ARDUINO INTERMEDIO

Miguel Angel Ruiz Gálvez
Visita: miguelo.me
Material en: http://goo.gl/UO3xix

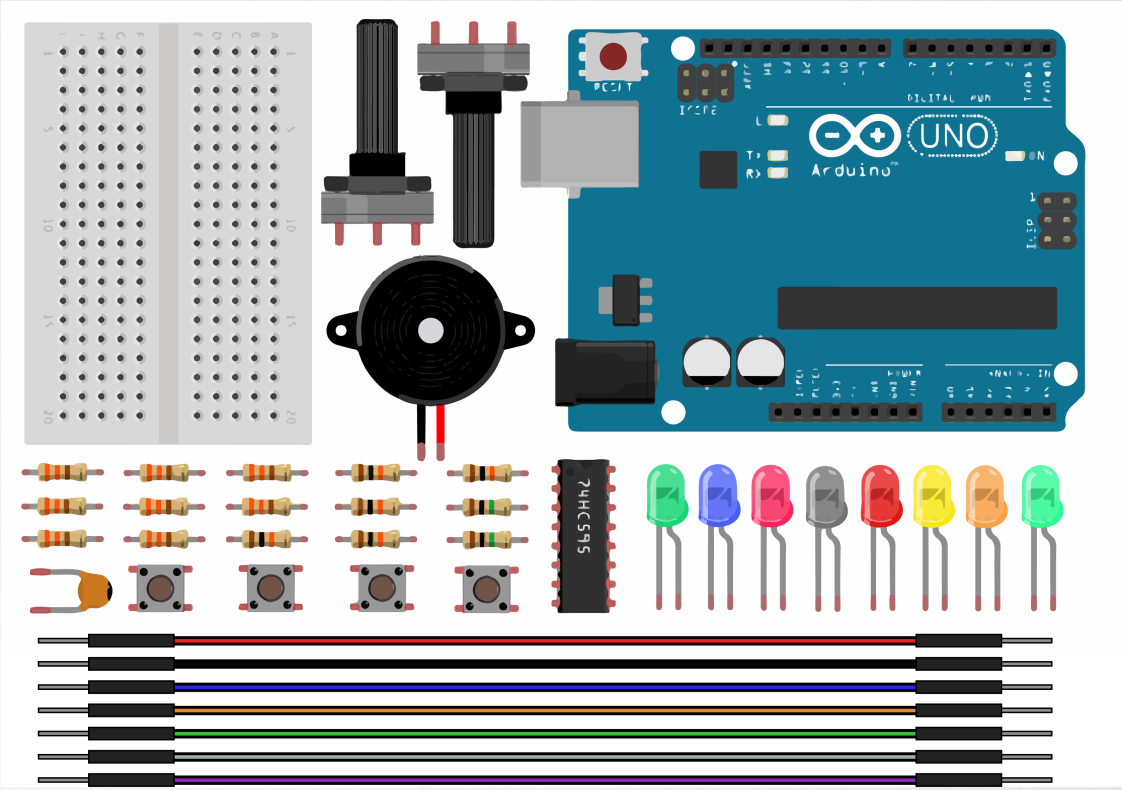
SOMEFI

MIGUELO RUIZ MECHATRONIC ENGINEER

cc BY NC SA

MATERIAL A UTILIZAR

- Tarjeta de desarrollo Arduino UNO y cable.
- 1 Protoboard
- 8 LED´s
- 8 Resistencias de 300 Ω, ¼ [W]
- 4 Pushbottons
- 4 Resistencias de 10 KΩ, ¼ [W]
- 2 Resistencia de 10 MΩ, ¼ [W]
- 2 Potenciometros de 10 kΩ, ¼ [W]
- 1 Capacitor cerámico de 100 pf
- 1 Buzzer
- 1 74HC595
- 1 Cable Caimán-Caimán
- 1 Pedaso de papel aluminio de cocina
- 20 Jumpers



MIGUEL RUIZ
MECHATRONIC ENGINEER

REFERENCIA DE ADC

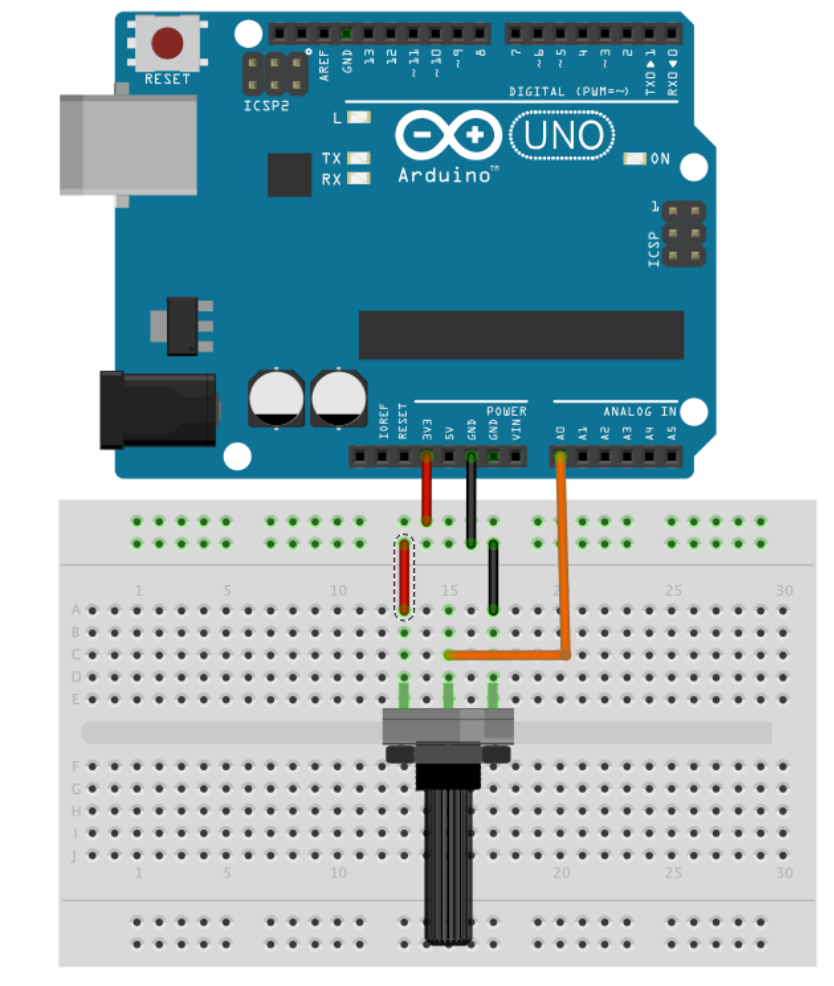
```
analogReference(tipo);
```

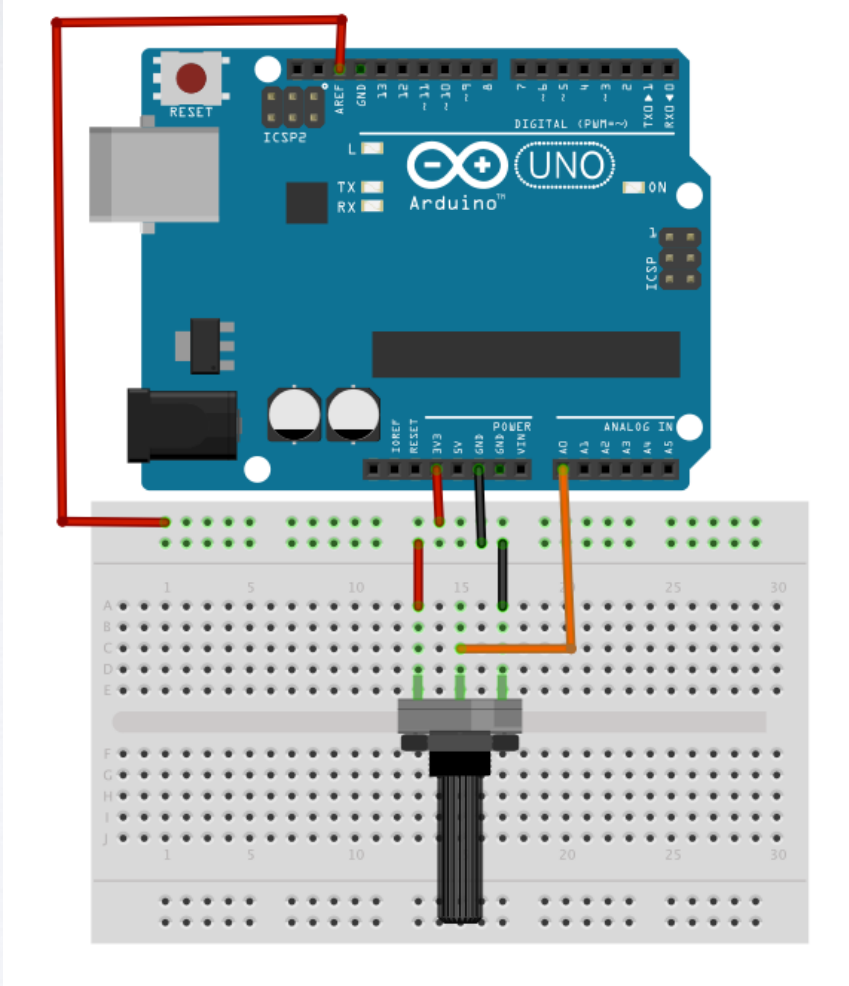
```
valor máximo = 1024-0 = 2^10 = Vref
Medición minima(mV) = Vref/(2^10)
```

- Después de cambiar la referencia analógica, las primeras lecturas de analogRead () puede no ser exacta.
- Advertencia: No usar menos de 0 V o más de 5V para referencia externa.
- Si utiliza una referencia externa, debe establecer la referencia analógica a EXTERNAL antes de llamar analogRead (). De lo contrario, dañara el microcontrolador.

DEFAULT: La referencia por defecto de 5 volts.
Internal: 1.1 volts en el ATmega168 o ATmega328. Diferente para MEGA.
EXTERNAL: La tensión aplicada al pin AREF (0 a 5 V) se utiliza como la referencia.

REFERENCIA ANALOGICA



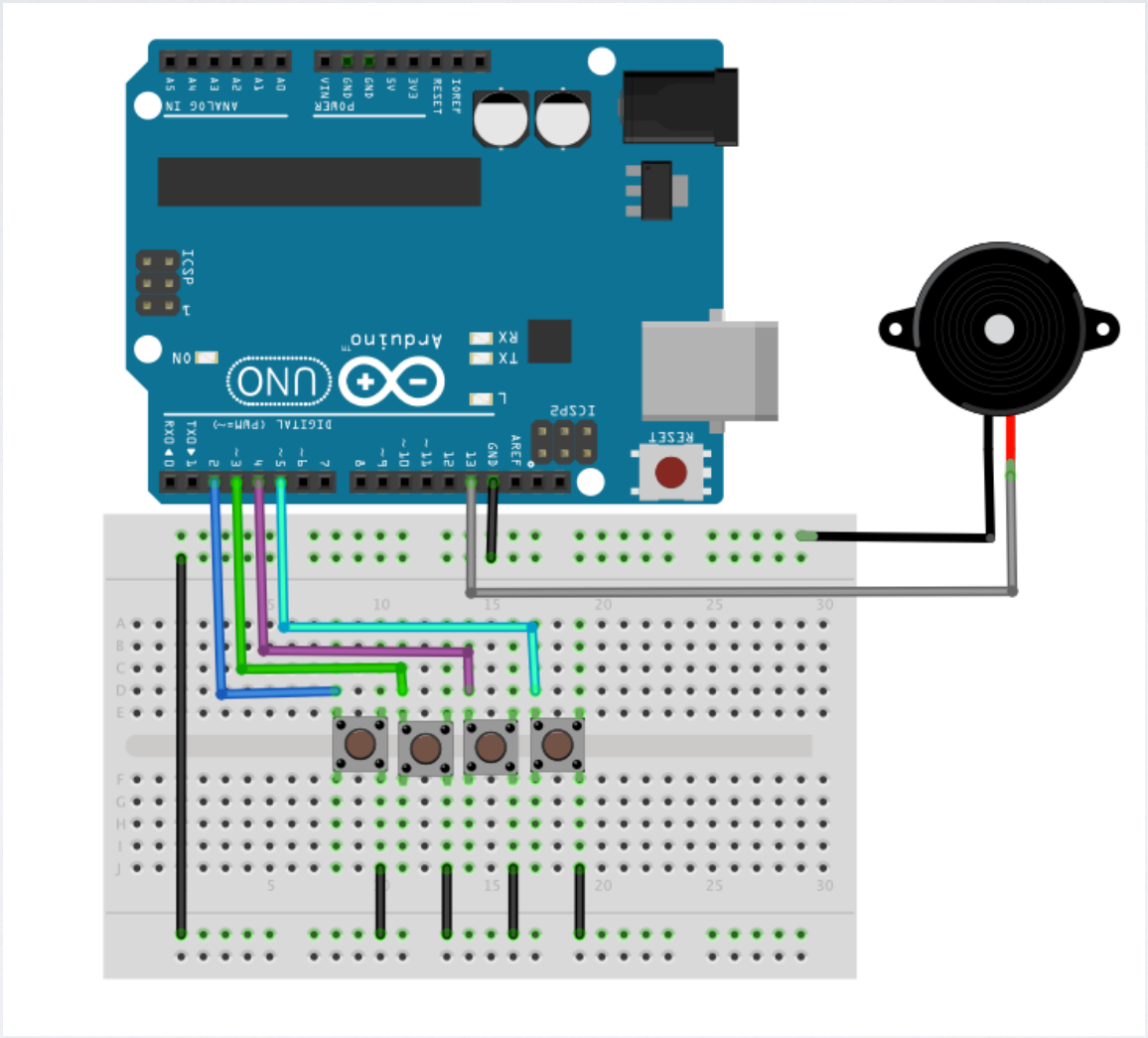


SONIDOS

```
tone(pin, frecuencia);
tone(pin, frecuencia, tiempo);
noTone( );
```

- Genera una onda cuadrada de 50% de ciclo de trabajo y frecuencia especificada.
- Se puede especificar un duración, de lo contrario la onda continúa hasta que una llamada a `noTone();`
- Sólo un tono puede ser generado a la vez. Si se llama la función en un pin diferente no tendrá efecto. Si se llama la función en el mismo pin, se establece la nueva frecuencia.
- El uso de la función `tone();` interferirá con salida PWM en los pines 3 y 11.
- No es posible generar tonos inferior a 31Hz.

PIANITO



MIGUEL RUIZ
MECHATRONIC ENGINEER

FUNCIONES

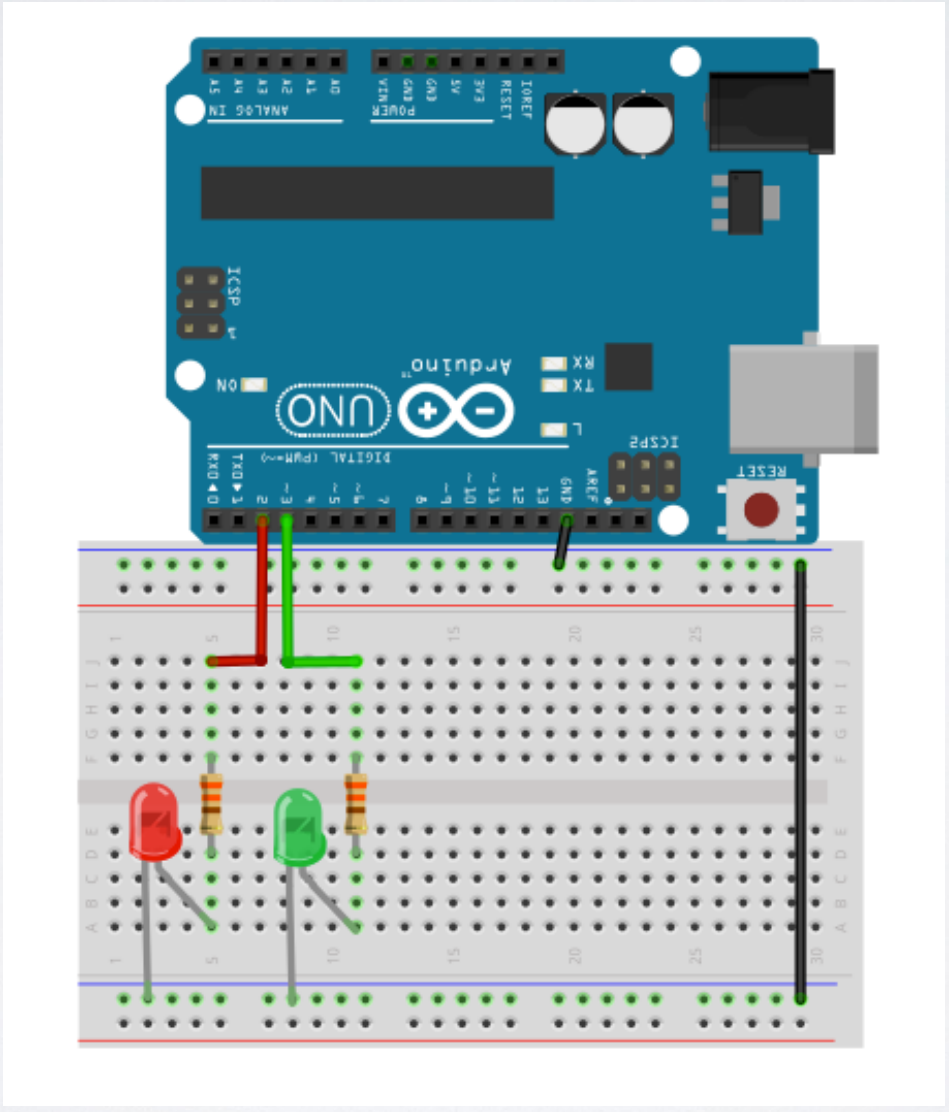
- La segmentación de código en funciones permite al programador crear piezas modulares de código que realizan una tarea definida. Se acostumbra hacer una función cuando hay que realizar la misma acción varias veces en un programa.

- Nuevas funciones deben crearse fuera de `setup()` y `loop()`.

```
void funcion(parametros){
   // tu código
}
```

```
datatype funcion(parametros){
   // tu código
   return datatype
}
```

PARPADEO



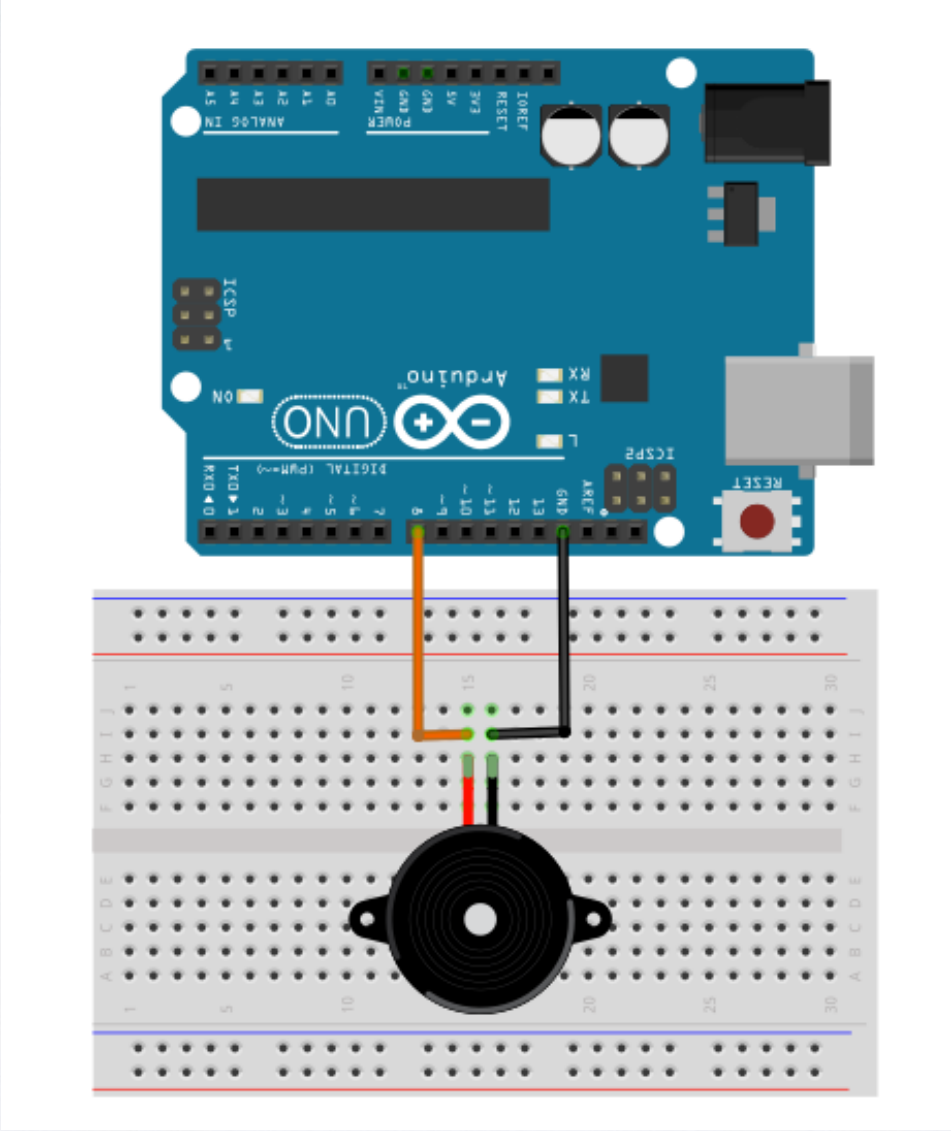
MIGUEL NUIZ
MECHATRONIC ENGINEER

ARREGLOS

```
int lista[6];
int lista[] = {2, 4, 8, 3, 6};

char cadena[6] = “String”;

int tamaño = sizeof(variable);
```

- Un arreglo es una colección de variables a las que se accede con un índice, los arrays de Arduino heredan las características de los arrays de C.
- El indice de los arreglos siempre empieza en 0;
- Se puede utilizar la función `sizeof( )` para obtener la dimensión del arreglo.

MARIO



MIGUEL RUIZ
MECHATRONIC ENGINEER

INTERRUPCIONES

- Se basa en el ISR(Interrupt Service Routine) del microcontrolador
- Se activa cuando se produce una **interrupción** y activa la rutina.
- La mayoría de las placas Arduino tienen dos interrupciones externas: Números 0 (en pin digital 2) y 1 (el pin digital 3).

Tarjeta	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega	3	3	21	20	19	18
Leonardo	3	2	0	1	7	

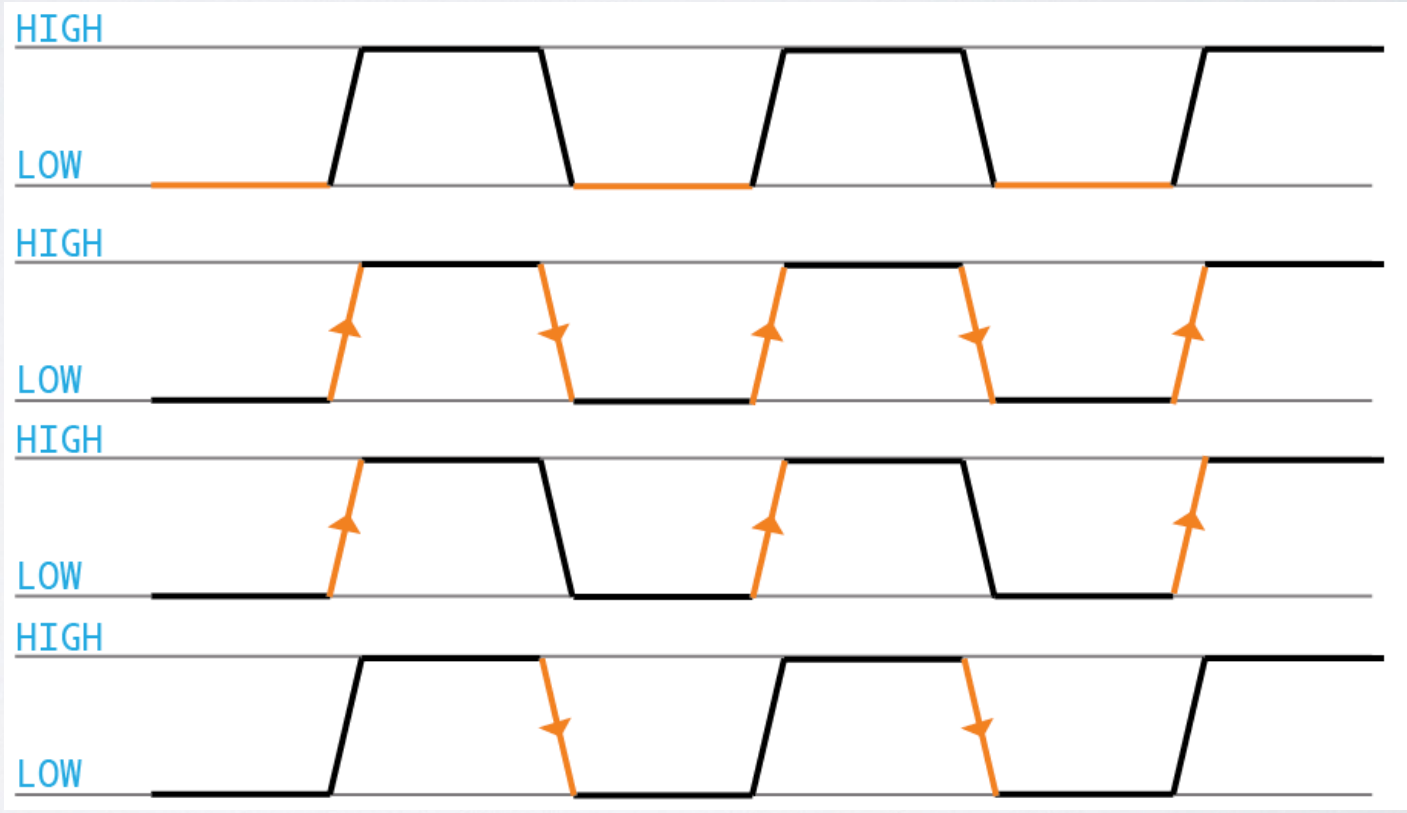
```
attachInterrupt(interrupt, función, mode);
attachInterrupt(pin, función, mode);

void función()
{
  // tu código
}
```

LOW : cada vez que cambie a bajo.

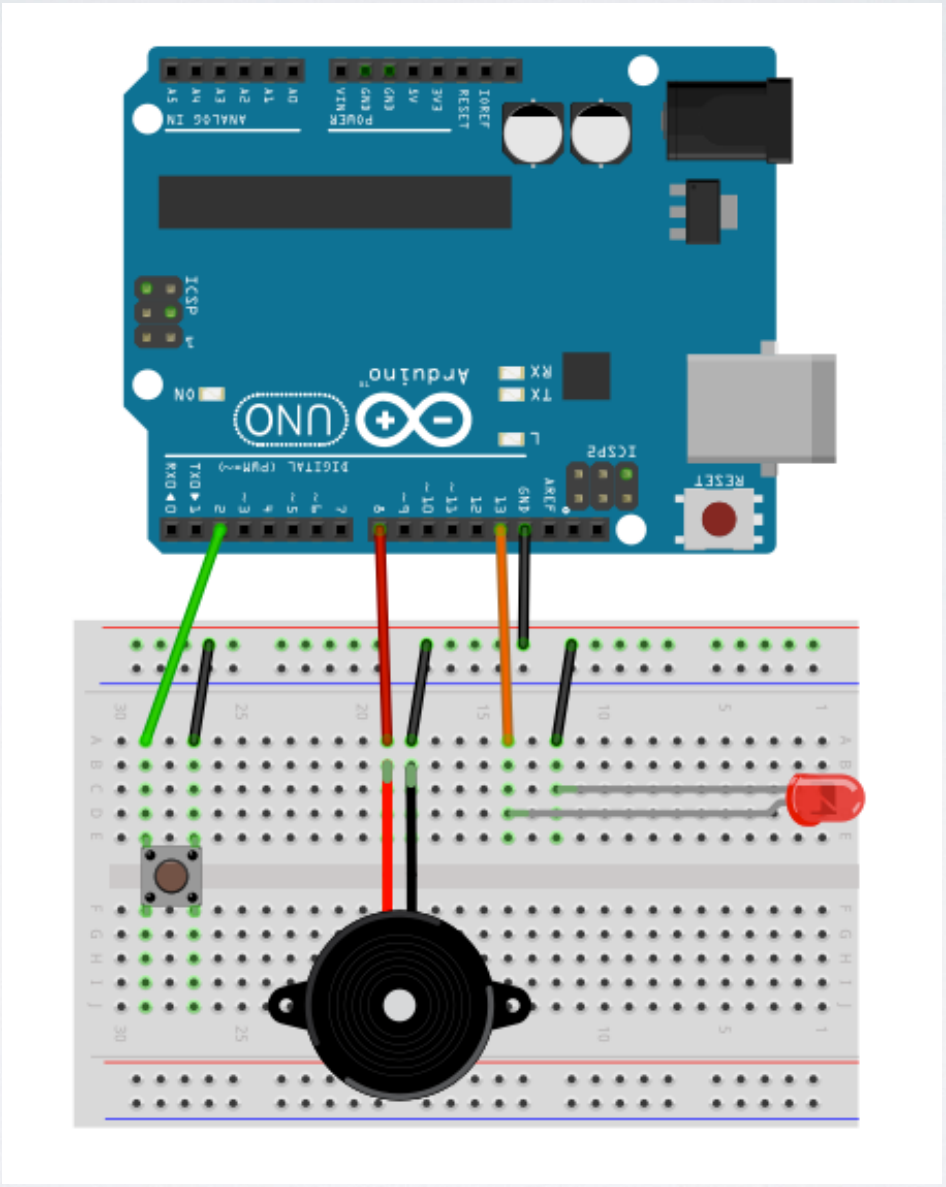
CHANGE : cada vez que cambie a de valor.

RISING : cada vez que cambie a de bajo a alto.

FALLING : cada vez que cambie a de alto a bajo.

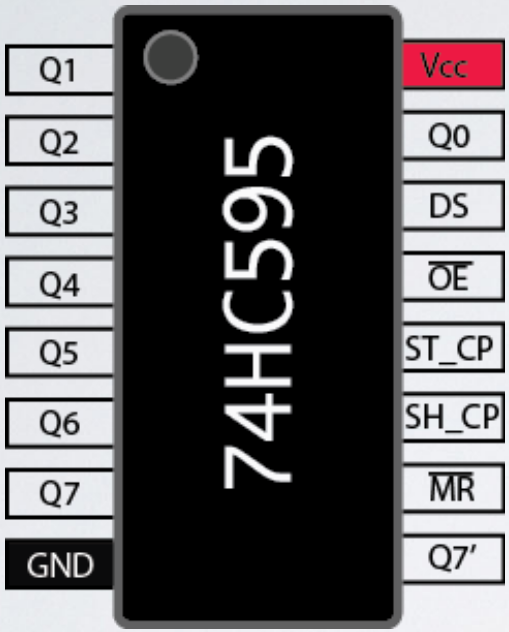


SIN PARAR A MARIO



MIGUEL RUIZ
MECHATRONIC ENGINEER

REGISTRO DE DESPLAZAMIENTO



Pin	Función
Q0-Q7	Salidas
DS	Envia los datos al reistro
ST_CP SH_CP	ST_CP: Debe ser puestos en HIGH para guardar la información. SH_CP: Cuando pasa a HIGH activa el desplazamiento.
!MR	Si se manda a LOW borra el registro.
Q7'	Pin de acarreo.
!OE	Al están el LOW activa las salidas.

- El circuito 74HC595 es un registro de desplazamiento, el cual te permite controlar ocho salidas digitales con únicamente 3 pines y si es necesario incluso más salidas si se ponen los registros en serie;

- Arduino ofrece un comando simple llamado `shiftOut( )` para controlar el registro.

```
shiftOut(DSPin,SHCP_Pin,mode,valor);
```

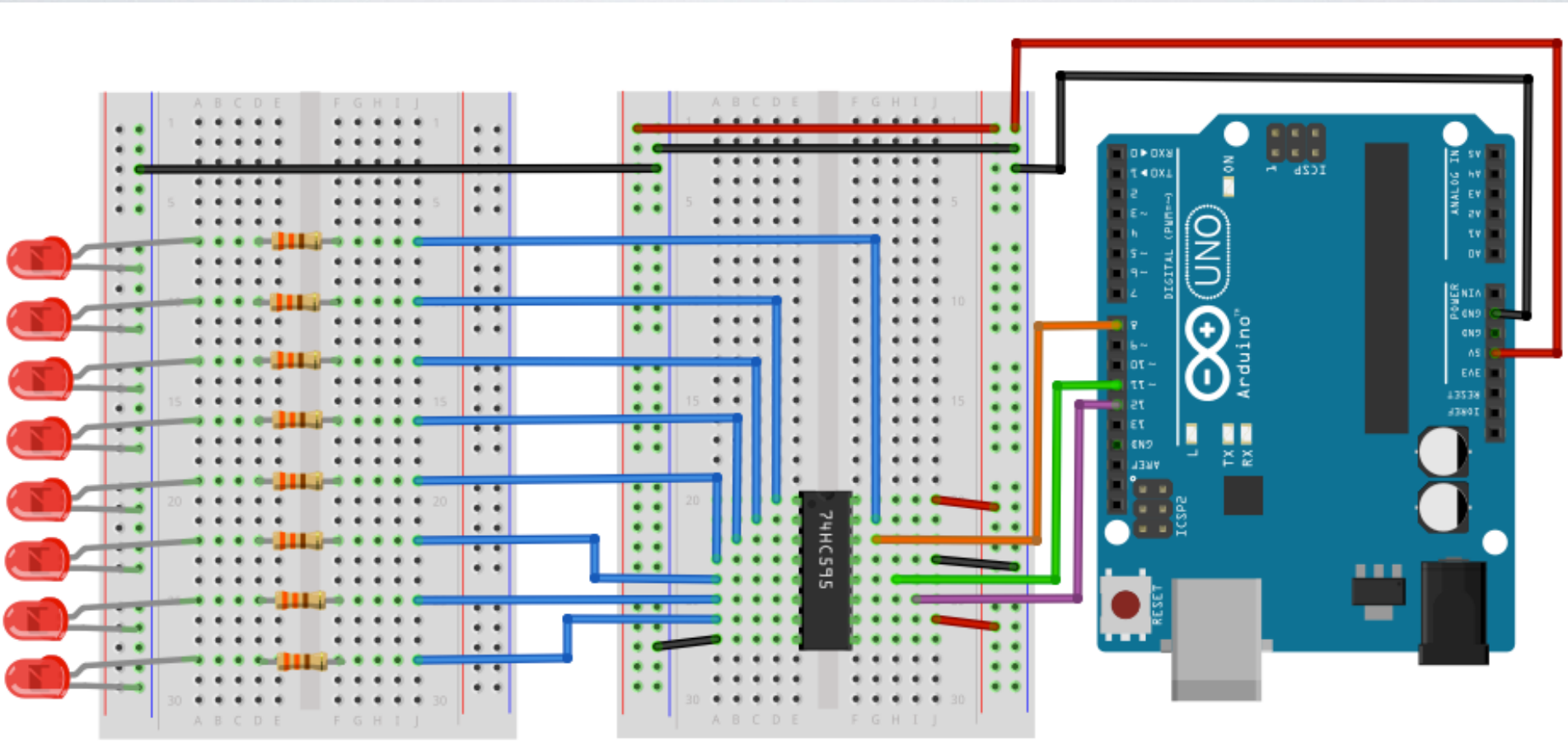
- DSPin = pin configurado como salida, se conecta a DS.
- SHCP_Pin = pin configurado como salida, se conecta a SH_CP.
- mode = MSBFIRST, LSBFIRST.(Most Significant Bit First, or, Least Significant Bit First)
- Nota: el pin de ST_CP de sebe configurar como salida y activar manualmente

MANEJO DE BITS

```
byte num = B01001000;
```

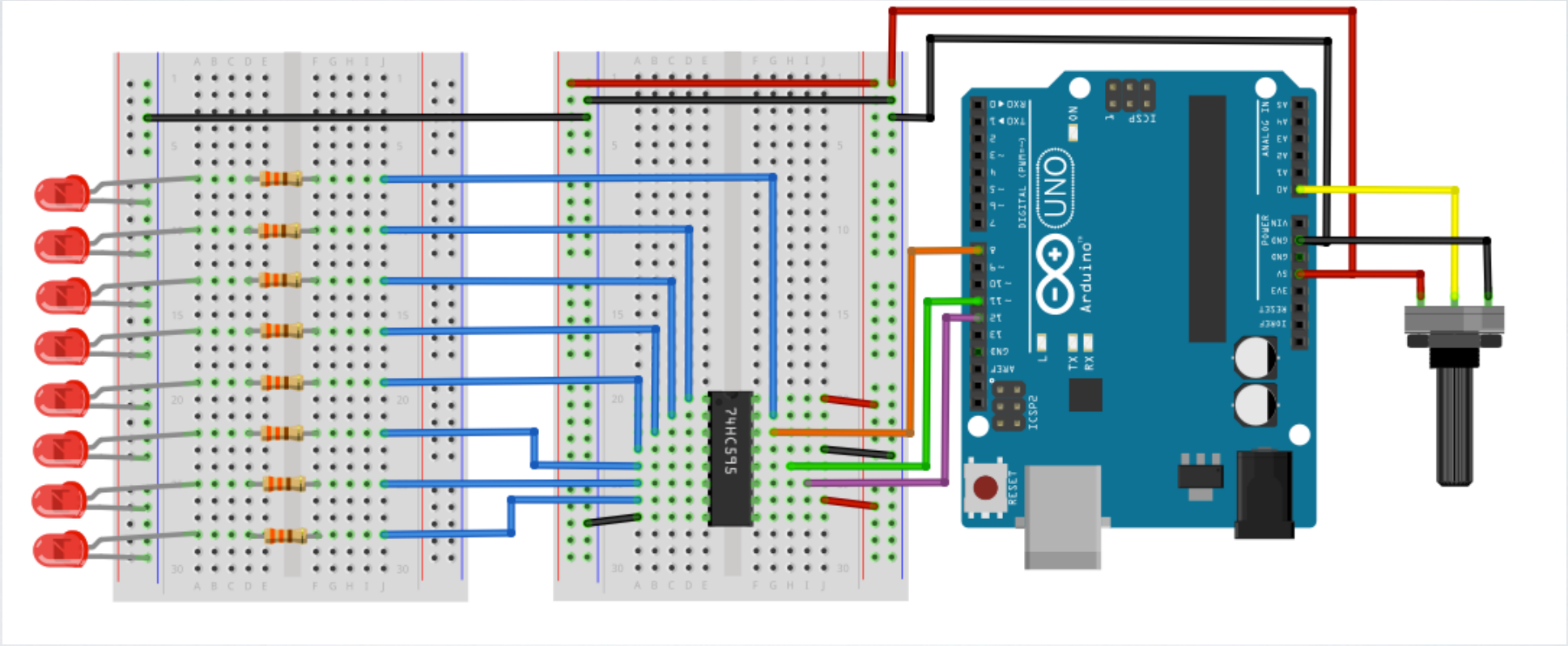
Función	Acción	Ejemplo
bitSet(byte, pos);	Escribe un 1 en la posición seleccionada del byte	bitSet(num, 5);
bitClear(byte, pos);	Escribe un 0 en la posición seleccionada del byte	bitClear(num, 3);
bitWrite(byte, pos, val);	Escribe 0 o 1 en la posición seleccionada del byte	bitWrite(num, 0, HIGH);
int valor = bitRead(byte, pos);	Arroja el valor 0 o 1 de la posición seleccionada del byte	int val = bitRead(num, 0);

KIT EL AUTO INCREÍBLE



MIGUEL NUIZ
MECHATRONIC ENGINEER

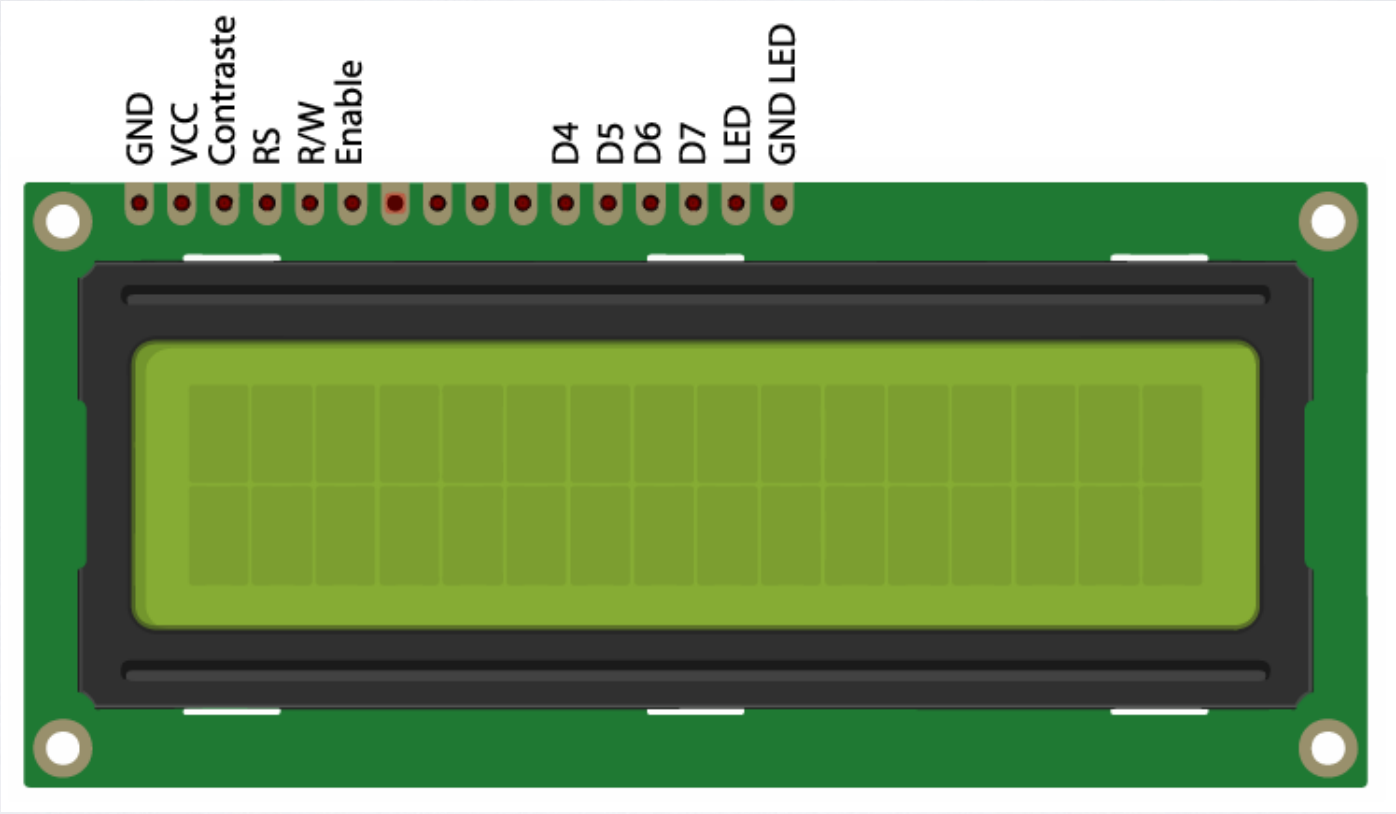
BARRA DE CARGA



MIGUEL NUIZ
MECHATRONIC ENGINEER

PANTALLA LCD

La librería `LiquidCrystal` permite controlar las pantallas LCD que sean compatibles con el controlador Hitachi HD44780.



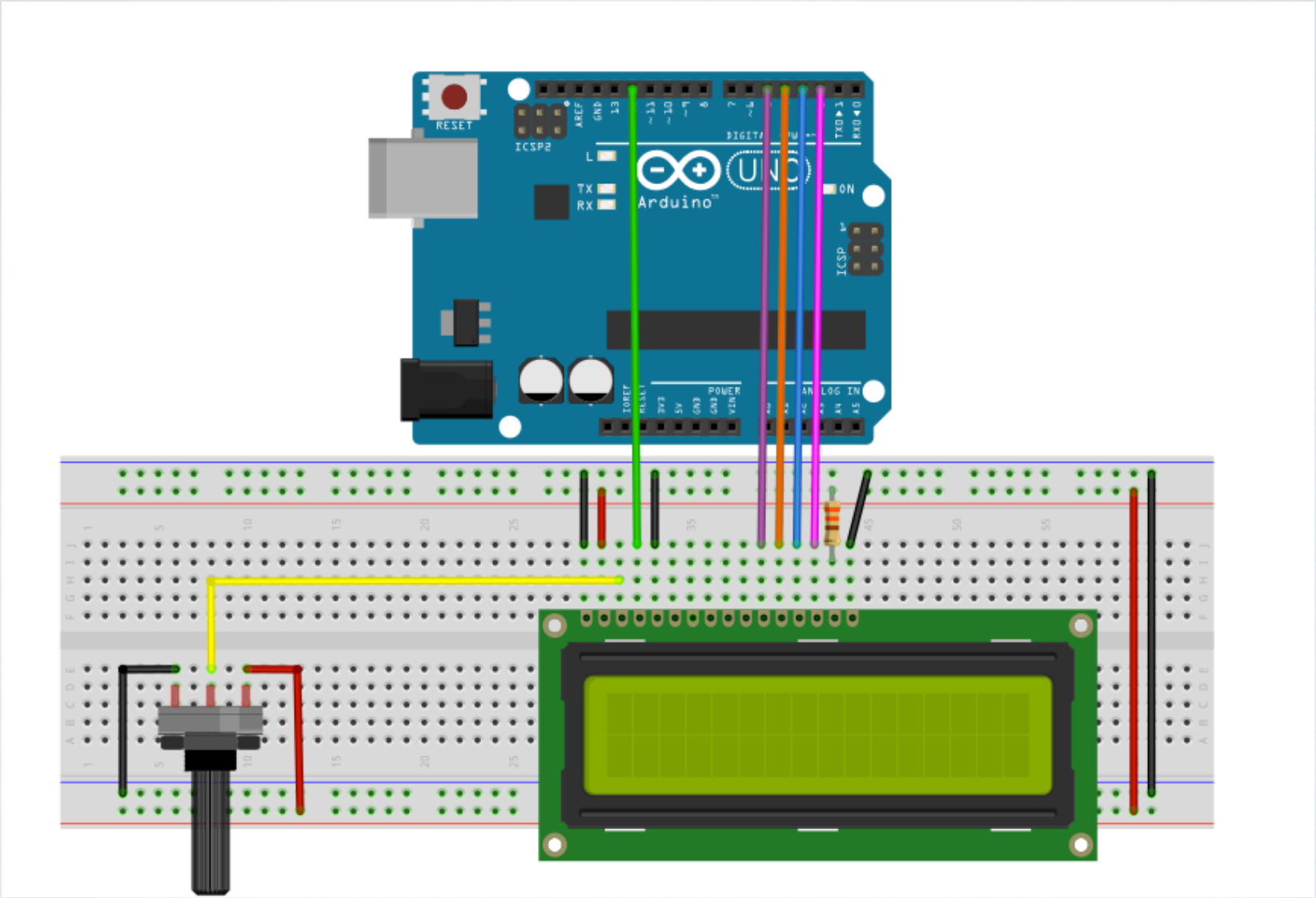
PANTALLA LCD

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

```
lcd.begin(n_columnas , n_lineas); //16x2;
lcd.print(texto);
lcd.setCursor(columna, linea);
```

- `begin`: Inicializa la pantalla LCD, y especifica las dimensiones de la pantalla. `begin()` debe ser llamada antes de cualquier otro comando de la librería LCD.
- `print`: Imprime el texto de la pantalla LCD. Comienza en la linea 0 columna 0.
- `setCursor`: Establecer la ubicación en la que se mostrará el texto.

PANTALLA LCD

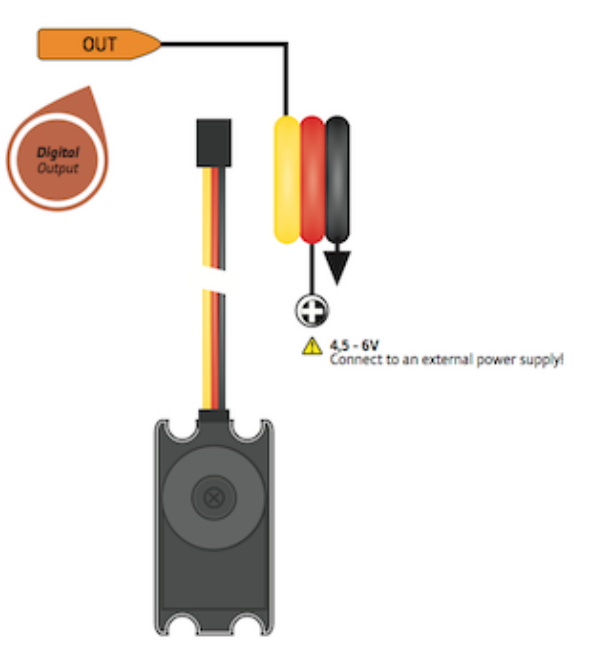


MIGUEL NUIZ
MECHATRONIC ENGINEER

SERVO MOTOR

Un servomotor es un moto-reductor acoplado a un potenciometro, este le permite saber la posición en la que se encuentra y así poder controlarla.

- Para controlar el servomotor analógico se le envían pulsos cada 20 ms (50Hz). El ancho de determina la posición angular.
- Dependiendo del tamaño del servo varia el consumo de corriente, en ese casi es necesaria una alimentación en ese caso es necesario una fuente de 5V independiente para poder mov externa de 4.5V a 6V.
- Se selecciona un servomotor según su la carga que soporta, genialmente en unidades Kg/cm



SERVO

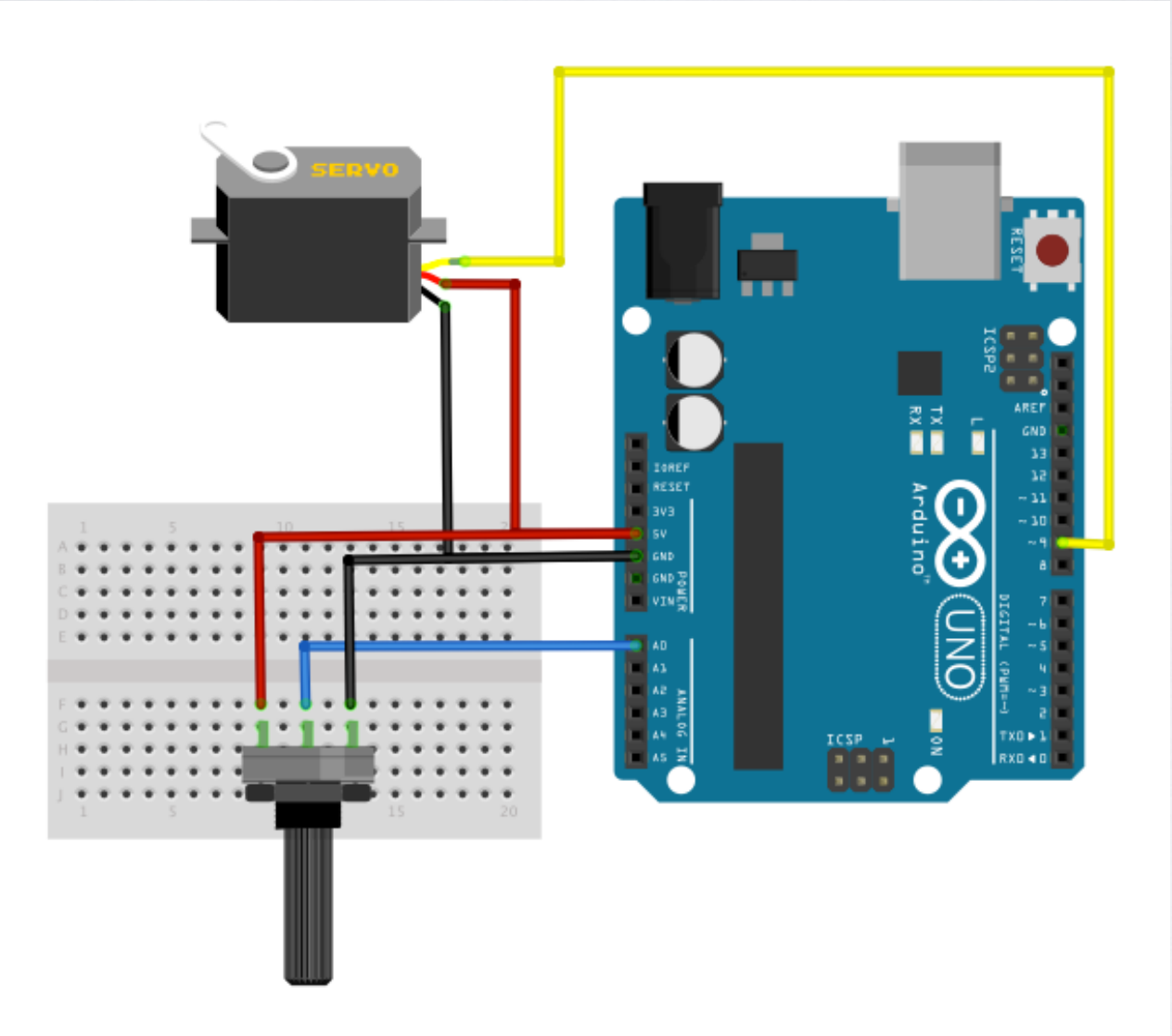
```
#include <Servo.h>

Servo.miServo;
```

```
miServo.attach(pin);
miServo.write(angulo);
```

- `attach`: Es la primer función que se debe de llamar, se utiliza para iniciar al servo, se pueden usar cualquier `pin` digital.
- `write`: Se utiliza para controlar el servomotor. el valor del `angulo` puede ir de 0 a 179.

Nota: la librería servo inutiliza la función de PWM de los pines 9 y 10.

SERVO

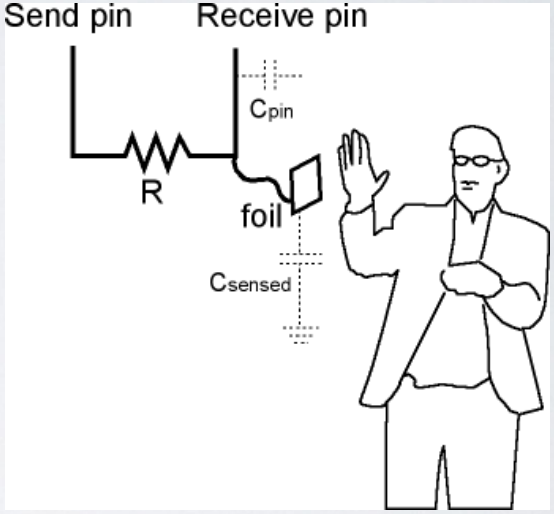


MIGUEL NUIZ
MECHATRONIC ENGINEER

SENSOR CAPACITIVO

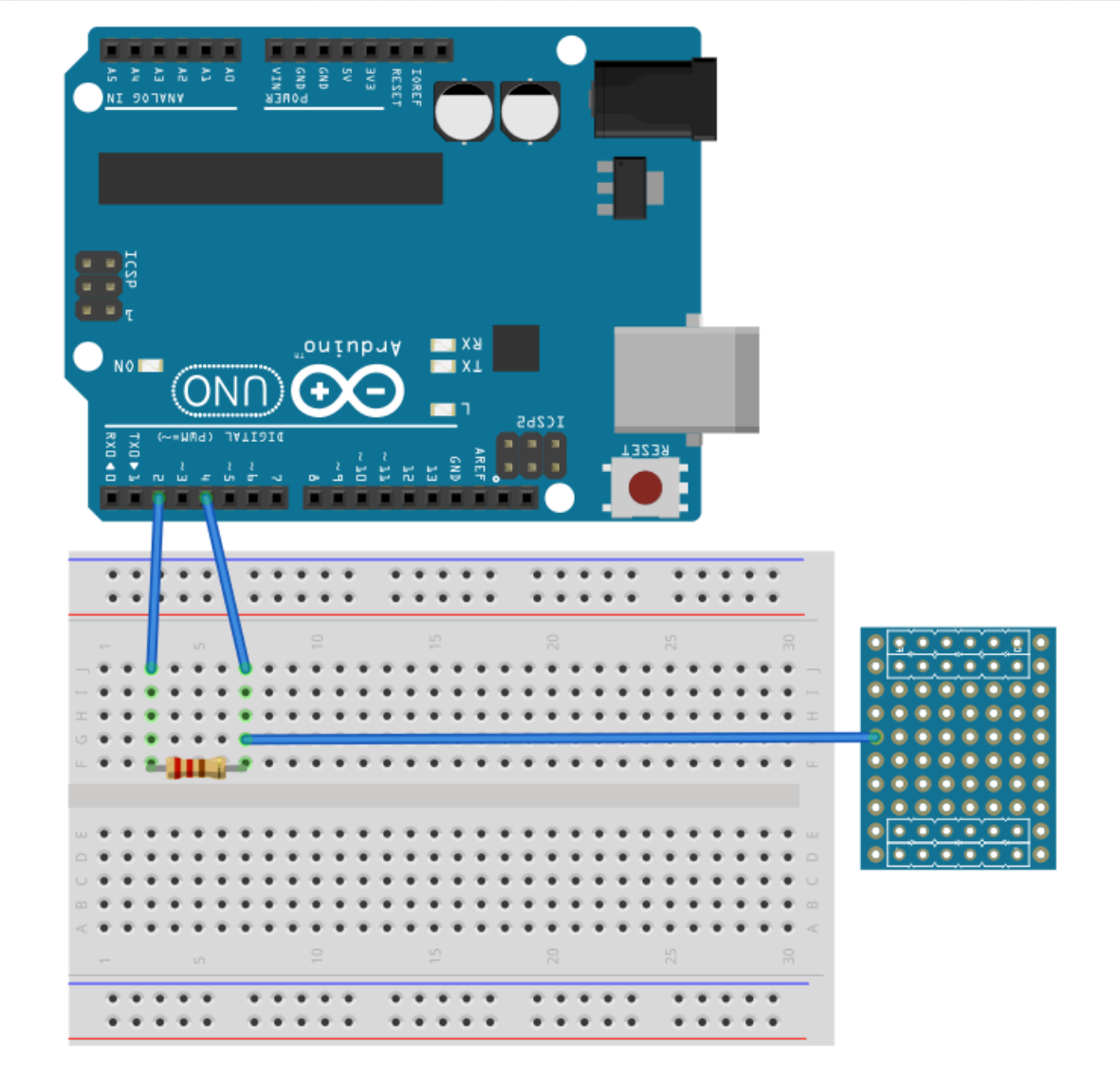
```
#include <CapacitiveSensor.h>
CapacitiveSensor sensor = CapacitiveSensor(pin send , pin receive);
```

```
long sensor =  cs_4_2.capacitiveSensor(muetras);
```



- La librería capacitiveSensor transforma dos o más pines de Arduino en un sensor capacitivo, que puede detectar la capacitancia eléctrica del cuerpo humano. Solo es necesaria una resistencia de valor alto y un pequeño pedazo de material conductivo.
- En su forma más sensible, el sensor comenzará a sentir una mano a pulgadas de distancia del sensor.
- Se puede calibrara el número de muestras que se realizan cada 20 segundos.

SENSOR CAPACITIVO

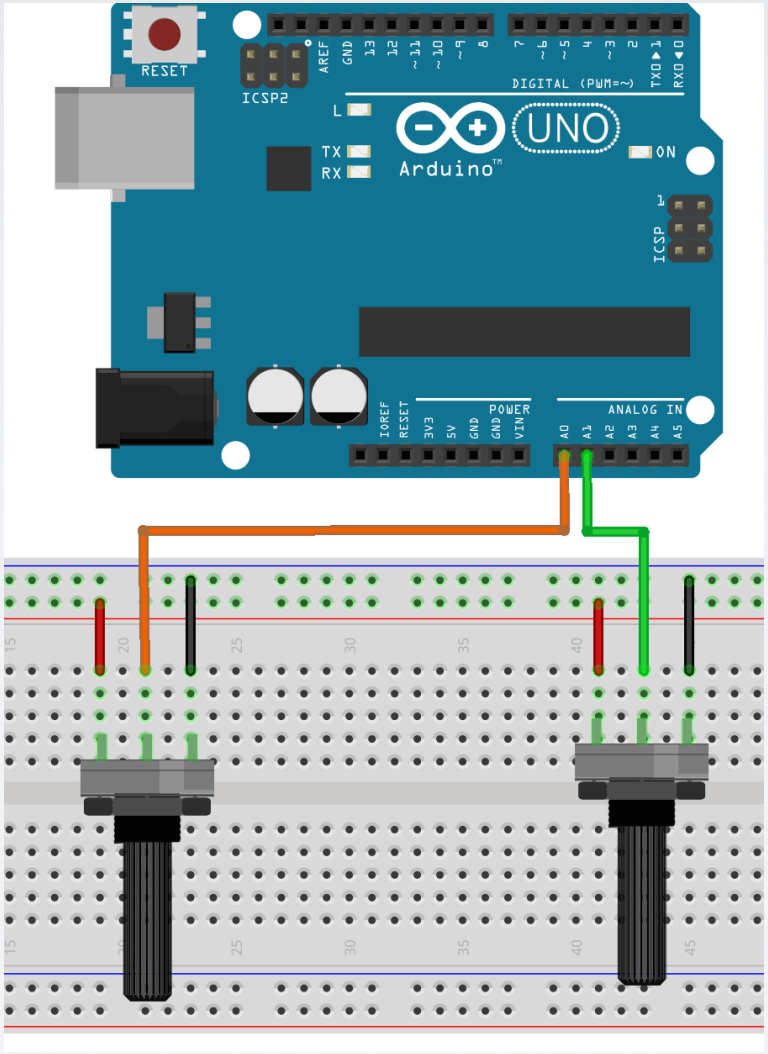


MIGUEL RUIZ
MECHATRONIC ENGINEER

PROCESSING: TABLA MAGICA

MIGUEL ANGEL RUIZ
MECHATRONIC ENGINEER

PROCESSING: TABLA MAGICA



MIGUEL RUIZ
MECHATRONIC ENGINEER

Creado por: Miguel Angel Ruiz Gálvez
Contacto: miguelo.me

Agradecimientos a:

- Massimo Banzi
- Hernando Barragan
- David Cuartielles
- Brett Hagman
- pighixxx.tumblr.com
- David A. Mellis
- Limor Fried
- Tom Igoe
- Paul Badger

MIGUELO RUIZ MECHATRONIC ENGINEER

creative commons