Robot seguidor de línea con Raspberry Pi

Kevin Pérez

Curs Creació de prototius d'IoT amb Raspberry

CIFO La Violeta

Noviembre 2025

Índice

1.Resumen

La idea principal es desarrollar de forma experimental un sistema de control inteligente para un tren a escala, utilizando una Raspberry Pi 3, con el propósito principal de aprender y comprender los fundamentos de control, la automatización y la electrónica aprendida en este curso.

Más que buscar un resultado final completamente funcional, el trabajo se basa en el proceso de diseño, construcción y aprendizaje, aplicando lo aprendido en este curso, desde la integración de hardware (motor, sensores...etc) hasta la programación de Python y el control lógico de comportamiento, donde todo lo nombrado anteriormente es nuevo para mí.

2.Introducción

El desarrollo de este trabajo se basa en poder controlar un tren de forma inalámbrica, a través de un mando, el cual estará formado por tres botones, verde (acelerar), amarillo (bajar velocidad), rojo (detener).

El objetivo es que un niño pueda entender el funcionamiento de un semáforo aplicándolo al uso de un tren y de esa manera pueda aprender mediante el juego.

Empiezo a intentar diseñar el modelo del tren en 3D, buscando modelos al cual pueda incorporar motores, sensores… después de buscar e intentar obtener algún diseño, con la ayuda de Joan (el profesor del curso), empezamos a enfocar el trabajo desde otro ángulo. Me recomienda que empiece a entender el funcionamiento de motores, sensores…etc más que en el propio diseño. Así que siguiendo su consejo empiezo a entender el funcionamiento y añado un sensor de seguimiento de líneas, el cual hará que pueda moverse de manera autónoma siguiendo la línea que se diseñe en el suelo.

A medida que avanzo con el trabajo, me doy cuenta que no será viable hacer el tren, así que empiezo a pensar otra idea. La cual es hacer un coche con 4 motores, que siga una línea marcada en el suelo. Esta idea viene gracias a un proyecto de años anteriores en este centro CIFO La Violeta, en el cual un compañero hizo un coche (G1) controlado a través de un joystick. Este proyecto me ha servido de gran ayuda para realizar el mío.

Así que primeramente empiezo a probar los motores, controlando velocidad, giro, etc

Añado los sensores de seguimiento de línea, para poder controlar el movimiento de los motores, (adelante, izquierda, derecha, detenerse)

Una vez controlado los 4 motores y sensores de seguimiento de línea añado el receptor bluetooth, cambio la idea de hacer un control remoto físico, ya que por falta de tiempo no podría hacerlo y decido vincular la Raspberry a través de bluetooth con una aplicación móvil, la cual tiene que ser compatible con Android, ya que con IOS, no he sido capaz de encontrar alguna.

Una vez vinculado, se modifica el código principal en el que solo se movía el coche a través de la línea, para poder controlarlo manualmente a través de la aplicación, si el coche en algún momento se sale de la línea, poder rectificar el movimiento. Para ello, importamos una librería para poder controlar el receptor que es la *pyserial,* en el diseño del sistema, explicare como hacer la instalación.

A demás, he hecho otro código, para poder controlarlo solamente de forma manual y así tener una manera más divertida y libre de controlar el coche.

A la hora de empezar con el montaje, intento aprovechar material del aula, para ello necesito piezas de soporte para los motores. Hago el diseño en 3D y las consigo, con la dificultad de tener que hacer los agujeros para que después pueda introducir los tornillos. Tuvimos que buscar unos adaptadores que finalmente no pude usar, porque

hacían el agujero de la pieza mas grande y no quedaban dentro para después poder colocar el tornillo. Así que acabo colocando tornillos directamente y consigo que se sujeten bien los motores a la estructura. En cuanto a los motores, tuve que soldar un par de cables en cada motor, añadiendo cables macho para poder hacer la conexión directa a la protoboard y así evitar perder potencia haciendo empalmes innecesarios.

Finalmente, por desgracia, un motor en los últimos 2 días se ha estropeado y no he podido reemplazarlo, así que no he podido probar el coche en el suelo. Tampoco he sido capaz de encontrar una fuente de alimentación que pueda alimentar a los 4 motores ni a la Raspberry y de esa manera conseguir que el coche pueda ser autónomo.

3.Teoria de los componentes

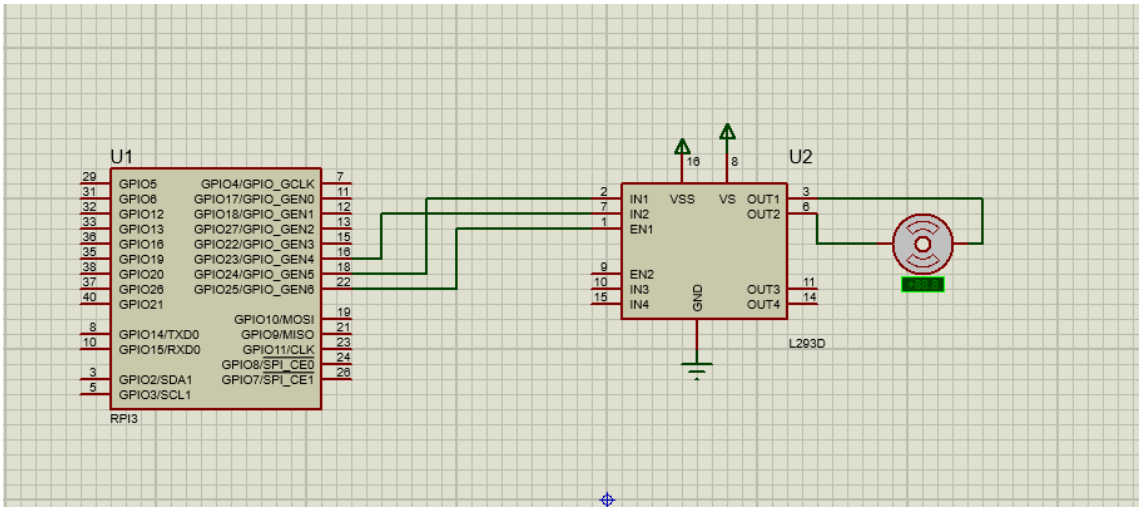
A continuación, detallare los materiales utilizados y que función tiene cada uno de ellos.

Componente	Descripción	Función en el proyecto
Raspberry Pi 3B	Microcomputadora con procesador ARM Cortex-A53 de 64 bits, 1.4 GHz, 1 GB de RAM, Wi-Fi, Bluetooth y 40 pines GPIO.	Actúa como unidad principal de control. Procesa la información de los sensores IR, envía señales al módulo L298N para manejar los motores y gestiona la comunicación Bluetooth con el móvil.
Módulo L298N (Driver de motores)	Módulo controlador basado en el integrado L298N que permite manejar dos motores DC en ambos sentidos usando señales PWM.	Controla la velocidad y dirección de los motores. Recibe las señales de control de la Raspberry Pi y suministra la corriente necesaria para mover los motores.
Sensores infrarrojos (IR) (2 unidades)	Módulos emisores y receptores de luz infrarroja que detectan la reflexión del color del suelo (blanco o negro).	Permiten detectar la línea negra sobre la que debe desplazarse el robot. Cuando el sensor detecta negro, el robot se detiene o corrige su dirección.
Motores DC (4 unidades)	Motores de corriente continua de 3 a 12 V, conectados a las ruedas del chasis.	Proporcionan el movimiento del robot hacia adelante, atrás o en giros, según las órdenes del driver L298N.
Módulo Bluetooth HC-05	Módulo de comunicación inalámbrica basado en Bluetooth clásico (versión 2.0).	Permite la comunicación entre la Raspberry Pi y un dispositivo móvil. Se puede usar para enviar comandos manuales o recibir datos del robot.
Resistencias (varias unidades)	Componentes pasivos que limitan la corriente eléctrica.	Protegen los sensores IR o los pines de la Raspberry Pi evitando sobrecorriente.
Chasis con ruedas	Estructura del robot que sostiene todos los componentes.	Proporciona soporte físico y movilidad al sistema. Generalmente incluye soporte para sensores y motores.
Protoboard (breadboard)	Placa de plástico con filas de agujeros conectados internamente, que permite montar circuitos sin necesidad de soldar.	Facilita la conexión de sensores, motores, resistencias y la alimentación del robot de manera provisional, permitiendo hacer pruebas, cambios rápidos y verificar el funcionamiento del circuito antes de hacer un montaje definitivo.

4.Diseño del sistema

El primer paso, es entender el funcionamiento de los motores. Empiezo con simulaciones en Proteus, para probar si soy capaz de hacerlos funcionar y seguidamente empezar a controlar el comportamiento de ellos.

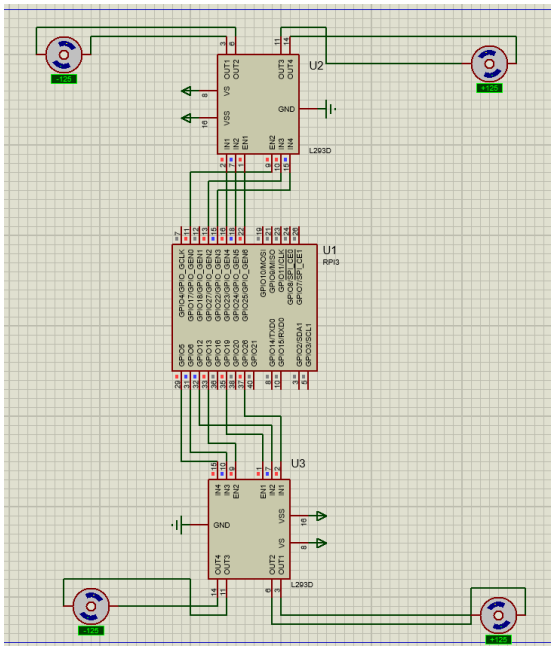
Primera simulación, con un solo motor, donde podemos observar que utilizo el chip L293D, es el que me permite controlar dos motores a la vez:



Primer código, añado función PWM, donde el motor hará la acción de acelerar y desacelerar de forma controlada.

```
import time
import RPi.GPIO as GPIO

IN1 = 23
IN2 = 24
EN1 = 25
TEMPS = 5
PAUSA = 5

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(EN1, GPIO.OUT)

p = GPIO.PWM(EN1, 50)
p.start(0)

def antihorari(temps):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    for dc in range(0, 101, 1):
        p.ChangeDutyCycle(dc)
        time.sleep(0.05)
    time.sleep(temps)
    for dc in range(100, -1, -1):
        p.ChangeDutyCycle(dc)
        time.sleep(0.05)

def horari(temps):
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)
    for dc in range(0, 101, 1):
        p.ChangeDutyCycle(dc)
        time.sleep(0.05)
    time.sleep(temps)
    for dc in range(100, -1, -1):
        p.ChangeDutyCycle(dc)
        time.sleep(0.05)

def peripheral_loop():
    antihorari(TEMPS)
    time.sleep(PAUSA)
    horari(TEMPS)
    time.sleep(PAUSA)

def main():
    try:
        while True:
            peripheral_loop()
    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Segunda simulación donde incluyo los 4 motores, aquí añado otro chip L293D, para poder controlar 2 motores más y tener un total de 4 motores controlados.



Código con el control de los 4 motores y el sensor de seguimiento de línea. El siguiente código lo hare más detallado para dejarlo lo más claro posible. Ya que en este código añado sensores sin probar en el Proteus porque no he podido encontrarlos.

Empezamos como siempre con la configuración de pines (motores y sensores):

```
import RPi.GPIO as GPIO

import time


GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)


SENSOR_IZQ = 18

SENSOR_DER = 4

GPIO.setup(SENSOR_IZQ, GPIO.IN)

GPIO.setup(SENSOR_DER, GPIO.IN)
```

```
IN1, IN2, EN1 = 23, 24, 25
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(EN1, GPIO.OUT)
p1 = GPIO.PWM(EN1, 1000)
p1.start(0)
```

Aplico las funciones de movimiento, aquí decido hacer los giros hacia izquierda y derecha, reduciendo la velocidad de los motores según interés de giro, de esta manera puedo tener un giro más controlado. La otra opción era detener motor, pero de esa manera el giro será más brusco.

```
def avanzar():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    p1.ChangeDutyCycle(70)

def detener():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    p1.ChangeDutyCycle(0)

def girar_izquierda():
    GPIO.output(IN1, GPIO.HIGH); GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH); GPIO.output(IN4, GPIO.LOW)
    GPIO.output(IN5, GPIO.HIGH); GPIO.output(IN6, GPIO.LOW)
    GPIO.output(IN7, GPIO.HIGH); GPIO.output(IN8, GPIO.LOW)

    p1.ChangeDutyCycle(velocidad_giro_interno)
    p2.ChangeDutyCycle(velocidad)
    p3.ChangeDutyCycle(velocidad_giro_interno)
    p4.ChangeDutyCycle(velocidad)
```

```
def girar_derecha():
    GPIO.output(IN1, GPIO.HIGH); GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH); GPIO.output(IN4, GPIO.LOW)
    GPIO.output(IN5, GPIO.HIGH); GPIO.output(IN6, GPIO.LOW)
    GPIO.output(IN7, GPIO.HIGH); GPIO.output(IN8, GPIO.LOW)

    p1.ChangeDutyCycle(velocidad)
    p2.ChangeDutyCycle(velocidad_giro_interno)
    p3.ChangeDutyCycle(velocidad)
    p4.ChangeDutyCycle(velocidad_giro_interno)
```

Seguidamente, la lógica de los sensores de seguimiento de línea:

aquí indicamos con 1→ detecta línea

con 0 →No detecta la línea.

```
while True:
    izq = GPIO.input(SENSOR_IZQ)
    der = GPIO.input(SENSOR_DER)

    if izq == 1 and der == 1:
        avanzar()
    elif izq == 1 and der == 0:
        girar_izquierda()
    elif izq == 0 and der == 1:
        girar_derecha()
    else:
        detener()
```

```
    time.sleep(0.05)
```

Por último, hacemos limpieza de los pines para tener una salida limpia:

```
except KeyboardInterrupt:
    detener()
    GPIO.cleanup()
```

Una vez comprobado el funcionamiento en el simulador, empiezo a llevarlo a la práctica.

Primero empiezo con dos motores, el funcionamiento es correcto los motores responden a lo que el sensor les transmite, pero la velocidad de giro parece insuficiente. La fuente de alimentación utilizada es la Elegoo Power MB de 5V.

→adjunto enlace al video:

https://drive.google.com/file/d/1znDe4wb2bOJLVXb8MK-KrVOqJpWKov4X/view?usp=drive_link

Al usar 4 motores, la fuente es insuficiente para mover todos los motores. Empiezo a investigar si en el código he puesto un valor demasiado bajo y por eso no consigo mover los motores. Subo la frecuencia de 50hz y veo que sigue sin funcionar. Compruebo la potencia que tiene cada motor, 100mA y como necesito 4 motores, necesito una fuente de alimentación que me de 5V, 2ª con 10w. Así que cambio la fuente de alimentación.

→adjunto enlace al video:

https://drive.google.com/file/d/1n_gWcZ1S8VKkXknNp4bJ69MIOmWA09V9/view?usp=drive_link

INSTALACION DE LA LIBRERIA PYSERIAL

Con en el siguiente proceso, podremos conectar el receptor bluetooth a la raspberry.

1. **Conexión de pines:**

- **VCC** → **5 V** de la Raspberry
- **GND** → **GND** de la Raspberry
- **TX (HC-05)** → **GPIO 15 (RX)**
- **RX (HC-05)** → **GPIO 14 (TX)** con divisor de tensión (2 kΩ + 1 kΩ)
- **GND común** es obligatorio.

Si el LED parpadea **rápido (2 veces por segundo)** → está esperando conexión.
Si parpadea **lento (1 vez cada 2 s)** → está conectado.

2. **Abrir la terminal y ejecutar:**

Ejecuta en la terminal:

```bash
sudo raspi-config
```

Luego:

```pgsql
→ Interface Options
→ Serial Port
→ Would you like a login shell to be accessible over serial? → NO
→ Would you like the serial port hardware to be enabled? → YES
```

Guarda, sal y reinicia:

```bash
sudo reboot
```

3. **Comprobar que puerto usar:**

Después del reinicio, escribe:

```bash
ls /dev/tty*
```

Si ves `/dev/ttyS0` o `/dev/serial0`, usa **ese** en tu código Python:

```python
bt = serial.Serial('/dev/ttyS0', 9600, timeout=1)
```

4. **Emparejar y conectar desde Android:**

1. En **Configuración** → **Bluetooth**, elimina el HC-05 si ya estaba guardado.
2. Apaga y vuelve a encender el módulo.
3. Vuelve a emparejarlo (PIN = `1234` o `0000`).
4. Abre la app **Bluetooth RC Controller**.
5. Pulsa **Connect** → **HC-05**.
    - Si el LED parpadea lentamente → conectado correctamente.
    - Si no cambia, no logró conectar (pasa al paso siguiente).

5. **Si después de todo esto no se conecta, hará falta hacer un ultimo paso:**

Si la app sigue sin conectar, prueba con un **test directo**:

```bash
sudo minicom -b 9600 -o -D /dev/ttyS0
```

(minicom se instala con `sudo apt install minicom` si no lo tienes).

Mientras estás ahí, intenta conectar la app Bluetooth.

Si al tocar botones ves caracteres en pantalla → la conexión funciona.

La aplicación que he utilizado ha sido *Serial Bluetooth Terminal*, una vez conectado, podía usar los comandos F,B,L,R,S desde el teclado de la aplicación y los motores respondían correctamente.

Dejo enlace de los códigos completos:

1r código, control bluetooth con los sensores de líneas:

https://drive.google.com/file/d/1RGtZ0wNOfkF7JEDEPUkfAc2ltwct4dI2/view?usp=drive_link

2do código, control solo por bluetooth:

https://drive.google.com/file/d/1xs_CRWxbnOK_sA6YywfYhiysyoimcXnP/view?usp=drive_link

5.Conclusiones

La idea desde el principio no era tener un proyecto totalmente acabado, pero si puedo decir que durante el proceso he aprendido a desarrollarme ante las dificultades y encontrar los resultados que buscaba. El hecho de haber aprendido a controlar motores, tener que soldar, buscar como conectar un receptor bluetooth. Todo eso eran retos nuevos y puedo decir con creces que he conseguido aprender de todo el proceso.

Al empezar el curso, mi conocimiento sobre robótica y programación era cero, con la ayuda de los compañeros y el profesor Joan Masdemont, superar este nuevo reto a sido más fácil.

Quiero dar las gracias a Joan, por su dedicación a mantener el grupo unido y siempre a demostrarnos que somos capaces de todo lo que nos propongamos. También manteniendo los pies en el suelo. Y a los compañeros, siempre muy respetuosos y con una gran amabilidad.

6.Bibliografias y referencias

La mayoría de información a sido obtenida por el ChatGPT.

Proyecto G1 del compañero de un anterior curso:
https://drive.google.com/file/d/1SRs7r9nBlUA_QvBzb0CGPBiLDISqpP11/view?usp=drive_link

Control de motores:

https://drive.google.com/file/d/18pA4LplFIl1uEts6emGGavhR-pIIl3de/view?usp=drive_link

https://drive.google.com/file/d/1OTjxde1oHYnMgeaRh6smqKzTwBym8hVU/view?usp=drive_link