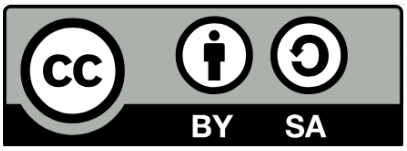
Universidad
Rey Juan Carlos

Grado de Ingeniería en Robótica Software

Sistema empotrados y de tiempo real

Práctica 4

Sigue Linea en Arduino basado en sistemas RT



Sumario

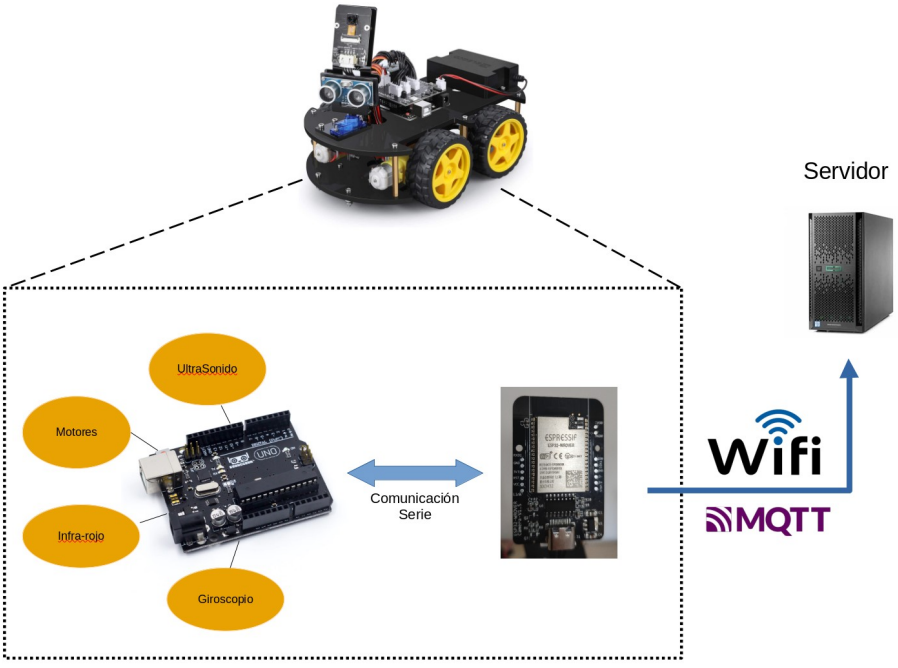
1. Descripción

La práctica se realizará por grupos de 2 estudiantes. Se utilizará el kit de Smart Robot Car Kit v4.0 compuesto por un arduino uno y un ESP32 camera

La evaluación y objetivos de esta práctica son:

- Sigue la línea lo más rápido posible sin salirse.
- Comunicación IoT a través de MQTT
- Comunicación serie entre ESP32 y Arduino UNO
- Si el robot pierde la línea se permite realizar una búsqueda de la línea de nuevo.
- Detección de obstáculos

2. Esquema



3. Montaje

Consulta el manual que viene con el kit y el siguiente vídeo para realizar el montaje correctamente. Es el primer paso de la práctica.

4. Arduino

El robot incorpora un Ardunio UNO que actúa de cerebro del sistema controlando todos los sensores y actuadores.

4.1 Ultrasonido

Los sensores de ultrasonido los puedes controlar uitlizando los siguientes pines.

```
#define TRIG_PIN 13
#define ECHO_PIN 12
```

4.2 Sensor Infra-rojo

Los sensores infra-rojos, son sensores analógicos. Usa toda su potencia y rango continuo de valores. **No los utilices como sensores digitales**

```
#define PIN_ITR20001-LEFT   A2
#define PIN_ITR20001-MIDDLE A1
#define PIN_ITR20001-RIGHT  A0
```

4.3 Motores

El Smart Robot Car incorpora 4 motores DC que se controlan a través de una pequeña placa controladora incorporada en la placa de extensión. Las velocidades las comandaremos por grupos de motores (los del lado derecho, y los del lado izquierdo). A continuación se muestran los pines de control, velocidad y dirección.

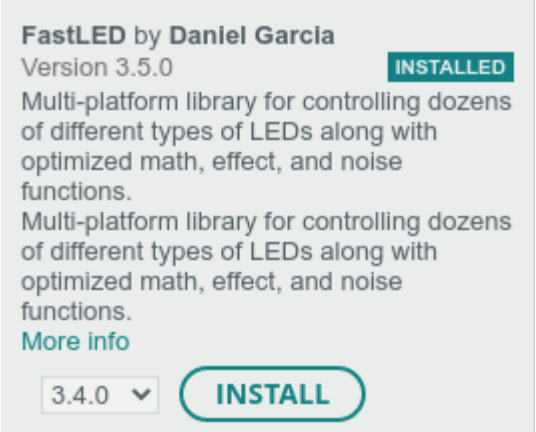
```
// Enable/Disable motor control.
//  HIGH: motor control enabled
//  LOW: motor control disabled
#define PIN_Motor_STBY 3

// Group A Motors (Right Side)
// PIN_Motor_AIN_1: Digital output. HIGH: Forward, LOW: Backward
#define PIN_Motor_AIN_1 7
// PIN_Motor_PWMA: Analog output [0-255]. It provides speed.
#define PIN_Motor_PWMA 5

// Group B Motors (Left Side)
// PIN_Motor_BIN_1: Digital output. HIGH: Forward, LOW: Backward
#define PIN_Motor_BIN_1 8
// PIN_Motor_PWMB: Analog output [0-255]. It provides speed.
#define PIN_Motor_PWMB 6
```

4.4 LED

La placa de expansión contiene un LED RGB que podemos utilizar para plasmar colores según los estados de tu comportamiento (muy útil para depuración en el circuito).

IMPORTANTE: Es obligatorio que siempre que tu robot detecte la línea (con cualquiera de los 3 sensores), debe mostrar el color verde en el LED. Si no detecta la línea, debe mostrar el LED en rojo.

Es necesario instalar la siguiente librería (FastLED):



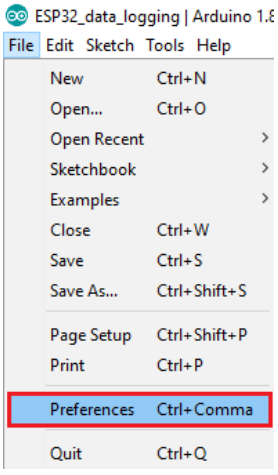
Puedes consultar el siguiente código de ejemplo para entender el funcionamiento DemoLed

5. ESP32

5.1 Arduino-IDE y librerías

El robot incluye un modelo ESP32 CAM, que nos permite comunicarnos a través de cualquier red WiFi. Utilizaremos Arduino-IDE para programarlo, pero es necesario realizar las siguientes acciones:

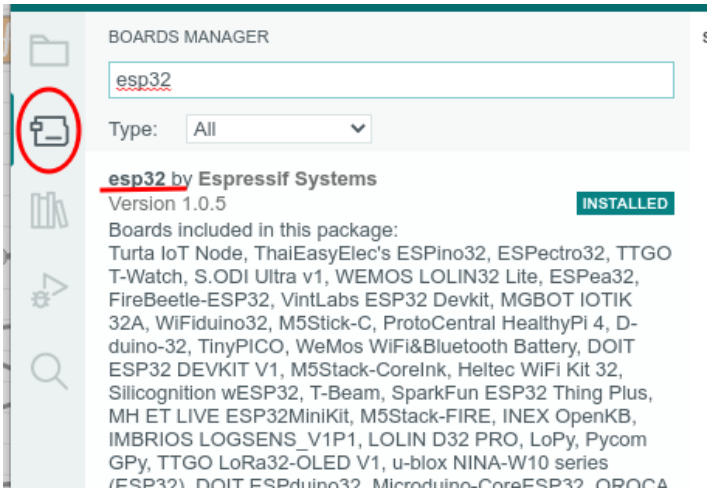
- Primero, añade el repositorio para ESP32 dentro de tu Arduino IDE



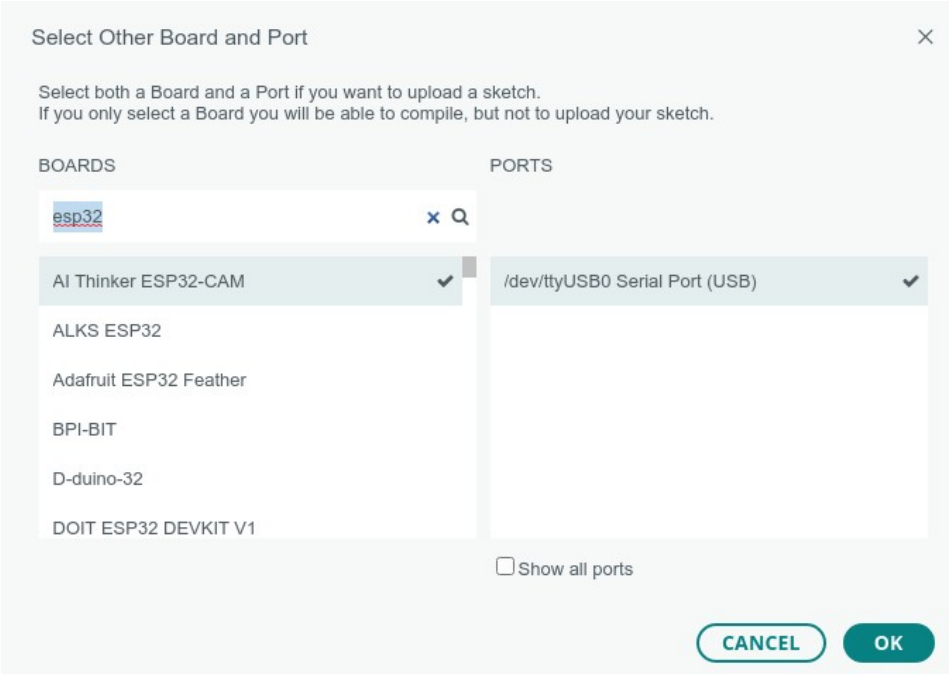
Introduce la siguiente URL en "Additional Board Manager URLs"

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Asegurate que tienes instalado el paquete ESP32 dentro de "Boards Manager" en tu arduino IDE



Asegurate de configurar correctamente el modelo de placa "AI Thinker ESP32-CAM"



Por último asegurate que en los laboratorios, te aparecen 2 puertos detectados (ttyS4 y ttyUSB0). Utiliza éste último para realizar la programación del ESP32.

5.2 Comprobar la conexión WiFi

- Puedes comprobar la conexión WiFi (ESP) y la IP asignada. Para ello utilizaremos la red wifi eduroam. Puedes utilizar de base el código de ejemplo de **eduroam** del repositorio.

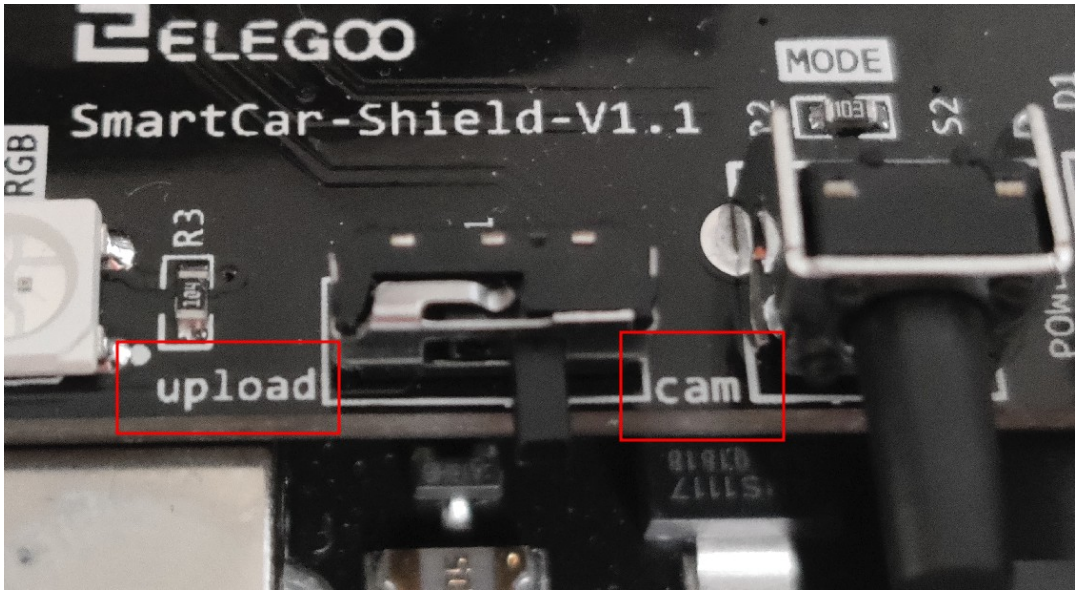
Si ves que la wifi no te da IP, puedes realizar pruebas con la wifi que levantes en tu smartphone (Hotspot) o con cualquier otra wifi (incluso la de tu casa, si haces pruebas allí).

6. Comunicación Serie

Revisa el código de ejemplo que encontrarás en SerialCommunication para entender como es posible comunicar los puertos serie del ESP y de Arduino. Este ejemplo muestra comunicación en ambos sentidos y hace uso de LedFast.

La comunicación serie puede ser bidireccional, es decir, podemos mandar mensajes del ESP al arduino y viceversa. El comportamiento sigue línea NUNCA puede comenzar hasta que el ESP confirme que tiene WiFi y está conectado el servidor MQTT.

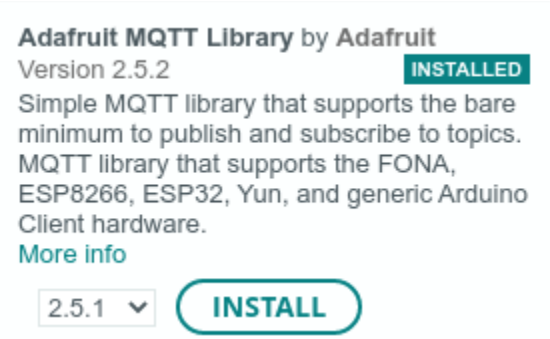
IMPORTANTE: Para que la comunicación serie entre el ESP y Arduino funcione correctamente, el switch S1 de la placa de expansión debe estar en la posición "CAM". Recuerda que ese switch debe estar en la posición "UPLOAD" para cargar el programa en la placa Arduino.



7 Comunicación IoT

7.1 MQTT

Desde el ESP32 tendrás que conectarte y mantener la conexión abierta para mandar mensajes al servidor según vayas completando el circuito. Para ello necesitarás instalar la librería **Adafruit-MQTT**



Aquí es necesario configurar un servicio MQTT publico para el acceso de los estudiantes. Se deja un pequeño manual de como realizarlo.

7.2 Mensajes

Tu robot debe mandar los siguientes mensajes siempre conectando al servidor MQTT y utilizando obligatoriamente el siguiente TOPIC

```
/SETR/2023/$ID_EQUIPO/
```

Para comprobar los mensajes que envía tu robot a través de MQTT puedes utilizar el siguiente comando que debes ejecutar en los ordenadores del laboratorio. Básicamente es un subscriptor a un topic determinado que mostrará todo lo que llega a ese topic.

```
mosquitto_sub -v -h 193.147.53.2 -p 21883 -t /SETR/2023/$ID_EQUIPO/
```

Para realizar pruebas puedes probar a publicar de la siguiente manera:

```
mosquitto_pub -h 193.147.53.2 -p 21883 -t /SETR/2023/$ID_EQUIPO/ -m "hello world!"
```

Mensaje de inicio de vuelta

- Descripción: Este mensaje debe enviarse siempre justo antes de empezar la vuelta al circuito. Por tanto debe realizarse sólo 1 vez. **IMPORTANTE: La vuelta nunca podrá comenzar si no hay conexión a la red WiFi ni a MQTT.**

- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "START_LAP"
}
```

Mensaje de Fin de vuelta

- Descripción: Este mensaje debe enviarse siempre al finalizar una vuelta, que lo sabrás porque tendrás un obstáculo delante. Por tanto debe realizarse sólo 1 vez. Presta atención al campo "time", debe contener el tiempo (en milisegundos) transcurrido desde que enviaste tu START_LAP hasta que envías tu END_LAP. La vuelta ha finalizado justo después de detectar un obstáculo.
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "END_LAP",
        "time": 000000
}
```

Mensaje Obstáculo Detectado

- Descripción: Este mensaje debe enviarse siempre que detectes un obstáculo en el camino de la línea. El coche se debe detener entre 5 y 8 cm antes del obstáculo. En el campo distancia debes enviar la distancia detectada en centímetros.
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "OBSTACLE_DETECTED",
        "distance" : 000
}
```

Mensaje Línea Perdida

- Descripción: Este mensaje debe enviarse siempre que tu robot se haya salido de la línea.
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "LINE_LOST"
}
```

Mensajes PING

- Descripción: Mensaje de estado mandado cada **4 segundos** (empezando en 0). El campo time debe reflejar el tiempo (en milisegundos) desde que comenzó la vuelta (START_LAP). Debe priorizar el comportamiento sigue linea al envío de mensajes PING.
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "PING",
        "time": 000000
}
```

Mensaje Inicio Búsqueda de Linea (OPCIONAL)

- Descripción: Opcionalmente tienes la posibilidad de implementar un comportamiento "encuentra línea" una vez que te hayas salido del camino (esto te permitirá reducir la penalización).
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "INIT_LINE_SEARCH"
}
```

Mensaje Fin Búsqueda de Linea (OPCIONAL)

- Descripción: Si has implementado la búsqueda de línea una vez que la has perdido, deberás mandar este mensaje en el momento que la hayas encontrado y por tanto finalizado el comportamiento "encuentra línea"
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "STOP_LINE_SEARCH"
}
```

Mensaje Línea Encontrada (OPCIONAL)

- Descripción: Este mensaje debe enviarse siempre que tu robot haya encontrado la línea. Además este mensaje sólo se debe enviar si anteriormente se ha enviado el mensaje **LINE_LOST**
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
```

```
        "action": "LINE_FOUND"
}
```

Mensaje Estadísticas Línea Visible (OPCIONAL)

- Descripción: Este mensaje debe enviarse al finalizar la vuelta, es decir, cuando encuentres el obstáculo. Debes enviar el % de veces que has detectado la línea utilizando los infra-rojos. Es decir, si durante la vuelta has leído 1000 veces el infrarojo, y 900 has detectado la línea con alguno de los sensores, y 100 veces no la has detectado, deberás mandar un 90.0 %
- Payload JSON:

```
{
        "team_name": "$TU_NOMBRE_DE_EQUIPO ",
        "id": "$ID_EQUIPO",
        "action": "VISIBLE_LINE",
        "value": 0.00
}
```

8 Evaluación

8.1 Día de Test y Día de Examen

El **Martes 19 de Diciembre** en horario de clase cada equipo dispondrá de al menos 5 minutos para probar su solución en el circuito real y con la parte de comunicaciones realizada. Podrá comprobar así que los mensajes MQTT están bien formados y son correctos.

El **Jueves 21 de Diciembre** en horario de clase cada equipo dispondrá de máximo de **2 rondas/tests** para realizar el circuito. Entre ronda y ronda está permitido modificar el código del programa.

El circuito será idéntico para ambos días.

8.2 Bonificaciones y Penalizaciones

- Cuando el coche se salga del circuito, ese test se dará por finalizado y se declarará nulo.
- Para aprobar la práctica el coche debe terminar el circuito correctamente y mandar los mensajes de comunicación establecidos mediante MQTT. El no envío de mensajes MQTT supondrá tener la práctica suspensa.
- El coche debe parar en el rango [5-8] cm antes del obstáculo. Por cada cm fuera de rango se aplicará una penalización de un 1% al tiempo final.
- Si el coche pierde la línea tiene un máximo de 5 segundos para encontrarla. Si es capaz de recuperarla correctamente y mandar los mensajes MQTT correspondientes obtendrá una rebaja del tiempo final del 2%.

8.3 Nota Final

La nota final dependerá de:

- La posición final en la tabla de tiempos
- Código arduino y mecanismos utilizados.
- Mensajes MQTT enviados y periodicidad (PING)
- Post en el blog explicando la solución.

IMPORTANTE: La práctica estará suspensa si:

- El coche no es capaz de realizar 1 vuelta completa.
- El coche tarda más de 20 segundos en completar la vuelta.