INTERNET DE LAS COSAS

PROYECTO MOIS & MARIO KARTS FOR IOT

MARIO LOSANTOS ALBACETE 04847925-P
mla4@um.es
MOISÉS FRUTOS PLAZA 48488132-S
moises.frutos@um.es

MÁSTER EN BIG DATA

ENTREGADO EL 4 DE MAYO DE 2017

Índice

Introducción

El objetivo de este proyecto es dar aplicación práctica, en un proyecto de algo más de envergadura, que el realizado en las prácticas de clase, con los conocimientos adquiridos en las mismas, e intentando, en la medida de lo posible, ir un poco más allá y proponer nuestras propias aportaciones e ideas.

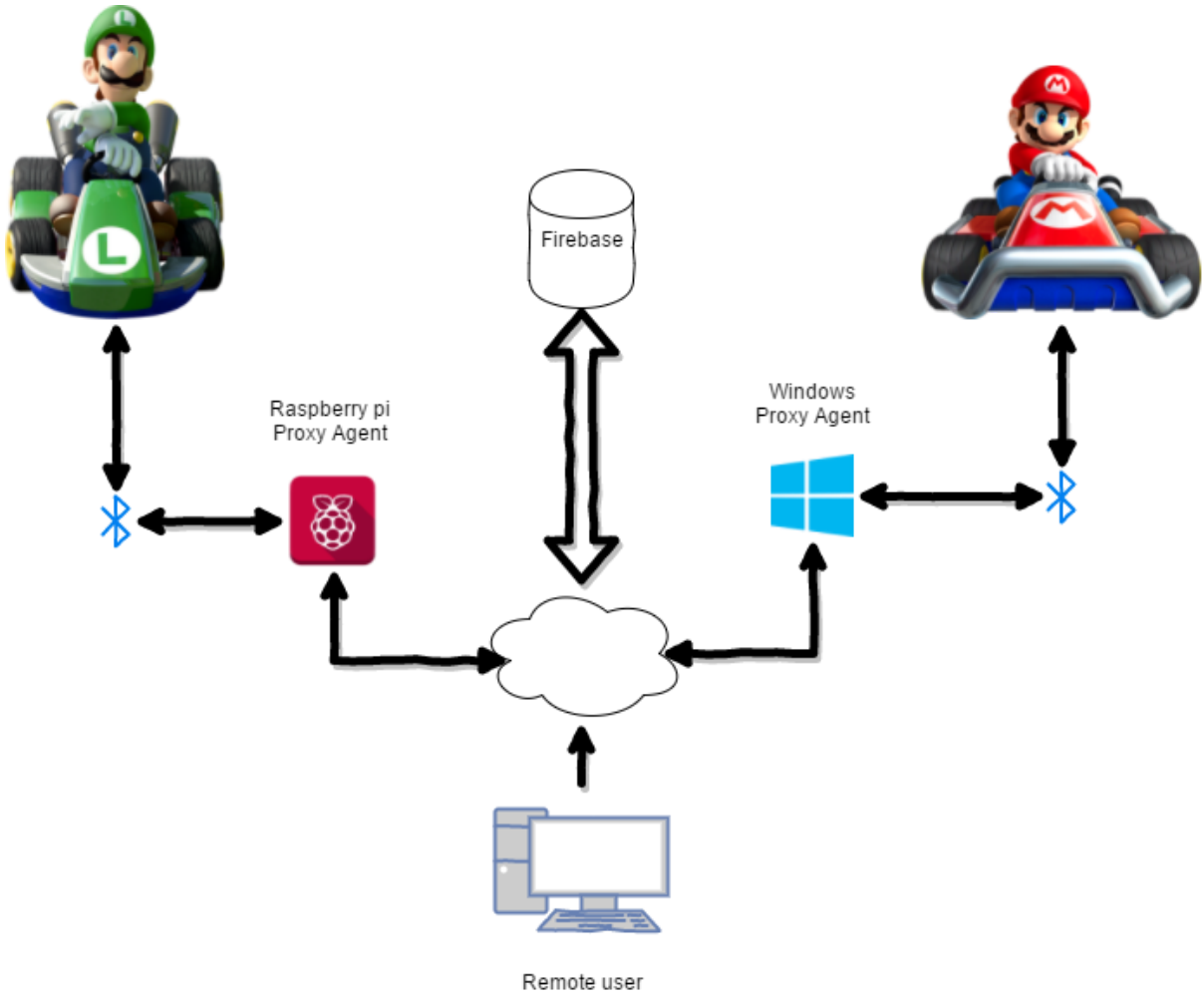
Inspirados en el famoso videojuego de Nintendo, Mario Kart, hemos querido hacer una versión IoT con nuestros coches robot basados en Arduino. El propósito es recopilar datos de los mismos (marcas de tiempo y estados), y al mismo tiempo darles órdenes desde un Frontend final basado en web, y un Backend basado en Node.js. Todo ello pasando, por supuesto, por una plataforma IoT en la nube.

Para explicar el proyecto con el mayor rigor posible, dentro del tiempo que tenemos, hemos decidido dividir esta memoria en cinco apartados principales:

I. Arquitectura propuesta
II. Montaje del hardware de los equipos (karts)
III. Enlace entre los equipos (karts) y los proxy agent
IV. Enlace entre el Proxy Agent y la plataforma de IoT en la nube
V. Extracción y visualización de datos y emisión de órdenes
VI. Posibles mejoras

I. Arquitectura propuesta

A continuación explicaremos brevemente la arquitectura propuesta.



El diagrama muestra la arquitectura desarrollada. Podemos diferenciar claramente ambos karts que se comunican mediante un protocolo de bajo consumo y rendimiento como es bluetooth. La conexión a la plataforma de almacenamiento y procesamiento en la nube corre a cargo de una raspberry pi y de un equipo windows, ambos corriendo una aplicación desarrollada en nodejs. El mecanismo normal de funcionamiento consiste en que un usuario remoto se conecte a una página web y desde allí planifique qué ruta quiere que ejecute cada vehículo, le mande ejecutarla y vea las telemetrías emitidas por éste.

II. Montaje del hardware de los equipos (karts)

Introducción

Ambos equipos, a partir de ahora karts, son del mismo modelo y fabricante (Smart Robot Car Kit V2.0 de Elegoo[1]) por lo que la explicación del montaje de uno de ellos sirve para ambos ya que son idénticos. Y en eso nos vamos a centrar en este apartado en explicar por pasos el montaje de uno de los karts.

Cada kart está compuesto por:

- un microcontrolador Arduino UNO R3 compatible.
- un shield encargado de controlar los distintos sensores o actuadores.
- una placa encargada de suministrar y distribuir la energía a los motores.
- un módulo bluetooth para comunicaciones.
- un sensor de ultrasonidos para detectar obstáculos (por determinar su uso).
- un servomotor para hacer girar el detector de ultrasonidos (por determinar su uso).
- un detector de IR (Infrarrojos) (no se va a usar).
- tres sensores de tracción y detección de línea (no se va a usar).
- un compartimento para las baterías de alimentación.
- dos baterías de alimentación tanto para la placa como para las ruedas.
- dos placas acrílicas para chasis del coche.
- cuatro motores tractores.
- cuatro ruedas para el movimiento del coche..

Pasos del Montaje

1. Identificar cual es la placa de chasis inferior y cual la superior, para empezar el ensamblaje del robot desde la base y no al revés. Esto puede parecer trivial, pero no lo es, ya que es muy importante para un correcto ensamblaje del kart empezar de abajo hacia arriba.

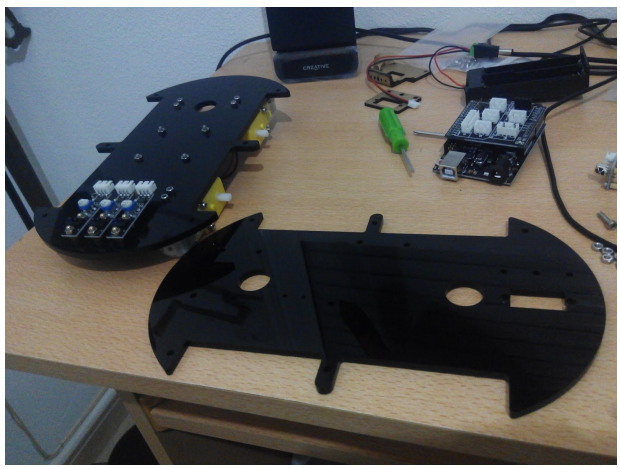


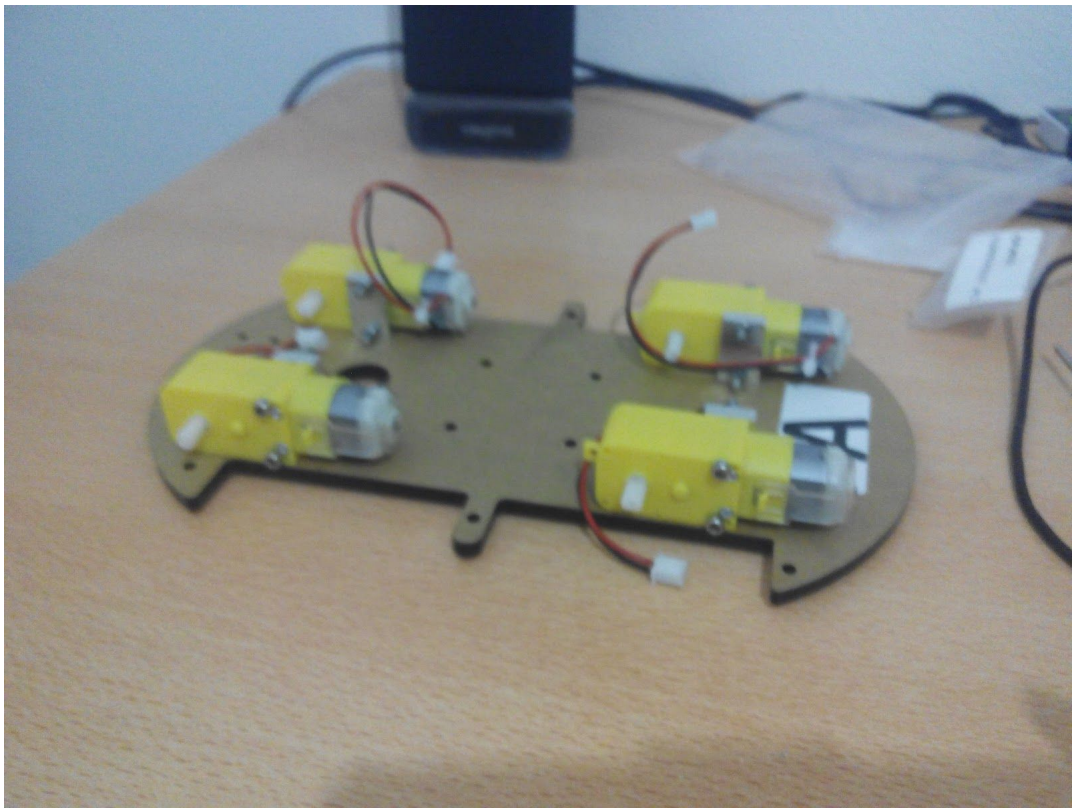
Foto de los dos chasis: el chasis inferior es el de arriba en la foto y el chasis superior el de abajo en la foto

[1] https://www.elegoo.com/product/elegoo-uno-project-upgraded-smart-robot-car-kit-v2-0/

2. Una vez identificada la placa de chasis inferior, se montan los cuatro motores, se acoplan al chasis, y se acopla también su placa encargada de suministrarles y distribuirles energía.

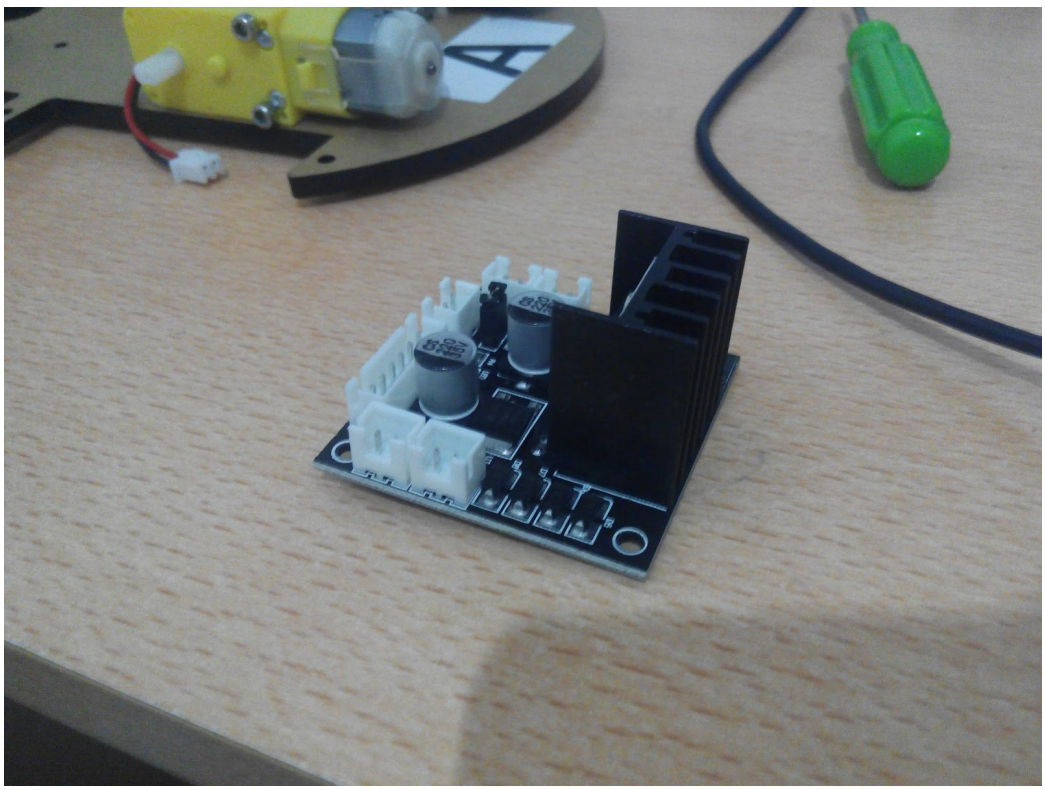
2.1: Los cuatro motores y su tornillería

2.2: Montaje de los motores

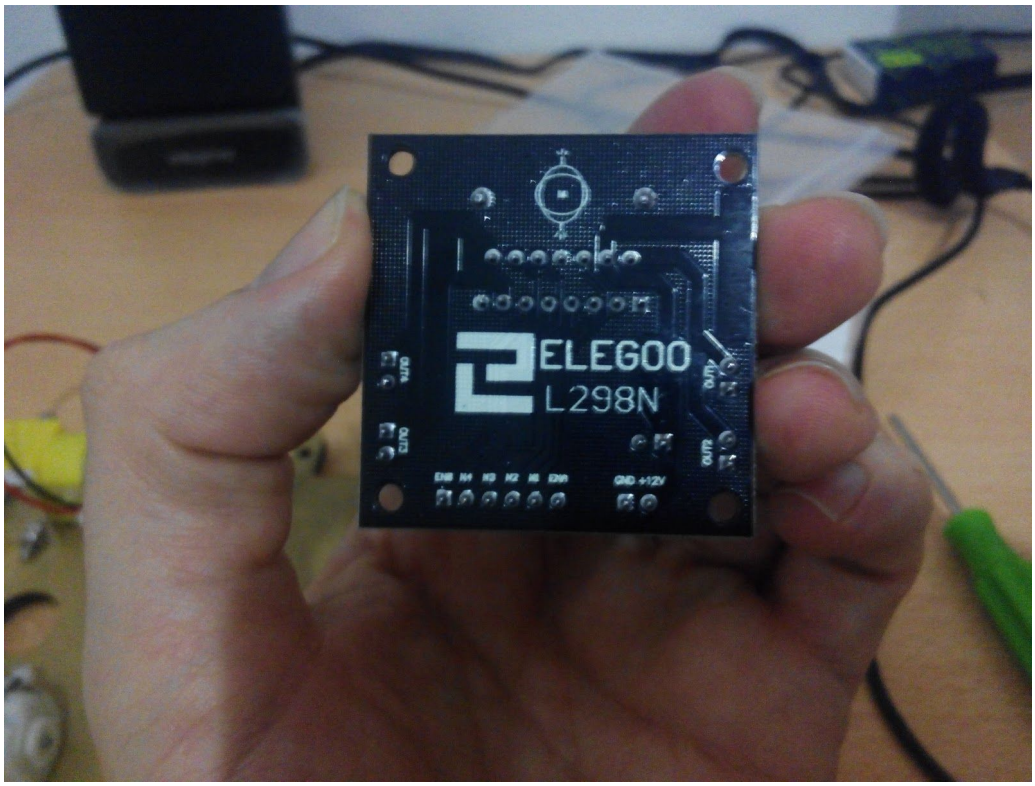


2.3: Acomplamiento de los motores al chasis

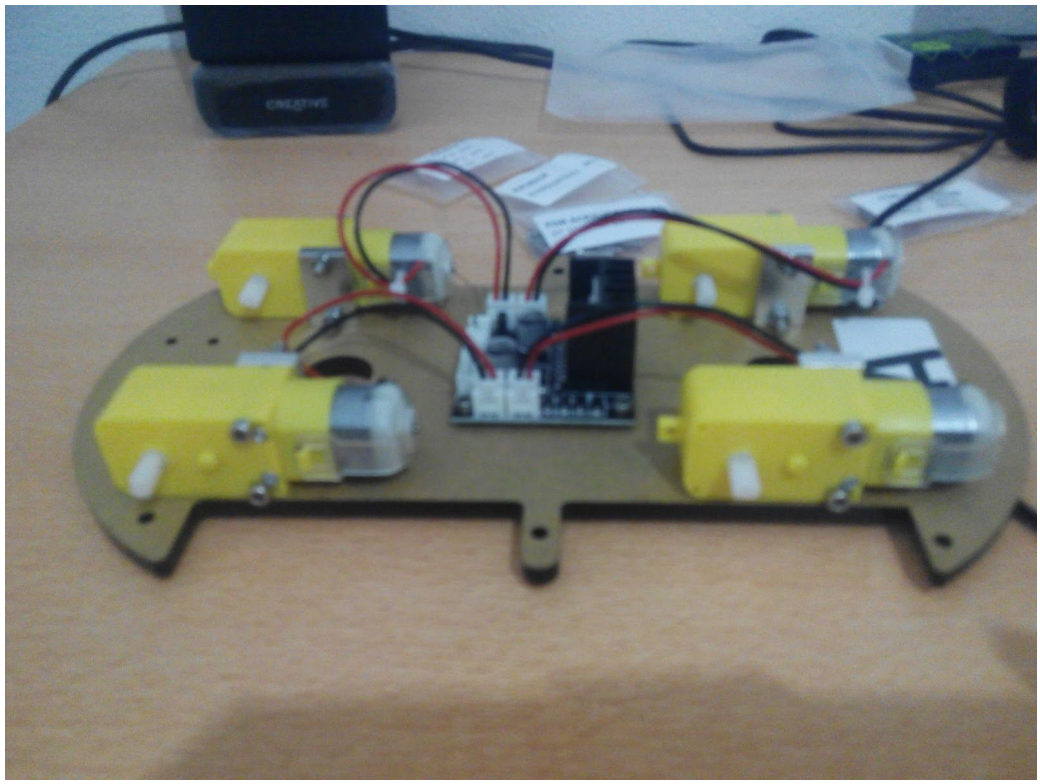
***Fe de errata**: Nos equivocamos en la dirección de acomplamiento de los dos motores traseros, pero lo corregimos después poniéndolos en la dirección correcta.*



2.4: Parte superior de la placa controladora de los motores

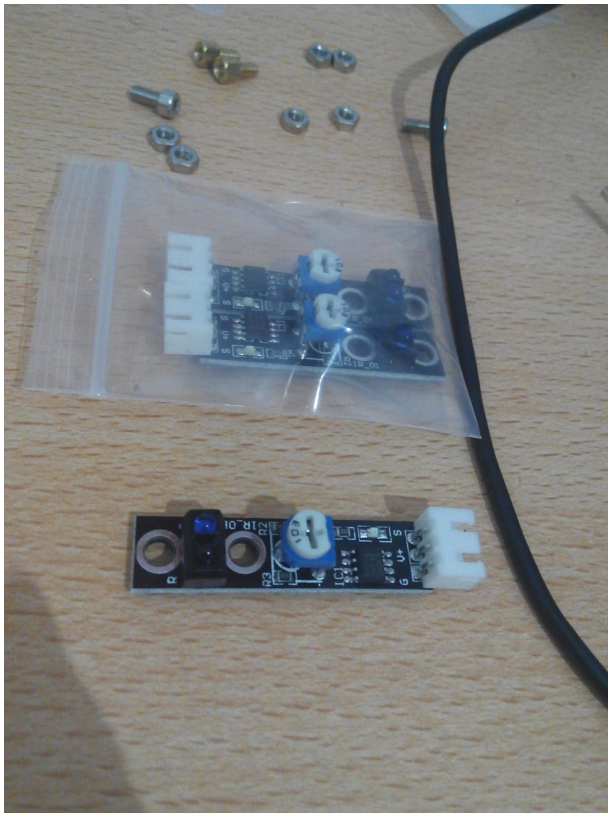


2.5: Para inferior de la placa controladora de los motores



2.6: Resultado después del acoplamiento de los 4 motores y la placa controladora de los mismos

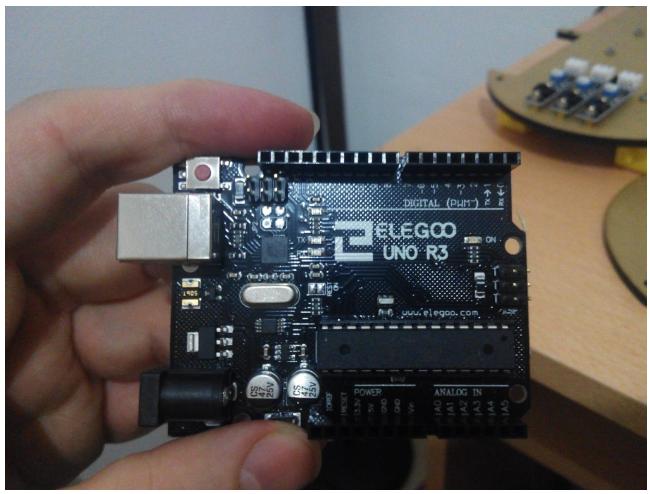
3. Colocación de los sensores de tracción y detección de línea en el chasis inferior.



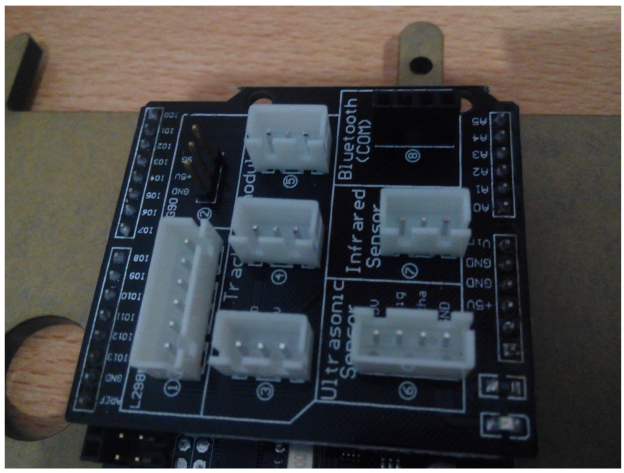
3.1: Sensores de tracción y detección de línea.

3.2: Acoplamiento de los mismos al chasis inferior

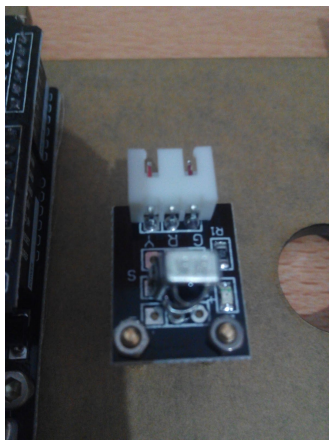
4. Acoplamiento de la placa Arduino UNO R3 en el chasis superior, su shield para controlar los sensores y actuadores del robot, el sensor de infrarrojos, el módulo de comunicaciones bluetooth, el servomotor y su sensor de ultrasonidos en la parte delantera, y el compartimento de baterías y la inclusión de las mismas en las parte trasera.



4.1: Placa Arduino UNO R3 compatible.

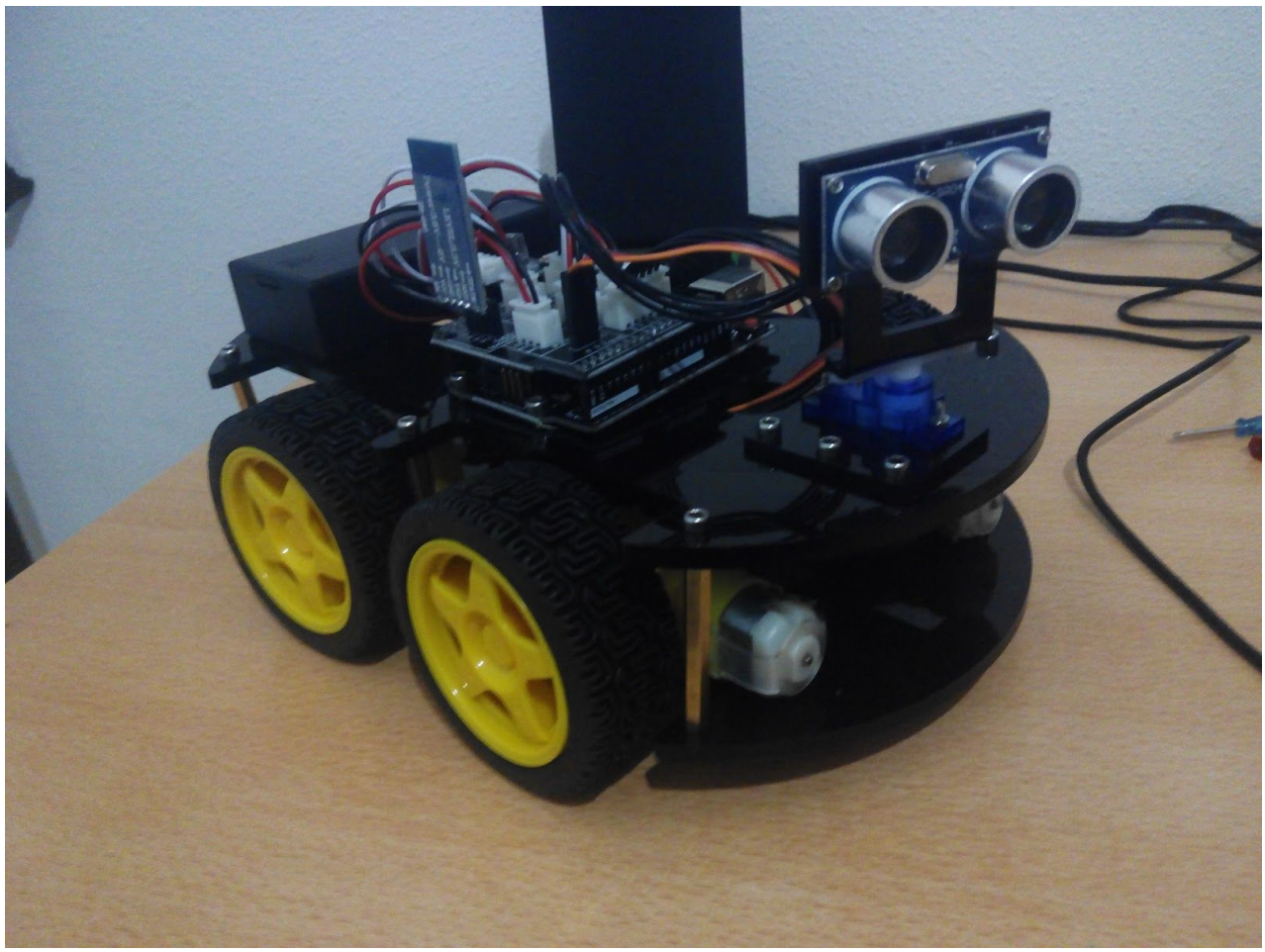


4.2: Shield controlador de sensores y actuadores del robot



4.3: Detector de IR

5. El último paso es conectar todos los cables en su sitio, acoplar las dos placas chasis de modo que la estructura del kart quede completamente ensamblada, y por último colocar las cuatro ruedas.



Resultado final del montaje del kart

III. Enlace entre los equipos (karts) y los proxy agent

Introducción

El objetivo en este segundo apartado del proyecto es enlazar vía Bluetooth los karts con un *Proxy Agent*, con más potencia de cálculo y de procesamiento que los microcontroladores Arduinos, pero que tampoco consumiera mucha energía, por lo que pensamos en la Raspberry Pi 3 como mejor opción, ya que su consumo es inferior a los 10Wh aproximadamente[2], que en comparación con un portátil, por ejemplo, que consume 30Wh, estamos disminuyendo el consumo energético a un tercio, y aún cumple perfectamente con su función de recopilar los datos y enviarlos a la plataforma IoT en la nube.

La función del *proxy agent* es, por tanto, la de suministrar el conjunto de órdenes a ejecutar por parte de los kart y de recopilar y tratar la información recibida de estos, así como enviarla a la nube.

Código a cargar en el microcontrolador arduino

El código de la aplicación que se ejecuta en cada kart es distinto para cada uno de ellos y ambos se encuentran disponibles en el repositorio https://github.com/Loksly/cartrack

El código está basado en parte en el código de ejemplo que venía con el kit adquirido[3], aunque mejorado en varios aspectos y adaptado para emitir la telemetría objeto del proyecto. En los siguientes apartados podemos ver la codificación de esa telemetría[4] y en qué ha consistido finalmente. Según qué versión se consulte se podrá ver modificaciones en la forma en la que se interpreta el conjunto de órdenes recibidas por el puerto serie o cómo se ha empaquetado el código relativo al control del vehículo en un fichero separado.

El código que se ejecuta en el kart asignado a mario está en el directorio con ese nombre y del mismo modo existe un directorio con el nombre de luigi.

[2] Se alimenta con un transformador de corriente continua que suministra 2 amperios a 5 voltios.
[3] Disponible en: http://bit.ly/2pLSFst
[4] Por ejemplo: https://github.com/Loksly/cartrack/blob/master/mario/bluetooth_car.ino#L25

Aplicación proxy agent desarrollada

La aplicación a ejecutar en el hardware que actúa de proxy ha sido implementada en nodejs. Se encuentra publicada en *github* en https://github.com/Loksly/fire-serial-proxy junto con las instrucciones necesarias para su instalación tanto en windows como en raspberry pi. Para su ejecución es necesaria una versión reciente de *nodejs*.

El funcionamiento de la aplicación es relativamente sencillo. Una vez que la conexión bluetooth está establecida a nivel de sistema operativo como puerto COM, éste es utilizado de manera bidireccional para enviar el listado de órdenes a ejecutar por parte del kart y recibir su telemetría y reenviarla a la plataforma de IoT en la nube.

Protocolo de comunicación entre Arduino y el proxy agent

Arduino recibe sus órdenes en texto plano, codificando cada letra una orden. El vocabulario de órdenes que recibe es el siguiente:

‘A’: Conmutar el estado del LED.
‘f’: Avanzar (inicial de forward).
‘b’: Retroceder (inicial de backward).
‘s’: Parar (inicial de stop).
‘l’: Rotar en el sentido antihorario (inicial de left).
‘r’: Rotar en el sentido horario (inicial de right).
‘z’: Rotar a la izquierda el sensor de ultrasonidos.
‘x’: Rotar a la derecha el sensor de ultrasonidos.

Arduino codifica su telemetría en JSON y la envía al proxy agent para que la trate, comprima o lo que considere necesario previo a su envío a la nube.

En este caso tenemos como propuestas de telemetría este formato codificado en JSON:

```
{
  “c”: [último comando ejecutado],
  “d”: [distancia según el sensor de ultrasonidos],
  “l”: [estado del LED],
  “s”: [ángulo del servomotor del sensor de ultrasonidos],
  “t”: [marca de tiempo[5]]
}
```

[5] Añadida por el proxy agent posteriormente.

Este formato es un ejemplo para demostrar que funciona el envío de información desde el arduino hasta la nube y, posteriormente hasta el usuario remoto, pero podría enviarse cualquier lectura de un sensor disponible en el arduino o el estado interno de cualquier variable del programa.

Enlace entre el kart de Luigi (Arduino) y Proxy Agent Raspberry Pi mediante bluetooth

Como la configuración del bluetooth en la Raspberry Pi ha requerido varios pasos de configuración procedemos a detallarlos aquí:

1. Después de instalar la última versión de Raspian con kernel wheezy, ya tenemos habilitado tanto el wifi como el bluetooth internos de la placa de la Raspberry Pi 3, por lo que ya podemos abrir un terminal y operar con el dispositivo bluetooth interno directamente.

2. Lo primero de todo es hacer el kart de Luigi visible mediante su módulo bluetooth, el cual lo hace por omisión al encenderse la placa Arduino.

3. En la terminal de Raspberry Pi ponemos el siguiente comando:

```
$hcitool scan
```

4. El dispositivo bluetooth de la Rpi3 se pondrá a escanear en busca de dispositivos disponibles. Entre los dispositivos descubiertos debemos de copiar con ctrl+shift+c la MAC correspondiente al dispositivo HC-06.

5. A continuación vinculamos el dispositivo a la Rpi3 creando y montando un puerto de serie llamado *rfcomm0* con el siguiente comando:

```
$sudo rfcomm bind /dev/rfcomm0 [N°MAC] [N°Canal]
```

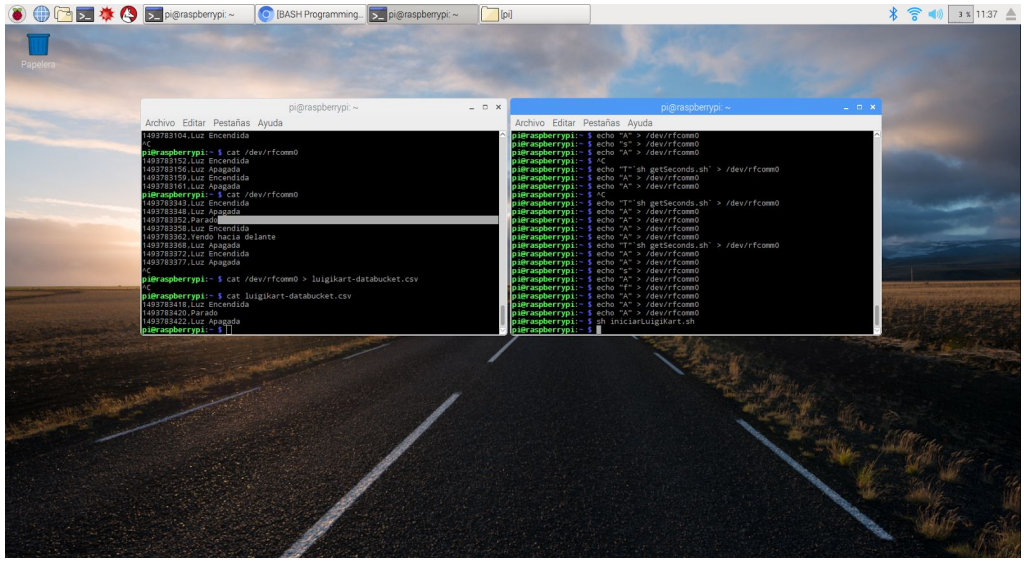
6. Con la orden anterior ha establecido un vínculo mediante bluetooth con el kart de Luigi tratándolo como un puerto serie de comunicaciones, por el cual podemos tanto enviar como recibir datos.

7. En este momento sería recomendable comprobar que efectivamente el puerto está montado y operativo con la orden:

```
$ls -l /dev/rfcomm0
crw-rw---- 1 root dialout 216, 0 may  4 20:23 /dev/rfcomm0
```

8. Si como resultado muestra algo similar a lo anterior, es que el puerto está montado y listo para las comunicaciones, si da error de que no encuentra el archivo o directorio es que algo no ha hecho bien. Para leer basta con un *'cat'* y para enviarle datos basta con un *'echo'*. No obstante, cada vez que se reinicia la Rpi3 el puerto se desvincula y se desmonta, para que esto no pase hay que incluir la orden del paso 5 al final del fichero .bashrc mediante:

```
$sudo nano .bashrc
```



Captura de pantalla recibiendo y enviando datos por terminal

IV. Enlace entre el Proxy Agent y la plataforma de IoT en la nube

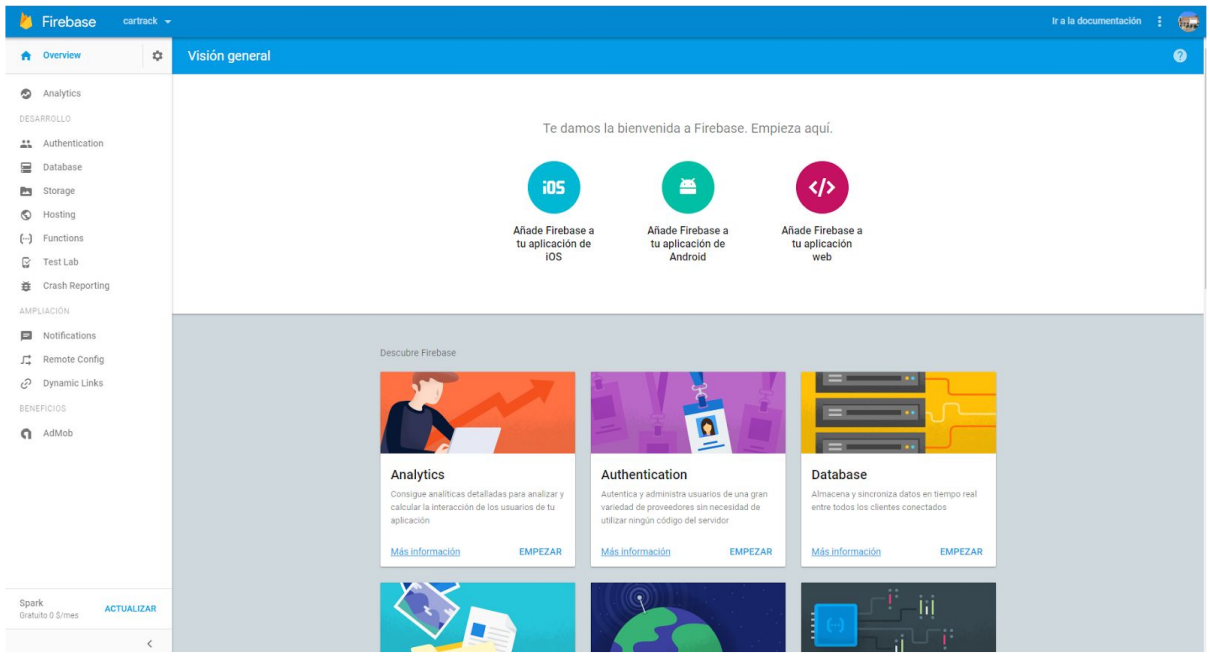
Plataforma en la nube: Firebase

Para el almacenamiento de datos y control de los dispositivos hemos elegido una plataforma llamada Firebase[6]. Firebase es una plataforma adquirida y evolucionada por Google cuya característica más destacable es la propagación de los cambios en tiempo real a todos los dispositivos suscritos a un conjunto de objetos o recursos. Teniendo implementaciones (bindings) a diversos lenguajes y plataformas como Java, IOS, Android o Javascript (en diversas plataformas y arquitecturas) es comprensible elegirlo como opción de interconexión multisistema. Firebase se ofrece con una capa gratuita de prueba con capacidad suficiente para gestionar las pruebas derivadas de esta práctica. El motivo para elegir esta plataforma sobre otras fue la experiencia previa en su uso por parte de uno de

[6] https://firebase.com

nosotros. Es sencillo realizar una conexión desde una web o desde el proxy agent. Un listado detallado de sus características más relevantes puede obtenerse en su propia página web[7] o en la página de la wikipedia dedicada al proyecto[8].

Los pasos necesarios para crear un proyecto en *firebase* son sencillos y tan sólo es necesario tener una cuenta de google. Una vez dentro se elige el plan gratuito, un nombre para el proyecto y un país/región. En nuestro caso elegimos "cartrack" y España, y el sistema nos asignó *cartrack-d6610*. Una vez dentro el panel de control nos muestra diversas opciones como podemos ver a continuación:



Captura de pantalla de la plataforma con diversas opciones que muestra en su menú.

La relativa a los parámetros de conexión desde nuestras aplicaciones es accesible en el menú "*Añade Firebase a tu aplicación web*". Es necesario utilizar esos datos para configurar la aplicación web para el usuario remoto y la aplicación del proxy agent.

Instalación de la librería

La comunicación entre el proxy agent y firebase se produce mediante las librerías proporcionadas por firebase, en su versión para nodejs, disponible en npmjs[9], el repositorio por excelencia en nodejs.

Para instalar ese código tan sólo es necesario teclear:

```
npm install firebase
```

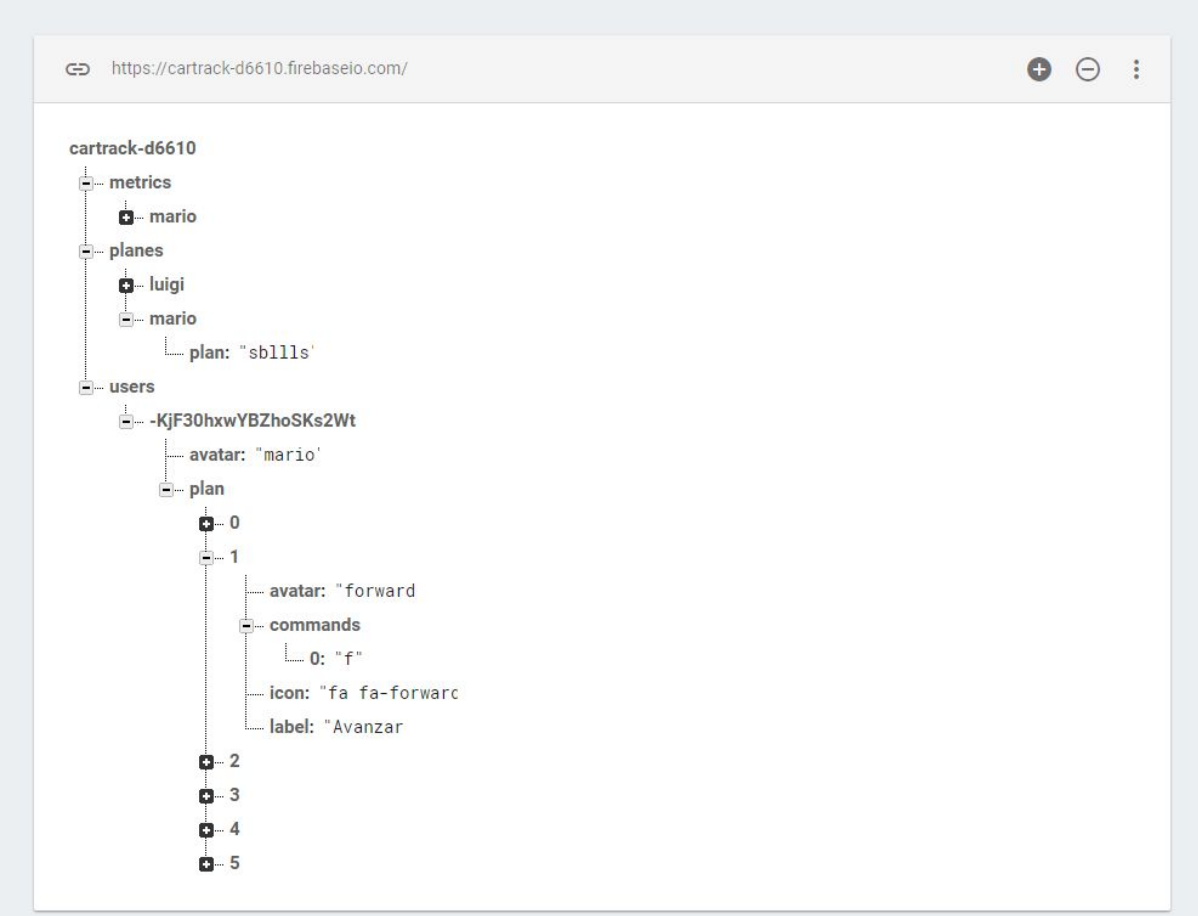
[7] https://firebase.google.com/features/
[8] https://en.wikipedia.org/wiki/Firebase
[9] https://www.npmjs.com/package/firebase

Supuesto que estuviera instalado nodejs y npm en el equipo destino.

Queremos destacar aquí que existe una versión para arduino[10] proporcionada por firebase pero se encuentra en una versión muy temprana y no es estable. Para usarla sería necesario contar con una versión o shield dotado de wifi o ethernet.

Esquema de datos en la nube de firebase

Los datos almacenados en la nube de firebase pueden reflejarse en la siguiente imagen:



Captura de pantalla de la base de datos durante una de las fases de desarrollo tal y como es navegable desde el panel de control de firebase.

Como podemos ver tenemos 3 colecciones, la primera *metrics* es la destinada a almacenar la información reportada por el kart (lo que hemos llamado *telemetría*), la segunda *planes* es la que sirve para indicar al kart qué recorrido realiza, la última de ellas *users* sirve durante la planificación del recorrido para guardar el listado de comandos que el usuario elige que el kart realice.

[10] https://github.com/firebase/firebase-arduino/

V. Extracción y visualización de datos y emisión de órdenes

Para el control de los karts hemos desarrollado una web desde 0, utilizando frameworks y librerías de software libre, entre ellos:

- referentes a funcionalidad: angularjs[11] y angularfire[12].
- referentes a estilo: bootstrap[13] y font-awesome[14].

Como la aplicación estará en producción en https://cartrack-d6610.firebaseapp.com/ es sencillo ver el código fuente de la misma añadiendo en el navegador el prefijo “view-source:”

El mecanismo de funcionamiento de un usuario es sencillo. En la parte superior se puede seleccionar cada uno de los “jugadores” que están disponibles. Hay 7 disponibles, aunque la única limitación fue la de encontrar iconos adecuados para cada uno de ellos. Nuestros karts están configurados para usar como identificadores “mario” y “luigi” aunque podrían definirse otros nuevos con relativa sencillez.

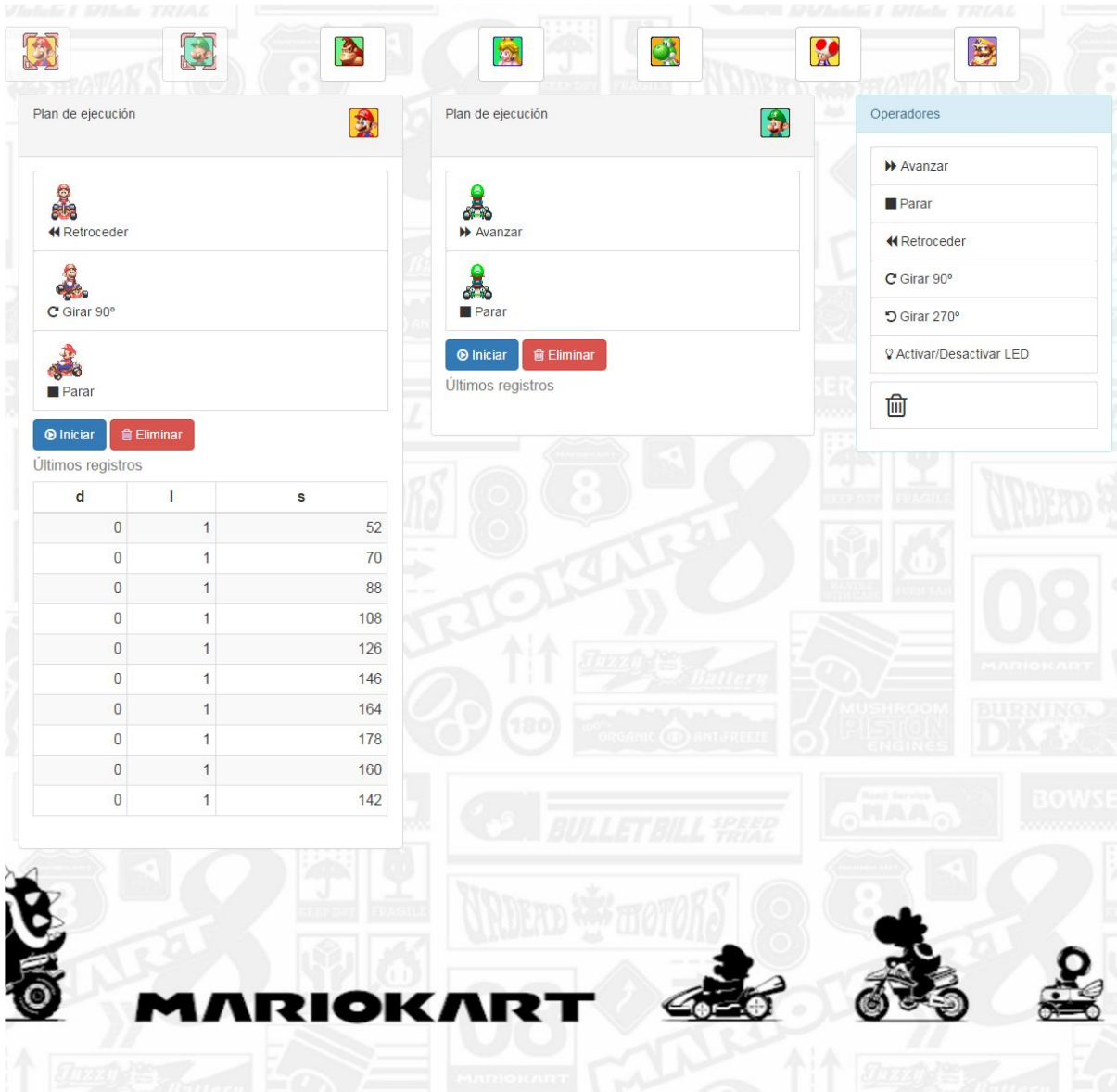
En la parte derecha de la pantalla se pueden ver el conjunto de operadores que hemos definido, mecanismo igualmente extensible[15]. Para elegir qué debe hacer cada kart lo único que hay que hacer es arrastrar y soltar cada funcionalidad al coche correspondiente. Posteriormente pulsar en Iniciar. Conforme la ejecución tenga lugar se irá viendo debajo la telemetría enviada por el vehículo. Lo mostrado es un ejemplo y podría ser extendido con otros parámetros como la velocidad de cada rueda o cualquier lectura de algún nuevo sensor.

A continuación mostramos las constantes utilizadas en el programa y que reiteramos pueden ser ampliadas:

```
  'OPERACIONES' = [
      {"label": "Avanzar", "commands": ["f"], "icon": "fa fa-forward",
"avatar": "forward"},
      {"label": "Parar", "commands": ["s"], "icon": "fa fa-stop", "avatar":
"forward"},
      {"label": "Retroceder", "commands": ["b"], "icon": "fa fa-backward",
"avatar": "backward"},
      {"label": "Girar 90°", "commands": ["r"], "icon": "fa fa-repeat",
"avatar": "right"},
      {"label": "Girar 270°", "commands": ["l"], "icon": "fa fa-undo",
```

[11] https://angularjs.org/
[12] https://github.com/firebase/angularfire/
[13] http://getbootstrap.com/
[14] http://fontawesome.io/
[15] Consultar fichero con código disponible en: https://cartrack-d6610.firebaseapp.com/script.js

```
"avatar": "left"},
      {"label": "Activar/Desactivar LED", "commands": ["A"], "icon": "fa
fa-lightbulb-o", "avatar": "forward"},
]
  'CHARACTERS' = ['mario', 'luigi', 'donkeykong', 'peach', 'yoshi', 'toad',
'wario']
```



Captura de la aplicación web en una de las fases de su desarrollo.

Recomendamos el acceso a la web indicada para comprender este apartado.

VI. Posibles mejoras

Debido a limitaciones en el tiempo para el desarrollo del proyecto y, aunque cumple con los objetivos de explorar vías de comunicación bidireccional entre los microdispositivos y entornos reales obteniendo incluso elementos que puedan desarrollarse en la práctica real, nos han quedado algunas propuestas que nos habría gustado explorar. Podríamos destacar entre ellas:

Proyecto Google Street View Indoor: disponemos de una cámara 360º que podría ser fácilmente acoplable al coche. Nuestra intención era ponerle a recorrer un circuito y sacar una foto de 360º cada cierto número determinado de segundos. Obtener esa foto mediante conexión en tiempo real en wifi y desarrollar un portal para mostrarla. Llegamos a investigar y encontrar librerías para la visualización de las instantáneas pero la dificultad surgió al intentar automatizar las capturas. El protocolo de conexión que utiliza es propietario y no está documentado así que realizamos diversas tentativas, entre ellas: decompilar la aplicación android o capturar el tráfico mediante wireshark. Desgraciadamente, el tiempo necesario para desarrollar un software que reemplazara el propietario y permitiera la automatización excedía con creces el tiempo que debería ser destinado a la asignatura.

Proyecto conducción automática mediante redes neuronales: esta opción era interesante pues combinaba varias asignaturas. La intención era entrenar redes neuronales fuera del microcontrolador y cargar el valor de las capas intermedias en el arranque. Esto permitía ejecutar en el microcontrolador una cantidad de código predecible y cuantificable al ejecutarse la red neuronal en tiempo constante. A la vez que se realizara el recorrido se enviaría la telemetría para posterior aprendizaje. Este proyecto no se descartó por no ser viable sino por el atractivo que tenía la competición del mario kart.