Sistemas con Microprocesadores

Práctica 1

Título: Primeros diseños con Arduino y el simulador *Tinkercad*. Entradas/Salidas.

Objetivo:

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

Entregar:

Hacemos públicos los diseños Tinkercad de los ejercicios 5 y 7.
Entregamos un fichero PDF de 1 página aproximadamente que contenga:

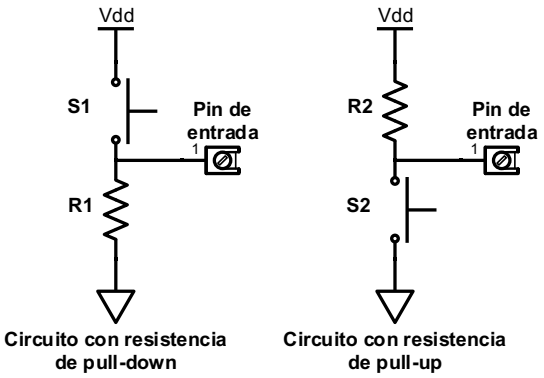
- Nombre y apellidos
- Dirección web del proyecto (o proyectos) Tinkercad
- Captura de pantalla del funcionamiento con los botones sin *pull-up* externa.
- Captura de pantalla del funcionamiento del mando y receptor de infrarrojos.
- Respuesta a las preguntas que se plantean en los siguientes apartados.

Desarrollo:

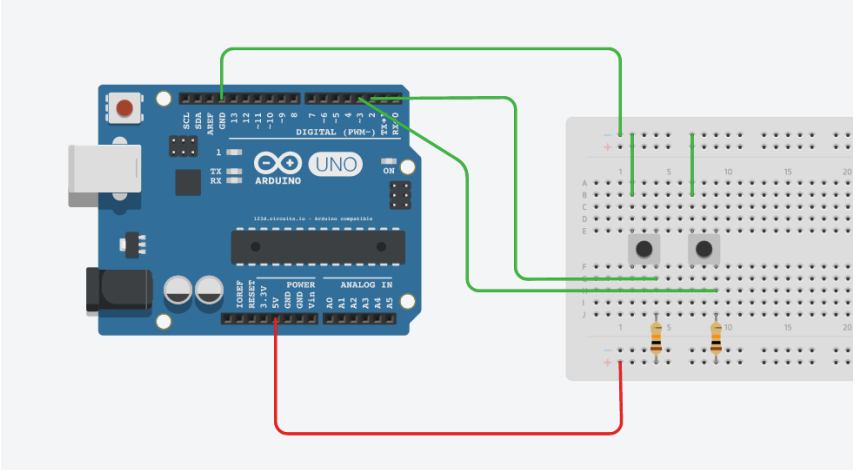
Esta práctica está dividida en varios apartados:

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

Conexión de un botón a una entrada en Arduino. Conceptos de *pull-up* y *pull-down*.



Conectamos dos botones en Arduino con *pull-up*, uno al pin 2 y otro al pin 3, ambos pines configurados como entrada.



2) Programa de lectura de 1 botón.

```
#define PIN_BOTON 2
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 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 pulsamos MAYÚSCULAS al hacer clic sobre un botón, éste se mantiene pulsado hasta el siguiente clic.

4) ¿Cómo podemos comprobar los dos botones, aunque uno de ellos permanezca pulsado?

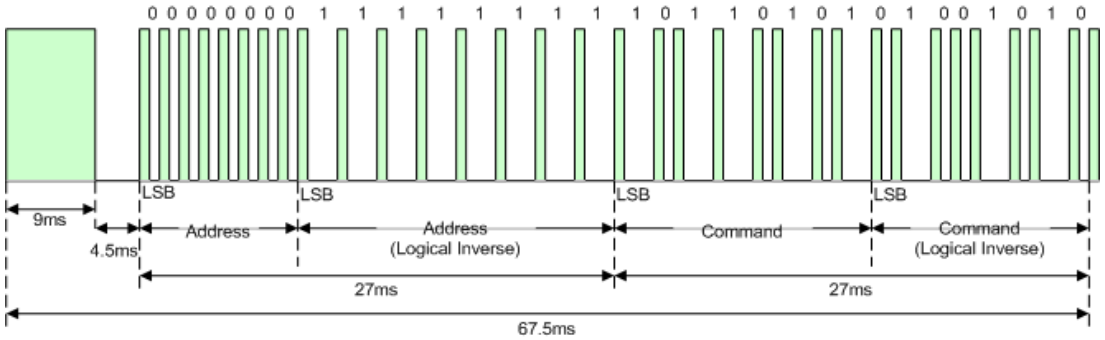
Modificamos el programa anterior para añadir esta funcionalidad. Para esto podemos almacenar el estado de los dos botones en sendas variables booleanas globales (`bool`). Las inicializamos en la función `setup`. En la función `loop` comprobamos si el estado de alguno de los dos botones ha cambiado y en caso afirmativo actualizamos la variable global correspondiente.

5) Cambio en el diseño hardware. ¿Podemos eliminar las resistencias de *pull-up* externas?

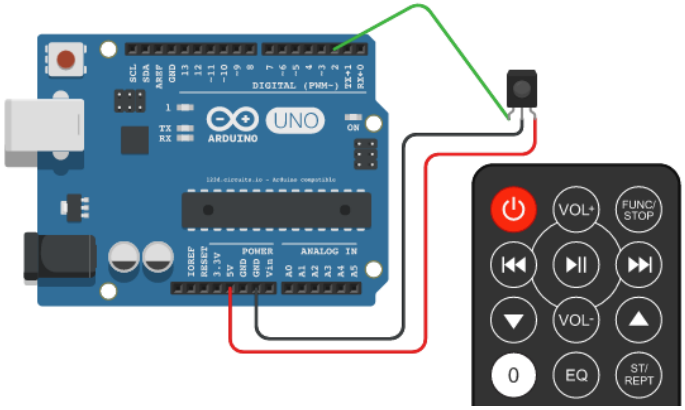
Consultar la ayuda de la función `pinMode()`.

6) Lectura de un mando a distancia por infrarrojos. Utilizar IR Sensor e IR Remote.

NEC IR es un protocolo que utilizan algunos mandos a distancia:
- Codificación de bits:
  - Inicio: Pulso de 9 ms seguido de un espacio 4,5 ms
  - 0 lógico: pulso de 562,5 µs seguido de un espacio 562,5 µs, con un tiempo de transmisión total de 1,125 ms
  - 1 lógico: pulso de 562,5 µs seguido de un espacio 1,6875 ms, con un tiempo total de transmisión de 2,25 ms
- Trama completa: Dirección + Dirección complementada + Orden + Orden complementada



Incluimos el receptor de infrarrojos y un mando a distancia y probamos el siguiente programa.




```
#include <IRremote.h>
#define PIN_SENSOR_IR 2 // Sensor conectado al pin 2
```

```
void setup()
{
  Serial.begin(9600);
  // Inicializa el receptor
  IrReceiver.begin(PIN_SENSOR_IR, ENABLE_LED_FEEDBACK);
}

void loop()
{
  if (IrReceiver.decode()) // ¿Hay dato?
  {
    uint32_t dato_ir;
    dato_ir = IrReceiver.decodedIRData.decodedRawData;
    Serial.println(dato_ir, HEX);

    if(dato_ir == 0xF906BF00)
      Serial.println("Avanza");
    else if (dato_ir == 0xFB04BF00)
      Serial.println("Retrocede");
  }
  IrReceiver.resume();
}
```

¿Con qué botones del mando se imprimen los mensajes de texto en la consola?

7) Modificación del programa de lectura del mando a distancia.

Ampliamos el programa anterior para que también se impriman mensajes de texto cuando se pulsan los botones de volumen del mando.