

# Análisis de la herramienta Cylon.js



Marc Garcia Ferrer  
Alejandro Vicent Micó  
Jose Francisco Gómez Alemany

# Índice

<b>1. Introducción y definición del contexto</b>	<b>3</b>
<b>2. Objetivos del proyecto</b>	<b>4</b>
<b>3. Metodología</b>	<b>5</b>
<b>4. Descripción y análisis del proyecto realizado</b>	<b>6</b>
4.1 Alcance y restricciones	6
4.1.1 Plataformas soportadas	6
4.1.2 Drivers soportados	8
4.1.3 Restricciones	9
4.2 Consideraciones económicas del proyecto	9
4.3 Funcionamiento de Cylon.js	10
4.4 Caso práctico realizado	11
4.4.1 Hardware	12
4.4.2 Software	14
4.5 Consideraciones sobre los aspectos de seguridad y privacidad del proyecto	17
<b>5. Conclusiones</b>	<b>18</b>
<b>6. Bibliografía</b>	<b>19</b>

# 1. Introducción y definición del contexto

---

Hoy en día el término IoT o *Internet Of Things* está en boca de todos. Este concepto hace referencia a la interconexión de diferentes dispositivos a través de una red donde todos ellos pueden interactuar y comunicarse entre sí. Estos dispositivos pueden ser de todo tipo, desde pequeños sensores, hasta grandes electrodomésticos como neveras o lavadoras. Normalmente, la red que se utiliza es el propio Internet y esto ha hecho que cosas con las que soñábamos hace unos años como por ejemplo encender las luces a golpe de palma o la posibilidad de ver videos desde un reloj sea una realidad a día de hoy.

Cylon.js es un framework de Javascript que está directamente enfocado a la robótica y al IoT. Esta herramienta da soporte a un gran número de plataformas, algunas de ellas muy conocidas como Raspberry Pi o Arduino, y permite la conexión entre ellas. Sin Cylon, si quisieramos conectar varios dispositivos de este tipo tendríamos que crear un software o hardware específico que sirviera de puente para permitir la comunicación. Además, da soporte a protocolos como HTTP o MQTT para recibir y mandar datos y proporciona una API desde la que podemos controlar el hardware de manera sencilla y cómoda.

Las características que se han mencionado hacen de Cylon.js una herramienta interesante para el desarrollo e investigación en el ámbito del IoT. En este proyecto ahondaremos en las posibilidades que ofrece en este campo y estudiaremos las limitaciones que presenta la herramienta con el fin de poder justificar su uso en la creación de servicios novedosos de IoT.

## 2. Objetivos del proyecto

---

Los objetivos de este proyecto se pueden resumir en estos cuatro puntos:

- Investigar sobre el IoT y su funcionamiento.
- Aprender a utilizar el framework Cylon.js
- Analizar las posibilidades que ofrece Cylon.js
- Crear un servicio de IoT con Cylon.js a pequeña escala que pueda llevarse a un ámbito más grande.

### 3. Metodología

---

Para cumplir los objetivos establecidos en este proyecto, se han seguido diversos pasos. En el caso del primer objetivo, para formarnos sobre *Internet of Things* y cómo funciona se ha buscado en diversas fuentes de información cómo son artículos online, estudios realizados o los propios contenidos de esta asignatura.

Una vez entendido el funcionamiento de IoT y su importancia tanto en la sociedad actual como futura, se ha procedido a aprender a utilizar el framework Cylon.js y analizar las posibilidades que este ofrece. El modo de hacer esto ha sido leyendo la documentación oficial en su página web, dónde se explica qué es lo que hace y para qué se puede usar. Además, se ha alcanzado una comprensión mayor a través de la realización de varios ejemplos encontrados tanto en su web como en otras páginas. Estos ejemplos, aunque simples, nos ofrecen una experiencia práctica en la utilización de Cylon y facilitan su uso.

Por último, en lo que respecta al objetivo de crear un servicio a pequeña escala que se pueda llevar a un ámbito más grande, lo que se ha decidido hacer es una representación a pequeña escala de una vivienda con las bombillas representadas por leds y la inclusión de más representaciones de electrodomésticos con la utilización de un arduino. Además de la implementación de una web para el control de los “electrodomésticos” de la representación. Así, se observa cómo funcionaría en el caso de ser aplicado al IoT en una *smart home*.

# 4. Descripción y análisis del proyecto realizado

---

## 4.1 Alcance y restricciones

Para analizar las posibilidades que ofrece Cylon.js es importante definir qué es lo que puede hacer y con qué es compatible, pero también qué es lo que no puede hacer. Por eso, empezaremos hablando de las plataformas con las que Cylon.js es compatible, seguidamente definiremos los drivers que soporta y para finalizar este apartado hablaremos de las restricciones que presenta.

### 4.1.1 Plataformas soportadas

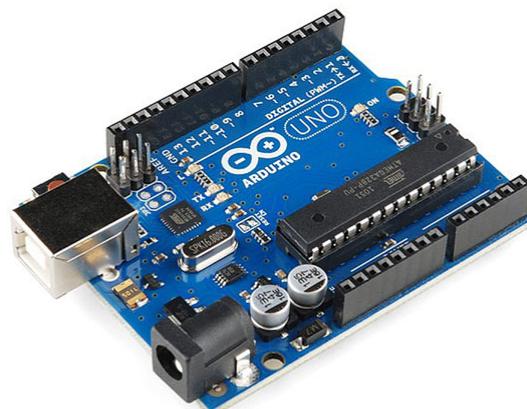
Una de las principales fortalezas de este framework es el gran abanico de plataformas a las que da soporte. Gracias a esto, las oportunidades que ofrece para la creación de servicios basados en IoT son muy amplias.

Comenzaremos viendo algunas de estas plataformas y explicando lo que puede aportar Cylon.js a cada caso.

#### Arduino

Arduino es una placa basada en un microcontrolador ATMEL. Este microcontrolador tiene una interfaz de entrada desde donde podemos conectar diversos dispositivos y una de salida para mover la información procesada en el Arduino a otros periféricos. Es uno de los líderes del mercado gracias a su flexibilidad y su facilidad de uso ya que permite realizar todo tipo de proyectos utilizando esta placa, desde diferentes juguetes electrónicos hasta sistemas precisos de detección con sensores.

Cylon.js permite realizar una conexión a los Arduinos instalando, en primer lugar, el protocolo firmata en estos últimos. Este protocolo permite la comunicación entre los microcontroladores y el software instalado en un ordenador. Es un protocolo bastante común y ya viene instalado en muchos sistemas operativos como puede ser Windows 10. Utilizando Cylon.js podemos crear la conexión definiendo por ejemplo el puerto de conexión del Arduino al ordenador y utilizando la API que nos proporciona este framework podemos realizar diferentes operaciones sobre los elementos conectados a la placa. Veremos un ejemplo de esto en el caso práctico que hemos realizado.



## **Intel Edison**

Intel Edison es la propuesta de un microcontrolador enfocado al IoT capaz de rivalizar con Arduino. Tiene un sistema Linux embebido sin embargo es capaz de ejecutar los Sketches de Arduino sin problema. También cuenta con conectividad Bluetooth y Wi-Fi.

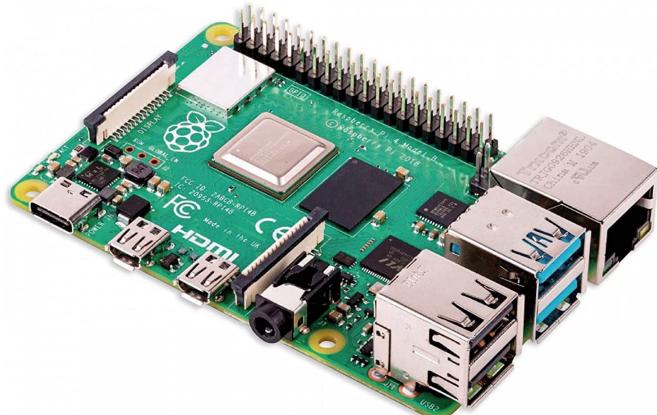
Cylon.js permite realizar actividades similares a las que podríamos realizar con un Arduino pero al tener conectividad Bluetooth y Wi-Fi es importante realizar la configuración de red pertinente para poder trabajar con esta placa sin el uso de una conexión por cable. Una vez se ha realizado esto, podemos utilizar Cylon para llevar a cabo tareas de dificultad variable, desde encender o apagar un LED, hasta la detección de colisiones.



## **Raspberry Pi**

La Raspberry Pi es una placa de microordenador, también llamados SoC (System On Chip), con componentes como son una CPU, RAM, puertos de I/O etc. Estos dispositivos han tenido mucho éxito gracias a todo lo que nos permite hacer con un precio bastante asequible. Se le pueden dar infinidad de usos como por ejemplo usar una de media center, de sistema de emulación de videoconsolas retro, como un propio ordenador con un sistema operativo Linux o como proyecto de IoT como puede ser una pequeña estación meteorológica.

Dadas las características de estos dispositivos, se debe instalar Cylon.js directamente sobre la Raspberry si se está usando un sistema operativo Linux como por ejemplo Raspbian. La instalación es algo más delicada que en los casos anteriores ya que se deben modificar varios archivos internos y dar distintos permisos en el sistema operativo para que Cylon pueda funcionar correctamente. Una vez se ha realizado esto, ya se puede usar el framework para trabajar con los componentes soportados por la Raspberry Pi.



## Speech

Cylon.js también ofrece una funcionalidad de *text-to-speech*, es decir, que si se le da un texto puede transmitirlo en voz alta. Esto tiene muchas funciones de accesibilidad y puede facilitar la vida de personas con dificultades para hablar. Si se les da un dispositivo con esta función incorporada pueden escribir el texto que quieran transmitir y el robot de Cylon.js lo expresará en voz alta para ellos.

Para usar esta funcionalidad se necesita instalar el módulo cylon-speech, además de la utilidad espeak en el propio ordenador. Una vez hecho esto ya se puede usar cylon-speech como cualquier otro robot de Cylon.js.

## MQTT

MQTT es un protocolo de red ligero basado en el modelo *publisher-subscriber* para transportar mensajes entre distintos dispositivos. Un buen uso de Cylon.js es como un cliente que se subscribe a un broker MQTT. Se pueden realizar diferentes funciones haciendo uso de la flexibilidad de plataformas que ofrece este framework. Por ejemplo se puede hacer algo similar a lo que realizamos en las prácticas de la asignatura, hacer que Cylon se suscriba a los datos meteorológicos y que cuando el valor de la temperatura sea inferior a 0 grados suene una pequeña alarma conectado a un Arduino. La posibilidad de poder realizar operaciones de este tipo abre todo un mundo de posibilidades en el mundo IoT.

### 4.1.2 Drivers soportados

Cylon.js no sería demasiado útil si diera soporte a muchas plataformas pero no soportara los elementos y dispositivos que dan vida a estas plataformas. Por suerte esto no es así y el framework tiene drivers para todo tipo de estos elementos. Por este motivo se verán los drivers más interesantes y los que hemos utilizado para realizar nuestro caso práctico.

En primer lugar, es importante destacar que la gran mayoría de los drivers de los que hemos hecho uso entran dentro de la categoría GPIO (General Purpose Input/Output). Esto hace referencia a los elementos que se conectan mediante pines de I/O. En nuestro caso, al haber trabajado con un Arduino nos ha venido especialmente bien. Cylon.js tiene la ventaja de que se trabaja igual con elementos de esta categoría sin importar el dispositivo que se use mientras estos elementos se conecten mediante estos pines, es decir, si hubiéramos

usado un Intel Edison en vez de un Arduino, el código para interactuar con los elementos como un LED sería exactamente el mismo.

Los LEDS son uno de los elementos más básicos y útiles con los que hemos trabajado. Cylon.js no ofrece una API con la que podemos acceder al LED y realizar operaciones como encender y apagar o cambiar el brillo de una manera muy sencilla. Otro elemento que hemos utilizado es el motor. Haciendo uso del pin a los 5V que ofrece el Arduino podemos conectar un motor y controlarlo desde código. No solo podemos encenderlo si no que además Cylon nos permite tener control sobre la velocidad del motor. De esta categoría un driver muy interesante es el llamado *Direct Pin*. Nos ofrece la posibilidad de mandar un valor digital al pin. Esto hace que Cylon nos permita encender y apagar dispositivos que a priori no son soportados si tienen la característica de que se enciendan con un valor de 1 en el pin y se apaguen con un valor de 0. En nuestro caso esto se ha realizado para hacer funcionar un pequeño *buzzer*. Este componente no está directamente soportado por el framework pero esta API más genérica abre muchas puertas para otros componentes similares.

Finalmente es importante recordar que Cylon ofrece APIs para realizar operaciones sobre dispositivos BLE (Bluetooth Low Energy) o dispositivos que se conectan mediante el protocolo I2C como puede ser un LCD, entre muchos otros. Otros dispositivos hardware soportados interesantes son los AR.Drone para poder controlar los drones de esta marca y los sensores Tessel que se pueden usar para recabar todo tipo de datos y son muy útiles para crear sistemas IoT de todo tipo.

#### 4.1.3 Restricciones

Respecto a las restricciones que tiene Cylon.js, es importante destacar que la última actualización que recibió este framework fue en 2016. Por eso, es posible que no sea compatible con las últimas iteraciones de algunas plataformas. Además, dado el tiempo que ha pasado desde la última actualización, no se espera que haya nuevas versiones así que la cantidad de estas incompatibilidades aumentará con el tiempo.

Además, como hemos dicho en el apartado 4.1.1, Cylon.js soporta el *Speech*, pero no soporta el reconocimiento de voz. Esto es un problema porque elimina la posibilidad de hacer un asistente de voz como pueden ser Siri de Apple o Alexa de Amazon.

Estas restricciones no son impedimentos totales ya que, como veremos más adelante, Cylon.js ofrece la posibilidad de incluir hardware propio, lo que significa que los problemas mencionados anteriormente se podrían superar, aunque fuera necesitando más trabajo del que idealmente se haría.

### 4.2 Consideraciones económicas del proyecto

A la hora de usar Cylon.js para un proyecto, hay que tener en cuenta no solo el coste de desarrollo sino además el coste de los dispositivos usados tanto para las propias funcionalidades de IoT como para el control de estos.

Para el caso práctico que explicaremos más adelante, al ser una representación, es suficiente con un Arduino y algunos componentes sencillos con un precio total menor a 50€. Sin embargo, si se escalara este proyecto a una vivienda real añadiendo más dispositivos de IoT, considerando todos los componentes necesarios, el precio ascendería a un valor entre 5000 y 10000€.

## 4.3 Funcionamiento de Cylon.js

Cylon.js funciona mediante el uso de robots. A la hora de hacer un programa de Cylon se crea un robot con conexiones a una plataforma y a dispositivos. Una vez hecho esto se crean métodos que controlan el funcionamiento de los dispositivos en situaciones concretas. Por ejemplo, este es un ejemplo con la conexión de un LED y un botón a un Arduino:

```
var Cylon = require('cylon');

Cylon.robot({
  connections: {
    arduino: { adaptor: 'firmata', port: 'COM5' }
  },

  devices: {
    led: { driver: 'led', pin: 13 },
    button: { driver: 'button', pin: 2 }
  },

  work: function(my) {
    my.button.on('push', function() {
      my.led.toggle()
    });
  }
}).start();
```

Como se puede observar en la foto, la función hace que el LED se encienda si está apagada cuando se pulsa el botón y viceversa. Además, se pueden conectar varios Arduinos para controlar distintos dispositivos. El ejemplo anterior utilizando dos Arduinos sería así:

```
var Cylon = require('cylon');

Cylon.robot({
  connections: {
    arduino_A: { adaptor: 'firmata', port: 'COM4' },
    arduino_B: { adaptor: 'firmata', port: 'COM5' }
  },

  devices: {
    led: { driver: 'led', pin: 13, connection: "arduino_A" },
    button: { driver: 'button', pin: 2, connection: "arduino_B" }
  },

  work: function(my) {
    my.button.on('push', function() {
      my.led.toggle()
    });
  }
}).start();
```

De esta forma, esto funciona de la misma forma que el ejemplo anterior pero el primer Arduino controla el led mientras que el segundo controla el botón. Cylon.js ofrece muchas funcionalidades aparte de controlar LEDs y botones, como se ha explicado en los apartados anteriores. Un ejemplo sería la utilización de Cylon.js para manejar drones:

```
var Cylon = require('cylon');

Cylon.robot({
  connections: {
    ardrone: { adaptor: 'ardrone', port: '192.168.1.1' }
  },

  devices: {
    drone: { driver: 'ardrone' }
  },

  work: function(my) {
    my.drone.takeoff();

    after((10).seconds(), function() {
      my.drone.land();
    });

    after((15).seconds(), function() {
      my.drone.stop();
    });
  }
}).start();
```

Con este programa, el dron conectado a Cylon despegue, se mantiene en el aire durante 10 segundos, aterriza y después de 15 segundos en el suelo se detiene del todo. Así, vemos que aún conectando dispositivos más complejos como un dron, Cylon.js es un framework intuitivo, simple y fácil de utilizar. Por esto, es un buen framework para ser utilizado en muchas funciones diversas manteniendo la simplicidad y sin volverse excesivamente complicado.

## 4.4 Caso práctico realizado

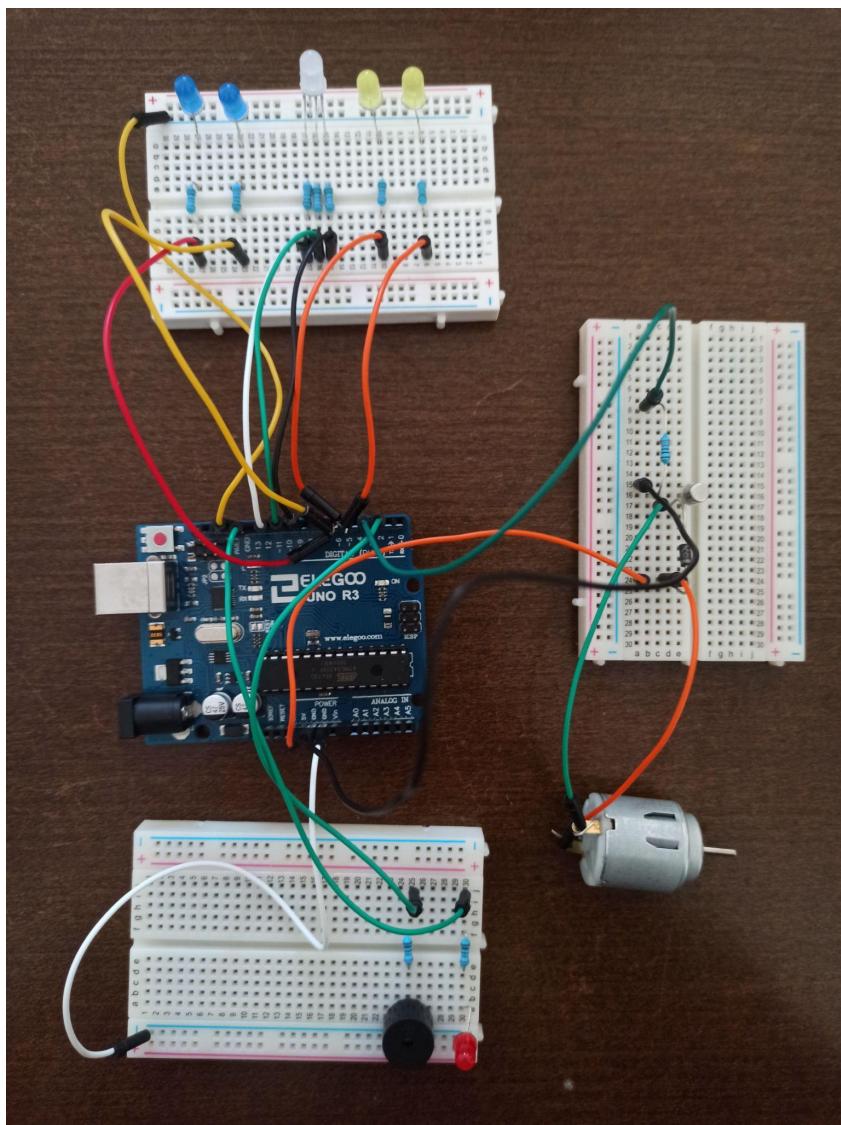
Después de estudiar las posibilidades ofertadas por Cylon.js y de probarlas con algunos ejemplos, se ha decidido realizar un proyecto que muestre el funcionamiento de este framework en un caso real. Para ello se ha construido con una caja una casa que simbolice una casa real, pero a pequeña escala. Utilizando un Arduino, se ha montado sobre esta casa un circuito con diversos componentes, tales como LEDs, un motor o un *buzzer*. Además, se ha desarrollado una aplicación web con la que controlar los dispositivos de la casa mediante una interfaz intuitiva y fácil de entender.

#### 4.4.1 Hardware

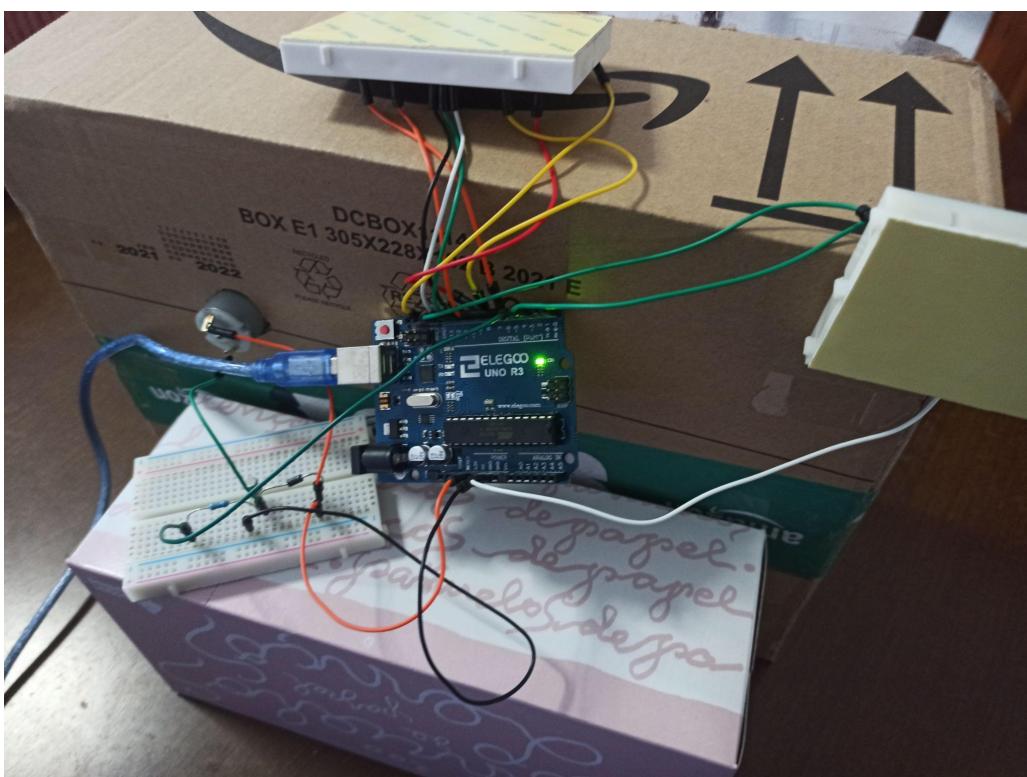
Como podemos observar en la imagen inferior, el circuito está formado por un Arduino y tres *breadboards* conectadas a este mediante una serie de cables. En la placa superior se han conectado cinco LEDs con sus respectivas resistencias, que simularán la iluminación de la casa a pequeña escala. Las dos LEDs azules de la izquierda están conectadas al pin 6 del Arduino, las LEDs amarillas de la derecha están conectadas al pin 5 y la LED RGB del centro tiene las patas conectadas a los pins 9, 10 y 11 para los colores azul, verde y rojo, respectivamente.

La placa de la derecha se ha utilizado para conectar un motor, que simulará el ventilador de la casa. El motor necesita una resistencia, un diodo (de color gris en la foto) y un transistor (de color negro), y está conectado al pin 3 del arduino.

La placa inferior tiene un buzzer que emite sonido y una LED roja, con sus resistencias, que se enciende cuando el buzzer está activado. El buzzer simula un altavoz de la casa para poner música y está conectado al pin 2 del arduino, mientras que la LED roja se conecta al pin 13.



A continuación se puede observar el interior de la casa, con los componentes ya instalados. Arriba a la izquierda, con el círculo rojo, están el buzzer y su LED roja. En el centro del techo, con el círculo verde, están las LEDs de la iluminación y al fondo a la derecha, con el círculo azul, está el motor del ventilador. La siguiente imagen muestra la parte posterior de la casa, donde se puede observar cómo se ha instalado el circuito y los diferentes componentes.



#### 4.4.2 Software

Para poder controlar todos estos componentes, se ha desarrollado una aplicación web utilizando React. La pantalla principal de la aplicación tiene una serie de botones que se pueden pulsar para encender y apagar los dispositivos de la casa. Si el dispositivo está encendido, el botón estará en verde, y el botón en rojo significará que está apagado. Algunos componentes tienen funciones extra además de encender y apagar, que se mostrarán una vez esté encendido dicho componente. Finalmente, la etiqueta de la parte inferior proporcionará *feedback* al usuario de las acciones que se están realizando.



La aplicación se divide en dos partes, el front-end y el back-end. El front-end está situado en la carpeta cylon-rse, está desarrollado con React (Javascript y css) y es el encargado de mostrar al usuario la aplicación, así como de obtener los inputs de este y enseñarle los outputs. El back-end está situado en la carpeta cylon-server y es el encargado de comunicarse con el arduino para realizar acciones sobre los dispositivos de la casa. La conexión entre ambas partes se lleva a cabo mediante una serie de llamadas http. En el back-end, se crea un robot con Cylon.robot(), y se le asignan los dispositivos conectados, especificando su tipo y el puerto al que están conectados. Una vez creado el robot, se inicializa con start().

```
var robot = Cylon.robot({
  connections: {
    arduino: { adaptor: "firmata", port: "COM5" }
  },

  devices : {
    blueLeds: { driver: "led", pin: 6},
    yellowLeds: { driver: "led", pin: 5},
    multicolorRedLed: { driver: 'led', pin: 11 },
    multicolorGreenLed: { driver: 'led', pin: 10 },
    multicolorBlueLed: { driver: 'led', pin: 9 },
    buzzer: { driver: "direct-pin", pin: 2},
    buzzerLed: { driver: "led", pin: 13},
    motor: { driver: "motor", pin: 3},
  },
}).start();
```

A continuación, se va a explicar el funcionamiento de los dispositivos disponibles y cuáles son las acciones que podemos realizar con ellos:

### **Luces azules y luces amarillas**

El funcionamiento de las luces azules y las amarillas es el mismo, por lo que se va a explicar solo para las azules. Cuando presionamos el botón de luces azules, se comprueba en el estado de la aplicación si estaban encendidas o no. En caso afirmativo, se llama al back-end con un fetch a *blueLightsOff* para que se apaguen. En caso de que estuviesen apagadas, se hará un fetch a *blueLightsOn* para encenderlas.

En el back-end, router.get("/blueLightsOn") recibirá la llamada del front-end y encenderá las leds con el método de Cylon turnOn(). Establecerá la luminosidad a un valor predeterminado de 125 y responderá al front-end con "Luces azules encendidas". Esta string se mostrará en pantalla principal mediante la etiqueta explicada anteriormente. Por otro lado, router.get("/blueLightsOff") apagará las luces con turnOff().

Cuando las luces estén encendidas, el usuario podrá subir o bajar su intensidad en un rango de [0, 9] con los botones + y - que aparecerán en pantalla. Estos botones llamarán a /blueLightsRaiseIntensity y /blueLightsLowerIntensity, que utilizarán el método brightness() para modificar la luminosidad de las LEDs entre [0, 225], siendo una unidad de intensidad del front-end (entre 0 y 9) equivalente a 25 en el back-end.

Front-end	Back-end
<pre>const blueLightsHandler = () =&gt; {   if(!blueLightsOn) {     setBlueLightsIntensity(5)     fetch("http://localhost:9000/cylonRoute/blueLightsOn")     .then((res) =&gt; res.text())     .then((res) =&gt; setApiResponse(res))   } else {     setBlueLightsIntensity(0)     fetch("http://localhost:9000/cylonRoute/blueLightsOff")     .then((res) =&gt; res.text())     .then((res) =&gt; setApiResponse(res))   }   setBlueLightsOn(!blueLightsOn) }  const blueLightsRaiseIntensity = () =&gt; {   setBlueLightsIntensity(blueLightsIntensity === 9 ? 0 : blueLightsIntensity + 1)   fetch("http://localhost:9000/cylonRoute/blueLightsRaiseIntensity")   .then((res) =&gt; res.text())   .then((res) =&gt; setApiResponse(res)) }  const blueLightsLowerIntensity = () =&gt; {   setBlueLightsIntensity(blueLightsIntensity === 0 ? 9 : blueLightsIntensity - 1)   fetch("http://localhost:9000/cylonRoute/blueLightsLowerIntensity")   .then((res) =&gt; res.text())   .then((res) =&gt; setApiResponse(res)) }</pre>	<pre>router.get("/blueLightsOn", function (req, res, next) {   robot.devices.blueLeds.turnOn();   robot.devices.blueLeds.brightness(125);   res.send("Luces azules encendidas"); });  router.get("/blueLightsOff", function (req, res, next) {   robot.devices.blueLeds.turnOff();   res.send("Luces azules apagadas"); });  router.get("/blueLightsRaiseIntensity", function (req, res, next) {   robot.devices.blueLeds.brightness(     robot.devices.blueLeds.currentBrightness() === 225     ? 0     : robot.devices.blueLeds.currentBrightness() + 25   );   res.send("Intensidad azul aumentada"); });  router.get("/blueLightsLowerIntensity", function (req, res, next) {   robot.devices.blueLeds.brightness(     robot.devices.blueLeds.currentBrightness() === 0     ? 225     : robot.devices.blueLeds.currentBrightness() - 25   );   res.send("Intensidad azul disminuida"); });</pre>

### **Luz multicolor**

La LED RGB ofrece la posibilidad de escoger el color de la luz. Cuando se presiona el botón desde el front-end, comprueba el estado de la luz. Si estaba apagada, llama a /multicolorLightOn, que llama al método correspondiente del back-end. Se lleva a cabo un loop que asigna, cada segundo y mediante el método brightness, un valor aleatorio entre 0 y 255 a la intensidad de las patas roja, verde y azul del LED RGB. De esta forma, mientras la

LED esté encendida, tendrá un color diferente cada segundo, resultado de la mezcla de luz entre las tres patas.

En caso de que la luz estuviese ya encendida, si se pulsa el botón se llamará a /multicolorLightOff, que apagará la LED utilizando el método turnOff() sobre sus tres patas. Finalmente, enviará un mensaje al front-end para que se muestre en pantalla que la luz multicolor se ha apagado.

<b>Front-end</b> <pre>const multicolorLightHandler = () =&gt; {   if(!multicolorLightOn) {     fetch("http://localhost:9000/cylonRoute/multicolorLightOn")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   } else {     fetch("http://localhost:9000/cylonRoute/multicolorLightOff")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   }   setMulticolorLightOn(!multicolorLightOn) }</pre>	<b>Back-end</b> <pre>var multicolorLightOn = false;  router.get("/multicolorLightOn", function (req, res, next) {   multicolorLightOn = true;   every((1).second(), function() {     if(multicolorLightOn) {       robot.devices.multicolorRedLed.brightness(Math.random() * 255)       robot.devices.multicolorGreenLed.brightness(Math.random() * 255)       robot.devices.multicolorBlueLed.brightness(Math.random() * 255)     }   });   res.send("Luz multicolor encendida"); });  router.get("/multicolorLightOff", function (req, res, next) {   multicolorLightOn = false;   robot.devices.multicolorRedLed.turnOff()   robot.devices.multicolorGreenLed.turnOff()   robot.devices.multicolorBlueLed.turnOff()   res.send("Luz multicolor apagada"); });</pre>
--	---

## Música

Para simular un altavoz que reproduzca música en casa se ha utilizado un *buzzer*. Desde el front-end comprobamos si está encendido o no. En caso de que no, se llama a /buzzerOn, que utilizará el método digitalWrite(1) para encender el *buzzer*, además de encender la LED que tiene asociada con turnOn(). Devolverá la string “Música encendida” al front-end, que se mostrará por pantalla.

En caso de que esté encendido, se llamará a /buzzerOff, que apagará el dispositivo utilizando digitalWrite(0) y la LED con turnOff(). Comunicará que el proceso se ha ejecutado con éxito devolviendo “Música apagada”.

<b>Front-end</b> <pre>const buzzerHandler = () =&gt; {   if(!buzzerOn) {     fetch("http://localhost:9000/cylonRoute/buzzerOn")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   } else {     fetch("http://localhost:9000/cylonRoute/buzzerOff")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   }   setBuzzerOn(!buzzerOn) }</pre>	<b>Back-end</b> <pre>router.get("/buzzerOn", function (req, res, next) {   robot.devices.buzzer.digitalWrite(1);   robot.devices.buzzerLed.turnOn();   res.send("Música encendida"); });  router.get("/buzzerOff", function (req, res, next) {   robot.devices.buzzer.digitalWrite(0);   robot.devices.buzzerLed.turnOff();   res.send("Música apagada"); });</pre>
--	--

## Motor

Finalmente, se simulará un ventilador real mediante el uso de un motor. Si el motor estaba apagado, al pulsar el botón se llamará a /motorOn, que encenderá el motor con el método turnOn() y enviará al front-end el mensaje “Motor encendido”. En caso de que estuviese

encendido, se llamará a /motorOff, que apagará el motor con turnOff() y devolverá al front-end el mensaje “Motor apagado”.

Front-end	Back-end
<pre>const motorHandler = () =&gt; {   if(!motorOn) {     fetch("http://localhost:9000/cylonRoute/motorOn")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   } else {     fetch("http://localhost:9000/cylonRoute/motorOff")       .then((res) =&gt; res.text())       .then((res) =&gt; setApiResponse(res))   }   setMotorOn(!motorOn) }</pre>	<pre>router.get("/motorOn", function (req, res, next) {   robot.devices.motor.turnOn();   res.send("Motor encendido"); });  router.get("/motorOff", function (req, res, next) {   robot.devices.motor.turnOff();   res.send("Motor apagado"); });</pre>

## 4.5 Consideraciones sobre los aspectos de seguridad y privacidad del proyecto

IoT nos abre un mundo donde todos los elementos que nos rodean pueden comunicarse entre sí. El problema es que al estar conectado a todo, existen muchas más maneras de acceder a tu información. Por este motivo, aunque el IoT resulte apasionante es importante destacar su lado más oscuro, las posibles fallas de seguridad que puede provocar.

En el caso específico del proyecto Cylon.js, en dos versiones anteriores se encontraron múltiples vulnerabilidades que podrían haber causado mucho daño si se hubiera llevado a cabo un proyecto serio haciendo uso de este framework. Sin embargo, actualmente no se ha detectado ninguna vulnerabilidad en su última versión lo cual es una buena noticia para cualquier interesado en realizar algún tipo de proyecto con esta herramienta. Es importante recalcar que, aunque no se hayan detectado vulnerabilidades, la última versión data del año 2016 y es posible que amenazas recientes puedan afectar a este proyecto.

Version	0.20.2	0.21.2	0.22.2	1.2.0	1.3.0 04/2016
Release Date	11/2014	12/2014	04/2015	09/2015	
Direct Vulnerabilities	<span>O</span> <span>C</span>				
	<span>O</span> <span>H</span>				
	<span>O</span> <span>M</span>				
	<span>O</span> <span>L</span>				
Indirect Vulnerabilities	<span>O</span> <span>C</span>				
	<span>3</span> <span>H</span>	<span>3</span> <span>H</span>	<span>O</span> <span>H</span>	<span>O</span> <span>H</span>	<span>O</span> <span>H</span>
	<span>2</span> <span>M</span>	<span>2</span> <span>M</span>	<span>O</span> <span>M</span>	<span>O</span> <span>M</span>	<span>O</span> <span>M</span>
	<span>2</span> <span>L</span>	<span>2</span> <span>L</span>	<span>O</span> <span>L</span>	<span>O</span> <span>L</span>	<span>O</span> <span>L</span>
License Risk	<span>O</span> <span>H</span>				
	<span>O</span> <span>M</span>				
	<span>O</span> <span>L</span>				

## 5. Conclusiones

---

Como conclusiones de este proyecto, destacamos que Cylon.js es un framework que, siendo fácil de entender y de utilizar, ofrece resultados muy satisfactorios que se pueden ver a simple vista.

Para investigar Cylon.js hemos tenido disponibles muchos recursos. Tanto la propia web del proyecto, en la que se encontraba una gran cantidad de información explicada de forma clara, como otras páginas web que contenían mucha información sobre Cylon.js. Además, la web contiene también ejemplos prácticos que nos han ayudado a entender el funcionamiento de Cylon.js para poder aplicarlo a nuestro caso práctico.

Una vez entendidos los básicos de Cylon.js, consideramos que el caso práctico que hemos realizado ayuda a comprender su funcionamiento real y los posibles usos que se le puede dar para crear sistemas IoT a mayor escala

Como conclusión final, Cylon.js es un framework que, pese a todo lo que hemos dicho, es susceptible a vulnerabilidades de seguridad, como la mayoría de sistemas relacionados con IoT. Esto es difícilmente evitable pero a pesar de ello Cylon.js es un framework muy sólido que vale la pena usar en una gran variedad de proyectos.

## 6. Bibliografía

---

- [https://cylonjs.com/documentation/guides/what\\_is\\_cylon/](https://cylonjs.com/documentation/guides/what_is_cylon/)
- [https://programacion.net/noticia/cylon\\_js\\_-\\_una\\_forma\\_sencilla\\_de\\_programar\\_hardware\\_2178](https://programacion.net/noticia/cylon_js_-_una_forma_sencilla_de_programar_hardware_2178)
- <https://www.sitepoint.com/javascript-beyond-web-2014/>
- Lista de reproducción Cylon.js - Dev Mashup  
[https://www.youtube.com/playlist?list=PL9z6q1fnu6aBi\\_OKKbEfxtiJulXoyW0Yt](https://www.youtube.com/playlist?list=PL9z6q1fnu6aBi_OKKbEfxtiJulXoyW0Yt)
- <http://www.codefoster.com/edison-coding/>
- <https://blog.leapmotion.com/integrate-leap-motion-arduino-raspberry-pi/>