Taller 3: Actuación

Robótica Movil

2^{do} cuatrimestre 2025

A continuación se presentan los ejercicios a resolver. Se recomienda fuertemente que ante cualquier duda repasen las diapositivas de la clase.

Cada ejercicio debe realizarse primero en simulación usando Tinkercad y luego con un Arduino UNO y componentes electrónicos reales.

Introdución

En este taller se controlará un motor de corriente continua (CC), en primer lugar a lazo abierto, y a continuación a lazo cerrado. Además, deben bajar de la página de la materia un esqueleto del código, donde pueden encontrarse la configuración inicial para realizar los ejercicios y algunas funciones de utilidad detalladas más adelante. En este código hay una variable llamada `SIMULACION`, que debe ser seteada como 1 para trabajar en el simulador (Tinkercad) o como 0 para trabajar con el arduino y motor real.

Para realizar los ejercicios vamos a seguir la configuración de la Fig. 1. Para el desarrollo de los ejercicios, alimentar el motor con 12V.

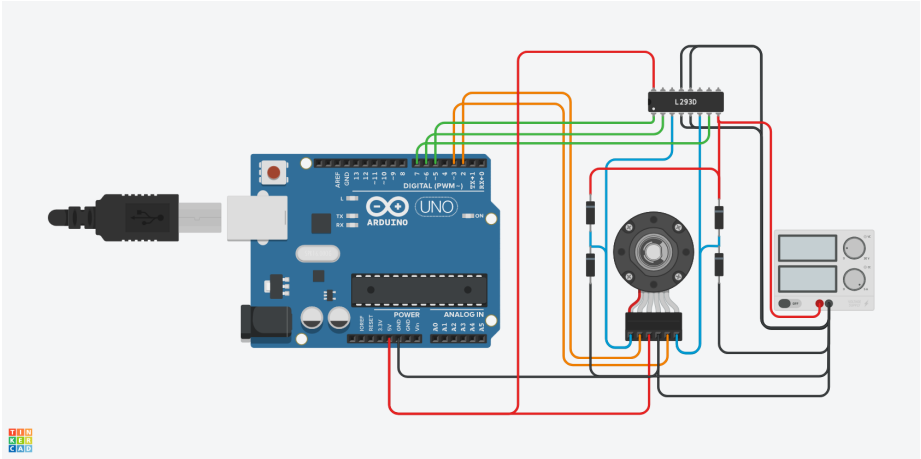

Figure 1: Configuración de Arduino UNO, motor CC y driver L293D.

Integrado L293D

En el simulador se utilizará el integrado L293D como driver del motor. Su modo de uso es el siguiente:

- El pin 1 (*Enable 1 & 2*) recibe la señal de `PWM` que permitirá controlar la velocidad del motor conectado a los pines 3 y 6 (*Output 1* y *Output 2*) del integrado.
- Los pines 2 y 7 (*Input 1* e *Input 2*) determinan la direción de rotación del motor. Para ello, se asignan valores de `HIGH` y `LOW` de forma contrapuesta para determinar el movimiento en sentido horario o antihorario.
- El pin 8 (*Power 2*) debe conectarse a la tensión de alimentación del motor. Ésta tensión depende del motor, nosotros recomendamos entre 5V y 9V para la simulación.
- El pin 16 (*Power 1*) debe conectarse a la misma tensión de alimentación del Arduino.
- Los pines 4, 5, 12 y 13 deben conectarse a tierra (*ground*).

Uso de `PWM` y cálculo de velocidad

Para dar un `PWM` a un motor se utilizará la función `analogWrite(pwm)`. Para leer la posición del encoder se usará la función `encoder_position()`. Esta función devuelve la posición del motor en cuentas de encoder (ce) respecto de la posición inicial. Además, se sabe que la resolución del encoder es de 480ce por vuelta (360^o) en el caso del encoder real y 900ce en el caso simulado (Tinkercad).

Recordar que, dados `pwm_min` y `pwm_max` calculados a partir de la zona muerta de un motor, se necesita definir `pwm_min_pid` y `pwm_max_pid` como cotas mínimas y máximas de saturación de la salida del PID (`pid_output`). Además, tener en cuenta que el valor del PWM con el que se alimentará el motor (`pwm_output`) estará dado por:

$$\texttt{pwm}_{\texttt{output}} = \texttt{pid}_{\texttt{output}} + \texttt{sig}(\texttt{pid}_{\texttt{output}}) \times \texttt{pwm}_{\texttt{min}}$$

donde $\texttt{sig}(\cdot)$ es la función signo.

Ejercicio 1: Control a lazo abierto

Se desea desarrollar un algoritmo que permita controlar un motor a lazo abierto, es decir, asignando un valor de `PWM` al motor sin verificar que la velocidad que sigue es realmente la deseada. De todas formas, queremos saber a que velocidad realmente funciona el motor para imprimir dicho valor en pantalla.

Para resolver el ejercicio es necesario completar en el esqueleto que bajan de la página de la materia la función `void set_motor_pwm(int pwm_consigna)`. Esta función debe asignar el valor del `PWM` correspondiente a los pines del motor, dado un valor de `pwm_consigna` utilizando la función `analogWrite(pwm)`.

Además, para medir la velocidad del motor, es necesario completar la función `float calcular_velocidad(double delta_t)`. Recuerde que puede utilizar la función `encoder_position()`. Dicha función debe devolver la velocidad del motor en (^{o}/ms), por lo que será necesario realizar un cambio de unidades.

Se pide:

a) Implementar las funciones pedidas. Probar el código obtenido asignando distintos valores de `PWM` al motor y visualizando la velocidad con el graficador.

b) Encontrar el valor de `PWM` máximo para el cual el motor no se mueve partiendo del reposo (`pwm_min`), es decir, en qué `PWM` comienza la zona muerta.

c) Encontrar la velocidad promedio en °/ms para los siguientes valores de PWM: -110, 120, -150, 180, -200, 255.

Ejercicio 2: Control a lazo cerrado

En este ejercicio se propone cerrar el ciclo de control, implementando un `PID`. Para ello deberán completar la función `void ciclo_control()` con el algoritmo de `PID` correspondiente. Para ello deberán utilizar la función `float calcular_velocidad(double delta_t)` para obtener la velocidad actual del motor y calcular el valor de `PWM` a asignar para llegar a la consigna. Observar el esqueleto existente de la función y completar donde se indica. Para evaluar el correcto funcionamiento utilice las siguientes constantes: $K_p = 20$, $K_i = 20$, $K_d = 30$. Verifique que el setpoint (referencia de velocidad) elegido tenga sentido. ¿Cúal es la máxima velocidad posible?

A continuación, se pide obtener el gráfico de velocidad versus tiempo para las constantes dadas a continuación y escribir una justificación del comportamiento observado por el motor.

Referencia [°/ms]	Kp	Ki	Kd
2	10	0	0
2	50	0	0
1	20	20	0
1	20	20	30
5	20	25	0