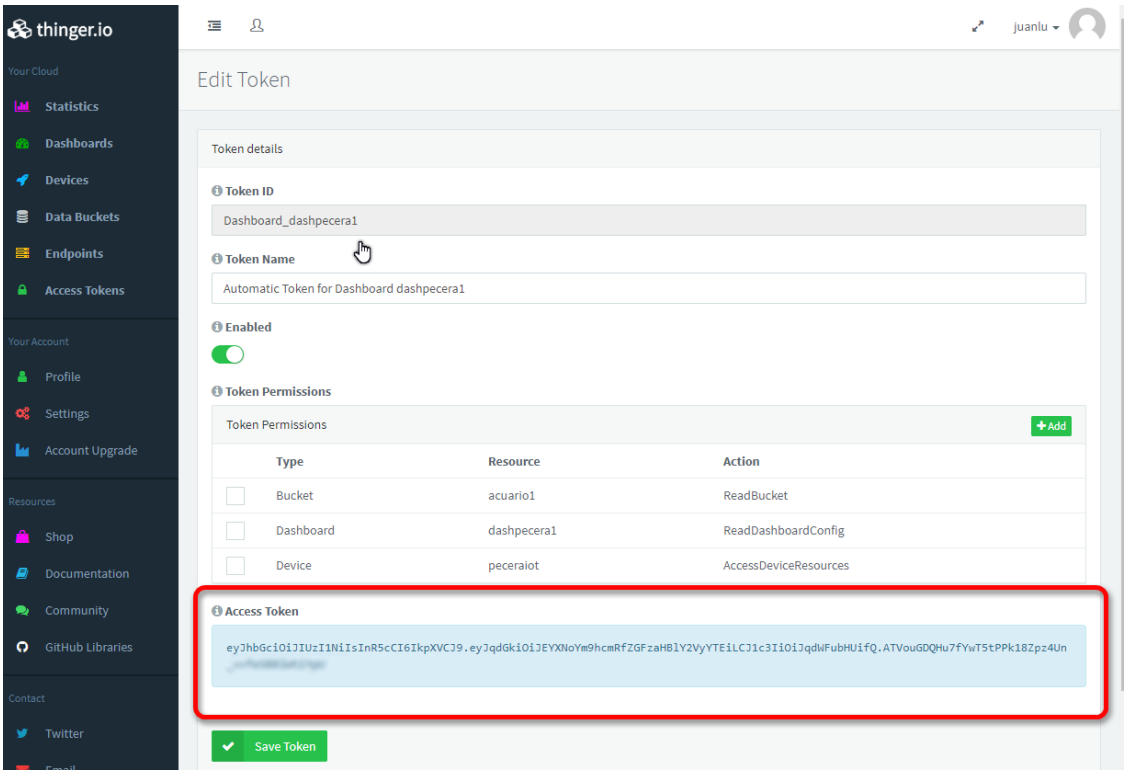


Ilustración 184 Autorización de acceso a un Token
Fuente: Captura de pantalla

8.5.9.1. Ejemplo de acceso a un dashboard creado

Abrimos el dashboard que hemos llamado dashpecera1 y copiamos la api rest que utiliza el navegador para solicitarle a thiger.io esta interfaz. Ver ilustración 185.

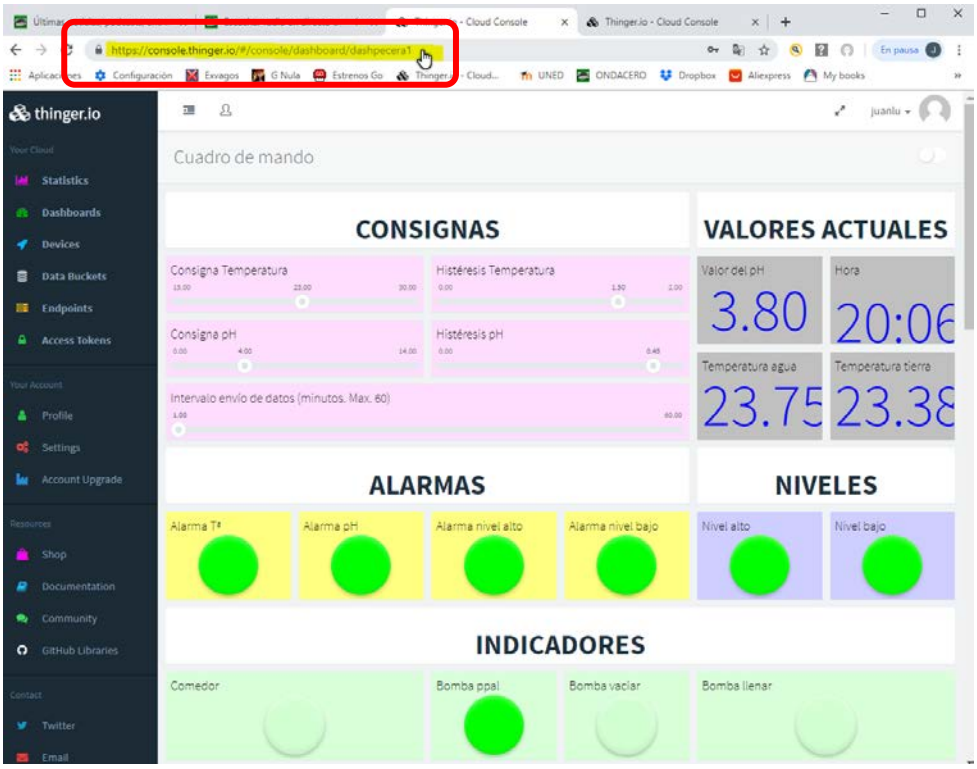


Ilustración 185 Dashpecera1
Fuente: Captura de pantalla

Ahora abrimos una nueva pestaña en el navegador y pegamos la api rest, y vemos que no nos deja acceder, nos pide el username y el password de acceso a la plataforma Thinger.io como vemos en la ilustración 186.

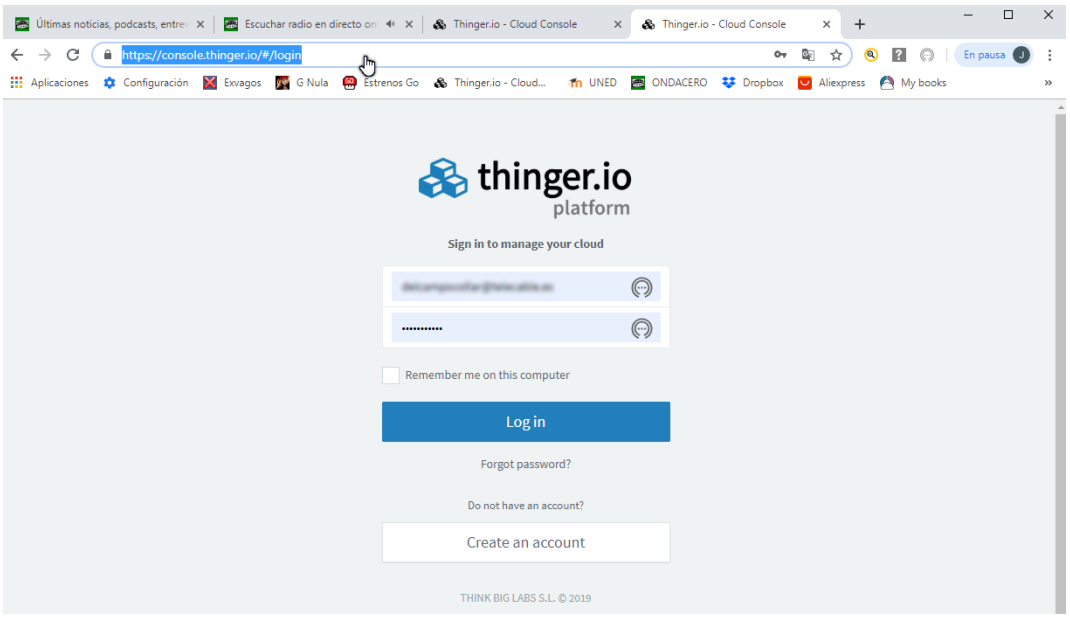


Ilustración 186 Solicitud de Username y Password para acceder al Dashpecera2
Fuente: Captura de pantalla

Para poder entrar hay que añadir a la api rest el texto " ? authorization = " más el Access token. En la ilustración 187 vemos como creamos la api rest con la autorización de acceso.

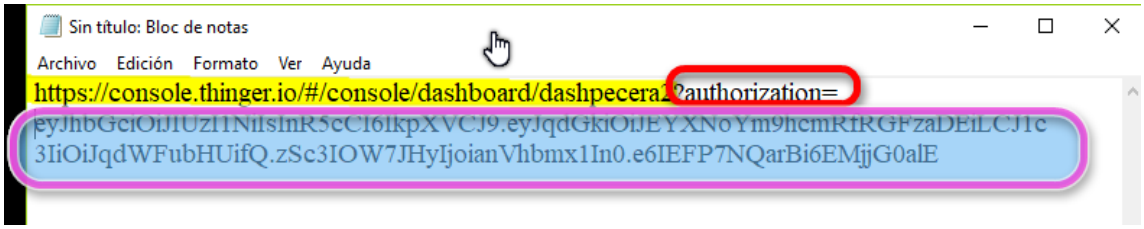


Ilustración 187 Api rest para acceder al Dashpecera2
Fuente: Captura de pantalla

Si ahora introducimos en la nueva pestaña del navegador este código vemos que tenemos acceso al dashboard, sin embargo, no se muestra ninguno de los datos. Esto es debido a que en el Access Token sólo hemos dado permiso de acceso al Dashboard. Pero el Dashboard toma los datos de un bucket al que no hemos dado acceso. Del mismo modo, si se intenta acceder a cualquier otro recurso de la cuenta no es posible por no tener la autorización correspondiente, mostrándonos el error que se muestra en la ilustración 188.

Ilustración 188 Mensaje de error de acceso
Fuente: Captura de pantalla

Para poder ver los datos solo hay que dar permiso de acceso al Bucket y para poder actuar con los dispositivos hay que dar acceso al Device. Estos accesos los podemos ver en la ilustración 189.

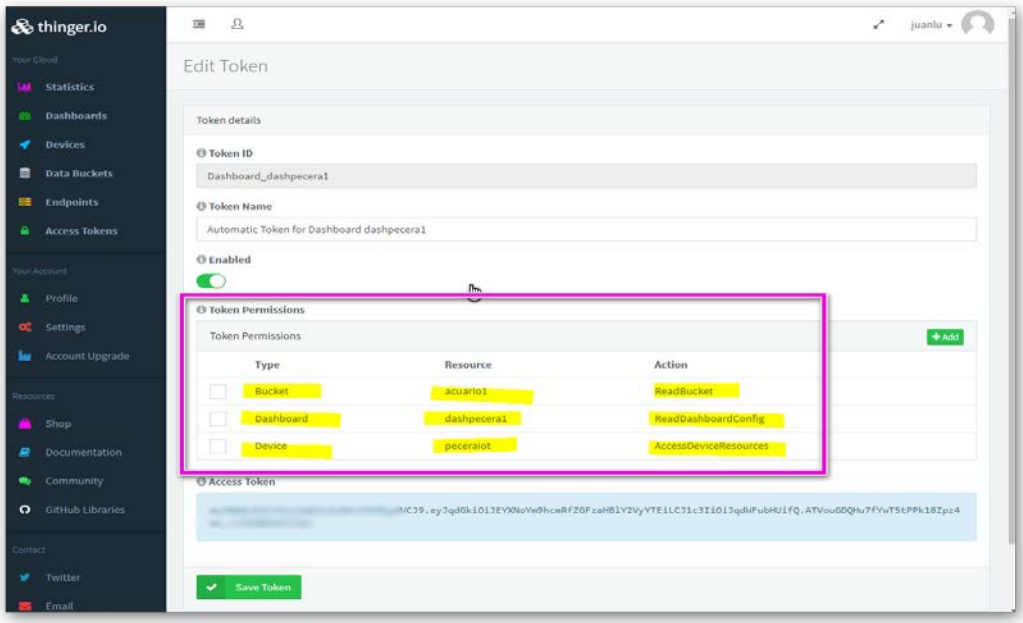


Ilustración 189 Todos los accesos para acceder al Dashpecera1
Fuente: Captura de pantalla

Si ahora introducimos en el navegador la api rest completa vemos que ya aparecen los datos y que podemos interactuar con los dispositivos. Ver ilustración 190.

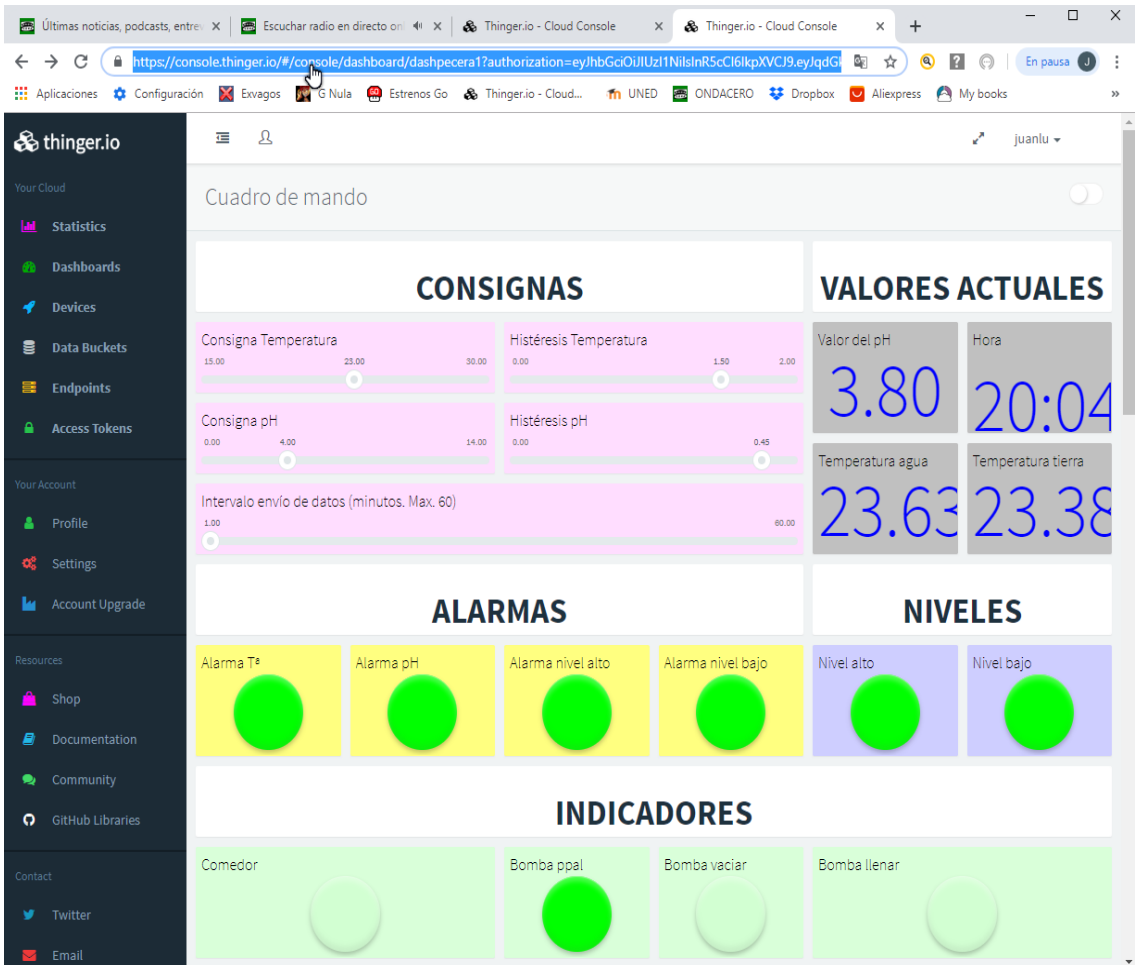


Ilustración 190 Dashpecera1
Fuente: Captura de pantalla

8.6. Anexo 6. Configuración plataforma IFTTT

IFFT es un servicio que permite crear conexiones entre diferentes plataformas de la web, y que vamos a utilizar para generar respuestas automáticas en nuestros proyectos a eventos exteriores o para enviar datos a otros servicios. El significado de IFTTT es “If This Then That”. IFTTT nos permite configurar un evento (trigger) que desencadena una acción (Action). Dentro de IFTTT encontramos casi todos los servicios que existen en internet. Solo hay que pensar que es lo que queremos hacer y buscar el servicio que mejor se adapte.

Vamos a poder integrar nuestro proyecto con la plataforma IFTTT, para que nuestros dispositivos puedan reaccionar ante ciertos eventos y también se pueden enviar eventos a la plataforma IFTTT.

En nuestro proyecto vamos a utilizar este servicio de IFTTT para la programación horaria de diversos eventos, como son:

- Encendido y apagado programado de luces, ocho horas diarias durante el día.
- Encendido y apagado programado de aireación, dos horas diarias por la noche.
- Cambio parcial de agua mensual
- Alimentación peces, dos veces al día.

El primer paso para utilizar IFTTT es en darse de alta en la plataforma. Para ello accedemos a su dirección web https://ifttt.com/join. Una vez dados de alta ya podemos crear nuestra primera receta, seleccionando “New Applet”, como vemos en la ilustración 191.

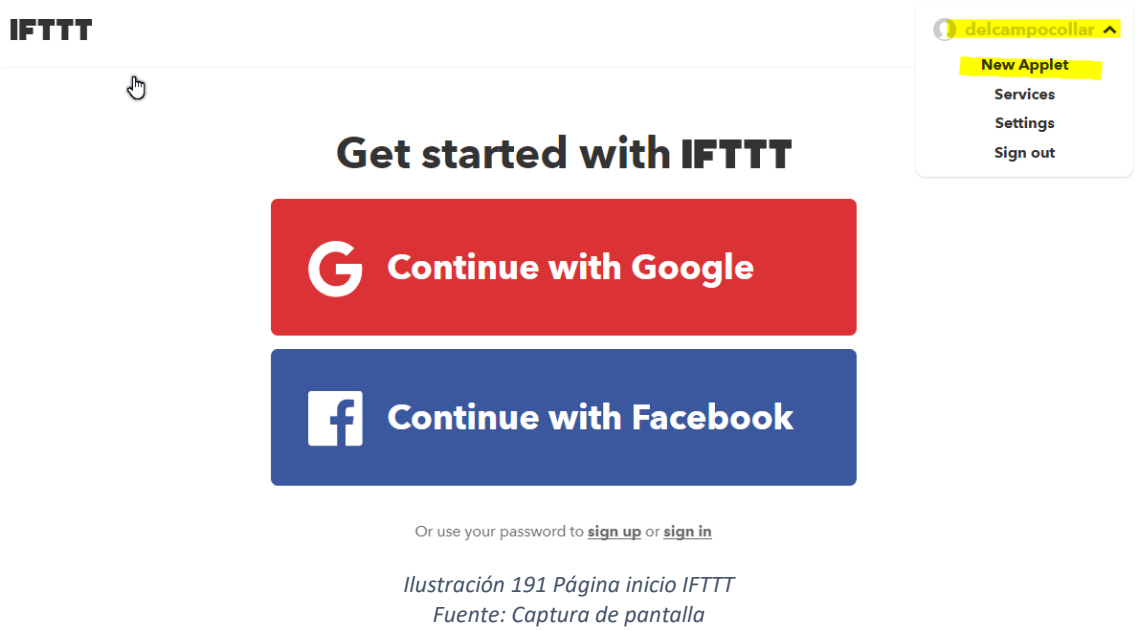


Ilustración 191 Página inicio IFTTT
Fuente: Captura de pantalla

Y accedemos a la pantalla principal, que se muestra en la ilustración 192, para crear una nueva receta.

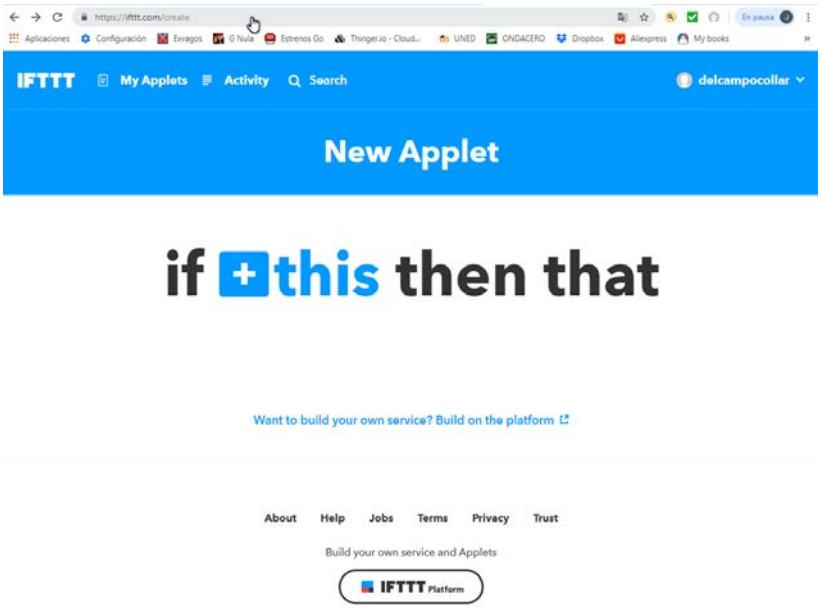


Ilustración 192 Pantalla New Applet de IFTTT
Fuente: Captura de pantalla

Como ejemplo de uso vamos a crear una receta para el encendido de la luz, conectada al relé 1, a las horas en punto y otra receta para que se apague a los quince minutos.

Lo primero que hay que hacer es seleccionar un evento. Hacemos clic en ´”+this” y nos sale la pantalla de la ilustración 193 donde podemos escoger el evento (trigger) que va a desencadenar el evento.

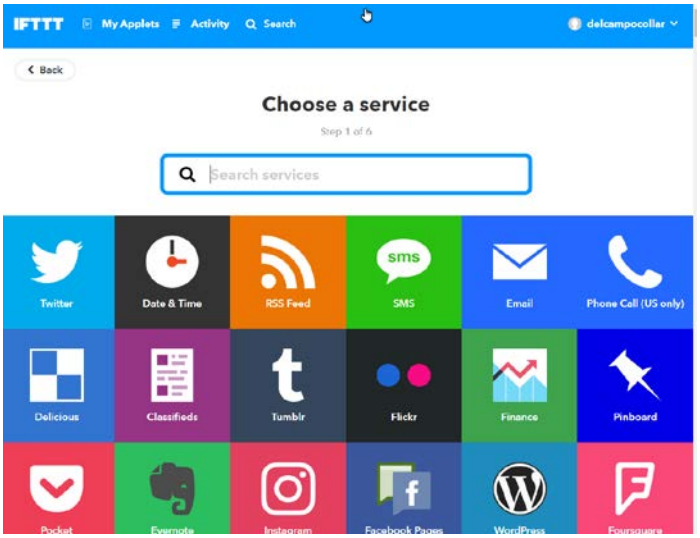


Ilustración 193 Pantalla para escoger un servicio
Fuente: Captura de pantalla

Buscamos el servicio “Date&Time”. Ver ilustración 194.

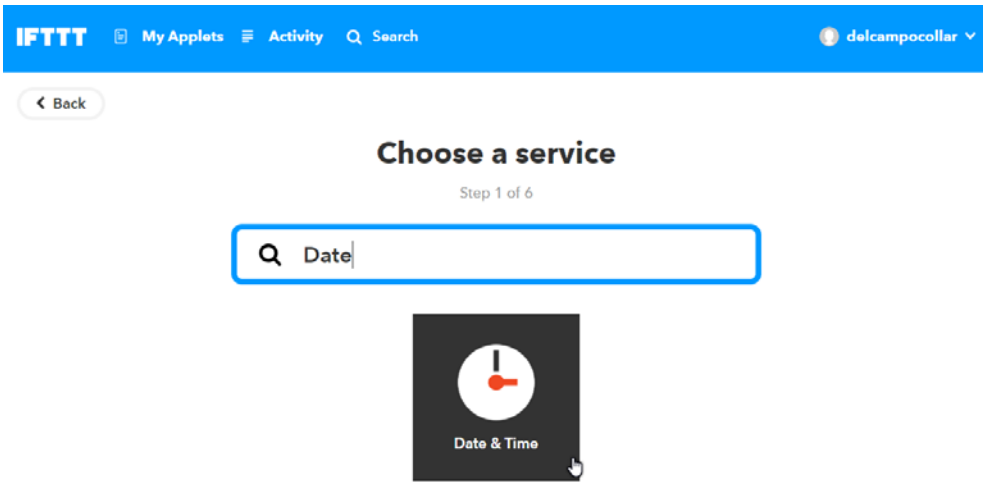


Ilustración 194 Búsqueda del servicio Date&Time
Fuente: Captura de pantalla

Hacemos clic en el icono de Date&Time y nos aparece otra pantalla para escoger el evento (trigger), como vemos en la ilustración 195. En este ejemplo vamos a escoger la opción “Every hour at” para que a todas las horas en punto se encienda la luz.

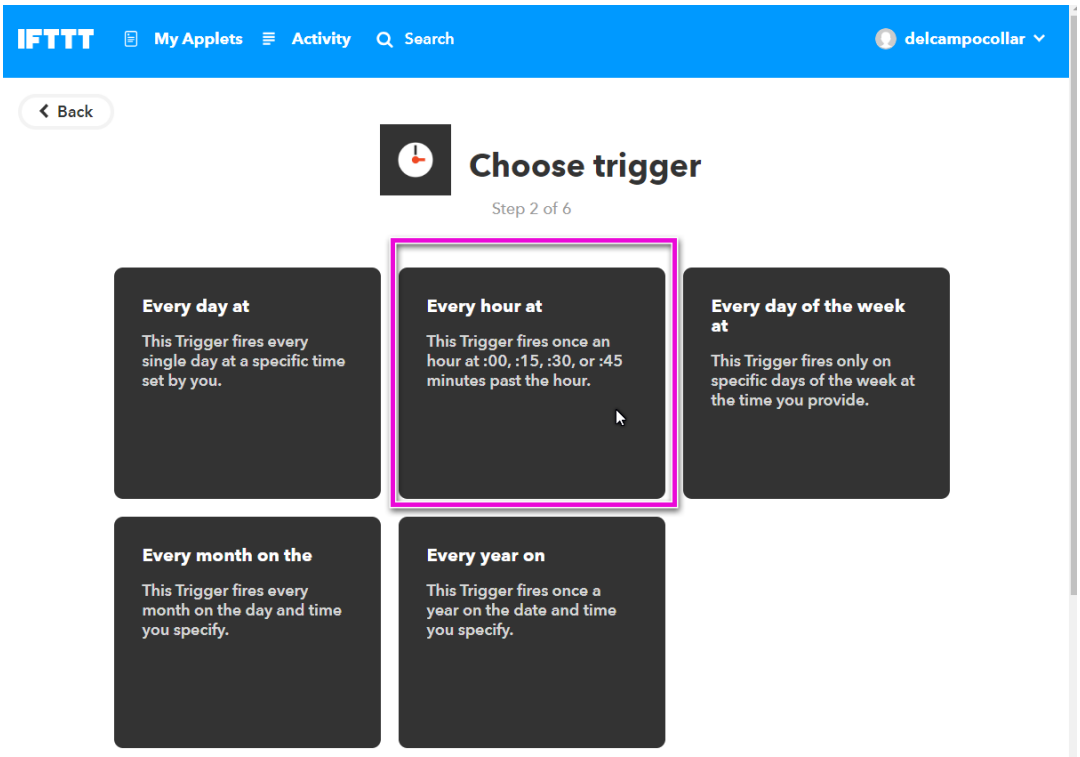


Ilustración 195 Escoger evento de Data&Time
Fuente: Captura de pantalla

Seleccionamos “00 Minutes past the hour” y hacemos clic en el botón “Create trigger”. Ver ilustración 196.

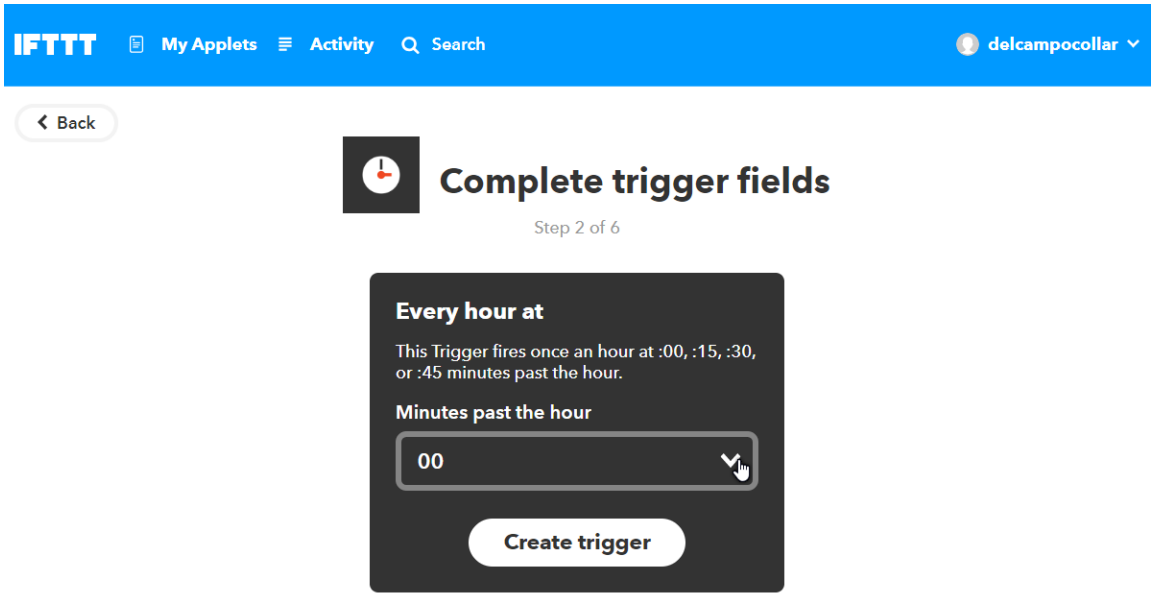


Ilustración 196 Cumplimentar campo del evento de Data&Time
Fuente: Captura de pantalla

Ahora hacemos clic en “+that”, como vemos en la ilustración 197, para seleccionar la respuesta.

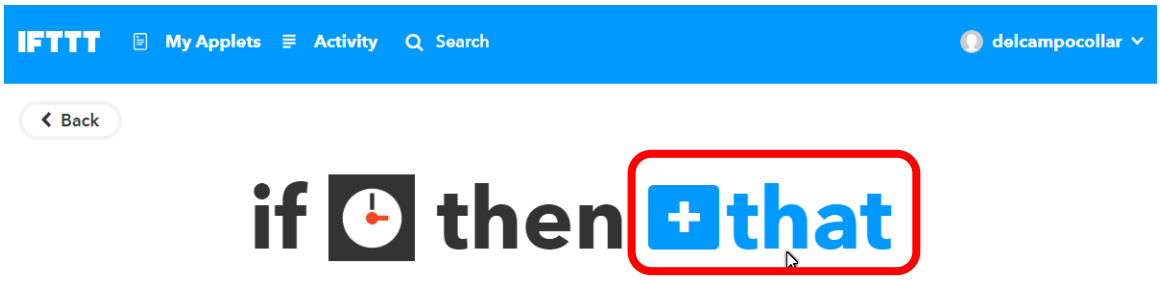


Ilustración 197 Seleccionar acción de respuesta
Fuente: Captura de pantalla

Ahora buscamos el servicio Webhooks, que nos permite hacer una solicitud web a una URL de acceso público. Ver ilustración 198.

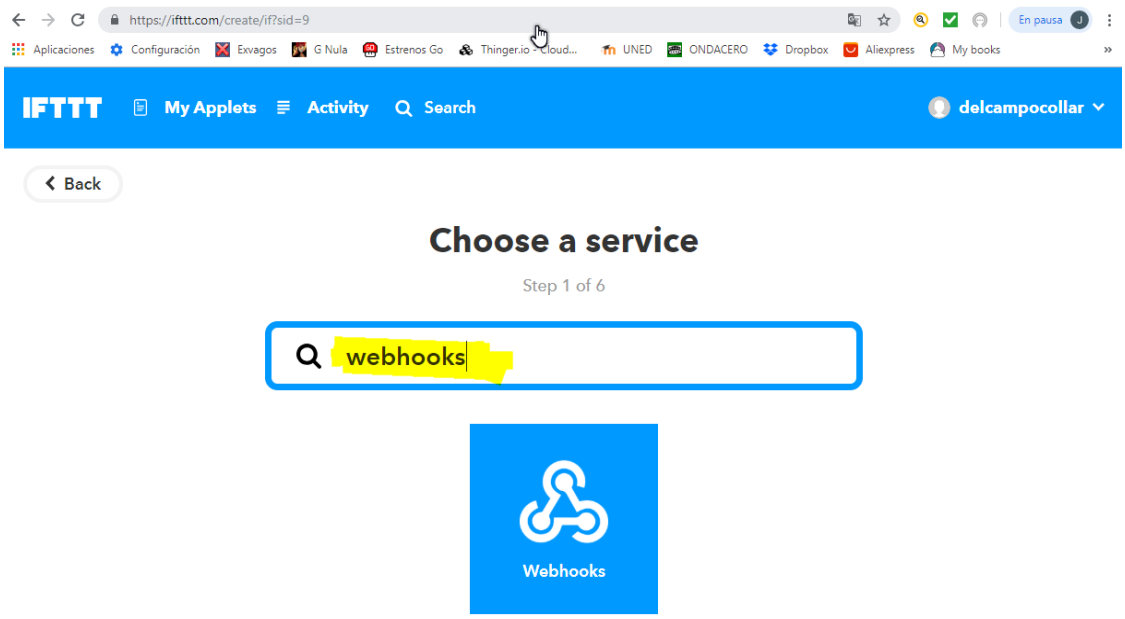


Ilustración 198 Servicio webhooks
Fuente: Captura de pantalla

Hacemos clic en el icono de Webhooks y nos aparece otra pantalla para escoger el evento, como vemos en la ilustración 199. En este caso solo hay una opción, "Make a web request", que es la que seleccionamos.

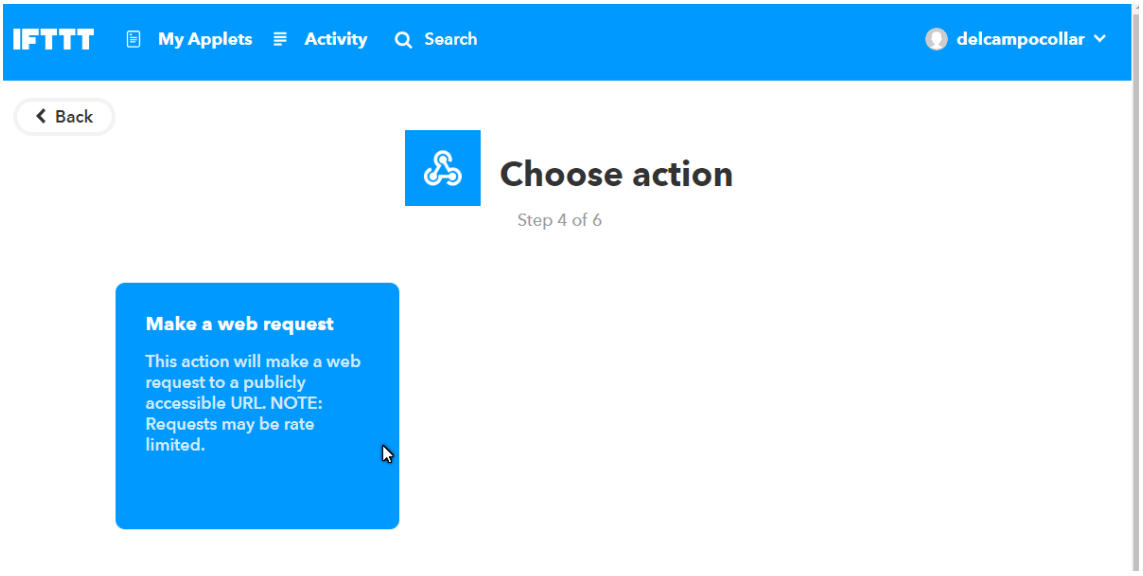


Ilustración 199 Acciones del webhooks
Fuente: Captura de pantalla

En la siguiente pantalla, ilustración 200, tenemos que cumplimentar los diferentes campos.

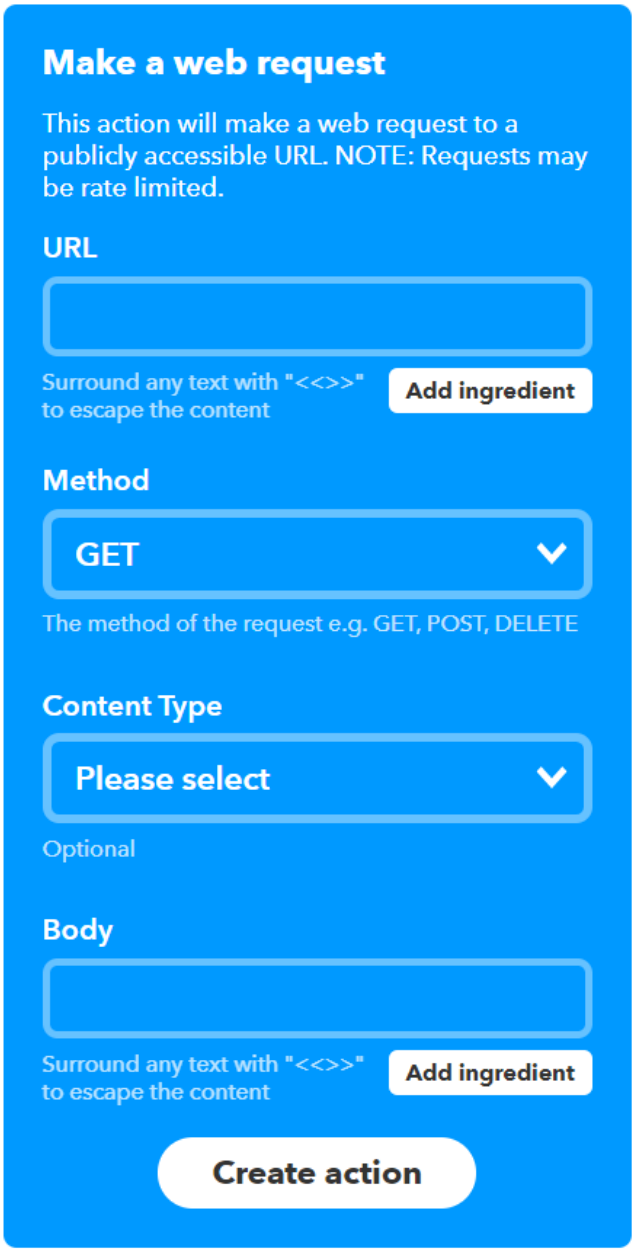


Ilustración 200 Campos a cumplimentar del webhooks
Fuente: Captura de pantalla

Los datos de la URL los obtenemos del Device Token “luzpecera1”, que previamente hemos creado en la plataforma Thinger.io, para permitir el acceso al rele1, que es donde se conecta la luz. Ver ilustración 201.

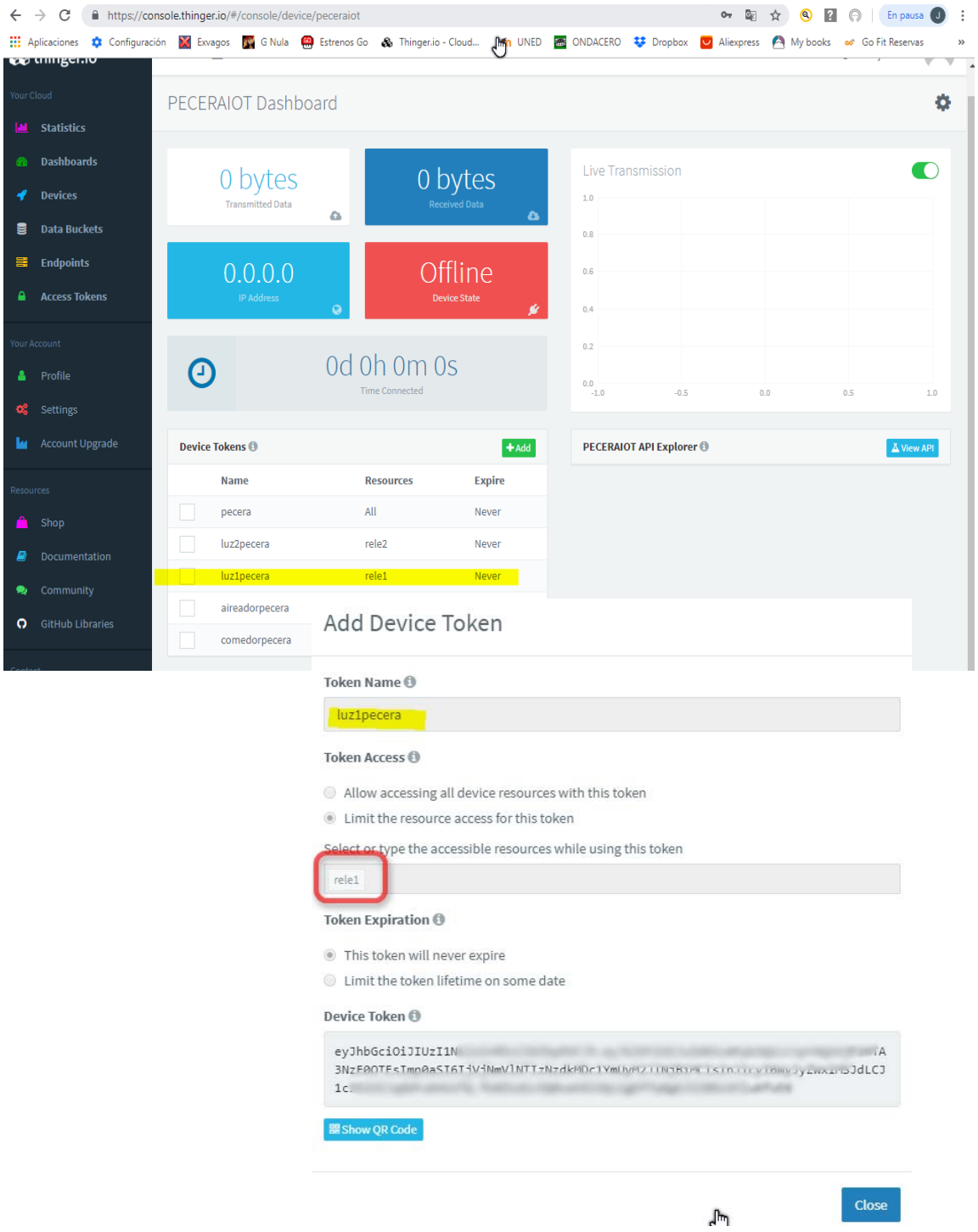


Ilustración 201 Token luz1pecera
Fuente: Captura de pantalla

En la ilustración 202 vemos el formulario una vez cumplimentado.

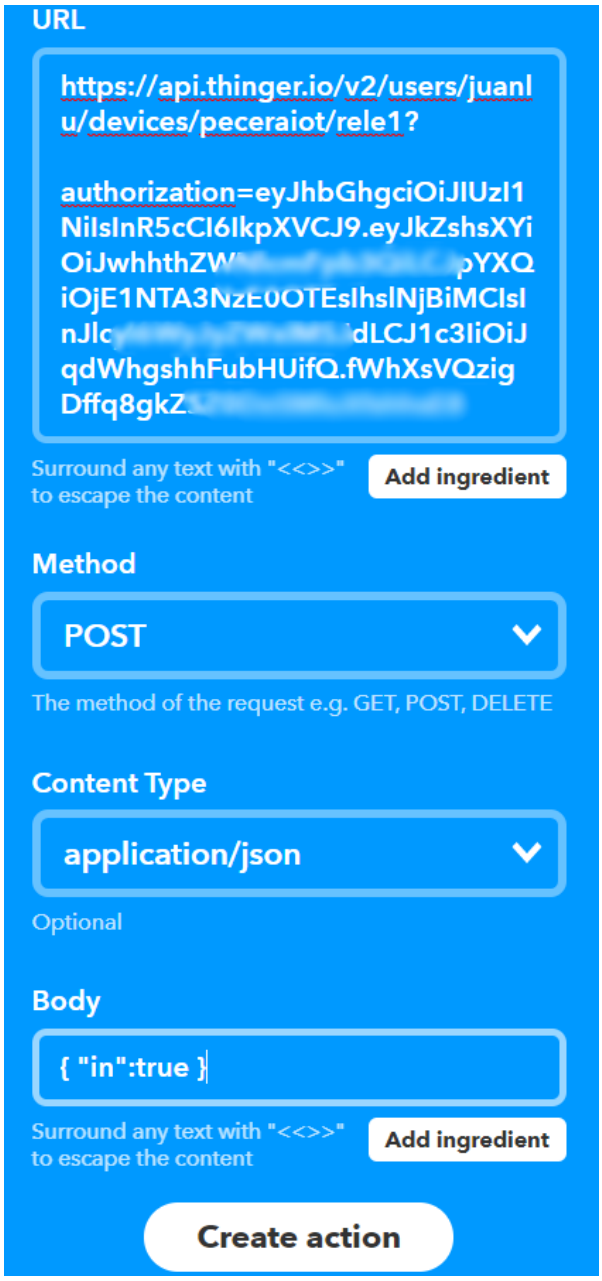


Ilustración 202 Formulario del webhooks cumplimentado
Fuente: Captura de pantalla

Para finalizar hacemos clic en el botón “Create action”. A partir de entonces a las horas en punto se va a encender la luz 1 del acuario.

Para crear la receta para que la luz 1 se apague a los quince minutos, lo que tenemos que hacer es seguir los pasos anteriores, haciendo los siguientes cambios. En el apartado “Complete trigger fields” seleccionamos “15 Minutes past the hour”, como vemos en la ilustración 203.

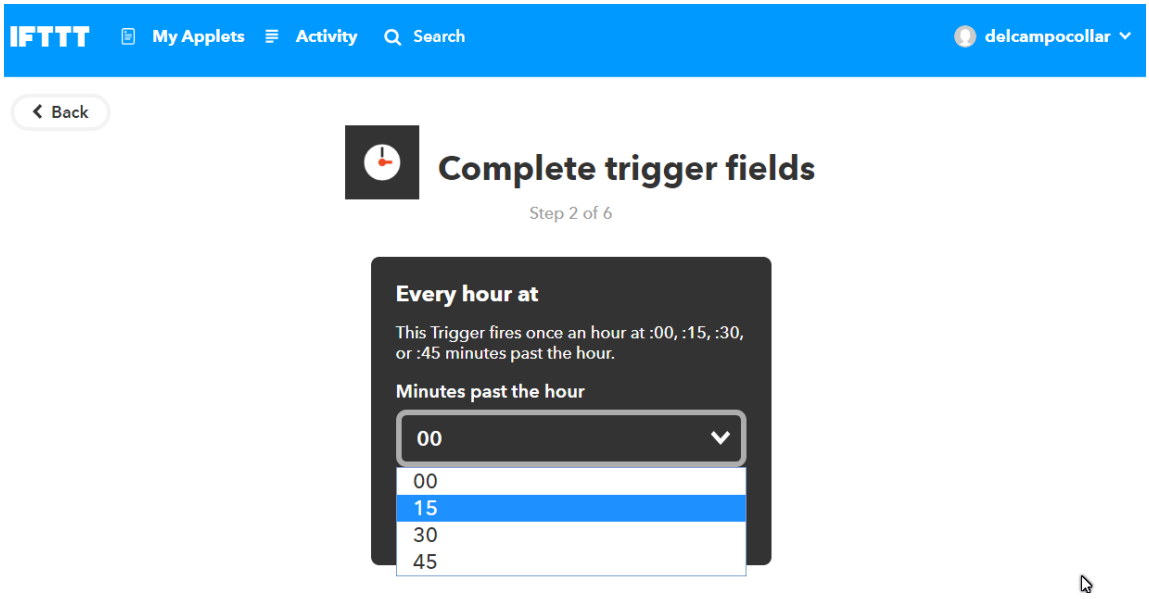


Ilustración 203 Cumplimentar campo del evento de Data&Time para apagar luz
Fuente: Captura de pantalla

y en el campo “Body” de “Complete action fields” escribimos { “in”:false } para que se apague la luz. Ver ilustración 204.

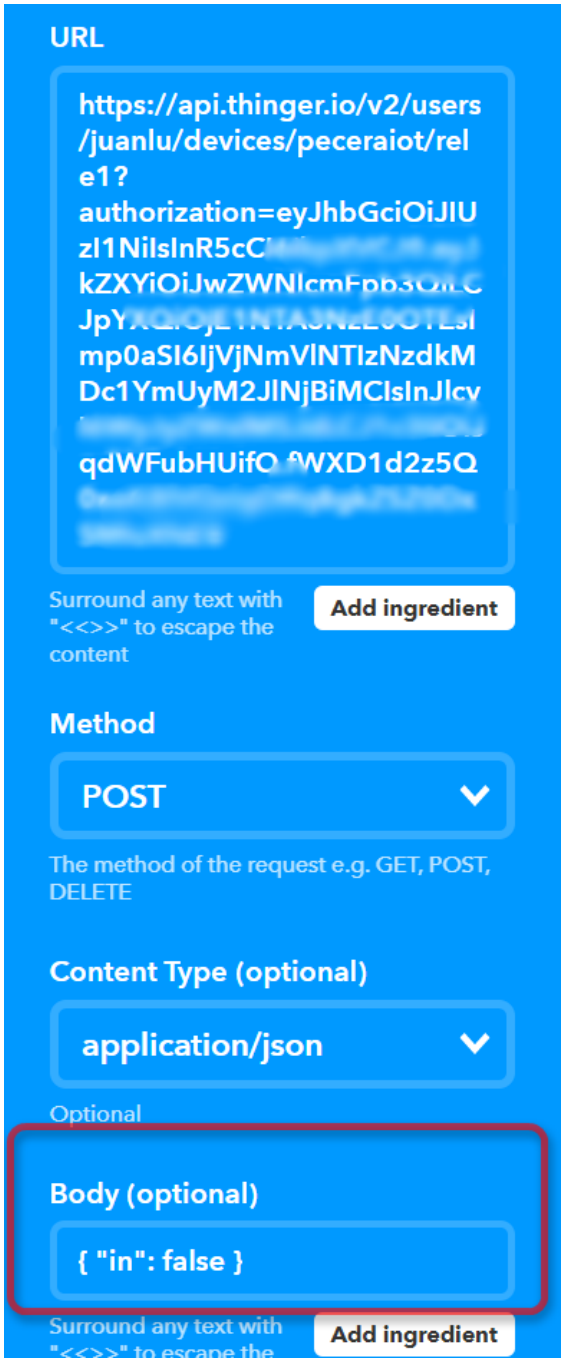


Ilustración 204 Formulario del webhooks cumplimentado para apagar la luz
Fuente: Captura de pantalla

En la ilustración 205 vemos las ocho recetas creadas para probar la programación horaria de diferentes dispositivos: luz 1, luz 2, aireador y comedor.

Se activa un dispositivo durante quince minutos y cuando este se desactiva, se activa el siguiente durante quince minutos y así sucesivamente.

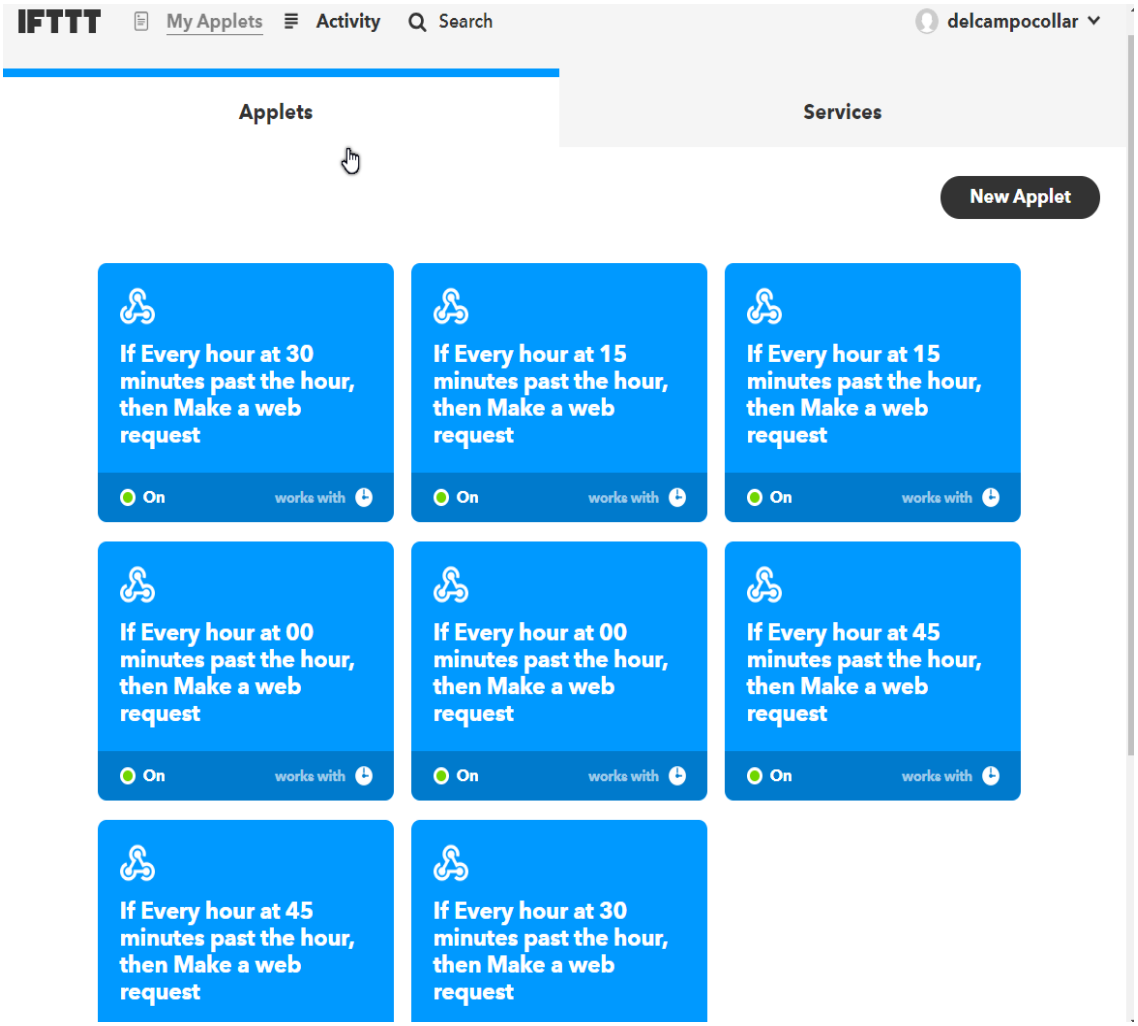


Ilustración 205 Recetas creadas para probar el proyecto
Fuente: Captura de pantalla

8.7. Anexo 7. Código fuente del Arduino

```
/*
 * PECERA  Mayo 2019
 * 04-05-2019 peceraiot_12_02_21
 */

// Para las RTDs
// negro tierra, rojo 5v, azul pin 5 (RTD)
// Pines 50, 51, 52 y 53 reservados para el bus SPI para la comunicación entre
// el Arduino Mega y el shield Ethernet
// Pin 4 reservado para la tarjeta SD
// Pin 10 reservado como SS

  #define _DEBUG_                  // Para la depuración a través del puerto serial

// CARGA DE LIBRERIAS
  #include <SPI.h>                 // Para la comunicación de dispositivos SPI con Arduino
  #include <Ethernet.h>            // Para funcionar con el shield Ethernet Arduino
  #include <ThingerEthernet.h>     // Para la plataforma Thinger
  #include <LiquidCrystal_I2C.h> // Para el control de la pantalla de cristal líquido LCD
  #include <OneWire.h>             // Para la comunicación mediante el protocolo I2C
  #include <DallasTemperature.h> // Para el control del sensor de temperatura DS18B20
  #include <Servo.h>               // Para el control del servomotor

// CONEXION CON THINGER
  #define USERNAME "****"       // Comunicación de la placa con Thinger
  #define DEVICE_ID "peceraiot" // Identificador del dispositivo
  #define DEVICE_CREDENTIAL "*******"   // Credencial

  ThingerEthernet thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);
                                   // Conecta el dispositivo con la plataforma

// DEFINICION DE CONSTANTES
  #define PinComedor 3             // Pin de salida para el control del servomotor (Comedor)
  #define PinTemperatura 5         // Pin de entrada para la comunicación de los sensores de Tª
  #define PinLedCambio 7           // Pin de salida para el led indicador del cambio de agua
  #define PinCambio 8              // Pin de salida para el control del cambio de agua
  #define PinManual 12             // Pin entrada selección modo de funcionamiento
  #define PinNivelAlto 16          // Pin de entrada para el detector de nivel alto agua
  #define PinNivelBajo 17          // Pin de entrada para el detector de nivel bajo agua
  #define PinAlarmaNivelAlto 22    // Pin de salida para la alarma de nivel alto agua
  #define PinAlarmaNivelBajo 23    // Pin de salida para la alarma de nivel bajo agua
  #define PinAlarmaTemp 24         // Pin de salida para la alarma de temperatura
  #define PinAlarmapH 26           // Pin de salida para la alarma de pH
  #define PinAcidez A0             // Pin de entrada analógico para la lectura del PH
  #define ErrorT1 0.0              // Error obtenido para el sensor de temperatura T1 (agua)
  #define ErrorT2 0.0              // Error obtenido para el sensor de temperatura T2 (tierra)
  #define ReleON 0                 // Estado de salida del relé en ON
```

```
#define ReleOFF 1                    // Estado de salida del relé en OFF
#define ErrorpH 0.0                  // Error obtenido para el lector de pH

const String Mensaje[]={"PECERA    pH:","T1:","T2:","MODO:"};  // Mensajes en
                                                                pantalla LCD
const String BorrarLinea={"                    "};          // Borra una línea del LCD
const byte PinRele[]={35,37,39,41,43,45,47,49}; // Pines de salida conectados a los relés
const byte PinWire[]={34,36,38,40,42,44,46,48}; // Pines entrada conectados a interruptores
const String TextoWireOn[]={"  COMEDOR      ON  "," LUZ PECES     ON  ",
"  LUZ PLANTAS   ON  ", "  AIREADOR     ON  ", "  CALENTADOR    ON  ",
"  BOMBA PPAL    ON  ", "  BOMBA LLENAR  ON  "," BOMBA VACIAR  ON  "};
                                     // Texto para el LCD al activarse un relé
const String TextoWireOff[]={"  COMEDOR      OFF  "," LUZ PECES     OFF  ",
"  LUZ PLANTAS  OFF  ", "  AIREADOR     OFF  "," CALENTADOR   OFF  ",
"  BOMBA PPAL   OFF  ", "  BOMBA LLENAR OFF  ",
"  BOMBA VACIAR OFF  "};      // Texto para el LCD al apagarse un relé

// LCD
LiquidCrystal_I2C lcd(0x3F,20,4);  // Dirección de la pantalla LCD, nº columnas 20, nº filas 4

// RTD
OneWire oneWireBus(PinTemperatura);  // Creamos el objeto oneWireBus indicando el pin de
                                        comunicación
DallasTemperature Sensors(&oneWireBus);   // Asociamos el objeto a un sensor de Dallas
DeviceAddress RTD1 = { 0x28, 0xFF, 0xE1, 0x7C, 0xA1, 0x16, 0x4, 0x9F };
                                     // Dirección del sensor de temperatura RTD1 (agua)
DeviceAddress RTD2 = { 0x28, 0xFF, 0xD7, 0x6A, 0xA1, 0x16, 0x4, 0xB6 };
                                     // Dirección del sensor de temperatura RTD2 (tierra)

// DEFINICION DE VARIABLES
unsigned long Contador = 0;          // Contador de tiempo del programa
byte ResolucionRTD = 11;             // Resolución para los sensores de temperatura
                                  // 9= 0,5 ºC, 10= 0,25 ºC, 11= 0,125 ºC, 12= 0,0625 ºC,
byte EstadoRele[]={1,2,3,4,5,6,7,8};   // Indica el estado ON/OFF de los relés
byte ValManual;                      // Selección modo de funcionamiento
byte Modo;                        // Modo de funcionamiento 0 = automático, 1 = manual
byte EstadoWire[]={1,2,3,4,5,6,7,8};   // Indica el estado ON/OFF de los interruptores
float TemperaturaT1,TemperaturaT2;   // Temperatura de los sensores RTD1 y RTD2
float ConsignaT = 23;                // Consigna de temperatura del agua
float HisteresisT = 1.5;             // Histéresis para la temperatura
float ConsignaPh = 7.0;              // Consigna del pH del agua
float HisteresisPh = 0.45;           // Histéresis para el pH
int retraso = 1;                     // Tiempo para el envió de la trama en minutos
int contador = 0;                    // Para retardar la alarma del Ph
int cambiomen = 0;                   // Para el cambio de agua
bool enviadoT=false;                 // Para el control del envío del email por alarma Tª
bool enviadoN=false;                 // Para el control del envío del email por alarma Nivel
bool enviadoPh=false;                // Para el control del envío del email por alarma pH
bool AlarmaTemperatura;              // Alarma de temperatura
bool EstadoBombappal, EstadoCalentador; // Estado de las bombas
```

```
  bool cambioagua = false;          // Para el cambio de agua
  bool paso1 = false;               // Para verificar el paso 1 del cambio del agua
  bool paso2 = false;               // Para verificar el paso 1 del cambio del agua
  bool extradatos = false;          // Para el envío extra de datos
  String TramaReles;                // Estado de los relés

// SERVOMOTOR
  Servo Comedor;                    // Declaración de la variable comedor de tipo Servo
  boolean Alimentar;                // Para el control de la alimentación

// NIVEL AGUA
  boolean NivelAlto, NivelBajo;     // Estado de los niveles del agua de la pecera
  boolean AlarmaNivelAlto, AlarmaNivelBajo; // Alarma de los niveles

// ACIDEZ
  unsigned int LecturaAcidez;       // Valor de la lectura del sensor de Acidez
  float Acidez;                     // Valor PH del agua


void setup() {

    // COMUNICACION SERIE
    Serial.begin(115200);           // Velocidad de comunicación en baudios
    Serial.println(Mensaje[0]);     // Envía el mensaje 0 al puerto serial

    // LCD
    lcd.init();                     // Inicializa la pantalla LCD
    lcd.backlight();                // Enciende la retroiluminación de la pantalla
    lcd.clear();                    // Borra la pantalla
    lcd.setCursor(0,0);             // Sitúa el cursor en la columna 0 de la fila 0
    lcd.print(Mensaje[0]);          // Escribe el mensaje 0 en la pantalla

    //SENSOR DE TEMPERATURA
    Sensors.setResolution(RTD1, ResolucionRTD);   // Establece la resolución para RTD1
    Sensors.setResolution(RTD2, ResolucionRTD);   // Establece la resolución para RTD2

    // DECLARACION PINES
    pinMode(PinComedor, OUTPUT);    // Declara PinComedor como salida
    for(byte i=0; i<sizeof(PinRele);i++)
    {
      pinMode(PinRele[i],OUTPUT);       // Declara los pines donde están conectados los relés
                                           como salida
      digitalWrite(PinRele[i],ReleOFF); // Desactiva todos los relés
    }
    for(byte i=0; i<sizeof(PinWire);i++)
    {
      pinMode(PinWire[i],INPUT);   // Declara los pines donde se conectan los interruptores
    }
```

```
// SERVOMOTOR
Comedor.attach(PinComedor);  // Vincula el objeto Comedor con el pin donde está conectado
Comedor.write(1);            // Controla el eje del servomotor. El parámetro ángulo en grados
Alimentar=false;             // Inicializa la variable alimentar a false

// NIVEL
pinMode(PinNivelAlto,INPUT);        // Declara el pin del detector de nivel alto como entrada
digitalWrite(PinNivelAlto,HIGH);    // Inicializa el pin del detector de nivel alto a nivel alto
pinMode(PinNivelBajo,INPUT);        // Declara el pin del detector de nivel bajo como entrada
digitalWrite(PinNivelBajo,HIGH);    // Inicializa el pin del detector de nivel bajo a nivel alto
pinMode(PinAlarmaNivelAlto,OUTPUT); // Declara el pin de la alarma de nivel alto
digitalWrite(PinAlarmaNivelAlto,LOW);// Inicializa el pin de alarma de nivel alto a nivel bajo
pinMode(PinAlarmaNivelBajo,OUTPUT); // Declara el pin de la alarma de nivel bajo
digitalWrite(PinAlarmaNivelBajo,LOW);    // Inicializa el pin de alarma de nivel bajo

// TEMPERATURA
pinMode(PinAlarmaTemp,OUTPUT);// Declara el pin de la alarma de temperatura como salida
digitalWrite(PinAlarmaTemp,LOW);  // Inicializa el pin de alarma de temperatura a nivel bajo

// pH
pinMode(PinAlarmapH,OUTPUT);     // Declara el pin de la alarma de pH como salida
digitalWrite(PinAlarmapH,LOW);       // Inicializa el pin de alarma de pH a nivel bajo

//THINGER OUTPUTS
thing["servo"] << servo(Comedor);                    // Permite controlar un recurso de entrada
thing["comedor"] << digitalPin(PinComedor);          // Permite un control sobre un pin digital
thing["rele0"] << invertedDigitalPin(PinRele[0]);    // por lo que puede actuar sobre los
thing["rele1"] << invertedDigitalPin(PinRele[1]);    // estados de encendido/apagado
thing["rele2"] << invertedDigitalPin(PinRele[2]);
thing["rele3"] << invertedDigitalPin(PinRele[3]);
thing["rele4"] << invertedDigitalPin(PinRele[4]);
thing["rele5"] << invertedDigitalPin(PinRele[5]);
thing["rele6"] << invertedDigitalPin(PinRele[6]);
thing["rele7"] << invertedDigitalPin(PinRele[7]);
thing["cambio"] << digitalPin(PinCambio);

thing["retraso"] << [](pson& in){     // Tiempo entre envío de datos
  if (in.is_empty())
    {
      in = retraso;
    }
    else
    {
      retraso = in;
    }
};
thing["ConsignaT"] << [](pson& in){       // Consigna de temperatura
  if (in.is_empty())
    {
      in = ConsignaT;
```

```
    }
    else
    {
      ConsignaT = in;
    }
};
thing["HisteresisT"] << [](pson& in){ // Histéresis para la temperatura
  if (in.is_empty())
    {
      in = HisteresisT;
    }
    else
    {
      HisteresisT = in;
    }
};
thing["ConsignaPh"] << [](pson& in){   // Consigna pH
  if (in.is_empty())
    {
      in = ConsignaPh;
    }
    else
    {
      ConsignaPh = in;
    }
};
thing["HisteresisPh"] << [](pson& in){   // Histéresis para el pH
  if (in.is_empty())
    {
      in = HisteresisPh;
    }
    else
    {
      HisteresisPh = in;
    }
};

//THINGER INPUTS
thing["acuario"]>>[](pson&out){             // Permite controlar un recurso de salida
  out["comedo"] = digitalRead(PinComedor);   // Comedor
  out["come"] = digitalRead(PinRele[0]);
  out["luz1"] = digitalRead(PinRele[1]);    // Luz 1
  out["luz2"] = digitalRead(PinRele[2]);    // Luz 2
  out["aire"] = digitalRead(PinRele[3]);    // Aireador
  out["cale"] = digitalRead(PinRele[4]);     // Calentador
  out["bbpp"] = digitalRead(PinRele[5]);   // Bomba principal de agua
  out["bbll"] = digitalRead(PinRele[6]);    // Bomba llenar
  out["bbva"] = digitalRead(PinRele[7]);    // Bomba vaciar
  out["retras"]= retraso;
  out["nivelalto"] = digitalRead(PinAlarmaNivelAlto);
```

```
    out["nivelbajo"] = digitalRead(PinAlarmaNivelBajo);
    out["alarmatemp"] = digitalRead(PinAlarmaTemp);
    out["alarmapH"] = digitalRead(PinAlarmapH);
    out["tagua"]= TemperaturaT1;
    out["ttierra"]= TemperaturaT2;
    out["pH"]= Acidez;
    out["ConsignT"]= ConsignaT;
    out["HisteresiT"]= HisteresisT;
    out["ConsignPh"]= ConsignaPh;
    out["HisteresiPh"]= HisteresisPh;
    out["ledcambio"] = cambiomen;
  };
}


void loop() {
  leerRele();                    // Obtener el estado de los relés
  leerNivel();                   // Obtener el estado de los sensores de nivel del agua
  leertemperatura();             // Obtiene la temperatura
  modotrabajo();                 // Obtener el modo de funcionamiento
  switch (Modo)
   {
    case 0:                      // Modo de funcionamiento AUTOMATICO seleccionado
      thing.handle();            // Llama al cliente de Thinger
      actuadores();
      cambiaragua();


      // ENVIO DE DATOS A LA PLATAFORMA
      if (millis() > Contador + (60000*retraso))  //Tiempo de envío entre datos
        {
         leeracidez();           // Obtiene el pH
         enviardatos();
        }

       // ENVIO EXTRA DE DATOS A LA PLATAFORMA
      if ((millis() > Contador + (61000)) && (extradatos==1))  //Tiempo de envío entre
                                                                 datos 61 segundos
        {
         enviardatos();
        }

    break;

    case 1:                      // Modo de funcionamiento MANUAL seleccionado
       leerInterruptor();        // Leer el estado de los interruptores
       actualizarRele();         // Activa/desactiva los relés en función del estado de los interruptores
    break;
```

```
    default:                        // Error en la selección del modo de funcionamiento
        Serial.println("Error en la selección del modo de funcionamiento");
    break;
   }
 }


void leertemperatura()
 {
  Sensors.requestTemperatures(); // Se prepara el sensor para la lectura
  TemperaturaT1=Sensors.getTempC(RTD1)+ErrorT1;   // Se lee la Tª de cada sensor
  TemperaturaT2=Sensors.getTempC(RTD2)+ErrorT2;          // y se añade su error
  if ((TemperaturaT1 == -127.00) or (TemperaturaT2 == -127.00))  // Error en la lectura
  {
   Serial.println("Error captura temperatura");  // Enviamos el mensaje de error al puerto serial
   lcd.setCursor(0,1);                 // Selecionamos la fila 2 del LCD
   lcd.print(BorrarLinea);             // Borramos la fila 2
   lcd.setCursor(0,1);                 // Selecionamos la fila 2 del LCD
   lcd.print("Error captura T1 T2");  // Enviamos el mensaje de error al LCD
  }
  else
  {
   lcd.setCursor(0,1);                 // Se visualiza en la segunda línea del LCD las temperaturas
   lcd.print(BorrarLinea);             // Borramos la fila 2
   lcd.setCursor(0,1);                 // Posicionamos el cursor
   lcd.print(Mensaje[1]);              // Enviamos el mensaje 1
   lcd.setCursor(4,1);                // Posicionamos el cursor
   lcd.print(TemperaturaT1);          // Enviamos la temperatura T1
   lcd.setCursor(10,1);               // Posicionamos el cursor
   lcd.print(Mensaje[2]);              // Enviamos el mensaje 2
   lcd.setCursor(14,1);               // Posicionamos el cursor
   lcd.print(TemperaturaT2);          // Enviamos la temperatura T2

   Serial.println("Capturando Temperatura");
   Serial.print("T1 = ");
   Serial.println(TemperaturaT1);
   Serial.print("T2 = ");
   Serial.println(TemperaturaT2);

   if ((TemperaturaT1 < (ConsignaT-(2*HisteresisT)) || (TemperaturaT1 >
    (ConsignaT+(2*HisteresisT)))) && (enviadoT==0))
     {
      Serial.println("Alarma Temperatura");
      enviadoT = true;
      digitalWrite(PinAlarmaTemp,HIGH);
      thing.call_endpoint("EndpNipecera",thing["acuario"]); // Envío email con alarma de Tª
      lcd.setCursor(0,3);                // Posicionamos el cursor
      lcd.print("Alarma Temperatura  ");      // Enviamos el aviso de alarma
      extradatos = true;
     }
```

```
    if ((TemperaturaT1 > ConsignaT-HisteresisT) && (TemperaturaT1 <
ConsignaT+HisteresisT) && (enviadoT==1))
       {
         Serial.println("Temperatura Ok");
         Serial.println(TemperaturaT1);
         enviadoT=false;
         digitalWrite(PinAlarmaTemp,LOW);
         lcd.setCursor(0,3);                    // Posicionamos el cursor
         lcd.print("Temperatura Ok      ");     // Enviamos el aviso
         Serial.println(enviadoT);
         extradatos = true;
       }
     }
  }


 void leerNivel()
  {
   NivelAlto=digitalRead(PinNivelAlto); // Obtiene el estado del sensor de nivel alto
   NivelBajo=digitalRead(PinNivelBajo); // Obtiene el estado del sensor de nivel bajo
   AlarmaNivelAlto=digitalRead(PinAlarmaNivelAlto); // Estado del sensor de nivel alto
   AlarmaNivelBajo=digitalRead(PinAlarmaNivelBajo); // Estado del sensor de nivel bajo
   if ((NivelAlto==LOW) && (AlarmaNivelAlto==LOW))
   {
    digitalWrite(PinAlarmaNivelAlto,HIGH);  // Activa la salida de alarma nivel alto
    Serial.println("Alarma nivel alto");
    lcd.setCursor(0,3);
    lcd.print("Alarma nivel alto   ");
    extradatos = true;
   }
  if ((NivelAlto==HIGH) && (AlarmaNivelAlto==HIGH))
   {
    digitalWrite(PinAlarmaNivelAlto,LOW);   // Desactiva la salida de alarma nivel alto
    Serial.println("Fin Alarma nivel alto");
    lcd.setCursor(0,3);
    lcd.print("Fin Alarma nivel    ");
    extradatos = true;
   }
   if ((NivelBajo==LOW) && (AlarmaNivelBajo==LOW))
   {
    digitalWrite(PinAlarmaNivelBajo,HIGH);  // Activa la salida de alarma nivel bajo
    Serial.println("Alarma nivel bajo");
    lcd.setCursor(0,3);
    lcd.print("Alarma nivel bajo   ");
    extradatos = true;
   }
   if ((NivelBajo==HIGH) && (AlarmaNivelBajo==HIGH))
   {
    digitalWrite(PinAlarmaNivelBajo,LOW);   // Desactiva la salida de alarma nivel bajo
```

```
    Serial.println("Fin Alarma nivel bajo");
    lcd.setCursor(0,3);
    lcd.print("Fin Alarma nivel    ");
    extradatos = true;
   }
   if ((NivelBajo==HIGH) && (NivelAlto==HIGH) && (enviadoN==1))
   {
    enviadoN = false;
   }
   Serial.println("Capturando nivel");
   Serial.print("Nivel alto = ");
   Serial.println(AlarmaNivelAlto);
   Serial.print("Nivel bajo = ");
   Serial.println(AlarmaNivelBajo);
 }


 void leeracidez()
 {
   LecturaAcidez = analogRead(PinAcidez);
   Acidez = (1023 - LecturaAcidez);
   Acidez = map(Acidez,0,1023,0,70);
   Acidez = (Acidez / 5.00) + ErrorpH;
   lcd.setCursor(15,0);                  // Posicionamos el cursor
   lcd.print(Acidez);                    // Enviamos el valor de la acidez
   Serial.println("Capturando pH");
   Serial.print("pH = ");
   Serial.println(Acidez);

   if ((Acidez < (ConsignaPh - HisteresisPh) || (Acidez > (ConsignaPh+HisteresisPh))
    && (enviadoPh==0)))
     {
      contador=contador +1;
      if (contador > 2 )                 // Alarma a la tercer lectura
        {
          Serial.println("Alarma pH");
          enviadoPh = true;
          digitalWrite(PinAlarmapH,HIGH);
          thing.call_endpoint("EndpNipecera",thing["acuario"]); // Envío email alarma de pH
          lcd.setCursor(0,3);           // Posicionamos el cursor
          lcd.print("Alarma pH          ");   // Enviamos el aviso de alarma
        }
     }

   if ((Acidez >= ConsignaPh-HisteresisPh) && (Acidez <= ConsignaPh+HisteresisPh)
    )
     {
       contador=0;
      Serial.print("pH Ok = ");
      Serial.println(Acidez);
```

```
      if (enviadopH==1)
          {
            enviadoPh=false;
            digitalWrite(PinAlarmapH,LOW);
            lcd.setCursor(0,3);                 // Posicionamos el cursor
            lcd.print("       pH  Ok       ");  // Enviamos el aviso
          }
    }
}


void alimenta()
{
  for (byte i=1;i<170;i++)            // Mueve el servomotor  paso a paso de la posición 1º a 170º
    {
            Comedor.write(i);
            delay(10);
    }
  Comedor.write(1);                   // Mueve el servomotor a la posición 1º
  delay(10);
}


void modotrabajo()
{
  ValManual=digitalRead(PinManual);    // pin 12    AUTOMATICO 0 / MANUAL 1
  if((ValManual == LOW) && (Modo!=0))
  {
    Modo = 0;                          // Modo automático
    Serial.println("Modo automatico");
    lcd.setCursor(0,2);
    lcd.print("MODO AUTOMATICO");
   }
if((ValManual == HIGH) && (Modo!=1))
  {
    Modo = 1;                        // Modo manual
    Serial.println("Modo manual");
    lcd.setCursor(0,2);
    lcd.print("MODO MANUAL    ");
   }
}


void leerInterruptor(){
   for(byte i=0; i<sizeof(PinWire);i++) // Lee el estado de los interruptores
   {
    EstadoWire[i]=digitalRead(PinWire[i]); // EstadoWire contiene el estado de los interruptores
   }
}
```

```
void actualizarRele(){
  for(byte i=0; i<sizeof(EstadoWire);i++)
  {
   if((EstadoWire[i] == HIGH) && (EstadoRele[i]==HIGH))  // Si el interruptor está
                                                          activado y el relé no
     {
     digitalWrite(PinRele[i],ReleON); // activa el rele
     Serial.println(TextoWireOn[i]);
     lcd.setCursor(0,3);
     lcd.print(TextoWireOn[i]);
     }
   if((EstadoWire[i] == LOW) && (EstadoRele[i]==LOW))    // Si el interruptor está
                                                          desactivado y el relé si
     {
     digitalWrite(PinRele[i],ReleOFF);// desactiva el relé
     Serial.println(TextoWireOff[i]);
     lcd.setCursor(0,3);
     lcd.print(TextoWireOff[i]);
     }
  }
  if((EstadoWire[0] == HIGH) && (Alimentar== false))      // Comprueba si hay que
                                                          alimentar en manual
    {
     alimenta();
     Alimentar=true;
     }
  if((EstadoWire[0] == LOW) && (Alimentar== true))
     {
     Alimentar=false;
     }
}


void leerRele(){
  TramaReles="";
  for(byte i=0; i<sizeof(PinRele);i++)          // Lee el estado de los relés
  {
   EstadoRele[i]=digitalRead(PinRele[i]);    // TramaReles contiene el estado de los relés
   TramaReles+=EstadoRele[i];
  }
}


void actuadores(){
      //BOMBA PRINCIPAL
      EstadoBombappal = digitalRead(PinRele[5]);    // Bomba principal de agua
      if (EstadoBombappal==1)
        {
          digitalWrite((PinRele[5]),LOW);  // Activa la bomba principal
```

```
      extradatos = true;              // Para enviar los datos a la plataforma
    }

  // CALENTADOR
  EstadoCalentador = digitalRead(PinRele[4]);    // Calentador
  if ((TemperaturaT1 < ConsignaT-HisteresisT) && EstadoCalentador==1)
    {
      Serial.println("Calentador ON");
      digitalWrite(PinRele[4],LOW);  // Activa calentador
      extradatos = true;
    }
  if ((TemperaturaT1 > ConsignaT+HisteresisT) && EstadoCalentador==0)
    {
      Serial.println("Calentador OFF");
      digitalWrite(PinRele[4],HIGH);  // Desactiva calentador
      extradatos = true;
    }

  // BOMBA LLENAR
  if ((AlarmaNivelAlto) && (enviadoN==0))
  {
    digitalWrite(PinRele[6],HIGH);  // Desactiva bomba llenar
    enviadoN = true;
    extradatos = true;
    thing.call_endpoint("EndpNipecera",thing["acuario"]); // Envió email con alarma de
                                                            nivel
  }

  // BOMBA VACIAR
  if ((AlarmaNivelBajo) && (enviadoN==0))
  {
    digitalWrite(PinRele[7],HIGH);                  // Desactiva bomba vaciar
    enviadoN = true;
    extradatos = true;
    thing.call_endpoint("EndpNipecera",thing["acuario"]); // Envío email con  alarma de
                                                            nivel
  }

  // COMEDOR
  if (Comedor.read()>20)
    {
      digitalWrite(PinRele[0],ReleON); // activa el relé
    }
  else
    {
      digitalWrite(PinRele[0],ReleOFF); // desactiva el relé
    }
}
```

```
void cambiaragua(){
// CAMBIO DE AGUA
        cambioagua = digitalRead(PinCambio);
        if ((cambioagua) && (not cambiomen))
          {
            cambiomen = 1;
            digitalWrite(PinLedCambio,HIGH); // activa el led indicador cambio de agua
            extradatos = true;
          }
        if (cambiomen==1)
        {
           if ((NivelBajo) && (not paso1) && (not paso2))
             {
               digitalWrite(PinRele[7],ReleON); // activa el relé de la bomba vaciado
               paso1 = true;
               extradatos = true;
             }
           if ((not NivelBajo) && (not paso2))
             {
               digitalWrite(PinRele[6],ReleON); // activa el relé e la bomba llenado
               paso2 = true;
               extradatos = true;
             }
           if ((not NivelAlto) && (paso1) && (paso2))
             {
               cambiomen = 0;
               paso1 =false;
               paso2 = false;
               digitalWrite(PinLedCambio,LOW);   // desactiva el led indicador cambio de agua
               extradatos = true;
             }
        }
}
void enviardatos(){
        thing.stream("acuario");   // Envío de los datos al data bucket según configuración
        Serial.println("Enviando datos");
        Serial.print("Retraso = ");
        Serial.println(retraso);
        Serial.println(ConsignaT);
        Serial.println(HisteresisT);
        Serial.println(ConsignaPh);
        Serial.println(HisteresisPh);
        Contador = millis();
        extradatos = false;
        Serial.println("Datos enviados");
}
```

8.8. Anexo 8. Análisis de datos

Nos va a permitir examinar los datos obtenidos para extraer la información útil que contienen.

Técnicas de análisis de datos:

- **Técnicas estadísticas.** Se realizan estudios de media, mediana moda, análisis de frecuencia, análisis de correlaciones o de regresión, estudios de variabilidad, estimaciones, etc. Existen multitud de herramientas preparadas para realizar análisis estadístico como:
  - **Excel.** Es un programa para trabajar con bases de datos de tamaño relativamente pequeño, pero con una gran versatilidad. Tiene funciones estadísticas avanzadas. (https://products.office.com/es-es/excel)
  - **Minitab.** Programa estadístico muy sencillo basado en la interfaz de Excel. Tiene mejor representación de gráficos. (http://www.minitab.com/es-mx/).
  - **SPS.** Ideal para el análisis estadístico y descriptivo de los datos. Dispone de una interface poco intuitiva. (https://www.ibm.com/analytics/es/es/technology/spss/)

- **Data mining.** Técnicas de análisis de datos que tratan de analizar de forma automática conjuntos de gran tamaño, para localizar patrones por medio de diferentes algoritmos. Ejemplo de herramientas que nos permiten realiza data mining, tareas de clasificación y análisis automático de conjuntos de gran tamaño son:

- **Weka.** Orientado al data mining. Contiene herramientas de preprocesado de datos, regresión, clasificación, clustering y visualización. Software de código abierto. (https://www.cs.waikato.ac.nz/ml/weka/)
- **Matlab.** Gran versatilidad. Es la herramienta más completa. Dispone de lenguaje de programación propio. (https://www.mathworks.com/products/matlab.html)

- **Técnicas de visualización.** La visualización de datos es una representación gráfica de la información y los datos. Mediante el uso de elementos visuales, como gráficos y mapas. La visualización de datos nos presenta de una manera fácil detectar y comprender las tendencias, los valores atípicos y los patrones en los datos. La visualización de los datos hace que los análisis sean más eficientes.
  Cuando vemos un gráfico, rápidamente identificamos las tendencias y los valores atípicos. Si podemos observar la información, nos resulta más fácil asimilarla.

8.9. Anexo 9. Seguridad en entornos IoT

Cada día es más frecuente encontrar noticias de ataques de ciberdelincuentes a redes de empresas u organizaciones a través de dispositivos de internet de las cosas. Los dispositivos IoT ofrecen a los ciberdelincuentes ejecutar ataques maliciosos aprovechándose de posibles vulnerabilidades. Estos ataques a dispositivos IoT empezaron hace años como se recoge en (27). Ver ilustración 206.

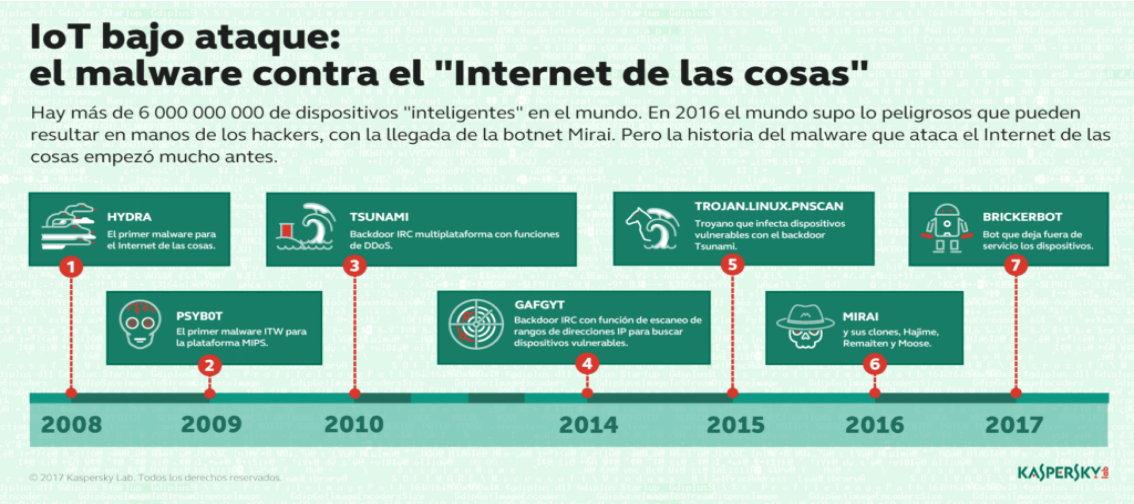


Ilustración 206 El malware contra el IoT
Fuente: http://www.mejor-antivirus.es/analisis/el-malware-contra-los-dispositivos-iot-se-ha-duplicado.html

Según (28) el número total de malware dirigido contra dispositivos IoT ha sobrepasado los 7.000, con más de la mitad de ellos apareciendo por primera vez en 2017, según los analistas de Kaspersky Lab y que podemos ver en la ilustración 207.

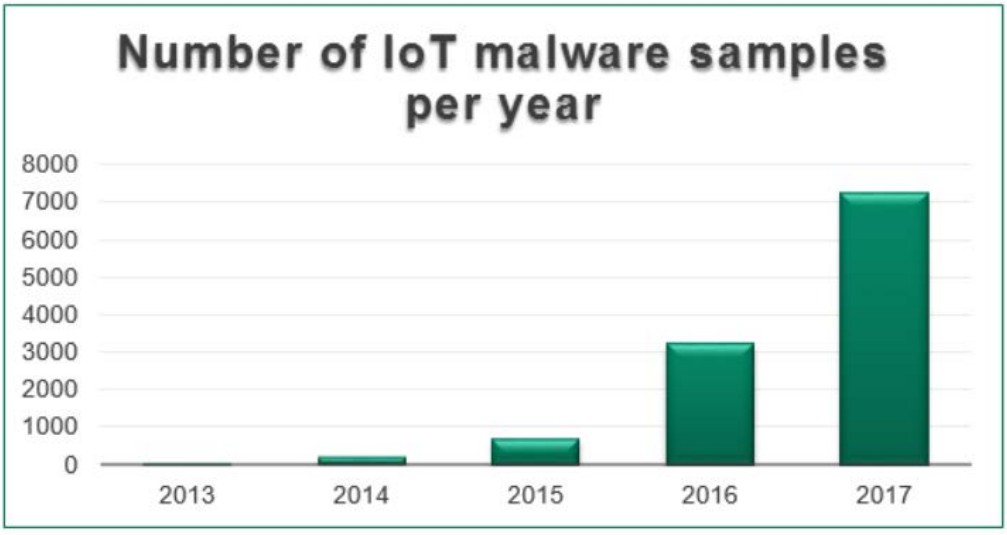


Ilustración 207 Número de malware para IoT por año
Fuente: https://www.kaspersky.es/about/press-releases/2017_malware-against-iot-devices-has-doubled-2017

Como ejemplo de estos ataques, y relacionado con nuestro proyecto, podemos mencionar el dado a conocer en la Reunión del Consejo de Londres del CEO que tuvo lugar el 12 de abril de 2018, por Nicole Eagan, directora ejecutiva de Darktrace, el caso de un casino que sufrió un robo de datos al acceder a su red tras detectar una vulnerabilidad en el termómetro del acuario del vestíbulo. Una vez dentro los atacantes robaron los datos privados de cientos de apostadores y los subieron a la nube (29).

Existen miles de dispositivos IoT que pueden ser atacados, ya que la mayoría no disponen de ningún tipo de defensa, por lo que es necesario implementar medidas de seguridad. Pero esto no es fácil, debido a las limitaciones de estos dispositivos y a que la implementación de medidas de seguridad exigiría mayor capacidad de procesamiento y mayor consumo de potencia.

Nosotros dejaremos para las ampliaciones futuras la realización de un estudio en profundidad para la aplicación de la seguridad a nuestro sistema.

8.9.1. Vulnerabilidades hardware

Parte de las vulnerabilidades por hardware surgen como resultado de la manipulación del contenedor del equipo, lo que permite acceder a los distintos componentes del hardware, o por tener acceso directo a alguna parte externa como por ejemplo las antenas, lo que dará lugar a una degradación de la funcionalidad del dispositivo.

Si el dispositivo dispone de un microcontrolador este puede ser manipulado, por lo que es recomendable implementar mecanismos que detecten la manipulación del hardware, como por ejemplo pulsadores o sensores de temperatura que detecten la apertura del contenedor y que provoquen la parada del equipo en caso de apertura.

También es importante que el dispositivo tenga los interfaces de comunicación (conector serie, USB, etc.) bien protegidos para evitar que un atacante se pueda conectar a ellos.

Para poder proporcionar ciertos niveles de seguridad en la transferencia de datos, se puede considerar la implementación de medidas Hardware y Software en un **SoC** (system on chip). Un SoC seguro es aquel que puede proveer una protección física, por ejemplo, almacenando las claves de encriptación en memorias tipo ROM seguras, que no se pueden leer, dentro del dispositivo en un lugar que no sean accesibles.

El reemplazo o movimiento de cualquier componente debe de ser imposible o sino ser capaz de inutilizar el equipo.

8.9.1.1. Vulnerabilidades en el Hardware de Arduino

Las placas Arduino en general no tienen una buena protección desde el punto de vista Hardware (30). Es más costoso el contenedor que la placa Arduino. Además, una vez que se accede a la placa es fácil acceder a cualquier elemento o inutilizar todo el circuito.

La realización por descuido de ciertas acciones, o el ataque malintencionado al Hardware, puede tener un resultado maligno para el equipo basado en una placa Arduino, como los que se muestran a continuación:

- Cortocircuitar un pin de E/S con tierra, esto sucede cuando configuramos un pin E/S como salida y lo establecemos en alto, si se conecta a tierra directamente produce sobrecarga en el pin y lo daña.

- Dos pines de E/S los establecemos como salida, activando uno en alto y al otro en bajo, y conectando los pines, se crea una sobreintensidad, dañándolos.

- Si aplicamos un voltaje superior a 5.5 V a un pin de E/S.

- Invertir la polaridad en la alimentación de la placa Arduino.

- Aplicar un voltaje de 6 V o más al pin de 5V de Arduino.

- Aplicar más de 3.6 V en el pin de 3.3 V de Arduino.

- Cortocircuitar el pin de alimentación Vin con GND.

- La corriente total procedente de todos los pines E/S, no debe superar los 200 mA.

Un ataque de Hardware importante es el **microprobing**, que consiste en acceder a los pines del chip directamente de manera que se puede observar, manipular e interferir el circuito integrado

Desde el punto de vista Hardware, es importante que el contenedor del dispositivo esté construido a prueba de manipulaciones, de manera que el acceso al circuito electrónico sea lo más complicado posible, algunas medidas a tomar pueden ser:

• Evitar dejar ninguna interfaz de acceso al dispositivo a la vista.

dotar al contenedor de mecanismos anti manipulación como por ejemplo el uso de tornillos unidireccionales.

• El contenedor debe de estar sellado de alguna forma, de manera que la ruptura o falta del sello implique la anulación de la garantía. deberíamos.

• Dotar al contenedor de mecanismos anti manipulación como por ejemplo el uso de tornillos unidireccionales.

• Incluir dispositivos como interruptores que detecten la apertura de la caja o el movimiento de algún componente, sensores que detecten un cambio operacional o medioambiental o circuitos que puedan detectar la rotura o el intento de modificar contenedor del dispositivo y que generen una señal de alarma o interrumpan el normal funcionamiento del equipo.

• Asegurarnos que los circuitos que forman parte de la placa Arduino estén recubiertos de resina epoxi o similar, para hacer que su extracción de la placa sea lo más complicada posible y produzca la destrucción del elemento que se quiera extraer.

• Evitar la posibilidad de acceso a los pines del microcontrolador, o de otros dispositivos presentes en la placa, sobre todo a los pines usados para las comunicaciones o para la alimentación.

8.9.2. Vulnerabilidades software

8.9.2.1. Vulnerabilidades en el Firmware de Arduino

Si se tiene acceso a la placa Arduino, es posible extraer el programa almacenado e incluso llegar a hacer algo de ingeniería inversa utilizando un programador ISP (in-system programmer) como el que se muestra en la ilustración 208.

Ilustración 208 Programador ISP
Fuente: https://vidaembebida.wordpress.com/2017/01/14/avr-grabar-el-firmware-dentro-del-microcontrolador-con-avrdude-linux-y-mac/

Podemos utilizar el programa avrdude, para obtener la información que está almacenada en la memoria del microcontrolador. Una vez que tenemos el contenido de la memoria flash se puede utilizar la herramienta exrays para obtener un lenguaje más legible. En la ilustración 209 se muestra cómo se realiza la conexión del programador a la placa Arduino Mega.

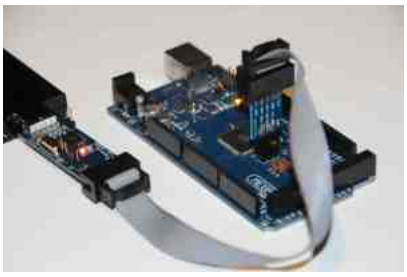


Ilustración 209 Conexión del Programador ISP al Arduino Mega
Fuente: https://petervanhoyweghen.wordpress.com/2015/12/02/the-usbasp-and-atmega2560-mystery/

Para evitar la lectura de un programa que este almacenado en la memoria de una placa Arduino, es necesario utilizar un programador ICSP (in-circuit serial programmer) para setear los bits de protección necesarios y haciendo uso de la herramienta avrdude usar el siguiente comando, que provoca el borrado de la memoria ante un intento de acceso:

avrdude -c usbtiny -p m328p -U lock:w:0xFC:m