Taller 3: Actuación

Introducción a la Robótica Movil

Segundo Cuatrimestre 2016

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

1 Introdución

En este taller se controlará un motor de corriente continua (CC) a lazo abierto, y a continuación, a lazo cerrado. Se utilizará un `ArduinoUNO` provisto por la cátedra. Además, se puede bajar de la página de la materia un esqueleto del código, donde pueden encontrarse la configuración inicial y algunas funciones de utilidad detalladas más adelante.

precaución: tenga cuidado de no apoyar el `ArduinoUNO` sobre ninguna superficie metálica o conductora, ya que puede poner en corto la placa.

1.1 Conectadose con un Arduino

Lo primero que debemos hacer es enchufar el `ArduinoUNO` a la computadora en algún puerto usb. A continuación es necesario chequear que la configuración del puerto serial sea correcta. Para ello vamos a verificar los siguientes items:

- En la pestaña `Tools`, vamos a verificar que el `Board` coincida con el arduino que vamos a programar, `Arduino/Genuino UNO` en nuestro caso.
- En la pestaña `Tools`, verificamos que el `Port` sea `/dev/ttyUSBX` ó `/dev/ttyACMX` (dependiendo de que `ArduinoUNO` usen), donde X corresponde al número del puerto en el cual conectamos el `ArduinoUNO`.

El chequeo de estos items tendremos que hacerlo cada vez que desconectemos y volvamos a conectar el cable `USB` del `ArduinoUNO`. Para verificar que toda la configuración es correcta, vamos a programar un *Blink*. Para ello vamos a la pestaña `File` y buscamos en `Examples -> 01.Basics -> Blink` . Por último compilamos y cargamos el código con los dos botones que estan arriba a la izquierda. Si todo la configuración es correcta veremos un cartel indicando que la carga fue exitosa y el led de el `ArduinoUNO` parpadeará.

1.2 Simulación

La cátedra proveé un esqueleto que deberá ser completado por los alumnos. Este archivo cuenta con una *MACRO* que permite seleccionar el modo de trabajo, es decir, si se va a simular el motor y el encoder (colocando `#define SIMULACION 1`) ó, por el contrario, si vamos a estar trabajando con el motor y el encoder real

(colocando `#define SIMULACION 0`). El código está estructurado de forma de que funcione para ambos modos.

En las dos formas de trabajo, para dar un `PWM` a un motor se utilizará la función `analogWrite(pwm)`. Por otro lado, 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 480 ce por vuelta (360^o).

1.3 Visualización

Para visualizar los resultados el `ArduinoUNO` envía por puerto serie la información de: velocidad real, consigna en función del tiempo y pwm de salida en función del tiempo. Para levantar y graficar estos datos utilizaremos el siguiente comando en una consola luego de cargar el código:
`./graficador.rb`

2 Ejercicio 1: Control 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 la función `void set_motor_pwm(int pwm_consigna)` en el esqueleto que bajan de la página de la materia. Esta función debe asignar el valor del `PWM` correspondiente a los pines del motor, dado un valor de `pwm_consigna` utilizando la funcionn `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:

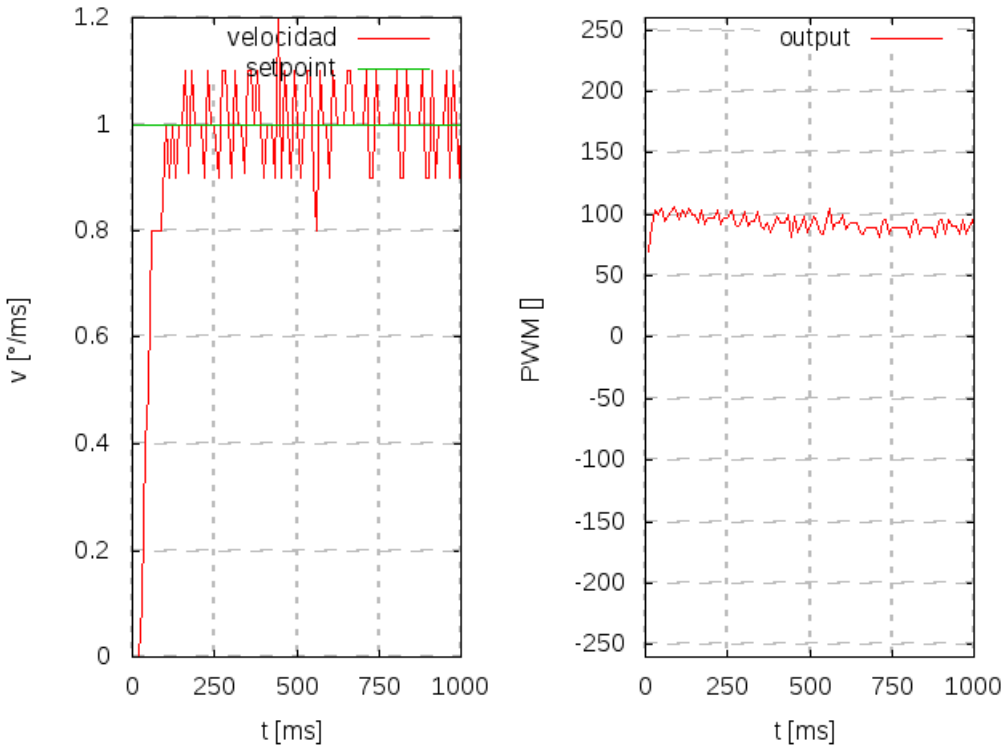
a) Implementar las funciones pedidas. Probar el código obtenido asignando distintos valores de PWM al motor y visualizando la velocidad con el graficador.

b) Encontrar el valor de `PWM` máximo para el cual el motor no se mueve partiendo del reposo ($\texttt{PWM}_{min}$), es decir, en qué `PWM` comienza la zona muerta.

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

3 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: $K_p = 20, K_i = 20, K_d = 30$.

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.

setpoint	Kp	Ki	Kd
2	10	0	0
2	50	0	0
1	20	20	0
1	20	20	30
5	20	25	0