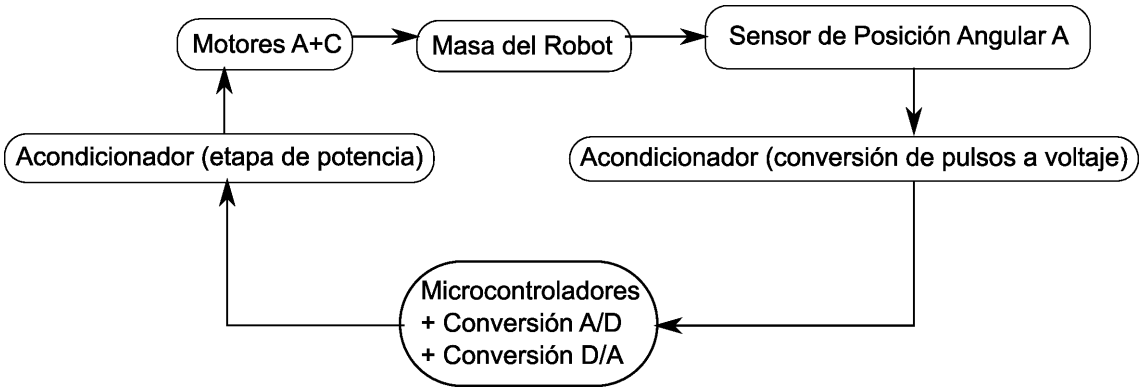
PROGRAMACIÓN DE ROBOTS

Programación de microbots
PRÁCTICA 2.- Control en bucle abierto/cerrado. Seguimiento de líneas

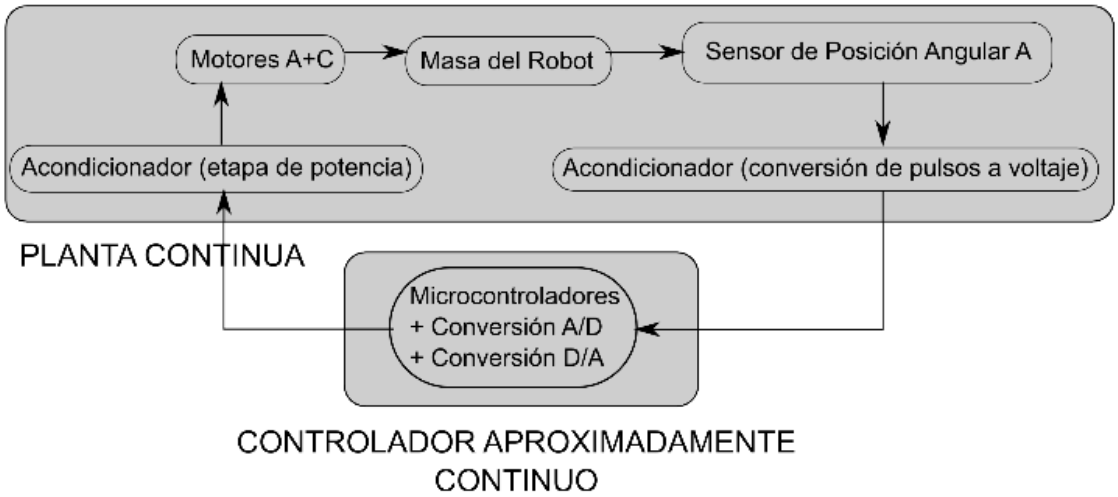
El objetivo de la práctica es doble. Por una parte, se experimentará con el control en bucle abierto y en bucle cerrado; por otra parte, se realizará un seguimiento básico de líneas.

1.- Control en bucle abierto vs. control en bucle cerrado

Vamos a controlar una planta que será el robot Lego para que se desplace en línea recta una cantidad de centímetros dados. La planta tendrá como entrada una fuerza de desplazamiento hacia adelante, producida por el actuador, que es el conjunto de los dos motores (a los dos les daremos la misma potencia para que actúen como un sólo actuador). Tendrá como salida lo que el robot ha avanzado en línea recta, pero para simplificar los cálculos tomaremos en su lugar como salida los grados que ha girado cualquiera de las ruedas del motor (por ejemplo la del motor A). La figura inferior muestra la planta a controlar con todos los elementos del robot Lego (acondicionadores, muestreo, mantenimiento, etc. que están en los microcontroladores y demás circuitería interna).



En realidad, en este sistema controlado, no tenemos acceso a la señal que los motores le dan a la masa del robot (fuerza), ni a la señal de posición que le da esta masa al sensor de posición angular (ángulo real de giro), con lo que es habitual agrupar actuador+planta+sensor en un sólo sistema compuesto y tratar a éste como la planta a controlar. En nuestro caso el agrupamiento es más amplio, porque tampoco tenemos acceso a la señal que produce la etapa de potencia ni a la señal de pulsos que produce el sensor de posición angular, con lo que también se pueden incluir en la planta a controlar los dos acondicionadores.
Por otra parte, podemos considerar que las etapas involucradas en la conversión A/D y D/A en el robot Lego (muestreador, cuantificador, etc.), no cometen un error apreciable en sus procesos correspondientes, y si además el algoritmo de control que corre en el microcontrolador empotrado se ejecuta lo suficientemente rápido (= el tiempo de muestreo es pequeño), llegamos a la conclusión de que se pueden tomar estos componentes como un sistema controlador prácticamente continuo. Con todas estas conclusiones, el sistema (continuo) a controlar quedaría como se muestra:

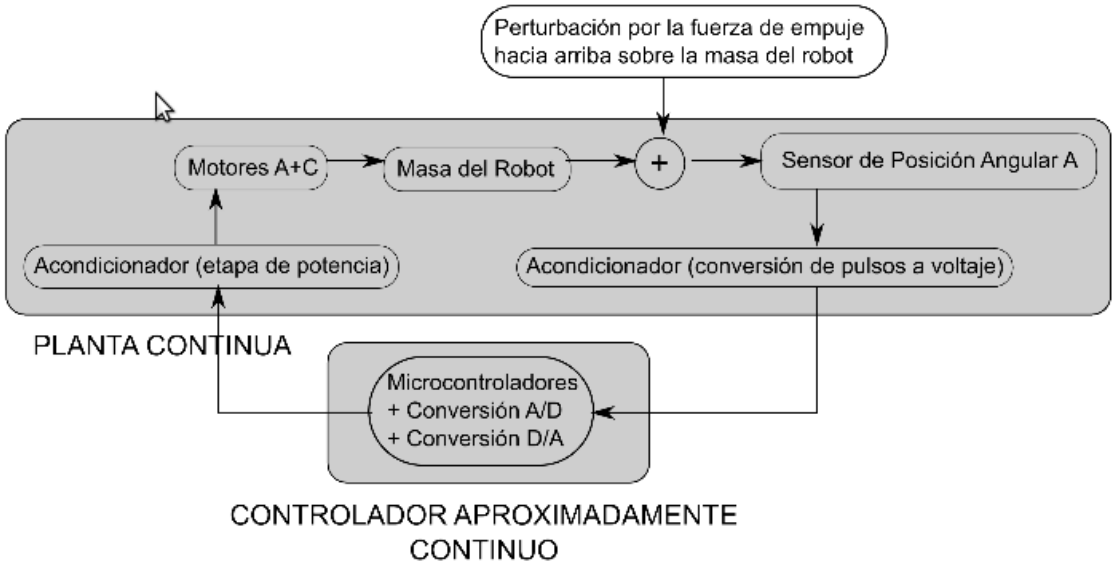


Por tanto, el objetivo de esta práctica es controlar por software la planta continua de la figura superior para que alcance (tenga unos grados recorridos finales en la rueda del motor A un valor de salida de 860 grados. Implanta los siguientes controladores:

a) *Control en bucle abierto (eliminamos de la figura superior la flecha que lleva la señal de salida de la planta hacia el controlador, para que el programa de control que ejecuta la CPU no lea el valor de esa salida a la hora de decidir su acción de control)*. Para esto escribe un programa en NXC que envíe una orden de potencia igual y constante a ambos motores (OnFwd(OUT_AC,<p, potencia de 0 a 100>) y la mantenga durante dos segundos, tras lo cual detendrá simultáneamente los dos motores y mostrará en pantalla el valor que lea en ese momento de la salida de la planta, es decir, del motor A. Muestra ese valor en pantalla durante tres segundos para que puedas observarlo bien. Nótese que esta única lectura de la salida de la planta no cierra el bucle de control: la lectura del sensor se hace después de controlar el motor por motivos de monitorización. Una vez escrito y volcado al robot el programa, pon el robot en el suelo y ajusta manualmente el valor de p hasta conseguir que cuando el robot se pare los ángulos girados sean los 860 requeridos más/menos un error que nos vamos a permitir de 10 grados. Después de obtener por primera vez un valor de p que logre eso, ejecuta tres veces más el mismo programa de control, con ese mismo valor, para asegurarte de que realmente has diseñado un controlador en bucle abierto correcto.

b) *Control en bucle cerrado (en este caso el programa de control que ejecuta el microcontrolador tiene acceso en todo momento al valor de la salida de la planta para decidir su acción de control).* Escribe ahora un segundo programa que actúe como controlador, consistente en un bucle software de 2000 iteraciones (es decir, sólo va a hacer control durante ese tiempo) que, en cada iteración, calcule el resultado de restar la salida deseada (860) de la salida leída en ese momento de la planta. Esa diferencia se le enviará directamente a ambos motores para cerrar el bucle de control, pero como éstos sólo admiten valores entre -100 y 100, si esa diferencia es mayor de 100, acótala a 100, y si es menor de -100, acótala a -100. Después envíasela directamente como orden de potencia a los dos motores a la vez: si es positiva, usa OnFwd(OUT_AC,<diferencia acotada>); si es negativa, usa OnRev(OUT_AC,-<diferencia acotada>) (el signo menos es para ponerla en positivo en este caso). Al final de cada iteración haz una pausa de 1 milisegundo para que la suma de todas las iteraciones corresponda más o menos a los 2 segundos que duraba el programa de control en bucle abierto del apartado a). Tras salir de todas las iteraciones, muestra en pantalla el último valor leído de la salida de la planta, y comprueba cuánto se desvía ese valor de la salida deseada de 860 grados: ejecuta tu programa situando el robot en el suelo y anota el valor final de esa salida, repitiendo esto tres veces. ¿Qué errores comete tu programa de control? ¿Mayores o menores que el controlador en bucle abierto? OBSERVA QUE NO HAS TENIDO QUE AJUSTAR NINGÚN PARÁMETRO A MANO EN TU CONTROLADOR DE BUCLE CERRADO PARA CONSEGUIR ESTE RESULTADO; SÓLO TENÍAS QUE CONOCER LA SALIDA DESEADA DE LA PLANTA.

c) Comparación y efectos de cada tipo de control ante perturbaciones imprevistas (ver figura inferior). Ahora vuelve a ejecutar el controlador de bucle abierto del apartado a) pero con el robot sostenido en el aire con la mano (equivalente a ejercer una fuerza hacia arriba que anula el peso del robot y por tanto le produce una perturbación en su movimiento horizontal -le hace andar más-puesto que los motores ya no tienen que arrastrar la masa). Ejecútalo tres veces y anota el valor final de posición angular que alcanza cada vez. ¿A qué crees que son debidos estos errores respecto a la salida deseada que sí conseguiste en el apartado a)? A continuación, ejecuta con el robot en el aire el controlador de bucle cerrado del apartado b). ¿Qué error tienes con este controlador cuando hay perturbaciones respecto a cuando no las tenías -robot en el suelo-? ¿Por qué?



2.- Seguimiento de líneas

Vamos a tratar de que el sensor de luz del robot se mueva en línea recta pero al mismo tiempo siguiendo lo más fielmente posible una franja de un cierto nivel de gris dibujada en la hoja. Esta franja producirá perturbaciones incesantes en la lectura del sensor de luz si el robot siguiera una línea recta, pues es una elipse; por tanto necesitamos un control en bucle cerrado para adaptar el movimiento del sensor de luz a esas perturbaciones. En nuestros experimentos, la banda de gris que el sensor queremos que mantenga debajo será la señal de referencia R=55.

El algoritmo de control (o ley de control) bang-bang genérico tiene dos parámetros: U y P (U es la "banda de histéresis", a la que daremos valores bajos, inferiores a 5-10; P es la "potencia ejercida"). En nuestro sistema, partiendo del error e(t)=R-L(t), la ley de control para este controlador será: si e(t)<-U, es que el sensor ve algo que es más oscuro de lo deseado, así que se le dará al motor izquierdo (mirando desde la parte trasera del robot hacia la delantera) una potencia negativa -P (OnRev(...,P)), y al derecho una positiva P (OnFwd(...,P)), para que el robot gire sobre sí mismo hacia donde hay valores más claros de luz; si e(t)>U se hará lo contrario, dando al motor izquierdo una potencia P y al derecho -P; en cualquier otro caso no hay perturbaciones y se le dará a los dos motores la misma potencia positiva P, para que el robot siga el nivel de gris alcanzado en línea recta. Tu algoritmo repetirá esta ley de control en un bucle durante 2000 iteraciones. Al final de cada iteración hará una espera de 10 milisegundos para que éste sea más o menos el período de muestreo. Después de salir de las iteraciones detendrá los dos motores y dará el experimento por concluido.

En cada iteración tienes además que guardar en un array el tiempo t (puedes medir el tiempo t en

milisegundos con CurrentTick(), restándolo del tiempo inicial para que el control empiece en el tiempo 0), y en otro array el valor leído por el sensor de luz. Cuando termine tu programa, grabará un fichero de texto con esos datos (t en la primera columna, luz en la segunda).

Una vez escrito el controlador bang-bang, grábalo en el robot y sitúa el sensor de luz sobre el centro (negro) del círculo, alineando el robot en el eje mayor de la elipse del papel. Sujeta bien el papel durante el experimento para que los giros del robot no lo muevan ni éste deslice, y ten cuidado de que no haya cables que puedan enrollarse cuando el robot gire. Haz varios experimentos de control ejecutando tu programa en esas condiciones iniciales, para valores de P de 15, 20, 30, 40, y 50. Después de cada experimento, pasa el fichero de muestras de tiempo y luz al PC.

¿Qué ocurre para valores bajos de P? ¿Y para valores altos?