B3 – Actuadores

Objetivo

El objetivo de esta sesión de práctica es aprender a realizar programaciones adecuadas para manejar varios tipos de actuadores: (i) un *microservo*, (ii) un *motor de corriente continua* y (iii) un *relé*.

Conceptos básicos sobre motores

La mayoría de los motores funcionan gracias al fenómeno de **inducción**: cuando un cable conduce corriente, se genera un campo magnético a su alrededor.

Existen varios datos a tener en cuenta cuando utilizamos motores en circuitos: uno de ellos es el **voltaje** al que pueden funcionar eficientemente, señalado por el fabricante. Con ese voltaje el motor podrá girar a su máxima velocidad. A más voltaje, corre el riesgo de quemarse. A menor voltaje, el motor girará a menor velocidad.

Otro dato importante es el consumo de **corriente** que tienen. Éste depende sobre todo de la carga que arrastran: a más carga, más corriente. Cada motor tiene una "**corriente de paro**", que es la corriente que consume cuando su giro se para por una fuerza opuesta. La corriente de paro es mucho más grande que la **corriente de giro**, que es la corriente consumida cuando no hay carga. Puede haber motores que consuman (durante un breve tiempo) casi su corriente de paro al arrancar, debido a la inercia de pasar de estar parados a moverse. Para evitar problemas, la fuente de alimentación debe poder ofrecer la corriente de paro y algo más. Si no puede ofrecerla, entonces se deben utilizar transistores o relés, para amplificar la corriente aportada por la fuente y así alimentar el motor convenientemente.

Todos los circuitos inductivos (como los motores), además de inducir ("crear") un campo magnético debido a la circulación de corriente eléctrica, también funcionan en el sentido contrario. Es decir, si existe un campo magnético *variable* alrededor de una bobina, se induce una corriente eléctrica a su través. Por tanto, si tenemos un motor girando y lo apagamos, la variación de campo magnético produce una corriente en sentido contrario al de la corriente utilizada para mover el motor. Esta corriente puede ser muy intensa y dañar la electrónica. Por ello, casi siempre se conecta un diodo (polarizado en inversa) en paralelo al motor, para parar esta corriente. Este diodo se suele llamar diodo "*fly-back*".

Servos de rotación limitada

Los *servos* son motores DC con engranajes (para limitar la velocidad y aumentar el torque) que también incorporan un potenciómetro y circuitería de control, para poder establecer la posición del eje del motor de forma precisa. El eje de un servo no gira libremente (como lo hace el de los motores DC convencionales) sino que rota un ángulo determinado, definido mediante una señal de control. Lo que hace especial a un servo es que podemos ordenarle que gire una cantidad de grados concreta, definida por la señal de control enviada en un momento dado.

Los servos disponen normalmente de tres cables: uno para recibir la alimentación eléctrica, otro para conectarse a tierra y otro para transmitir al servo, de parte del microcontrolador, los pulsos eléctricos que ordenarán el giro concreto de su eje. En la gran mayoría de servos, los pulsos de control son de frecuencia fija (50 Hz). El cable de alimentación se debe conectar a una fuente capaz de proporcionar 5 V y al menos 1 A. El cable de tierra se debe conectar a la tierra común del circuito. El cable de control se debe conectar a algún pin digital de la placa Arduino, por el cual se enviarán los pulsos que controlarán el desplazamiento angular del eje. (El pin digital donde se conecta el cable de control del servo NO tiene por qué ser de tipo PWM.)

El voltaje de trabajo de los *servos de tipo "hobby"* (los más pequeños) es de entre 5 V y 7 V, de manera que, en principio, los podemos alimentar con la propia placa Arduino. No obstante, el consumo eléctrico de un servo es proporcional a la carga mecánica que soporta su eje. Dependiendo de la carga que se coloque al servo, puede ser necesario utilizar una fuente externa de 5 V adicional, para proporcionarle una alimentación separada de la ofrecida a través de la placa Arduino (pero con la tierra común siempre). La fuente adicional también será necesaria cuando empleemos más de dos servos en nuestros circuitos, tengan la carga que tengan.

El desplazamiento angular del eje de un servo está determinado por la *duración* de los pulsos de la señal de control. Si el valor ALTO (5 V) del pulso se mantiene durante **1,5 milisegundos**, el eje del servo se ubicará en la **posición central** de su recorrido. Como los servos estándar permiten mover su eje en ángulos dentro de un rango entre 0 y 180 grados, esta posición central suele corresponder a **90 grados** respecto al origen. Es decir, si al servomotor se le envía una señal con pulso de 1,5 ms, el eje girará hasta estar situado en un ángulo de 90 grados respecto al origen (por tanto, a mitad de su recorrido total). Mientras la señal de control recibida por el servo sea siempre la misma, éste mantendrá la posición angular de su eje. Si la duración del pulso de la señal varía, entonces el servomotor girará hasta la nueva posición.

Si el ancho del pulso está **entre 1,5 y 2 milisegundos**, el eje del servo se moverá hasta una posición angular proporcional **entre 90 y 180 grados** del origen.

Si el ancho del pulso está **entre 1 y 1,5 milisegundos**, el eje del servo se moverá a una posición angular proporcional **entre 0 y 90 grados** del origen. Si queremos situarlo a 0 grados (origen), la longitud del pulso debe ser de 1 ms (duración mínima).

Servos de rotación continua

También existen *servos de rotación continua*. A estos servos especiales, no se les puede establecer su ángulo de giro, pero sí su *velocidad de giro*. Podemos cambiar su *sentido del giro* sin necesidad de ninguna circuitería extra.

La señal de control de un servo de rotación continua está formada por pulsos *cuadrados*, generalmente a una frecuencia de 50 Hz. Cuando la duración del valor ALTO (5 V) de un pulso se mantiene durante **1,5 ms**, un servo de rotación continua permanece **parado**. A medida que aumenta esa duración, su velocidad de giro aumenta *en un determinado sentido*, hasta llegar a la velocidad máxima cuando el valor ALTO del pulso llegue a durar un **máximo de 1,7 ms** (generalmente). Si, por el contrario, la duración del valor ALTO del pulso es menor de 1,5 ms, el giro se producirá en el *sentido contrario*, y su velocidad irá en aumento a medida que la longitud del pulso disminuya, hasta llegar a durar un **mínimo de 1,3 ms** (generalmente).

Librería *Servo*

La librería *Servo* sirve para facilitar al programador de la placa Arduino el control de motores servo. Permite manejar hasta 12 servos en la placa Arduino UNO (y hasta 48 en la Arduino Mega). El uso de esta librería en la placa Arduino UNO deshabilita la funcionalidad PWM de los pines 9 y 10 aunque no haya ningún servo conectado allí.

Lo primero que hay que hacer para poder controlar un servomotor utilizando esta librería es declarar un objeto de tipo *Servo*. Si suponemos que lo llamamos "*miservo*", esto se haría con la línea: ***Servo miservo;*** en la zona de declaraciones globales.

A partir de aquí, la primera función imprescindible que debemos escribir (normalmente dentro de "*setup*()") es:

miservo.attach()

Vincula el objeto "miservo" con el pin digital de la placa Arduino donde está conectado físicamente el cable de control del servomotor. Su parámetro es precisamente el número de ese pin.

A partir de aquí, ya podemos hacer uso del resto de funciones de la librería Servo para controlar los servomotores de nuestro circuito.

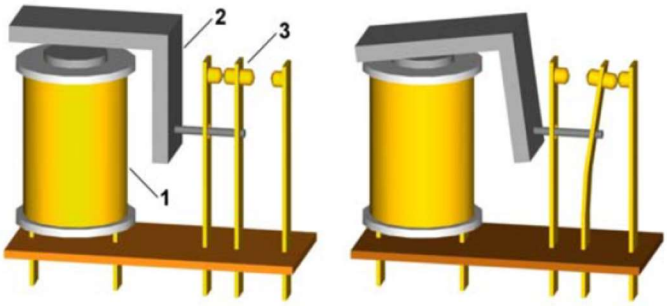
miservo.write()

Controla el eje del motor servo. Su único parámetro es el ángulo (en grados) donde se quiere situar dicho eje. Como resultado de ejecutar esta instrucción en un servomotor estándar, se obtendrá un movimiento del eje hasta alcanzar ese ángulo especificado. En un servomotor de rotación continua, en cambio, el valor de su parámetro representa la velocidad del eje: 0 equivale a la máxima velocidad en un sentido, 180 a la máxima velocidad en el sentido contrario y 90 equivale aproximadamente a inexistencia de movimiento. Esta función no tiene valor de retorno.

Relés

Un relé es un interruptor mecánico controlado eléctricamente. Puede controlar mucho más voltaje y corriente que un pin Arduino. Si quieres usar Arduino para controlar una bombilla de 120 V, una cafetera u otro dispositivo de alta potencia, un relé es una excelente manera de hacerlo. Debido a que el relé necesita más energía para conmutar que la que puede proporcionar un pin Arduino, se usa frecuentemente un transistor para controlar un relé.

Un relé consta de una bobina de hilo conductor y contactos de interruptor. Cuando se aplica energía a la bobina, se imana y (como consecuencia) se cierran los contactos del interruptor. Los contactos del interruptor están completamente aislados del Arduino, por lo que puedes usar un relé de forma segura para controlar voltajes que normalmente son peligrosos.



El relé tiene tres pines de contacto, COM (común), NC (*normally closed*) y NO (*normally open*). Cuando el relé no recibe energía, el pin COM está conectado al pin NC (*normally closed*). Cuando el relé recibe energía, el pin COM está conectado al pin NO (*normally open*).

Bloque repetitivo "*while*"

El bloque "*while*" es un bloque que implementa un bucle, es decir, repite la ejecución de las instrucciones que están dentro de sus llaves de apertura y cierre. ¿Cuántas veces? No hay un número fijo: se repetirán mientras la condición especificada entre sus paréntesis sea cierta ("true", 1). Su sintaxis es muy sencilla:

```
while (condición) {
      //Instrucciones que se repetirán mientras la condición sea cierta –"true",1–
}
```

Instrucción *Serial.available()*

El puerto serie de Arduino se puede utilizar para recibir y para enviar datos. Arduino almacena los datos que entran por el puerto serie hasta que esté listo para usarlos. La instrucción *Serial.available()* devuelve la cantidad de caracteres que recibió el puerto, pero que aún no ha utilizado el sketch. Cero significa que no ha llegado ningún dato.

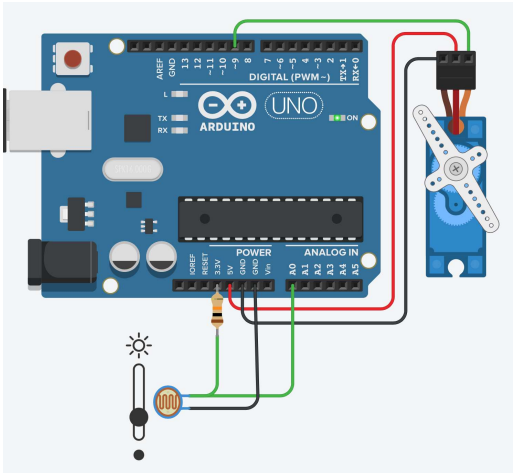
Instrucción *Serial.parseInt()*

Cuando se escriben caracteres numéricos en el puerto serie, la instrucción *Serial.parseInt()* extrae (*parses*) números enteros de los caracteres que recibe. Por ejemplo, si se escribe "1" "0" "0" en el puerto serie, la instrucción devolverá el número 100.

Prácticas a realizar

B3a – Microservo SG90

Implementar el montaje de la figura que emplea un microservo (que simula la regulación de la altura de una persiana) y una fotoresistencia LDR (que detecta las condiciones ambientales de iluminación).

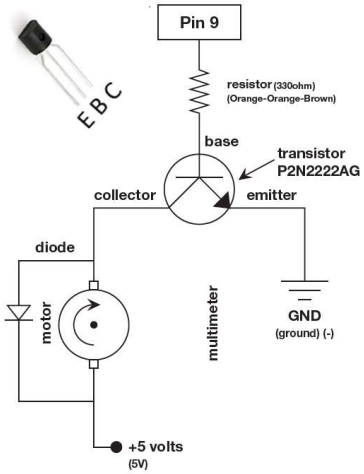


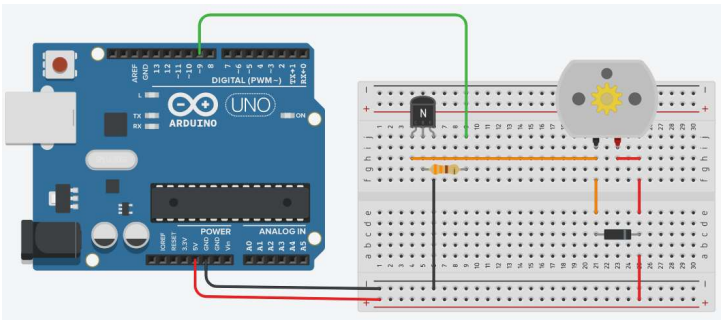
Se pide, mediante programación:

- Controlar el ángulo de giro del microservo (entre 0 y 170°) de manera que sea función de las condiciones ambientales de iluminación.
- El ángulo de giro del microservo debe ser 0° en condiciones de máxima iluminación y 170° en condiciones de mínima iluminación. Utilizar la función *map()*.

B3b – Motor de corriente continua

Se trata de hacer girar un motor de corriente continua de diferentes maneras, utilizando una señal PWM para controlarlo [función *analogWrite()*]. Es preciso usar un transistor (ver figura), para poder cortar y dejar pasar mayor cantidad de corriente que la que Arduino permite. También se utiliza un diodo (ver figura) para proteger el motor frente a polarización errónea.



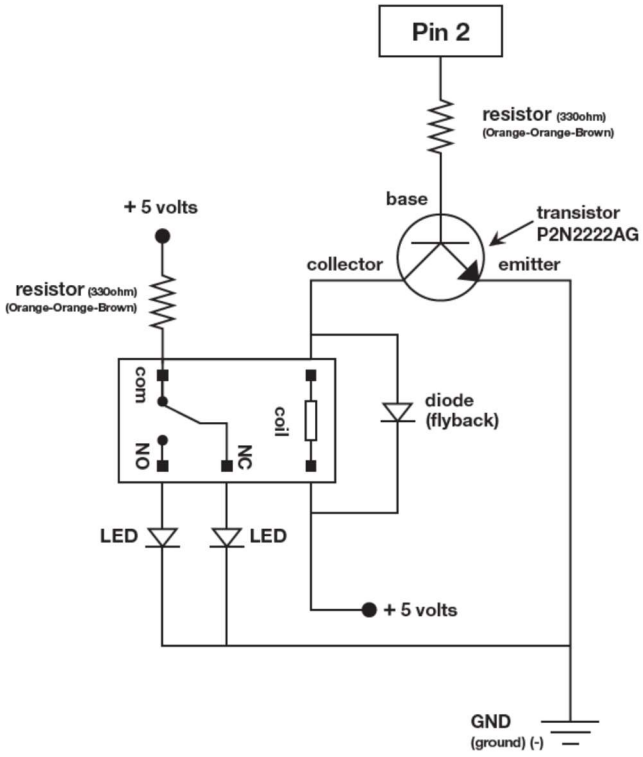


Se pide:

- Implementar el montaje de la figura. (Utilizar un trozo alargado de papel, a modo de aspas de un ventilador, para hacer visible el giro.)
- Utilizando funciones propias, programar los siguientes tres modos de funcionamiento del motor:
    - "Motor parpadeante". De forma similar al "led parpadeante", se trata de hacer que el motor gire y se pare, a intervalos, de forma repetitiva.
    - (Opcional) Se trata de "acelerar" el motor hasta máxima velocidad (incremento continuo de la velocidad) y "decelerarlo" hasta pararlo (decremento continuo de la velocidad).
    - (Opcional) Se trata de introducir un valor de velocidad (de 0 a 255) a través del puerto serie y hacer que el motor gire a velocidad proporcional.

B3c – Relé

Implementar el montaje de la figura.



Se pide, mediante programación:

- Hacer que se iluminen los LEDs alternativamente, a intervalos de 1 segundo. Se trata de un funcionamiento de tipo “relé parpadeante,” semejante al del “led parpadeante”. En funcionamiento, se debe poder escuchar el clic de los contactos del relé. (Utilizar la función *digitalWrite()* para conmutar el transistor.)