MANUAL DE INSTRUCCIONES DE USO DEL VEHICULO DEEP-RACER

Carga y encendido

El vehículo deep-racer cuenta con dos baterías. La primera, que se puede ver en la Figura 1, es una Power-Bank que alimenta el miniordenador, mientras que la segunda, que se puede ver en la Figura 2 es la que alimenta el motor. Las dos baterías se cargan con cargadores diferentes, y la batería de litio siempre se debe cargar con alguien presente.

En cuanto al encendido del vehículo tiene dos partes diferenciadas, por un lado el miniordenador, que además conectará los sensores como la cámara y el LiDAR. Para encender el miniordenador primero hay que conectar el cable de la Power-Bank, después encender la Power-Bank y finalmente el miniordenador. Este interruptor es un botón lateral fácil de encontrar. Por otro lado estará el interruptor del motor, que se encuentra oculto detrás de las ruedas, tal y como se puede ver en la Figura 3.

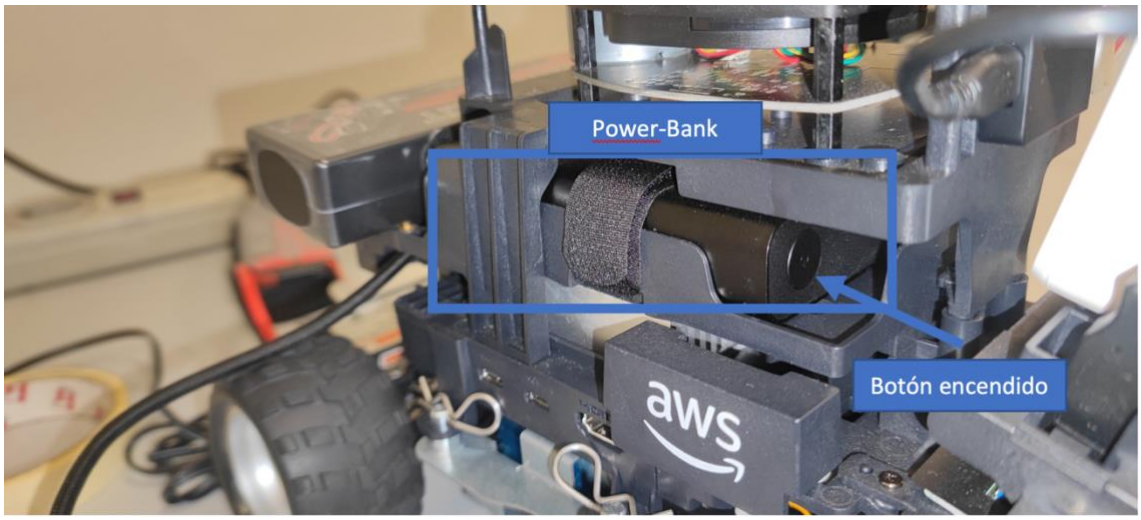

Figura 1. Power bank que alimenta el miniordenador del deep racer

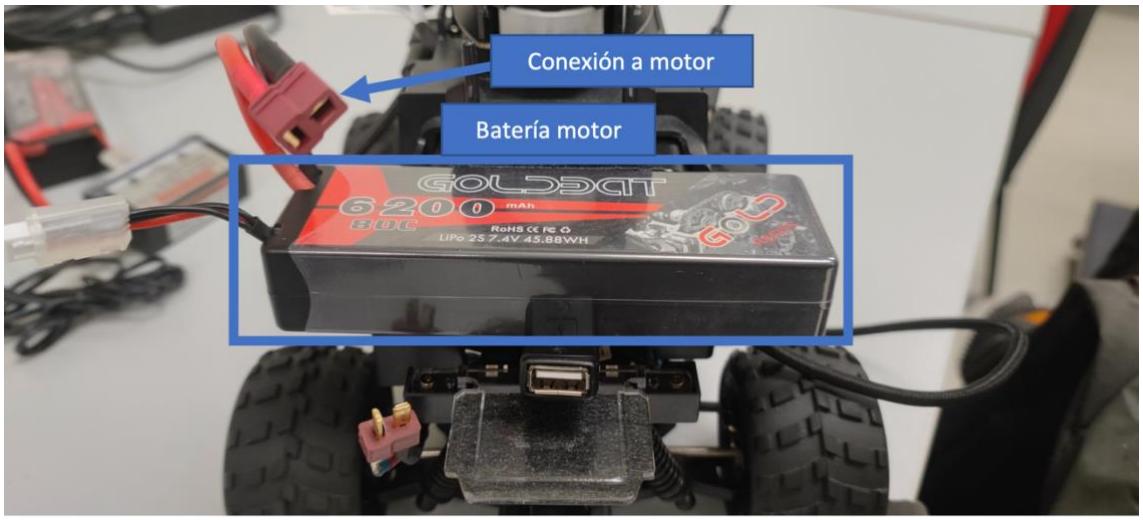

Figura 2 Bateria de Litio que alimenta el motor del deep racer

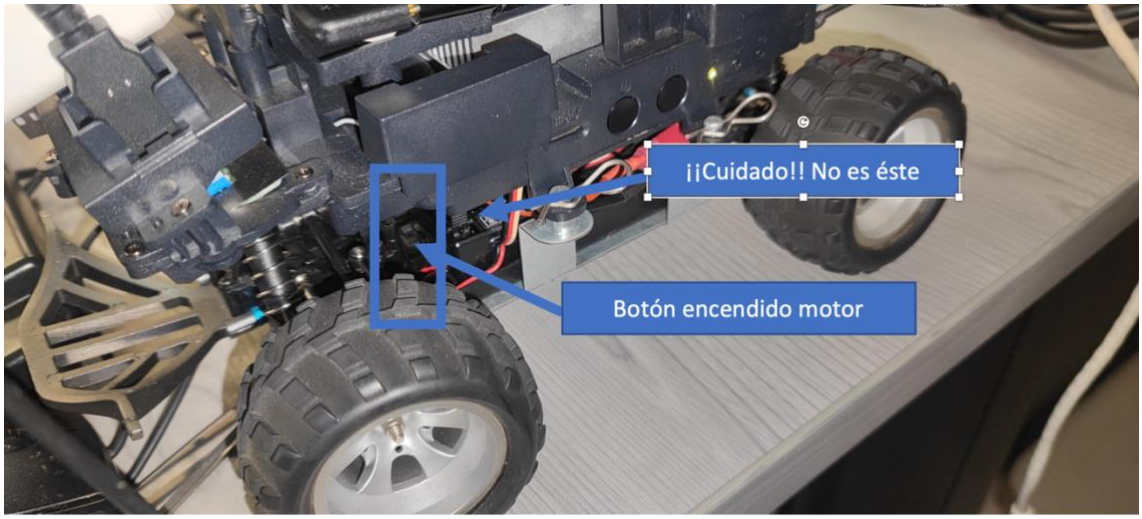

Figura 3 Posición de interruptor del motor

A la hora encender el vehículo, se recomienda empezar encendiendo solo la parte del miniordenador para aprender a configurar los servicios y acceder a ellos, tal y como se explicarán en las siguiente secciones. Una vez maneja correctamente esos servicios, puede pasar a trabajar con el vehículo en movimiento.

Por lo tanto, encienda el miniordenador del vehículo. Para ello conecte una pantalla a la entrada HDMI del vehículo, un teclado y un ratón para trabajar con mayor facilidad. Como el número de conexiones USB libres no son suficientes, desconecte la entrada USB del LiDAR para conectar ahí el teclado o el ratón.

Al arrancar verá que se trata de un Ubuntu 16.04.

El password es “elcochedejc”

La única cuestión que debe realizar en el vehículo es configurar de manera adecuada el fichero de configuración del mismo. Dicho fichero lo puede encontrar en ~/Soft4 /vehicle.conf.

IMPORTANTE: NO ACTUALICE EN NINGÚN MOMENTO EL SISTEMA OPERATIVO. EL SISTEMA ROS QUE TIENE INSTALADO EL VEHÍCULO SOLO FUNCIONA CON ESTA VERSIÓN DE LINUX. ACTUALIZAR CUALQUIER LIBRERÍA PODRÍA LLEVAR A QUE EL VEHÍCULO DEJARA DE FUNCIONAR DE MANERA APROPIADA. TAMPO ACTUALICE EL SISTEMA ROS AUNQUE EL VEHÍCULO SE LO PIDA.

IMPORTANTE: LA BATERÍA DEL MOTOR HAY QUE DEJARLA SIEMPRE DESCONECTADA. PARA ELLO, ASEGURESE DE APAGAR PRIMERO EL MOTOR Y LUEGO DESCONECTAR LA BATERÍA.

IMPORTANTE: LA POWER-BANK HAY QUE DESCONECTARLA DEL MINIORDENADOR UNA VEZ APAGADO ÉSTE PARA EVITAR QUE SE DESCARGUE.

Instalación de los servicios de control del vehículo DEEP-RACER

Le hemos subido una serie de archivos al campus virtual que se corresponden con los servicios de control del vehículo. La parte del vehículo ya está instalada, por lo tanto lo que tiene que instalar es estos servicios en su infraestructura de red, allí donde considere adecuado.

Lanzando el servicio de control en el vehículo

Por favor, revise la parte de configuración del vehículo, sobre todo la dirección IP donde se encuentra el servidor MQTT y el servidor *cloud*. Dicho fichero de configuración lo podrá encontrar en `~/SoftARTEMIS/vehicle.conf`

Para hacer efectivos estos cambios ejecute el comando:

```
sudo systemctl restart ARTEMIS.service
```

Lanzando el servicio de control en la nube

Suponiendo que ha seguido los pasos de instalación del servicio en el lugar deseado, pasamos ahora a lanzar el servicio.

Lo primero es que hay que instalar un servidor MQTT en algún sitio. Recomendamos instalar mosquitto, que es muy utilizado y tiene gran cantidad de documentación. El servidor mosquitto lo puede instalar allí donde le parezca más interesante. Si todo está bien configurado, en el momento en el que se active el servidor mqtt y se active los servicios en el vehículo el LED de estado del vehículo pasará de rojo a amarillo.

Para poder enviar ordenes al vehículo, lo ideal es instalarse una aplicación mqtt, como por ejemplo mymqtt en un smartphone o mqtt explorer en un ordenador.

Cuando el vehículo se conecta, veremos automáticamente en nuestro cliente MQTT los siguientes tópicos:

Tópico	mensaje
`vehicleID/type_of_conection`	4G/5G o WiFi
`vehicleID/public_ip`	IP del vehículo según se ve desde el exterior
`vehicleID/steering_calibration`	Parámetros con los que está calibrado la dirección del vehículo
`vehicleID/mqtt_server_ip`	IP del bróker MQTT
`vehicleID/mqtt_server_port`	Puerto del bróker MQTT
`vehicleID/cloud_server_ip`	IP del servidor que corre la aplicación de control del vehículo
`vehicleID/cloud_server_port`	Puerto del servidor donde se accede a la aplicación de control del servidor

Es interesante fijarse en el tópico `vehicleID`. Cada vehículo es identificado de manera distinta por este tópico, que será un número entre 1 y 4. Puede identificar con qué vehículo está trabajando viendo los clientes que publican tópicos en el servidor MQTT.

Una vez el vehículo está conectado, esté podrá pasar por diferentes estados dependiendo de los comandos mandados a través del tópico `vehicleID/command`.

En ese caso podemos lanzar los diferentes comandos de control del vehículo:

Estado	**Explicación**	**Comando para llegar al estado**	**Reacción del vehículo**
Encendido	El servicio ARTEMIS.service se ejecuta. Se utilizan los parámetros del fichero de configuración vehicle.conf, el sistema de giro es calibrado y la conexión al servidor MQTT se intenta establecer	Encender el vehículo presionando el interruptor del miniordenador y esperar un minuto aproximadamente	
Modo Autónomo - *Cloud*	El vehículo transmite imágenes al servidor configurado en el fichero vehicle.conf	`AM-Cloud`	
Modo autónomo - Off	El vehículo espera órdenes a través de MQTT	`AM-Off`	
Modo autónomo - Local	El vehículo comienza a procesar las imágenes y el control de vehículo según el algoritmo de guiado autónomo instalado en él	`AM-Local`	
Calibración	El vehículo calibra la dirección con los nuevos parámetros recibidos a través de MQTT	calibrate maxvalue midvalue minvalue	
Comando desconocido	El vehículo recibe un comando desconocido vía MQTT y lo ignora	Cualquier otro comando no reconocido	
Sin conexión	El vehículo no consigue establecer la conexión con el servidor MQTT y intenta repetidamente establecer la conexión	Vehículo desconectado de la red, archivo vehicle.conf erróneo o servidor mqtt desconectado	

Servicios en el *cloud*

Le hemos dado acceso a varios archivos .py que permiten controlar el vehículo para que actúe de manera autónoma. Dichos archivos pueden ser analizados para ver cómo se recibe la información de video y cómo se procesa para su propia aplicación. Los archivos son los siguientes:

1. `artemis_autonomous_car.py`: Clase que ejecuta incluye los algoritmos de guiado autónomo y que devuelve las instrucciones a ejecutar por el vehículo para la conducción.
2. `cloud_control_server.py`: Fichero que recibe las imágenes de "video" y la información del LiDAR desde el vehículo a través de UDP. Utiliza la clase artemis_autonomous_car.py para ejecutar los algoritmos de conducción autónoma y análisis de las imágenes. Este fichero contiene varias variables importantes. La primera es la tupla `server_address` que indica la IP y el puerto de escucha. Cada vehículo tiene asociado un puerto y hay que ajustar este puerto al vehículo. Si la variable booleana `show_image` está a `True`, el software mostrará en tiempo real las imágenes que le llegan y el algoritmo de ejecución. Si está fijado a `False` el software no mostrará nada. Es importante señalar que el camino que recorre el vehículo en la maqueta está controlado por un vector de números en la línea 18

```
auto_utils=artemis_autonomous_car.artemis_autonomous_car([x x x …],0)
```

   Utilizar un '1' significa que en el siguiente cruce debe tomar la izquierda, '2' recto y '3' derecha. Cuando se acaban los comandos de giro el vehículo para.

3. `Cloud_control_server_real_time.py`: Igual al anterior excepto que el sistema espera que le mandemos comandos por teclado. Los comandos son los siguientes:
   a. 'a': avanzar
   b. 'z': parar
   c. '1': girar a la izquierda en el próximo cruce
   d. '2': recto en el próximo cruce
   e. '3': girar a la derecha en el próximo cruce
4. `manual_control_server.py`: La estructura del programa es similar a los anteriores, excepto que permite controlar manualmente el vehículo. Puede ser de interés para comprobar si todo está funcionando correctamente. Las teclas de control son iguales a las de un videojuego de ordenador:
   a. 'w': marcha hacia delante
   b. 's': marcha hacia atrás
   c. 'a': girar a la izquierda
   d. 'd': girar a la derecha
   e. '2': marcha hacia delante gran velocidad (por favor, no utilizar excepto que la batería del motor esté prácticamente agotada)
   f. 'x': marcha hacia atrás a gran velocidad (por favor, no utilizar excepto que la batería del motor esté prácticamente agotada)