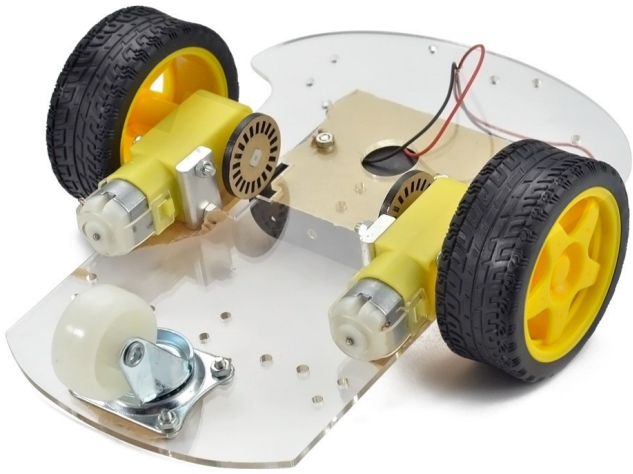
Control de motores DC con mando de PS3



Plácido Antonio López Ávila
placidoantoniolopez96@gmail.com

ÍNDICE

1. Introducción

En esta documentación se tendrá por entendido que conocemos la tecnología que arduino y raspberry proporciona, o al menos tenemos una cierta idea.

Este proyecto solo es una pequeña parte de varios desarrollos que se plantean para obtener una práctica más elaborada. En este en concreto, voy a centrarme en la conexión entre arduino y raspberry a través del puerto Serial para poder enviar información entre ambos. Dejando de lado si este desarrollo es o no el más óptimo, yo he optado por su desarrollo ya que separo el control de motores y sensores a arduino y el computo (este es muy sencillo, pero el de futuros proyectos se complica algo más) a raspberry.

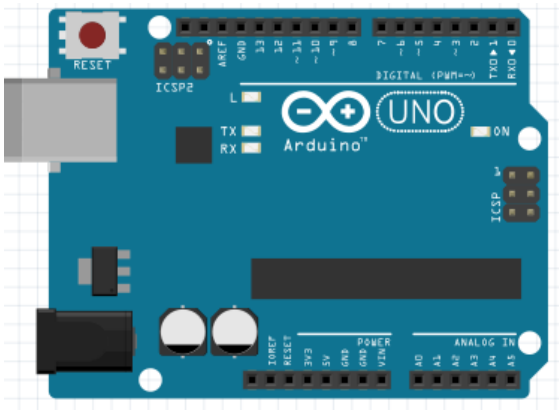
De esta forma, se busca controlar un coche o robot mediante un mando, en mi caso, de ps3. Cabe decir que cualquier dispositivo conectado por usb a la raspberry podría funcionar, un teclado, mandos de otro tipo, bluetooth… Aunque en este último caso sería algo más complejo.

2. Componentes necesarios

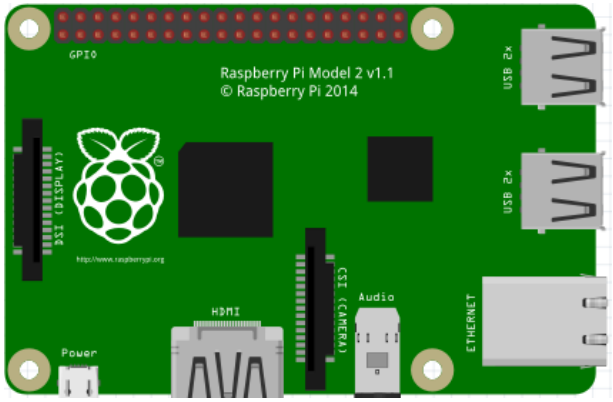
Para comenzar, deberemos disponer de los componentes necesarios para su montaje. En la siguiente lista recojo los empleados, con imágenes al final de la lista:

1. Raspberry Pi 3 B
2. Tarjeta micro-SD 8GB
3. Arduino UNO
4. Puente-H L298N
5. Kit siguelineas
6. 4 Pilas AA
7. Soldador de estaño
8. Switch
9. Mando usb PS3
10. Componentes varios (cables, destornilladores, placa de pruebas…)
11. Teclado, ratón y pantalla (si es la primera vez que conectas la raspberry)

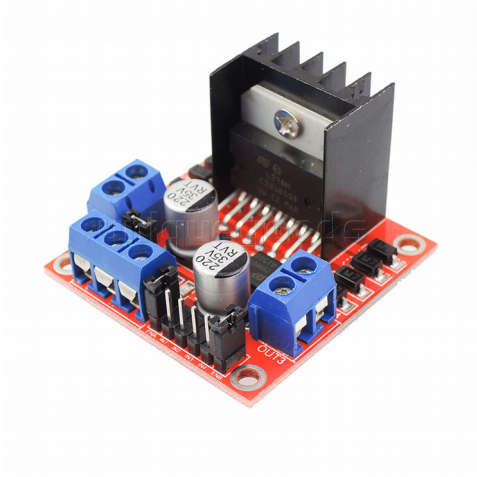
El último punto no es necesario si ya tienes la raspberry configurada con el sistema operativo y SSH activado para conectarte desde otro PC. Como he comentado, no voy a entrar en detalles sobre la instalación del SO y configuración de raspberry, para ello ya hay otros tutoriales.



Arduino UNO



Raspberry Pi 3 B



L298N

Kit siguelineas arduino

Algunas cosas a tener en cuenta son, en la imagen, la raspberry pi indica que es la 2, pero he introducido esa imagen por no encontrar la correspondiente a raspberry pi 3 B. Necesitaremos el cable usb de arduino para comunicarlo con la raspberry. El Kit del siguelineas puede venir con sensores de infrarrojos para la detección de lineas, pero esto en este proyecto no lo vamos a emplear.

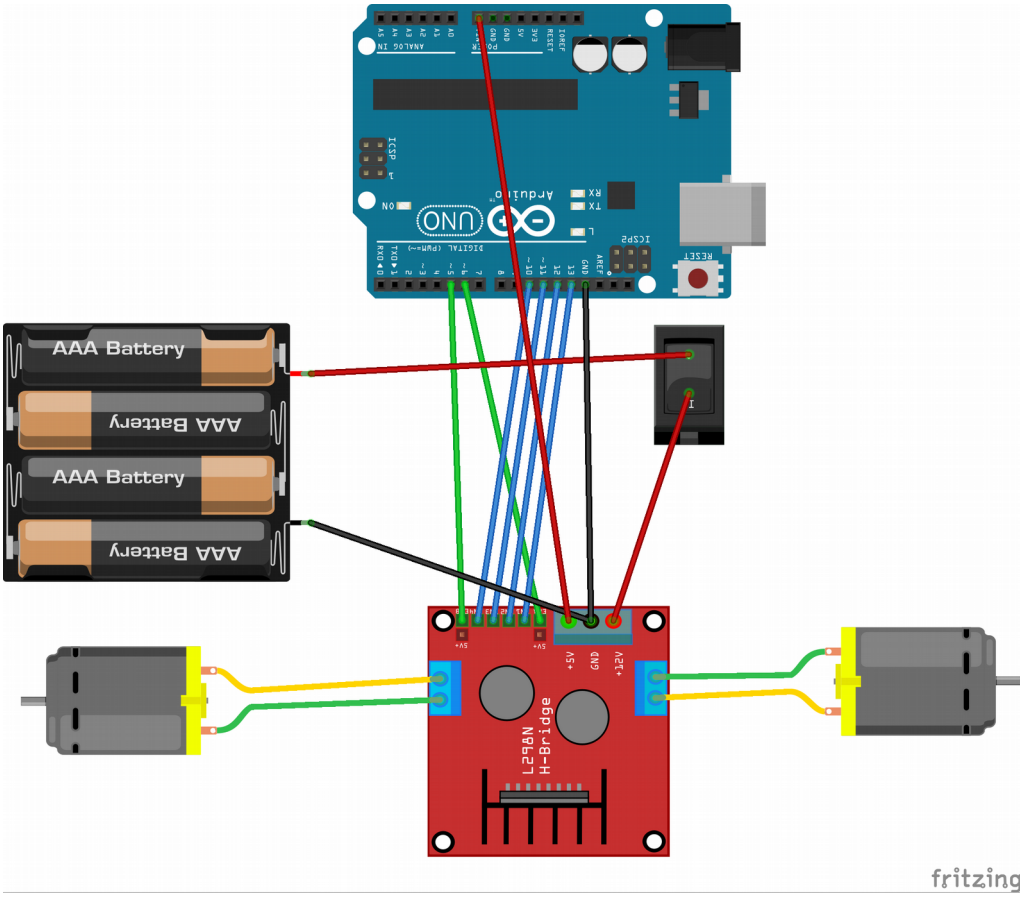
3. Software empleado

Como he indicado antes, la instalación del SO no la voy a comentar, pero si indicar que en mi caso he empleado el sistema operativo Raspbian para el uso de la raspberry y Linux-Mint en mi PC para la instalación del IDE de arduino y la carga del programa, aunque se puede instalar en la raspberry este IDE y cargarlo directamente desde esta.

Para la programación del script de control.py, como es evidente, se ha empleado python. En el dispositivo viene instalada la versión 2.7 pero se puede emplear las versiones posteriores sin ningún problema. Además es necesario instalar la librería evdev, que la instalamos con el siguiente comando.

sudo pip install evdev (si es python 3.*) sudo pip3 install evdev

4. Esquema de desarrollo



En este esquema podemos ver las conexiones realizadas entre arduino y el L298N. Voy a explicarlas brevemente para saber como utilizarlas en la práctica.

Los dos motores se conectan directamente a los extremos del L298N, no importa la polaridad, ya que esto se puede modificar en el código.

El L298N tiene una entrada de 12V a la que conectaremos las baterías, colocando un switch entre esta conexión, para así poder controlar cuando esta o no consumiendo de estas y así ser más sencillo de controlar. Dispone de una salida a tierra (cable negro) que va conectada a la toma de tierra del arduino directamente. También tiene una toma de 5V que la conectamos a la entrada VIN del arduino, pero en nuestro caso, con 4 baterías de este tipo no tenemos suficiente para hacer funcionar el circuito, aun así lo dejamos conectado. Como vamos a conectar el arduino con la raspberry por el puerto Serial esto no nos importa, ya que recibirá el voltaje necesario a través de ahí.

Los cables verdes se tratan de las conexiones con los pines ENA y ENB, que son los que proporcionan la velocidad a los motores, un voltaje de 255 aportan la máxima velocidad y 0 la mínima. Por otro lado los cables azules son las conexiones con los pines IN1… IN4, estos se encargan de proporcionar la dirección de giro a los motores, los IN1 y IN2 a un motor y los otros dos al otro.

Llegados a este punto en el que ya hemos conectado este circuito, queda conectar el arduino a la raspberry por el cable usb y el mando de PS3 a la raspberry por medio del usb.

5. Código empleado

En el github se dispone del código del proyecto, pero hay ciertas cosas que aclarar. Para comenzar en el script control.py:

```
dev = InputDevice('/dev/input/event4')
arduino = serial.Serial('/dev/ttyUSB0',9600)
```

Es necesario establecer la conexión tanto con el arduino (segunda línea) como con el mando de PS3 (primera línea). Cuando insertemos el cable usb del arduino en la raspberry es necesario comprobar el nombre del dispositivo, para hacerlo debemos entrar en el directorio /dev/ y ver el nombre de la conexión con el arduino. En mi caso es ttyUSB0, pero esta puede cambiar al conectar o desconectar el arduino, por lo que es necesario comprobarlo y cambiarlo en este caso.

Para la conexión del mando de PS3, debemos entrar a /dev/input/ y ver los eventos que se crean. Es decir, entramos al directorio sin conectar el mando y hacemos ls, veremos los eventos conectados, cuando conectemos el mando y hagamos ls podremos ver un nuevo evento, este es el evento que debemos indicar en el dispositivo, en mi caso es el event4.

```
If event.type == ecodes.EV_KEY:
```

En esta línea solo se comparan los eventos con codificaciones de la clase EV_KEY. Esto proboca que todos los botones del mando son reconocidos por el programa, a excepción de los joystick y de la cruceta de dirección, aunque para este proyecto me daba igual y he empleado los botones.

En el código de movimientoCoche.ino:

```
if (Serial.available()){
        lectura = Serial.read();
```

La comunicación con la raspberry se realiza a través del puerto Serial por lo que se escribe desde la raspberry en ese puerto y leemos en el arduino de ese mismo puerto.

```
Serial.begin(9600) ;
```

Es imprescindible que los baudios indicados al iniciar el puerto sean los mismos que los del script de python o no se realizará la comunicación de forma adecuada.

A continuación adjunto el código de ambos ficheros, para verlo comentado se puede acceder al github

https://github.com/PlacidoAntonio/RaspberryPi3/tree/master/Arduino%20%2B%20Raspberry%20Pi/Motor_control_with_ps3_control

Control.py

```
import serial

from evdev import InputDevice, categorize, ecodes
dev = InputDevice('/dev/input/event4')

arduino = serial.Serial('/dev/ttyUSB0',9600)

print("Starting...")
print("Control Menu-----------------------------------------------------")
print("1. R2 to go forward")
print("2. L2 to go back")
print("3. SQUARED to turn left")
print("4. CIRCLE to turn rigth")
print("5. START to exit")
print("----------------------------------------------------------------")

print(dev)

for event in dev.read_loop():

    if event.type == ecodes.EV_KEY:
        boton = categorize(event)

        if boton.scancode == 295:
            if boton.keystate == 1:
                print("R2 Clicked...")
                arduino.write("Adelante,")
            else:
                print("R2 Dropped...")
                arduino.write("Parar,")

        if boton.scancode == 294:
            if boton.keystate == 1:
                print("L2 Clicked...")
                arduino.write("Atras,")
            else:
                print("L2 Dropped...")
                arduino.write("Parar,")

        if boton.scancode == 291:
            if boton.keystate == 1:
                print("SQUARED Clicked...")
                arduino.write("Izquierda,")
            else:
                print("SQUARED Dropped...")
                arduino.write("Parar,")

        if boton.scancode == 289:
            if boton.keystate == 1:
                print("CIRCLE Clicked...")
                arduino.write("Derecha,")
            else:
                print("CIRCLE Dropped...")
                arduino.write("Parar,")

        if boton.scancode == 297:
            if boton.keystate == 1:
                print("START Clicked...")
                break
```

```
print("Conexion completed")
```

movimientoCoche.ino

```
#include <SoftwareSerial.h>
SoftwareSerial BT(2, 3);

int ENA = 6;
int IN1 = 13;
int IN2 = 12;

int ENB = 5;
int IN3 = 11;
int IN4 = 10;

int vel = 255;

char lectura;
String comando="";

void setup()
{
  Serial.begin(9600) ;
  pinMode (ENA, OUTPUT);
  pinMode (ENB, OUTPUT);
  pinMode (IN1, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (IN4, OUTPUT);

  comando.reserve(50);
}

void loop(){

  if (Serial.available()){
    lectura = Serial.read();

    if (lectura==','){
       if(comando == "Adelante"){
        Adelante();
        comando = "";
       }else if(comando == "Izquierda"){
        Izquierda();
        comando = "";
       }else if(comando == "Derecha"){
        Derecha();
        comando = "";
       }else if(comando == "Atras"){
        Atras();
        comando = "";
       }else{
        Parar();
        comando = "";
       }
    }else{
      comando += lectura;
    }
  }


}
```

```
void Derecha ()
{
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
  analogWrite (ENA, vel);
  //Direccion motor B
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, vel);
}

void Izquierda ()
{
  //Direccion motor A
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, vel);
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  analogWrite (ENB, vel);
}

void Atras ()
{
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
  analogWrite (ENA, vel);
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, HIGH);
  analogWrite (ENB, vel);
}

void Adelante ()
{
  //Direccion motor A
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, vel);
  //Direccion motor B
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, vel);
}

void Parar ()
{
  //Direccion motor A
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, LOW);
  analogWrite (ENA, 0);
  //Direccion motor B
  digitalWrite (IN3, LOW);
  digitalWrite (IN4, LOW);
  analogWrite (ENB, 0);
}
```

6. Bibliografía

https://www.arduino.cc/

https://www.raspberrypi.org/

https://www.prometec.net/robot-sigue-lineas/

http://python-evdev.readthedocs.io/en/latest/tutorial.html

https://geekytheory.com/arduino-raspberry-pi-raspduino

https://geekytheory.com/arduino-raspberry-pi-lectura-de-datos