Parte 2

Lección 3

Entradas Digitales

Resumen

En esta lección, usted aprenderá a utilizar los botones con entradas digitales para encender y apagar un LED.

Presionar el botón se encenderá el LED; pulsar el otro botón se apagará el LED.

Componente necesario:

(1) x Elegoo Uno R3

(1) x protoboard de 830 puntos de amarre

(1) x LED rojo de 5mm

(1) x resistencia de 220 ohmios

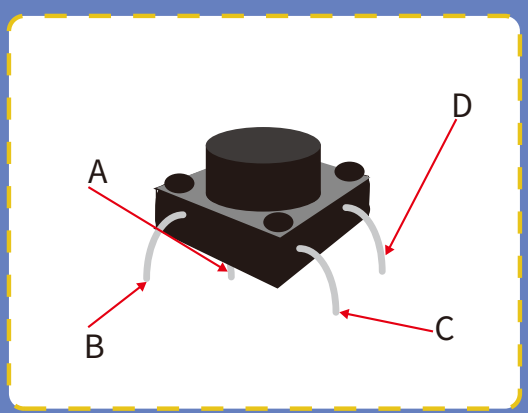
(2) x interruptores de presión

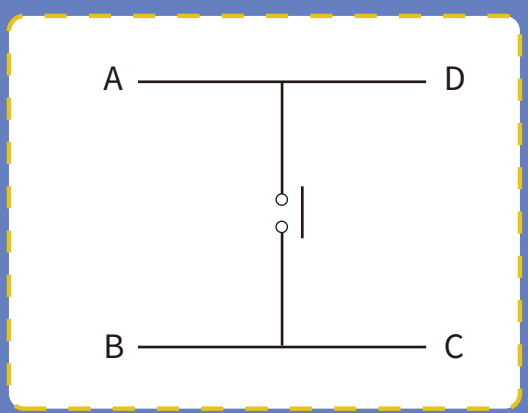
(7) x M M cables (cables de puente de macho a macho)

Introducción del componente

INTERRUPTORES DE EMPUJE:

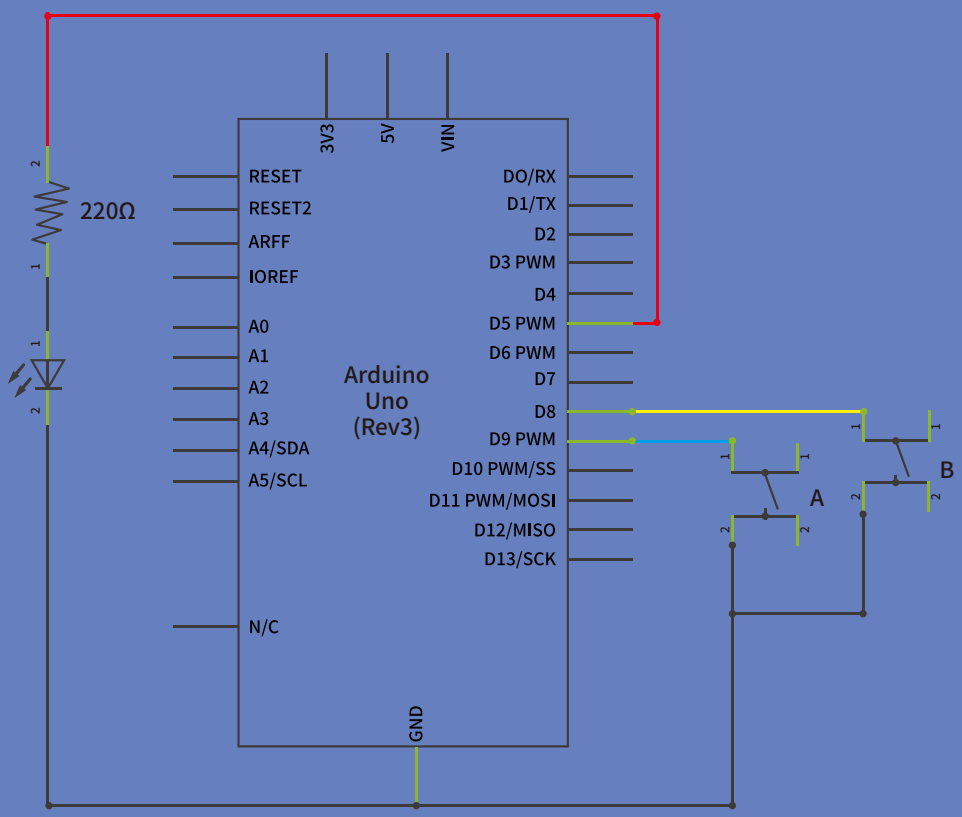
Los interruptores son componentes muy simples. Cuando pulse un botón o girar una palanca, conectan dos contactos para que la electricidad fluya a través de ellos.Los interruptores táctiles poco utilizados en esta lección tienen cuatro conexiones, que pueden ser un poco confusas

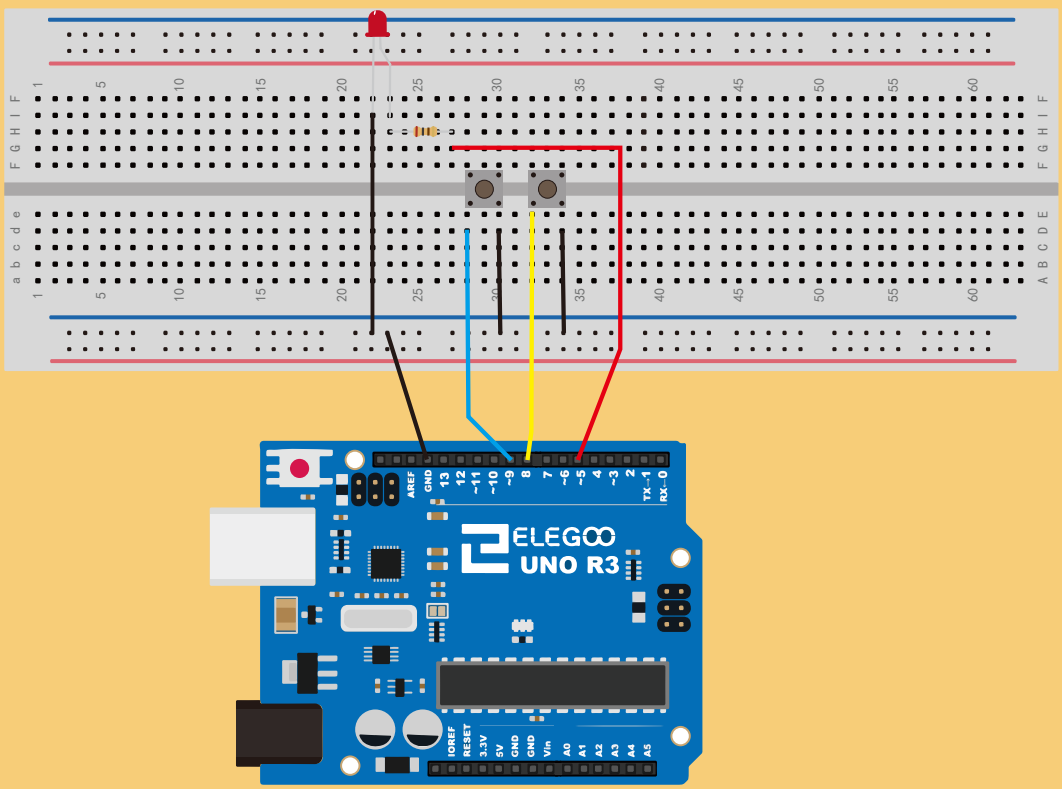




En realidad, hay realmente dos conexiones eléctricas. Dentro del paquete de interruptor, pins B y C se conectan entre sí, como son A y D.

Conexión Esquema

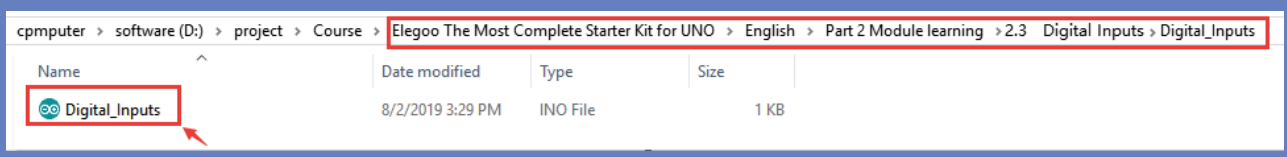




- Aunque los cuerpos de los interruptores son cuadrados, los pasadores sobresalen de los lados opuestos del interruptor. Esto significa que los pines sólo estarán lo suficientemente separados cuando se colocan correctamente en la placa de pruebas.
- Recuerde que el LED tiene que tener el cable negativo más corto a la derecha.

Código

Abra el programa.



- El bosquejo en su placa UNO de carga. Presionando el botón izquierdo se encenderá el LED mientras que pulsando el botón derechoapagará.
- La primera parte del proyecto define tres variables para ls tres patas que se van a utilizar. El 'ledPin' es el pin de salida y 'buttonApin' se refiere al interruptor más cerca de la parte superior de la placa y 'buttonBpin' para el otro interruptor.
- La función de 'configuración' define el ledPin como una salida normal, pero ahora tenemos las dos entradas para ocuparse. En este caso, utilizamos el conjunto el pinMode ser 'INPUT_PULLUP' como este:

```
pinMode (buttonApin,INPUT_PULLUP);
pinMode (buttonApin,INPUT_PULLUP)
```

- El modo pin de INPUT_PULLUP significa que el pin debe ser utilizado como una entrada, pero que si nada mas se conecta a la entrada, se debe 'sacarse' a alta. En otras palabras, el valor predeterminado de la entrada es alta, a menos que se tiró bajo por la acción de pulsar el botón.
- Por esta razón los interruptores están conectados a tierra. Cuando un interruptor se presiona, se conecta la clavija de entrada a la tierra, para que ya no es alta.
- Puesto que la entrada es normalmente alta y va sólo baja cuando se pulsa el botón, la lógica es un poco boca abajo. Nosotros nos encargaremos de esto en la función 'loop'
- En la función 'loop' hay dos declaraciones de 'si'. Uno para cada botón. Cada uno hace un 'digitalRead' en la entrada adecuada.
- Recuerde que si se presiona el botón, la entrada correspondiente será baja, si el botón A es bajo, entonces un 'digitalWrite' en el ledPin enciende.
- Del mismo modo, si se presiona el botón B, un bajo se escribe en elledPin.

```
void loop()
{
if (digitalRead(buttonApin) == LOW)
{
digitalWrite(ledPin, HIGH);
}
if (digitalRead(buttonBpin) == LOW)
{
digitalWrite(ledPin, LOW);
}
}
```

- ①If {} : La instrucción “IF” verifica una condición y ejecuta la instrucción o conjunto de instrucciones en base a si la condición es 'verdadera'.

```
Syntax
if (condición) {
  //declaracion(es)
}
```

- **Parámetros**

condition: una expresión booleana (es decir, puede ser verdadera o falsa).

- ②digitalRead(): Lee el valor de un pin digital especificado, que puede ser ALTO o BAJO.
- **Sintaxis**

digitalRead(pin)

- **Parámetros**

pin: Número de pin de la placa Arduino que quiere leer

- **Devuelve**

ALTO o BAJO

==

[Comparison Operators]

Description

Compares the variable on the left with the value or variable on the right of the operator. Returns true when the two operands are equal. Please note that you may compare variables of different data types, but that could generate unpredictable results, it is therefore recommended to compare variables of the same data type including the signed/unsigned type.

Syntax

```
x == y; // is true if x is equal to y and it is false if x is not equal to y
```

Parameters

x: variable. Allowed data types: int, float, double, byte, short, long.

y: variable or constant. Allowed data types: int, float, double, byte, short, long.