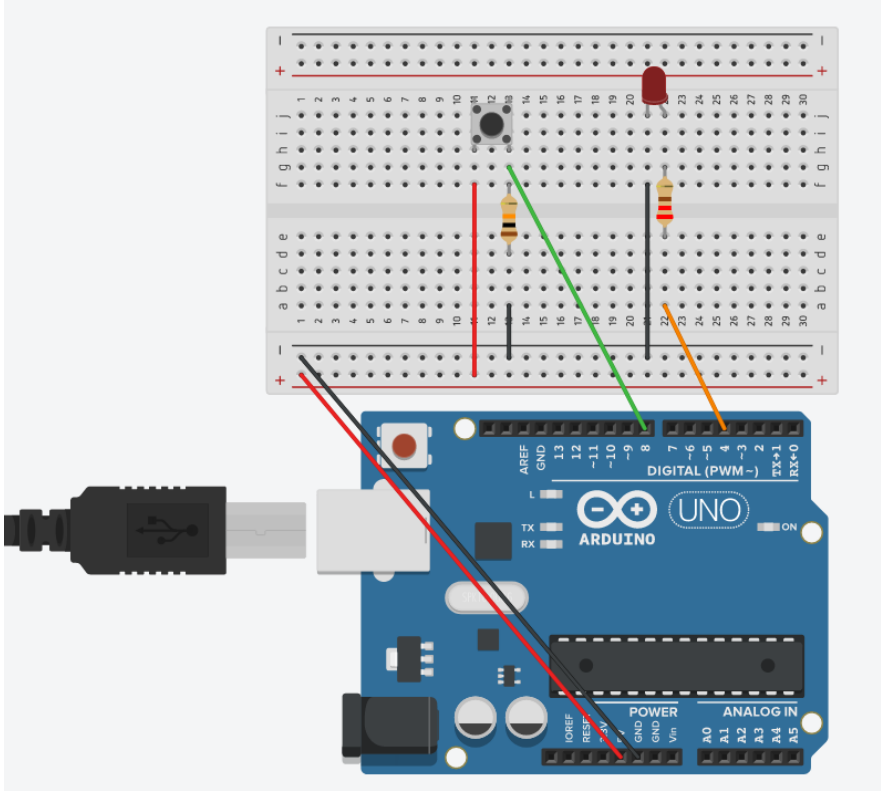
ARDUINO

Practica 4, ENTRADA DIGITAL



Se pretende controlar el encendido de un led mediante un pulsador.

Conectamos el circuito según el esquema: en las resistencias de nuestro kit escogemos una de colores:

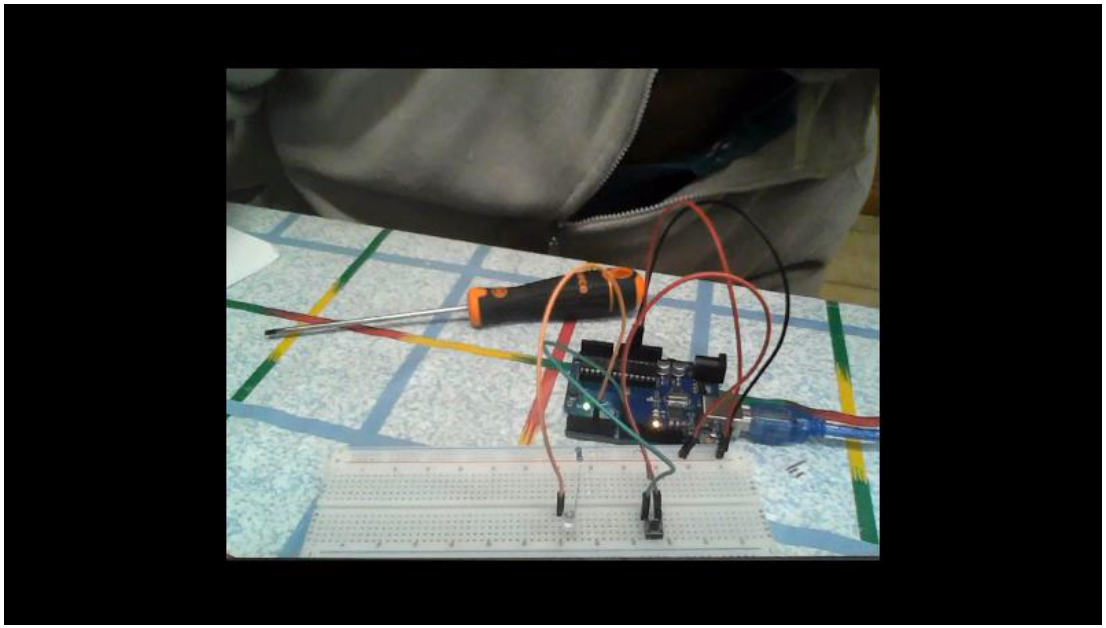
La resistencia del pulsador de: 10Kohmios (KΩ)
Marrón-negro-negro-rojo-tolerancia=
1_0_0x100 = 10,000 Ω = 10K Ω

La del diodo de: 220Ω
rojo-rojo-negro-negro-tolerancia=
2_2_0x1 = 220 Ω

Aclaraciones

La variable que habrá que declarar es el valor actual del pulsador, el cual puede variar entre 1 (5voltios) ó 0 (0voltios).

En el montaje se utiliza una resistencia de pull-down, para entender bien su función basta con desconectarla y observar como el led cambia de luminosidad y parpadea, aunque no hayamos pulsado el pulsador. Esto es debido a el ruido que capta la entrada digital cuando esta leyéndose, y no esta conectada a ningún sitio.



De la programación

if (condicion) {……..}

Comenzamos con la instrucción **if**. Seguido, y entre paréntesis, escribiremos una condición.
Si esta condición se cumple, será validada y por tanto se ejecutará el código que pongamos entre las llaves **{…….}**. En caso contrario, el programa continuará en la siguiente instrucción sin ejecutar lo encerrado entre llaves.

Estructura de control if – else

Una variante a la estructura de control if en arduino, es cuando usamos **else**, que nos permite la posibilidad de ejecutar unas instrucciones de programa concretas tanto si se cumple como si no, la condición que hayamos puesto inicialmente.

```
if (condicion) {.....
            instrucciones si se cumple
          }
      else  {.....
            instrucciones si no se cumple
          }
```

Estructura de control if () – else if ()– else if ()

```
 if (condición)
            {          instrucciones que se ejecutan si se cumple                    }
 else if (condición)
            {          instrucciones que se ejecutan si no se cumple la primera y se cumplen estas          }
 else if (condición)
            {          instrucciones que se ejecutan si no se cumple  la primera, ni la segunda y se cumple esta
}
```

De la programación: operadores lógicos

&& (AND lógico)
Verdadero si ambos operandos son verdaderos, por ejemplo

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) {   // lee dos interruptores
 // ...
 }
```

es verdadero solo si ambas entradas están en estado **HIGH**.

|| (OR lógico)
Verdadero si cualquiera de los operandos es verdadero, por ejemplo:

```
if (x > 0 || y > 0) { // ... }
```

es verdadero si x ó y es mayor que 0.

! (NOT lógico)
Verdadero si el operando es falso, por ejemplo:

```
if (!x) {   //... }
```

es verdadero si x es falso (por ejemplo, si x es igual a 0).

De la programación: operadores de comparación

Los operadores de comparación disponibles en Arduino son las siguientes

== (igual que)

!= (no igual que)

< (menor que)

> (mayor que)

<= (menor o igual que)

>= (mayor o igual que)

De la programación: tipos de datos

- **boolean** (8 bit)- lógico simple verdadero/falso.
- **byte** (8 bit)- número sin signo entre 0 y 255.
- **char** (8 bit)- número con signo, entre -128 y 127. En algunos casos el compilador intentará interpretar este tipo de dato como un caracter, lo que puede generar resultados inesperados.
- **unsignedchar** (8 bit)- lo mismo que ‘byte’; si es que eso es lo que necesitas, deberías usar ‘byte’, para que el código sea más claro.
- **word** (16 bit)- número sin signo entre, 0 y 65535.
- **unsignedint** (16 bit)- lo mismoque ‘word’. Utiliza ‘word’ por simplicidad y brevedad.
- **int** (16 bit)- número con signo, entre -32768 y 32767. Este tipo es el más usado para variables de propósito general en Arduino, en los códigos de ejemplo que vienen con el IDE.
- **unsignedlong** (32 bit)- número sin signo entre 0 y 4294967295. Este tipo se usa comúnmente para almacenar el resultado de la función millis(), la cual retorna el tiempo que el código actual ha estado corriendo, en milisegundos.
- **long** (32 bit)- número con signo, entre -2,147,483,648 y 2,147,483,647.
- **float** (32 bit)- número con signo, entre 3.4028235E38 y 3.4028235E38. El Punto Flotante no es un tipo nativo en Arduino; el compilador debe realizar varios saltos para poder hacerlo funcionar. Evítalo siempre que te sea posible. Hablaremos de eso más tarde; En una fecha próxima se publicará un tutorial más riguroso en el uso genérico de la matemática de punto decimal en Arduino.

El programa: condición IF – ELSE *SI - SINO*

```
                                        // declaramos las constantes que vamos a utilizar, el led y el pulsador
const int led = 4;                      // el led conectado a la salida 4, // con un igual = asignamos valores o posiciones
const int pulsador = 8;                 // el pulsador conectado a la entrada 8

                                        // y la variable del circuito
boolean estadoPulsador;                 // esta variable solo toma dos valores UNO ó CERO

void setup() {
  // put your setup code here, to run once:
 pinMode(led, OUTPUT);                  // configuramos led del pin4, como SALIDA
 pinMode(pulsador, INPUT);               // configuramos el pulsador del pin 8, como ENTRADA
                }

void loop()
                {
      estadoPulsador = digitalRead(pulsador);   // leemos el estado del pulsador (0 ó 1
                                        // ahora encendemos el led si el pulsador estaba pulsado, lo hacemos con la condición IF
                if (estadoPulsador == HIGH)     // con dos iguales == hacemos comparaciones de igualdad
                {
                 digitalWrite(led, HIGH);       // como se cumple la condición activamos el led
                }
                                else
                                {
                                digitalWrite(led, LOW);
                                }
                }
```