# Sistemas con Microprocesadores

# **Práctica 1:** Primeros diseños con Arduino y el simulador Tinkercad. Entradas/Salidas

## Resumen:

Realizamos una primera toma de contacto con la programación de placas Arduino y el simulador *Tinkercad* que se utilizará durante prácticas sucesivas.

## Entrega:

Entregamos un fichero PDF de 2 páginas aproximadamente que contenga:

- Nombre y apellidos
- Dirección web del proyecto (o proyectos) Tinkercad. Ver último apartado de este guion.
- Respuesta a la pregunta del guion.
- Captura de pantalla del funcionamiento del circuito con los botones sin resistencias de *pull-up* externas y con el led (ejercicio 6), y código fuente de este circuito.
- Cálculo del valor de la resistencia para el led.

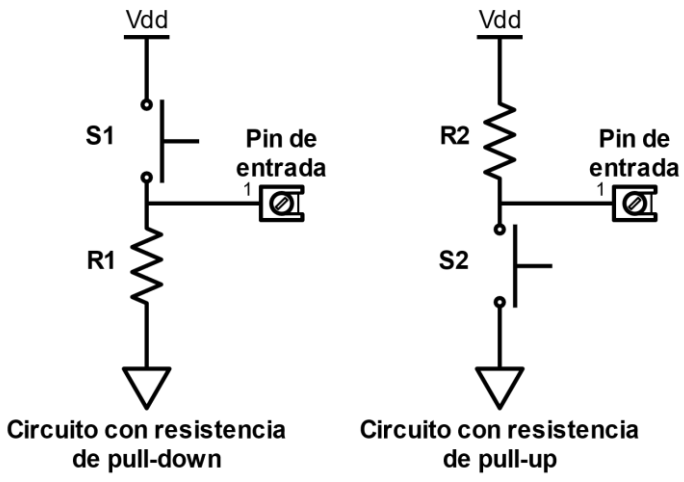
(Antes de pegar el código fuente en el documento PDF lo colorearemos, por ejemplo, utilizando la página: https://highlight.hohli.com/?language=cpp )

## Desarrollo:

Esta práctica está dividida en varios apartados:

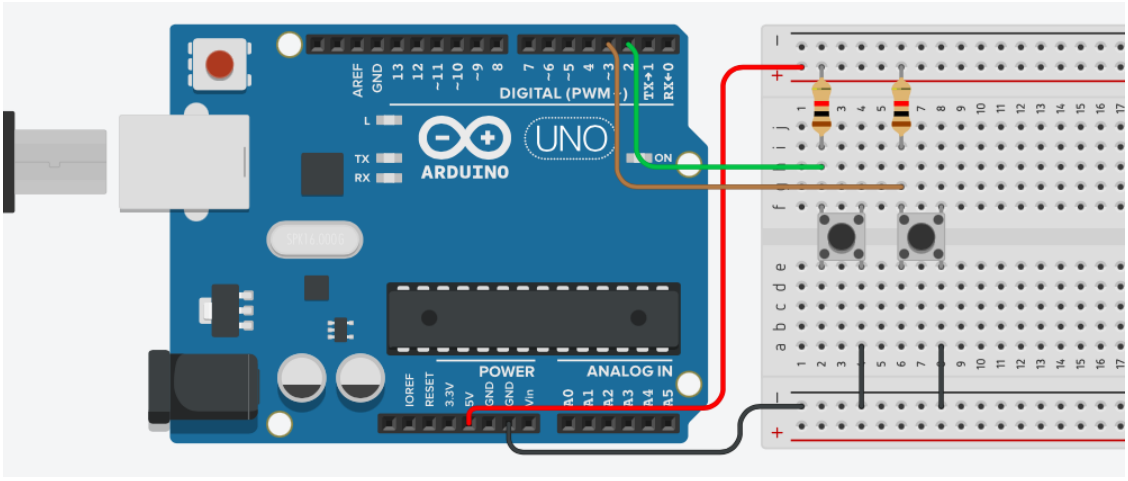
### 1) Creación del primer diseño: Conexión de 2 botones.

Conexión de botones a entradas en Arduino. Conceptos de *pull-up* y *pull-down*.



Las resistencias *pull-up* y *pull-down* son resistencias que se conectan a las entradas digitales del microcontrolador para fijar un valor por defecto. Es decir, cuando el otro dispositivo conectado a esta entrada (por ejemplo, un pulsador) no está estableciendo otro valor. Las resistencia de *pull-up* fijan un nivel alto ("1") y las de *pull-down* un nivel bajo ("0").

Conectamos dos botones en Arduino con *pull-up*, uno al pin 2 y otro al pin 3, ambos pines configurados como entradas. Si se quiere, se puede usar una placa de prototipado, como en la figura, para colocar los componentes electrónicos. Estas placas realizan conexiones entre los contactos (agujeros) internamente. En Tinkercad se puede colocar el puntero del ratón sobre un agujero para conocer sus conexiones.




## 2) Programa de lectura de 1 botón.

```
#define PIN_BOTON 2
unsigned int Pulsaciones;

void setup()
{
  pinMode(PIN_BOTON, INPUT);
  Serial.begin(9600);
  Pulsaciones = 0;
}

void loop()
{
  while(digitalRead(PIN_BOTON)) {}
  Serial.println(++Pulsaciones);
  while(!digitalRead(PIN_BOTON)) {}
}
```

## 3) Programa de lectura de 2 botones.

Modificamos el programa anterior cambiando la condición de los bucles `while`, de forma que cada vez que se modifique el estado del botón 1 o del botón 2 se incremente la variable `pulsaciones`.

¿Qué ocurre si mantenemos pulsado un botón y pulsamos otro?

En Tinkercad, cuando mantenemos pulsada la tecla MAYÚSCULAS al hacer clic sobre un botón, éste se queda pulsado hasta el siguiente clic.

## 4) Modificación del programa para comprobar los dos botones, aunque uno de ellos permanezca pulsado

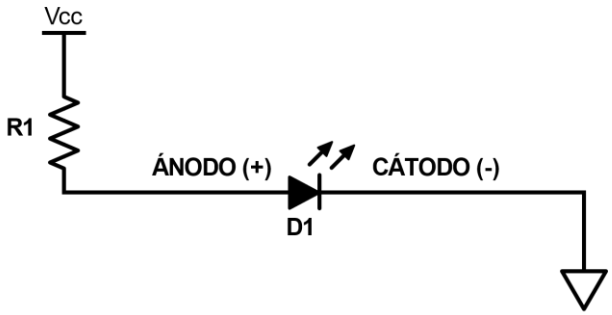
Modificamos el programa anterior para añadir esta funcionalidad eliminando los bucles `while`. Una posible implementación consiste en almacenar el estado de los dos botones en sendas variables booleanas globales (`bool`). Las inicializamos en la función `setup`. En la función `loop` comprobamos si el estado de alguno de los dos botones ha cambiado con respecto a lo almacenado en su variable de estado, y en caso afirmativo actualizamos dicha variable de estado. Comprobamos que nuestra implementación no pasa por alto pulsaciones. Para conseguir esto, es recomendable hacer solo una lectura de cada uno de los pines de los botones por cada ejecución de la función `loop` (iteración).

## 5) Modificación del diseño para eliminar las resistencias de *pull-up* externas

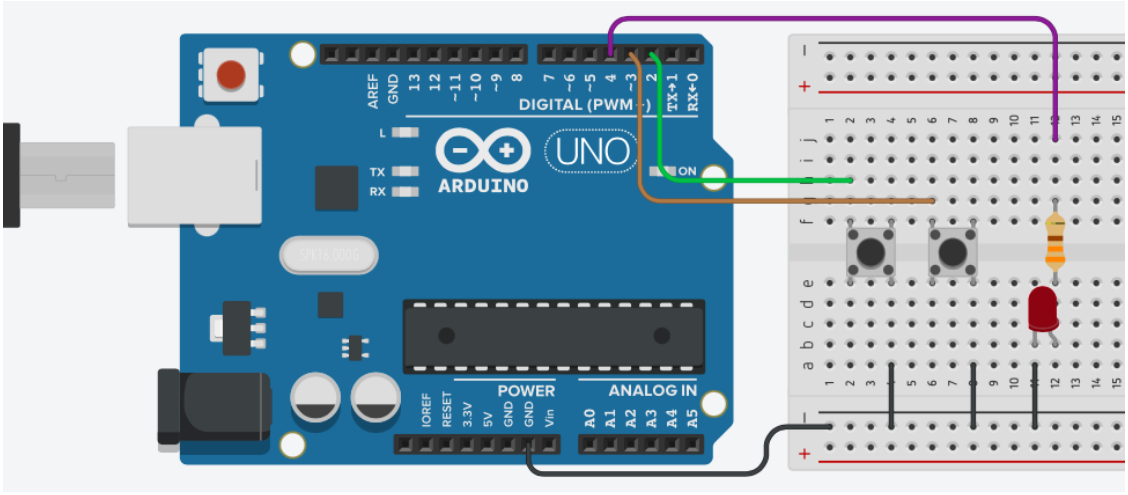
Consultamos la ayuda de la función `pinMode()` para ver cómo podemos substituir la resistencias externas por internas.

## 6) Conexión de un led

Añadimos al circuito anterior un diodo emisor de luz (led) que se debe encender cuando el número de pulsaciones que llevamos sea impar y apagarse cuando este número sea par.
Cuando se aplica un voltaje suficientemente positivo en el terminal ánodo del diodo con respecto al voltaje del cátodo el diodo conduce la corriente, y en el caso de un led, emite luz.




Este led lo conectaremos al pin 4 de la placa Arduino. Debemos tener en cuenta que un led no se debe conectar directamente a alimentación, sino que debemos intercalar una resistencia (en serie) que limite la corriente que pasa por él. Calculamos el valor de esta resistencia usando el Anexo 1 de este guion. Además, debido a que un led es un diodo y solo deja pasar la corriente en un sentido, debemos respetar la polaridad al conectarlo: su pata más corta (izquierda en la figura de Tinkercad) va conectada a masa.




Para alimentar el led con el pin 4 debemos configurar la correspondiente patilla del microcontrolador como salida en la función `setup()` como se muestra a continuación en el código fuente. Además, podemos modificar el estado de esta patilla con la función `digitalWrite()`:

```
#define PIN_LED 4 // Led conectado al pin 2
...

void setup()
{
  ...
  pinMode(PIN_LED, OUTPUT);
  ...
}

void loop()
{
  bool estado_led;

  ...

  digitalWrite(PIN_LED, estado_led);
}
```

## ANEXO 1: CÁLCULO DEL VALOR DE LA RESISTENCIA PARA ALIMENTAR UN LED

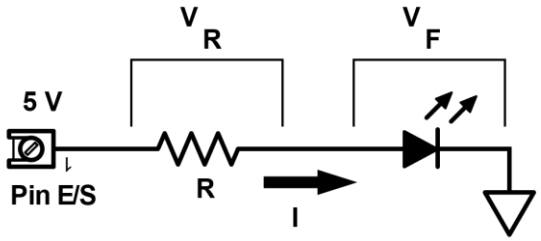
La vida útil estimada de un led suele estar entre las 20.000 a las 100.000 horas. En condiciones óptimas de disipación de calor se puede hacer circular una intensidad de corriente mayor para que ilumine más, haciendo al led funcionar estresado. En este caso, la vida útil se reduce a una fracción de la inicial en función de la intensidad de corriente y la temperatura que alcance.

Cálculo del valor de la resistencia

Vamos a calcular qué valor debe tener la resistencia en serie con el led de nuestro circuito para que funcione a un tercio de la intensidad máxima de corriente eléctrica que soporta. Para esto

supondremos que el led que estamos utilizando es el TLHR5400 del fabricante Vishay y seguimos los siguientes pasos:

- Descargamos el *datasheet* de este led, por ejemplo, del sitio web del fabricante (https://www.vishay.com).
- Localizamos en la sección Absolute Maximum Ratings del *datasheet* la corriente máxima (mA) que soporta el led (*DC forward current*).
- A continuación, calcularemos la tensión que caerá en el led en nuestro caso. Algunas de las características de led, como es esta caída de tensión (*forward voltage*, $V_F$), están especificadas en una tabla para una corriente típica de alimentación del led, en cambio, nosotros obtendremos esta característica de una gráfica. En concreto, buscaremos en la gráfica *Forward Current vs Forward Voltage* el valor aproximado de tensión (V) que caerá en el led para nuestro valor de *forward current* (un tercio de la *DC forward current* máxima). Debemos asegurarnos de que la gráfica que estamos consultando es la del led rojo (etiquetada como "red").




- Ahora, asumiendo un voltaje de salida de 5 V en las patillas del microcontrolador, podemos calcular el valor de tensión que caerá en la resistencia: $V_R = 5\ V - V_F$
- Como ya sabemos la tensión que cae en la resistencia y la intensidad de corriente que pasa por ella, podemos calcular su valor con la ley de Ohm: $V = R \cdot I \rightarrow R = V / I$. Debemos utilizar unidades enteras en la ecuación a la hora de hacer los cálculos (ohmios, voltios y amperios) para obtener un resultado correcto.
- Una vez tenemos el valor de la resistencia debemos buscar un valor que esté disponible comercialmente. Es decir, un valor normalizado. Nos restringiremos a la serie de resistencias E12, cuyos valores son muy comunes. En esta serie encontramos resistencias con los valores: 1,0 1,2 1,5 1,8 2,2 2,7 3,3 3,9 4,7 5,6 6,8 8,2 $\cdot 10^n$ ohmios donde n es un número entero. Es decir, encontramos resistencias de valores nominales como: 1 ohmio, 10 ohmios, 100 ohmios, 120 ohmios… Debemos buscar el valor de resistencia nominal ($R_N$) que coincida con la R calculada, o si no lo hubiera, que esté justo por encima (no utilizaremos una resistencia con un valor por debajo para no superar la intensidad de corriente I calculada). Utilizamos este valor, $R_N$, en el circuito a montar con Arduino en esta práctica.

Compartir captura de pantalla y enlace del proyecto:

- Para obtener la captura de pantalla del circuito podemos: en la pantalla de edición del circuito de Tinkercad utilizar la opción: Send To -> Picture of your design),
- Para obtener el enlace para compartir el proyecto creado: fuera de la pantalla de edición del circuito, en la pantalla con la lista de proyectos creados de Tinkercad, sitúe el puntero del ratón sobre el proyecto creado, pinche en el icono del engranaje que aparece -> Properties… -> Privacy -> seleccione Shared link y pulse Save changes. Después, en la pantalla con la lista de proyectos, pinche en el proyecto y en la ventana que aparece pulse el botón: Copy link.
  El enlace debe tener el formato: https://www.tinkercad.com/things/<código 1>?sharecode=<código 2>).
- Es recomendable comprobar que el enlace creado funciona.