MÓDULO 4: PROJECTO 12 " Mecanismo de bloqueo secreto"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Introducción

- Un zumbador piezoeléctrico normalmente crea vibraciones, pero esta vez lo va a usar como entrada para detectar cuántas veces alguien llama a tu puerta. Puede tomar las cosas un paso más allá y abrir automáticamente una caja con un servomotor si alguien toca el número correcto de veces.

- El altavoz piezoeléctrico puede ser utilizado como elemento de entrada. Cuando le suministramos una alimentación de 5V, el sensor puede detectar vibraciones que serán leídas por las entradas analógicas de Arduino. Para que funcione correctamente, deberemos conectar una resistencia de valor elevado (1 MΩ por ejemplo) como referencia a tierra.

- Cuando colocamos el altavoz contra una superficie sólida que pueda vibrar, como por ejemplo, una mesa de madera, nuestro Arduino puede notar como de intenso es un golpe o vibración. Utilizando esta información, podemos comprobar si se han dado una serie de golpes dentro de un rango aceptable. Dentro del código de programación, podemos contabilizar los golpes detectados y comprobar si corresponden a nuestros intereses.

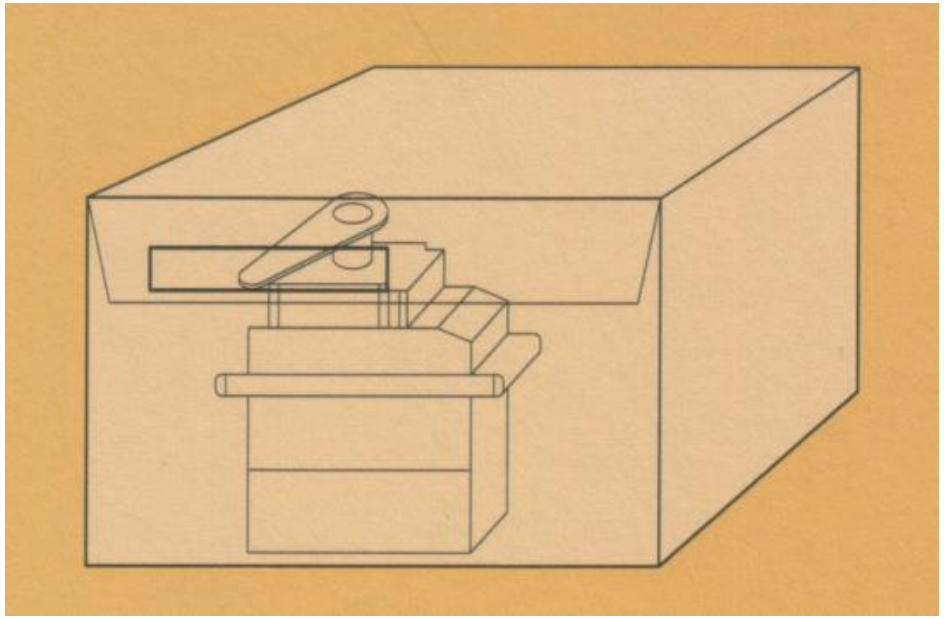
Introducción

- Un pulsador nos permitirá mantener al motor en su lugar. Por otro lado, usaremos tres leds para indicar el estado de la cerradura:
  - Led rojo: indica que la caja está cerrada.
  - Led verde: indica que la caja está desbloqueada.
  - Led amarillo: indica si se ha detectado un golpe de forma correcta.

- Escribiremos nuestra propia función para comprobar si el impacto recibido ha sido o demasiado fuerte o demasiado flojo. Escribiendo nuestra propia función conseguiremos ahorrar mucho tiempo de programación, ya que podemos reusar dicho código y no tener que estar reescribiéndolo una y otra vez. Las funciones pueden recibir argumentos y devolver valores. En nuestro caso, utilizaremos una función para detectar la intensidad del golpe y si dicho impacto está dentro de un rango preestablecido, incrementaremos el valor de una variable.

Introducción

- Podemos construir el circuito sin necesidad de acoplarlo a una caja o una puerta, pero es mucho más divertido ver como bloquea y abre dicho objeto. Si tenemos una caja de cartón la podemos usar para nuestro proyecto. Haremos una abertura que servirá para que un servo motor bloquee o desbloquee dicha caja.



Montando el circuito

1. Conectamos la alimentación y la tierra a la protoboard. Ponemos el pulsador en la protoboard conectando una de sus patillas a la alimentación de 5V y la otra a tierra mediante una resistencia de 10KΩ. Por último, hacemos una conexión entre la unión del pulsador con la resistencia y el pin digital 2 del Arduino.

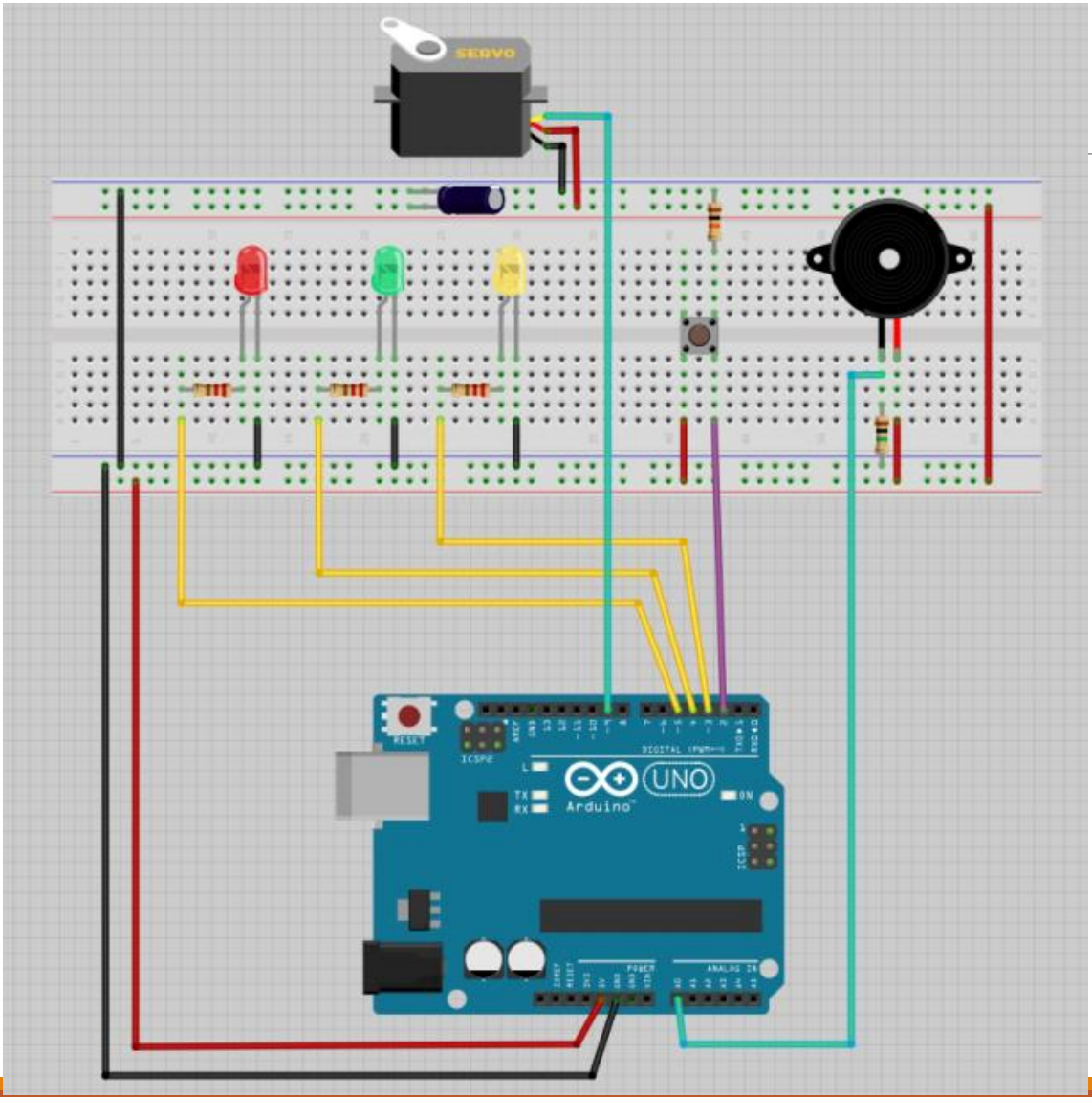
2. A continuación conectaremos el altavoz. Si dicho altavoz tiene un cable rojo o uno marcado con el símbolo "+", dicho cable debe ir unido a la alimentación. Si por el contrario, nuestro altavoz no tiene polaridad, da igual en que sentido lo conectemos. El cable que nos queda, irá unido al pin analógico A0. Finalmente, colocamos una resistencia de 1MΩ que vaya desde la unión del altavoz con el pin A0, hasta la toma de tierra. El hecho de que la resistencia tenga un valor tan elevado, es simplemente porque las resistencias de valores bajos hacen que el altavoz sea menos sensible a las vibraciones.

3. El paso siguiente es colocar los leds. Dichos componentes los conectaremos a tierra por el cátodo (la patilla corta). En cambio, por el ánodo los conectaremos en serie a una resistencia de 220Ω. A continuación y a través de su respectiva resistencia en serie, conectaremos el led amarillo al pin digital 3, el verde al pin digital 4 y el rojo al pin digital 5.

Montando el circuito

1. Ahora le toca el turno al servo motor. Tras conectarlo a la protoboard utilizando el adaptador, uniremos el cable rojo al positivo de la alimentación y el negro a tierra. Seguidamente, colocaremos el condensador de 100µF en paralelo al servo motor, o lo que es lo mismo, una patilla a la alimentación y la otra a tierra. Recordemos que dicho condensador tiene polaridad, por lo que tenemos que fijarnos bien a la hora de conectarlo. Al posicionar el condensador en paralelo al servo motor, conseguiremos evitar que este se vea afectado por irregularidades en la tensión. Por último, conectaremos el cable de datos al pin 9 de Arduino.

Montando el circuito



Componentes	
3	**Resistencias de 220Ω.**
1	**Resistencia de 10KΩ.**
1	**Resistencia de 1MΩ.**
3	**Diodos led (uno rojo,uno verde y uno amarillo).**
1	**Pulsador.**
1	**Altavoz piezoeléctrico**
1	**Condensador de 100µF.**
1	**Servo motor.**
1	**Adaptador para conectar el servo motor a la protoboard.**

Comentario del código

- Lo primero que vamos a hacer es importar la librería **Servo** y crear una instancia de la misma. De esta manera podremos beneficiarnos de las funcionalidades que nos proporciona.
- A continuación creamos unas constantes para hacer referencias a las entradas y a las salidas.
- Creamos variables para almacenar los valores proporcionados por el altavoz y el pulsador.
- Estas dos constantes nos servirán para establecer el nivel máximo y mínimo de los golpes a detectar.

Comentario del código

- La variable locked nos dirá si la cerradura se encuentra cerrada o no. Dicha variable será de tipo boolean (booleano en castellano).
- El tipo de dato **boolean** solo puede tener dos estados, cierto (valor 1) o falso (valor 0). Nosotros empezaremos con el mecanismo desbloqueado. La última de las variables de ámbito global será la encargada de almacenar el número de golpes correctos que se hayan detectado.
- Dentro de la función **setup()**, asociaremos el servo motor al pin 9. A continuación, estableceremos los pins de los leds como salidas y los del pulsador como entradas.

Comentario del código

- Lo siguiente que vamos a hacer es inicializar la comunicación serie con el ordenador, de esta manera podremos monitorizar el volumen de los impactos, cual es el estado actual de la cerradura y cuantos golpes restantes quedan. Activamos el led verde, movemos el servo motor a la posición de bloqueado y mostramos en el monitor serie el estado actual indicando que el circuito se encuentra en la posición de bloqueado.

- Dentro de la función **loop()**, lo primero que tenemos que hacer es comprobar si la caja está abierta o no. Esto determinará lo que ocurrirá en el resto del programa. Si esta cerrada, leeremos el valor del pulsador.

- Si el pulsador está cerrado (porque lo estamos presionando), cambiaremos el valor de la variable locked a true, indicando que la cerradura está cerrada. Apagamos el led verde y encendemos el rojo.

Comentario del código

- Si el monitor serie no se encuentra activado, sería de gran ayuda que lo estuviera, ya que así podemos ver el estado en el que se encuentra la cerradura. Movemos el servo a la posición de bloqueado y enviamos un mensaje al monitor serie indicando que la caja ahora está cerrada. Finalmente, añadimos un delay para que el servo motor tenga tiempo suficiente de posicionarse.

- Si la variable locked tiene el valor true y la cerradura está cerrada, leeremos el valor de vibración proporcionado por el altavoz y lo almacenamos en la variable **knockVal**.

- El siguiente bloque de código comprueba si tenemos menos de tres golpes válidos y si hay alguna vibración en el sensor (recordemos que el sensor es el propio altavoz piezoeléctrico). Si ambas condiciones son ciertas, comprobamos si el golpe actual es válido o no, e incrementamos la variable **numberOfKnocks**.

Comentario del código

- Llamaremos a nuestra función personalizada, cuyo nombre es **checkForKnock()**. El contenido de dicha función ya lo escribiremos al terminar el bucle **loop()**. De momento lo que debemos saber, es que a la función le preguntaremos si el golpe detectado ha sido valido o no, para ello le pasaremos como argumento la variable knockVal. Después de comprobar el valor devuelto por la función, mostraremos un mensaje por el monitor serie indicando el número de golpes válidos que aún nos hacen falta.

- Comprobaremos si tenemos tres o más golpes válidos. Si dicha condición es cierta, cambiaremos el valor de la variable locked a false y movemos el servo a la posición de desbloqueado. Esperamos unos milisegundos a que el motor comience a moverse y cambiamos el estado de los leds verde y rojo. Finalmente, mostraremos un mensaje en el monitor serie indicando que la caja está d esbloqueada.

Comentario del código

- Es el momento de implementar la función **checkForKnock()**. Cuando estamos creando funciones de nuestra autoría, tenemos que indicar si va a devolver un valor o no. Si no va a devolver ningún valor, declararemos la función como tipo **void**, del mismo modo que están declaradas las funciones **loop()** y **setup()**. Si la función va a devolver algún valor, debemos indicar de que tipo de valor se va a tratar (**int**, **long**, **float**, etc.). En nuestro caso, vamos a utilizar dicha función para comprobar si un golpe ha sido válido (true) o no (false), por lo que declararemos la función como tipo boolean.

- Nuestra función comprobará un número (la variable knockVal) para ver si es correcta o no. Para poder pasar el contenido de dicha variable a la función, crearemos un parámetro con un nombre al declarar dicha función. A dicho parámetro lo llamaremos **value**.

Comentario del código

- Dentro de la función **checkForKnock()**, siempre que hagamos referencia al parámetro **value**, estaremos haciendo referencia al número que hemos recibido como argumento desde el programa principal. O lo que es lo mismo, **value** tendrá el mismo contenido que **knockVal**. Comprobaremos si value es mayor que el límite de sonido mínimo (**quietKnock**) y menor que el límite de sonido máximo (**loudKnock**).

- Seguidamente, si el valor que acabamos de comprobar está dentro del rango que nos interesa, quiere decir que es un golpe válido. Haremos parpadear el led amarillo una vez y mostraremos el valor del golpe correcto por el monitor serie.

- Para decirle al programa el resultado de la comparación, usaremos el comando **return**. El comando **return** se utiliza para finalizar la ejecución de la función, una vez invocado, volvemos al programa principal.