# PRÁCTICA DE ROBÓTICA 1 : Motores
# Parte II : Programación de servomotores y plataforma robótica

Carlos Tardón Rubio
Marcos Herrero Agustín
Martín Fernández de Diego

21 de octubre, 2021

**Resumen**

Esta parte está centrada en el control de servomotores. Se realizaron dos experimentos sobre el tema. El experimento I consistió en accionar un servo mediante una señal PWM generada por la RaspBerry Pi y realizar una serie de observaciones. El experimento II requirió montar los servomotores sobre una plataforma y acoplarles sendas ruedas, con lo que obtuvimos la base de nuestro robot. Realizamos una serie de pruebas de desplazamiento en línea recta.

## 1. Objetivos

- Conocer el funcionamiento de los servomotores y el PWM.
- Montaje de los servos en el robot.
- Uso y control de servos en un robot móvil usando la Raspberry Pi.
- Diseño y construcción de una plataforma robótica.

## 2. Desarrollo

### 2.1. Experimento I: Uso de servomotor (actuador del robot) [1]

Primer objetivo: ejecutar un programa en la Raspberry Pi que mueva el servo variando sentido y velocidad con el fin de entender su comportamiento. Procedemos en este caso con el método de prueba y error. El programa que usamos para hacer pruebas con el servomotor se puede encontrar en el anexo A.

**Ejercicios**

**Conectar el motor anterior a la Raspberry Pi y moverlo mediante el programa creado. Verifique el valor necesario que debe utilizar en el programa para mover las ruedas (máximo dos velocidades) en ambos sentidos.**

Con los valores del intervalo abierto (0,14) obtenemos velocidad en sentido horario constante. Con los valores del intervalo abierto (14,100) obtenemos velocidades en sentido antihorario cada vez más lentas a medida que recorremos el intervalo en el orden natural.
Por tanto, tomamos el valor 2 en el sentido horario y los valores 16 y 99 en el sentido antihorario, siendo 16 rápido y 99 lento. Evitamos los valores más cercanos a los extremos 1 y 15 para asegurar el comportamiento deseado.

**Comprobar que se puede parar el motor desde el programa. ¿Se para totalmente?. Justifique los resultados obtenidos.**

Los enteros 0, 100 y 14 paran el servomotor. Los números 0 y 100 paran el servomotor totalmente, esto es por haber configurado previamente el 100 como cota superior. En cambio, el valor 14 no actuó correctamente en todos los casos de prueba: cuando el 14 era el primer valor escrito sobre el servomotor, actuaba como un entero del intervalo (0,14).

A continuación, el montaje.

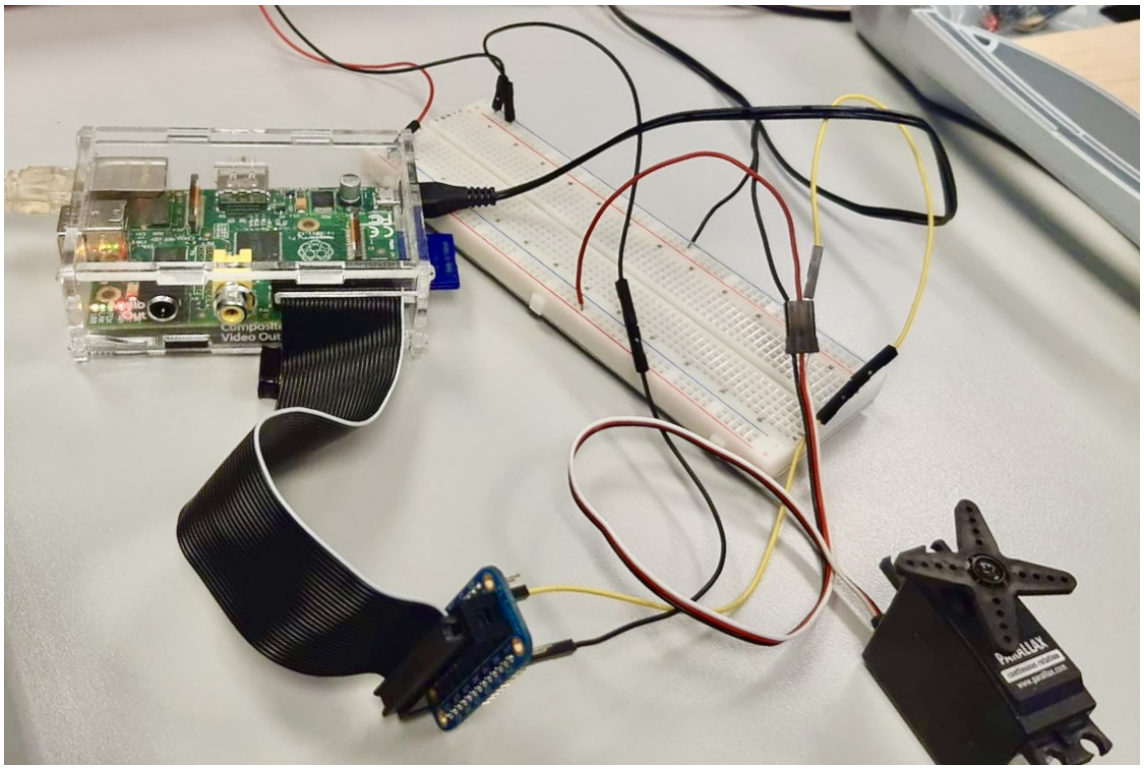

Figura 1: Montaje Raspberry Pi - Servo

En este caso, y para no alterar el rendimiento de la Raspberry Pi, conectamos el servomotor a la salida de 5V del entrenador(cable rojo de la foto). Por ello hemos unido las tierras, es decir, el cable negro que viene de la tierra del entrenador, con el cable negro que procede del GPIO de la Raspberry. La masa y los 5V del servo tienen los mismos colores, como se ve en la imagen(negro masa, 5V rojo). Por último el cable amarillo nos sirve para controlar sentido y velocidad del servo, y se conecta al pin de la raspberry.

**Problemas presentados**

- Al principio no entendíamos bien el programa de ejemplo(softPWM.c). Esto se solucionó estudiando detalladamente el código fuente del programa, y estudiando detalladamente el comportamiento del motor en cada uno de los bucles del programa.

## 2.2. Experimento II: Montaje de los motores en el robot

Hemos planeado como pensamos montar la primera versión de nuestro robot. Hemos adquirido los siguientes materiales:

- Una tabla de contrachapado, para hacer la base del robot
- Dos Power Bank, una para alimentar la raspberry y otra para los servos
- Velcro macho, para unir algunos componentes a la base
- Bridas de distintos tamaños para la sujeción de los motores.

La base del robot estará hecha de contrachapado y tendrá dimensiones 20 cm x 14 cm. Acoplaremos la RaspBerry, la placa de conexiones y las Power Bank a la base utilizando el velcro macho (y aprovechando el que ya viene en la Rasp y en la placa).

Unir los motores a la tabla será más complicado, ya que han de estar firmemente sujetos. Los servos constan de huecos para atornillarlos, pero para utilizarlos requerimos de soportes verticales (perpendiculares a la plataforma). Al final, optamos por

Una vez montado el robot según la descripción anterior, queda realizada el siguiente ejercicio:

**Coloque sobre la plataforma el circuito anterior y la Rasp Pi para mover los motores a la vez**

Para que el robot pueda moverse sin estar conectado, es necesario alimentar el robot y los servos mediante las Power Bank y pasar a controlar la RaspBerry por WiFi. Para extraer Vcc y tierra de las Power Bank conectaremos un cable USB cortado a cada una.

**Ejercicios**

**¿Puede girar también el ángulo que se desee?. Indique cómo lo haría.**

Para girar, el robot utilizará tracción diferencial. Es decir, si está parado, podremos hacer que el robot gire un cierto ángulo en una dirección activando el servo del lado opuesto (el conectado a la rueda opuesta) durante un cierto tiempo . De esta manera, el robot podrá girar el ángulo deseado en la dirección deseada. Si queremos realizar el giro en marcha, ambos servos están a velocidad alta(16), así que basta con disminuir la velocidad(a 99 por ejemplo) del servo de la dirección hacia la cual queremos girar.

# Referencias

[1] "Parallax continuous rotation servo," *https://docs.rs-online.com/870b/0900766b8123f8a1.pdf*. [Online]. Available: https://docs.rs-online.com/870b/0900766b8123f8a1.pdf

# 3. Anexo A: Código

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <wiringPi.h>
#include <softPwm.h>

#define RANGE           100
#define NUM_LEDS          1

int ledMap [NUM_LEDS] = { 0} ;
int values [NUM_LEDS] = {0} ;

int main (){
  int i, j ;
  char buf [80] ;

  wiringPiSetup ()  ;
  for (i = 0 ; i < NUM_LEDS ; ++i){
    softPwmCreate (ledMap [i], 0, RANGE) ;
    printf ("%3d, %3d, %3d\n", i, ledMap [i], values [i]) ;
  }
softPwmWrite (0, 14) ;
softPwmWrite (0, 1) ;
 fgets (buf, 80, stdin) ;
softPwmWrite (0, 0) ;
  fgets (buf, 80, stdin) ;

  for (i = 0 ; i < NUM_LEDS ; ++i)
    for (j = 0 ; j <= 100 ; ++j)
    {
      softPwmWrite (ledMap [i], j) ;
      delay (100) ;
        printf("%d\n",j);
    }

  fgets (buf, 80, stdin) ;
  for (i = 100 ; i >= 0 ; --i){
    for (j = 0 ; j < NUM_LEDS ; ++j)
      softPwmWrite (ledMap [j], i) ;
    delay (20) ;
    fgets (buf, 80, stdin) ;

  for (;;){
    for (i = 0 ; i < NUM_LEDS ; ++i)
      softPwmWrite (ledMap [i], values [i]) ;
    delay (50) ;
    i = values [0] ;
    for (j = 0 ; j < NUM_LEDS - 1 ; ++j)
      values [j] = values [j + 1] ;
    values [NUM_LEDS - 1] = i ;
  }
}
```