

LAB 2: Sensores y Actuadores

La presente evaluación será recibida sólo si se ha cumplido con los checkpoints de todas las preguntas en el **LABORATORIO** con el docente del curso.

La entrega debe constar de:

1. **Introducción** (incluir párrafo introductorio):
 1. **Marco Teórico** (definiciones con referencias)
 2. **Estado del Arte** (trabajos de investigación, artículos, tesis, páginas con referencias).
2. **Metodología** (explicación de componentes e implementos utilizados y diagramas de flujo/bloques).
3. **Desarrollo de la experiencia** (incluye discusión)
4. **Conclusiones** (una por cada experiencia realizada como mínimo)
5. **Referencias** (formato IEEE)

Consideraciones:

- Subir únicamente el PDF del informe a Canvas, usar un link para el repositorio, incluir fotos de paso a paso de cada implementación y un vídeo mostrando el funcionamiento de las experiencias. Sin embargo, para simulaciones, considere la placa Arduino UNO disponible en Tinkercad.
- Considere una placa de Arduino MEGA2560 tal cual existe en el laboratorio.
- Solo un miembro del grupo sube el informe.
- No olvidar colocar los nombres de todos los integrantes.

ÍNDICE

1. Encendido de un Motor DC con Arduino	3
1.1. Consideraciones	3
1.2. Responda:	4
1.3. Presentar:	4
2. Control de un motor con Driver	4
2.1. Consideraciones:	4
2.2. Responda:	4
2.3. Presentar:	5
3. Sistema de riego inteligente.	5
4. Riego de planta	6
5. IDLE	6

6. Alerta	6
7. Presentar:	6
8. Actuadores: Motores	7
9. Driver Puente H L293D	7
9.1. Conexión típica	8
10. Relés	9
10.1. Operación de un relé	9
10.2. Bloques de terminales de salida	11
10.3. Control de módulos	11
10.4. Ejemplo de Esquema de conexión (Deprecated)	13
11. Fotocelda o Resistor Dependiente de Luz (LDR)	13
11.1. Características eléctricas típicas	14
11.2. Principio de funcionamiento	14
11.3. Uso de un LDR con Arduino	14
12. LCD 2x16	16
13. Lectura Análoga con Arduino	17
14. Sensor de Humedad	19
15. Sensor de Temperatura	20
16. El uso de botones con Arduino.	21
17. Uso del monitor Serial.	22
18. Uso del teclado matricial	24
19. Diagramas de flujo	25
19.1. Simbología básica	26
19.2. Ejemplo de Diagrama de Flujo para un código simple.	27
20. Diagrama de Bloques	28
20.1. Objetivo de su Uso en el Diseño Electrónico	29
20.2. Componentes Típicos	29
20.3. Ejemplo	29

PARTE II: Actuadores

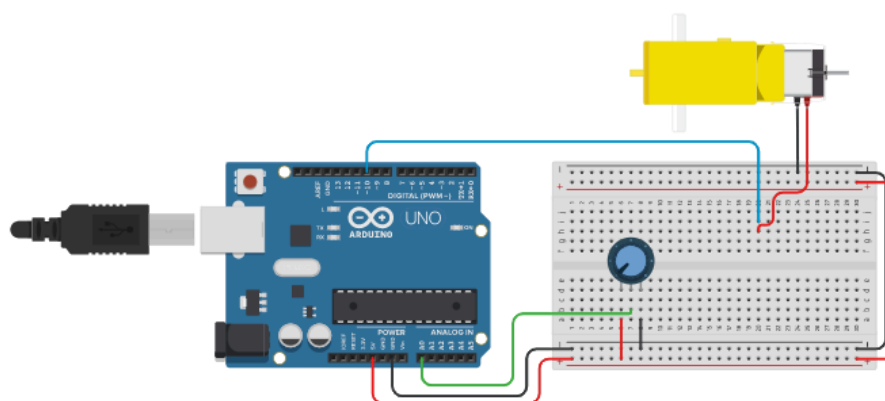
En esta parte de la experiencia realizará implementaciones de sistemas con sensores y actuadores. Los actuadores serán empleados principalmente con alimentación externa. Es decir, con una fuente de alimentación directa ya que los actuadores generalmente consumen más corriente al ser dispositivos inductivos.

1. Encendido de un Motor DC con Arduino

Este ejercicio es básico de encendido simple de un motor. Tenga en cuenta las siguientes consideraciones:

1.1. Consideraciones

- Conecte un **potenciómetro** al microcontrolador Arduino para emplearlo como medio de entrada analógica.
- Realice la lectura del valor analógico del potenciómetro mediante el pin correspondiente del Arduino (rango de 0 a 1023).
- Defina una variable denominada voltaje para almacenar el valor de voltaje resultante, convertido al rango de 0 a 5 voltios.
- Establezca una condición que verifique si el valor de voltaje es superior a 3 V y menor a 5 V.
- Si la condición se cumple, proceda a encender un motor de corriente directa (Motor DC).
- En caso contrario, asegúrese de que el motor permanezca apagado.
- Nota: En este caso no se emplea un driver ni alimentación externa por ser un ejercicio simple.



Esquema simplificado.

1.2. Responda:

1. ¿Cuál es la corriente que consume el motor durante su operación?
2. Si usted quisiera regular la velocidad del motor, ¿qué añadiría a su esquema de conexión?

1.3. Presentar:

- Funcionamiento del sistema en simulación.
- Diagrama de Bloques del sistema implementado.
- Respuesta de las preguntas.

[Checkpoint 1]

2. Control de un motor con Driver

Implementar el control de un motorreductor CC con driver L293D, dos pulsadores y un potenciómetro.

2.1. Consideraciones:

- El objetivo es que al pulsar el botón derecho gire a la derecha e Idem con el izquierdo.
- Mientras no se pulsa ningún pulsador, el motor permanece parado.
- Se utiliza un potenciómetro para controlar la velocidad de rotación del motor.
- La alimentación es externa (una pila de 9V) ya que los motores normalmente tienen un elevado consumo, y con los 40 mA y 5 V que ofrecen los pines de la placa Arduino son suficientes.

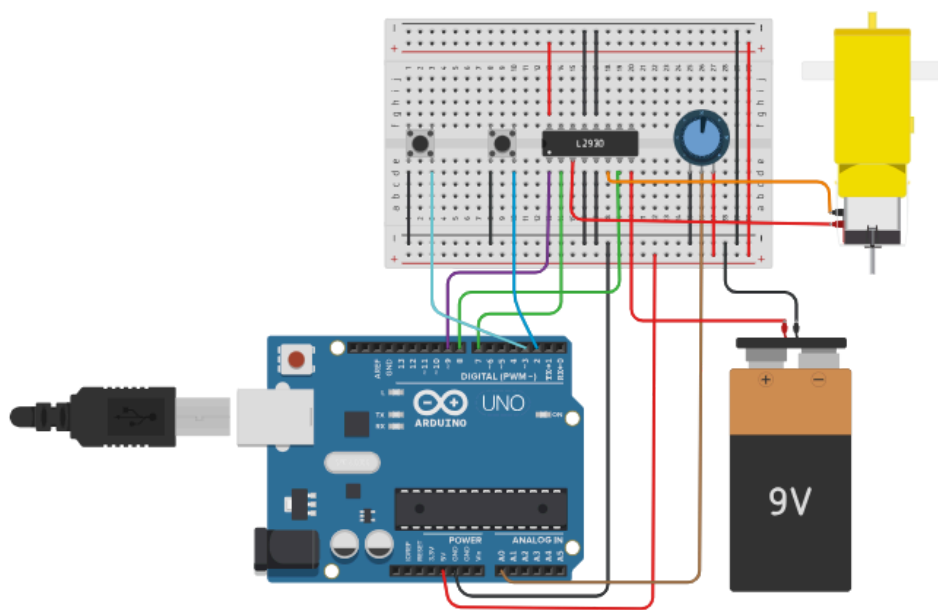
2.2. Responda:

- ¿Cuál es la función del potenciómetro en este sistema de control de motor y cómo se relaciona con la velocidad de rotación?
- Explique por qué se requiere una fuente de alimentación externa para el motor en lugar de usar directamente la salida de 5V del Arduino.

- Describa el comportamiento esperado del motor cuando se presiona únicamente el botón derecho, el izquierdo, y cuando no se presiona ninguno. ¿Qué sucede si se presionan los dos?

2.3. Presentar:

- Funcionamiento del sistema en simulación.
- Diagrama de Bloques del sistema implementado.
- Respuesta de las preguntas.



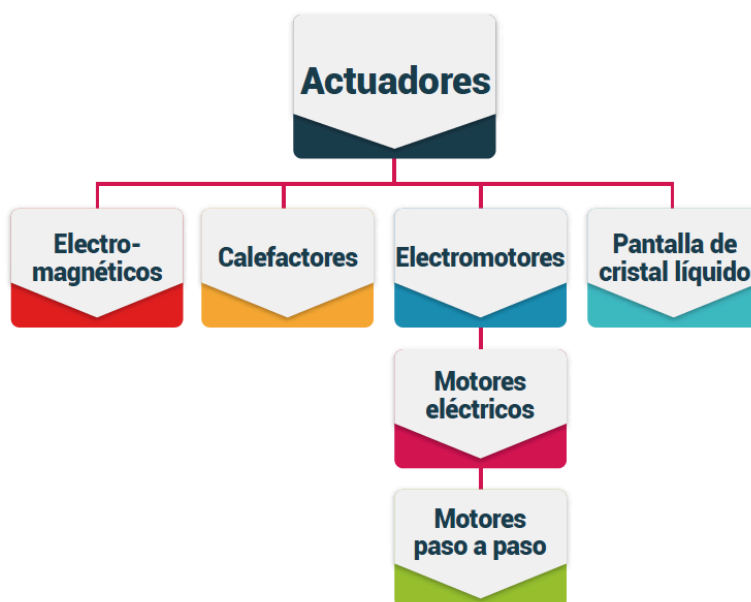
CHECKPOINT 2

ANEXOS

3. Actuadores: Motores

Un actuador es un dispositivo que recibe una entrada de energía y la convierte en movimiento o fuerza, y es un componente esencial en muchas tecnologías modernas y campos de la ingeniería. Desde la robótica hasta las energías renovables, los actuadores desempeñan un papel fundamental en el control y la automatización de diversos procesos y sistemas.

Hay varios tipos de actuadores: neumáticos, hidráulicos, eléctricos, magnéticos, térmicos y mecánicos, cada uno con sus ventajas e inconvenientes. El tipo de actuador utilizado en una aplicación depende de los requisitos específicos de dicha aplicación, como el nivel de fuerza, el tiempo de respuesta y la durabilidad necesarios.



Para más información de actuadores visitar el enlace:

[Actuadores Lineales con Arduino](#)

Abordaremos ahora algunos de los actuadores necesarios para el desarrollo de los laboratorios.

4. Driver Puente H L293D

El driver puente H L293D facilita el control de 2 motores DC con Arduino/PIC/RPIpico/ESP32. Al ser un driver dual, es decir de 2 canales, permite controlar de forma independiente 2 motores DC tanto en dirección de giro como en velocidad(PWM). Puede suministrar continuamente 0.6A y soporta picos de 1.2A por canal. Posee internamente diodos de protección para cargas inductivas como

motores o solenoides, también llamados diodos flyback. Su pequeño tamaño es ideal para ser utilizado en pequeños proyectos de robótica móvil como seguidores de línea, velocistas, laberinto.

El L293D internamente está conformado por 2 grupos de 2 medios puente-h o lo que es equivalente a 2 puente-h completos, esta característica nos permite controlar distintas cantidades y tipos de cargas dependiendo de la aplicación. Por ejemplo, puede controlar 4 motores DC uni-direccionalmente ó 2 motores DC en ambas direcciones(modos más comunes) ó también podría controlar un motor a pasos de bipolar o unipolar.

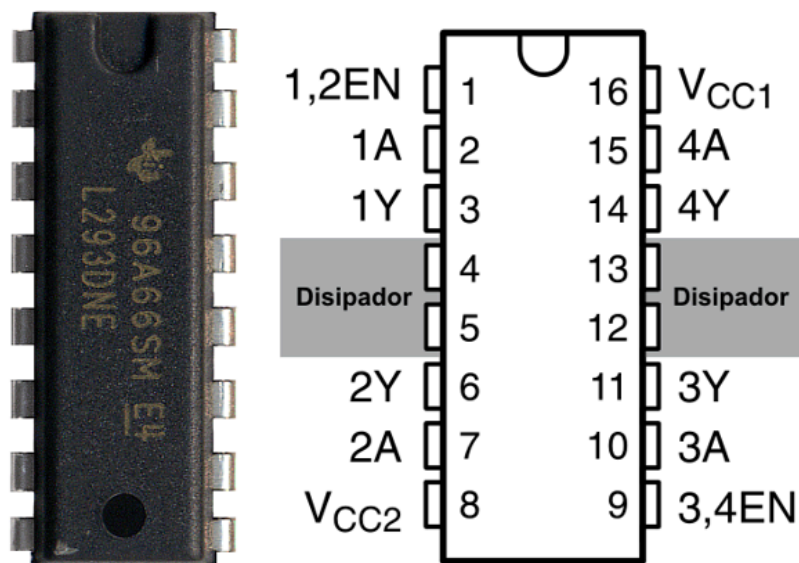
Una de las principales ventajas del controlador L293D es que permite una alimentación independiente para los motores (voltaje de potencia). Por ejemplo, se pueden controlar motores desde los 4.5 VDC hasta 36 VDC. Es importante mencionar que cuando se manejan potencias mayores a 5W ($P = V \cdot I$) es necesario utilizar buen disipador. El voltaje lógico de control es de 5V. A pesar de esta restricción, podría ser controlado con lógica de 3.3V. La desventaja de usar lógica de 3.3V es que aún requeriría una fuente de 5 VDC conectada al pin 16.

Cada puente H del L293D permite aplicar una diferencia de potencial variable en sentido directo o inverso a los terminales de un motor, posibilitando el giro en ambos sentidos. Además, el chip permite el control del encendido y apagado de los motores mediante pines de habilitación (**Enable**).

Para controlar la **velocidad** de rotación de un motor, se utiliza una señal PWM (modulación por ancho de pulso) aplicada a los pines de habilitación. De esta manera, la combinación de pines de entrada lógicos y PWM permite controlar tanto el **sentido de giro** como la **velocidad** del motor.

4.1. Conexión típica

- **IN1 / IN2 e IN3 / IN4:** Pines de control de dirección (desde el microcontrolador)
- **EN1 / EN2:** Pines de habilitación (usualmente conectados a PWM)
- **Vcc1:** Alimentación lógica (5 V)
- **Vcc2:** Alimentación del motor (p. ej., 9 V o 12 V)
- **GND:** Tierra común

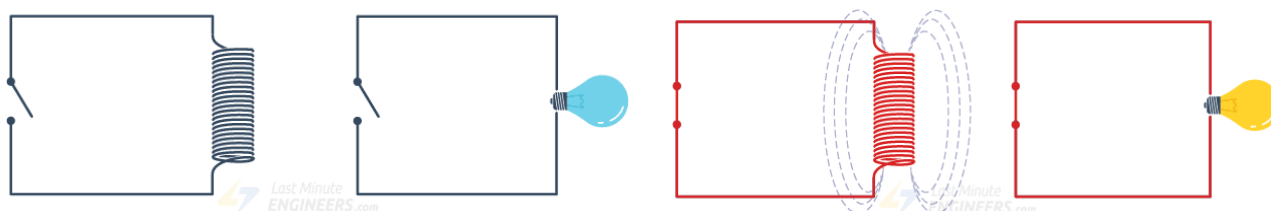


Información Adicional

<https://naylorlampmechatronics.com/drivers/223-driver-puente-h-l293d-1a-dip-16.html>

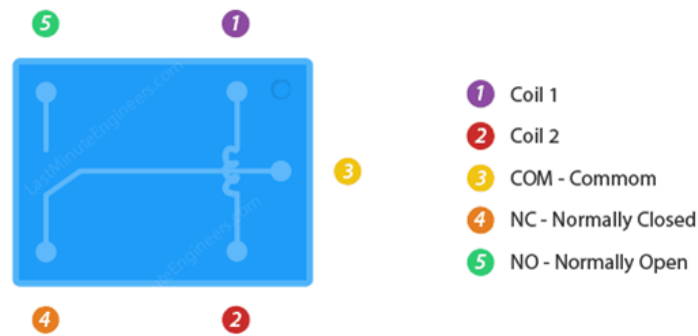
5. Relés

En el interior de un relé hay un electroimán (una bobina de alambre que se convierte en un imán temporal cuando la electricidad pasa a través de él). Se puede pensar en un relé como un interruptor que enciende una corriente relativamente pequeña y al activarse enciende otro dispositivo con una corriente mucho mayor.

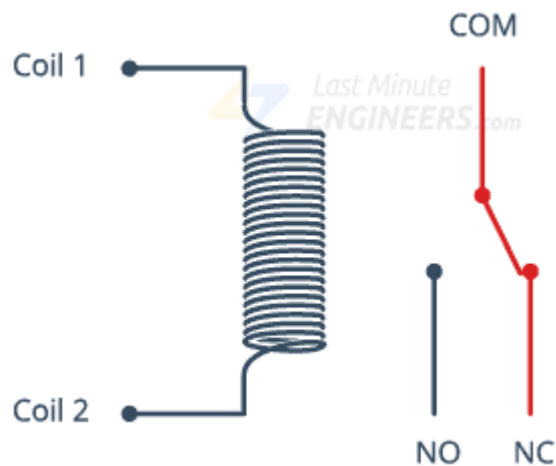


5.1. Operación de un relé

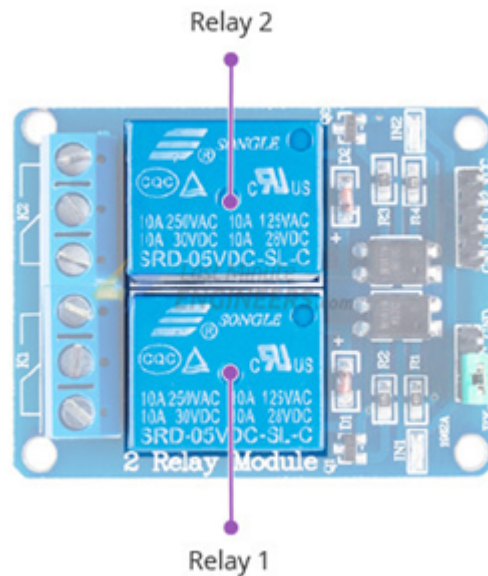
Un relé normalmente tiene cinco pines, tres de los cuales son terminales de alto voltaje (NC, COM y NO) que se conectan al dispositivo que se controla.



El dispositivo se conecta entre el terminal COM (común) y el terminal NC (normalmente cerrado) o NO (normalmente abierto), dependiendo de si el dispositivo debe permanecer normalmente encendido o apagado.

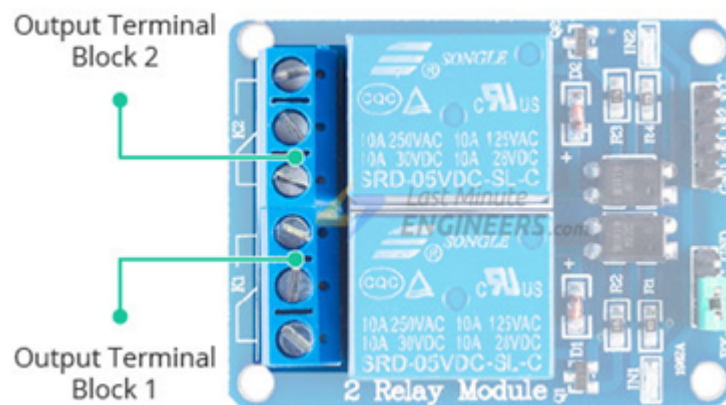


Entre los dos pines restantes (bobina 1 y bobina 2) hay una bobina que actúa como un electroimán. El módulo de relé de dos canales está diseñado para permitir que el Arduino controle dos dispositivos de alta potencia. Tiene dos relés, cada uno con una corriente nominal máxima de 10 A a 250 V CA o 30 V CC.



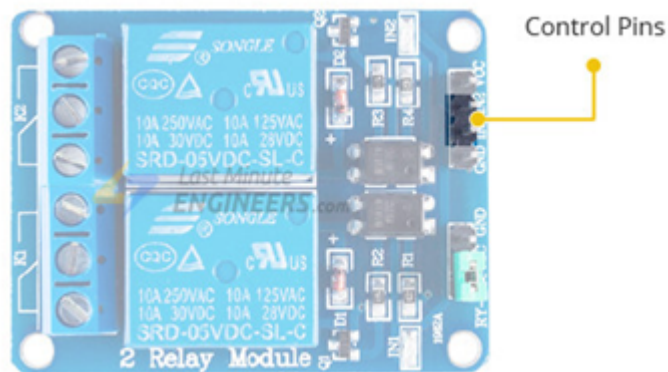
5.2. Bloques de terminales de salida

Los terminales de alto voltaje (NC, COM y NO) de cada relé se dividen en dos terminales de tornillo. El dispositivo que deseas controlar se puede conectar a través de ellos.



5.3. Control de módulos

En el otro lado del módulo, hay dos pines de entrada, IN1 e IN2, para controlar el relé. Estos pines son compatibles con la lógica de 5V, por lo que, si tienes un microcontrolador como un Arduino, puedes controlar un relé con cualquier pin de salida digital.



Los pines de entrada están activos bajos, lo que significa que una lógica BAJA activa el relé y una lógica ALTA lo desactiva.

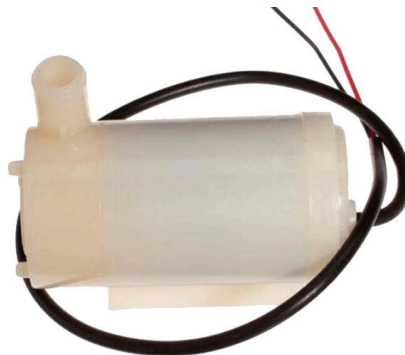
El módulo de relé tiene dos LED que indican el estado del relé. Cuando se activa un relé, se enciende el LED correspondiente.

Más información en

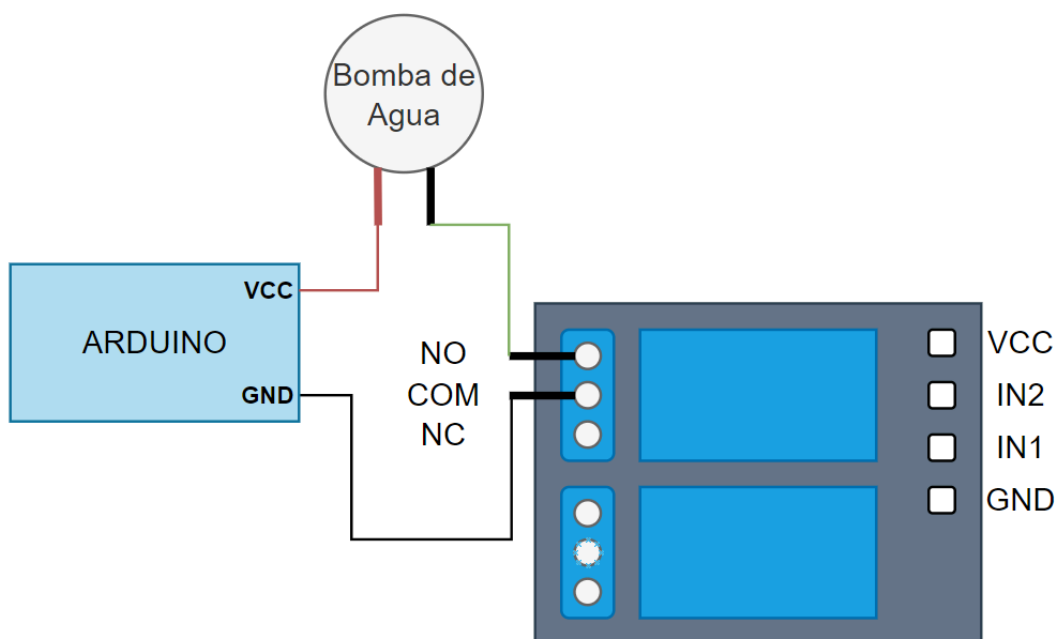
[Módulo de relé de dos canales con Arduino](#)

Bomba de Agua

La mini bomba es una pequeña bomba de agua que puede ser usada en acuarios, piletas, caño, sistemas hidropónicos o en cualquier otro lugar donde pueda ser útil. Esta bomba funciona sumergida en el agua y debe conectarse una manguera de 6mm de diámetro interno. Esta bomba de agua se alimenta con 3 a 5 VDC y requiere una corriente de 180 mA.

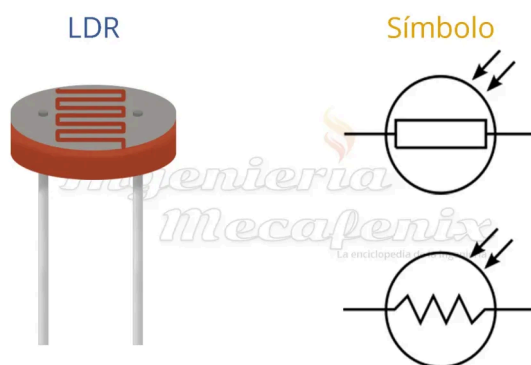


5.4. Ejemplo de Esquema de conexión (Deprecated)



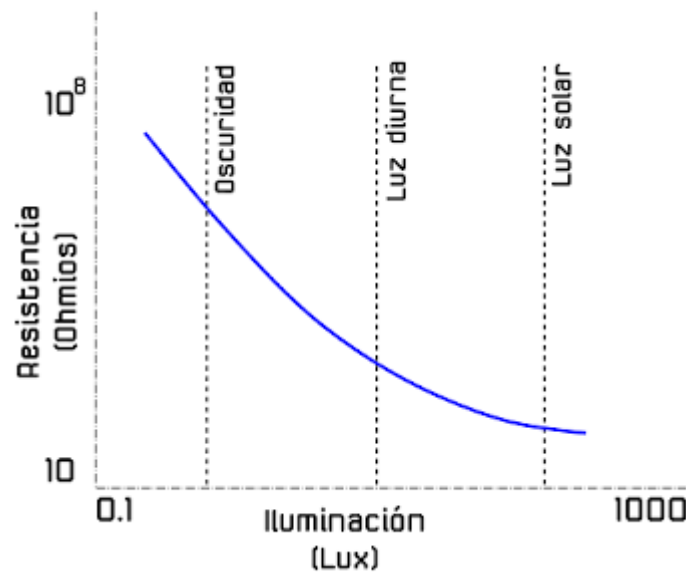
6. Fotocelda o Resistor Dependiente de Luz (LDR)

Un LDR (Light Dependent Resistor), también conocido como fotocelda o fotoresistor, es un dispositivo electrónico pasivo cuya resistencia eléctrica varía inversamente con la intensidad de la luz incidente sobre su superficie. Es decir, a medida que aumenta la luminosidad, la resistencia disminuye, y cuando la iluminación se reduce, la resistencia aumenta.



Ingeniería Mecafenix

Imagen y diagrama esquemático de un LDR



Curva de iluminación de un LDR: Resistencia vs. iluminación.

6.1. Características eléctricas típicas

- Rango de resistencia en oscuridad: $1\text{ M}\Omega$ o superior.
- Rango de resistencia bajo iluminación intensa: $100\ \Omega$ a $1\text{ k}\Omega$.
- Tensión máxima de operación: 150 V a 250 V (dependiendo del modelo).
- Corriente máxima: alrededor de 1 mA a 5 mA.

6.2. Principio de funcionamiento

El LDR está fabricado con materiales semiconductores, típicamente sulfuro de cadmio (CdS) o selenio, que presentan el efecto fotoeléctrico. Al incidir luz sobre el material, los fotones liberan electrones, aumentando así la conductividad del semiconductor y disminuyendo su resistencia.

6.3. Uso de un LDR con Arduino

Para medir la intensidad de la luz mediante un **Arduino**, el **LDR** se debe conectar formando un **divisor de voltaje** junto con una resistencia fija. El Arduino leerá la variación de voltaje a través de una de sus entradas analógicas.

6.3.1. Esquema de conexión

- Un terminal del LDR se conecta a **5V** del Arduino.
- El otro terminal del LDR se conecta a un punto común con:
 - Un extremo de la **resistencia de 10 kΩ**, cuyo otro extremo va a **GND**.
 - Una conexión hacia un pin **analógico** de Arduino (por ejemplo, **A0**).

Esto forma un **divisor de voltaje**, donde el voltaje leído en A0 depende de la iluminación.

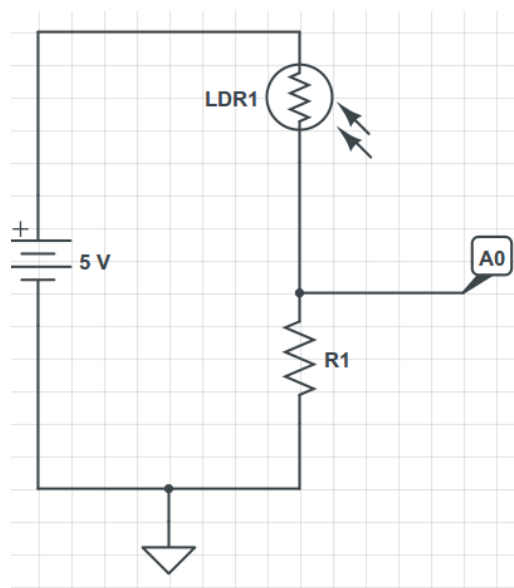


Diagrama de conexión de un LDR en configuración de divisor de voltaje.

6.3.2. Código básico de lectura:

```
int sensorPin = A0; // Pin conectado al divisor de voltaje
int sensorValue = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
}
```

```
    delay(500); // Medio segundo entre lecturas  
}
```

6.3.3. Funcionamiento

- **analogRead(A0)** devuelve un valor entre **0** (0V) y **1023** (5V).
- Mayor iluminación \Rightarrow menor resistencia LDR \Rightarrow mayor voltaje a la entrada analógica A0.
- Menor iluminación \Rightarrow mayor resistencia LDR \Rightarrow menor voltaje a la entrada analógica A0.

7. LCD 2x16

Para el correcto uso del LCD 2x16 se recomienda la guía disponible en línea de Naylamp.

Nota. Usar la librería LiquidCrystal.

[Guía de uso del LCD1602](#)

Funciones de la librería.

- **LiquidCrystal(rs, en, d4, d5, d6, d7)**

Función constructor, crea una variable de la clase LiquidCrystal, con los pines indicados.

- **begin(cols, rows)**

Inicializa el LCD, es necesario especificar el número de columnas (cols) y filas (rows) del LCD.

- **clear()**

Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda (posición (0,0)).

- **setCursor(col, row)**

Posiciona el cursor del LCD en la posición indicada por col y row (x,y); es decir, establecer la ubicación en la que se mostrará posteriormente texto escrito para la pantalla LCD.

- **write()**

Escribir un carácter en la pantalla LCD, en la ubicación actual del cursor.

- **print()**

Escribe un texto o mensaje en el LCD, su uso es similar a un Serial.print

- **scrollDisplayLeft()**

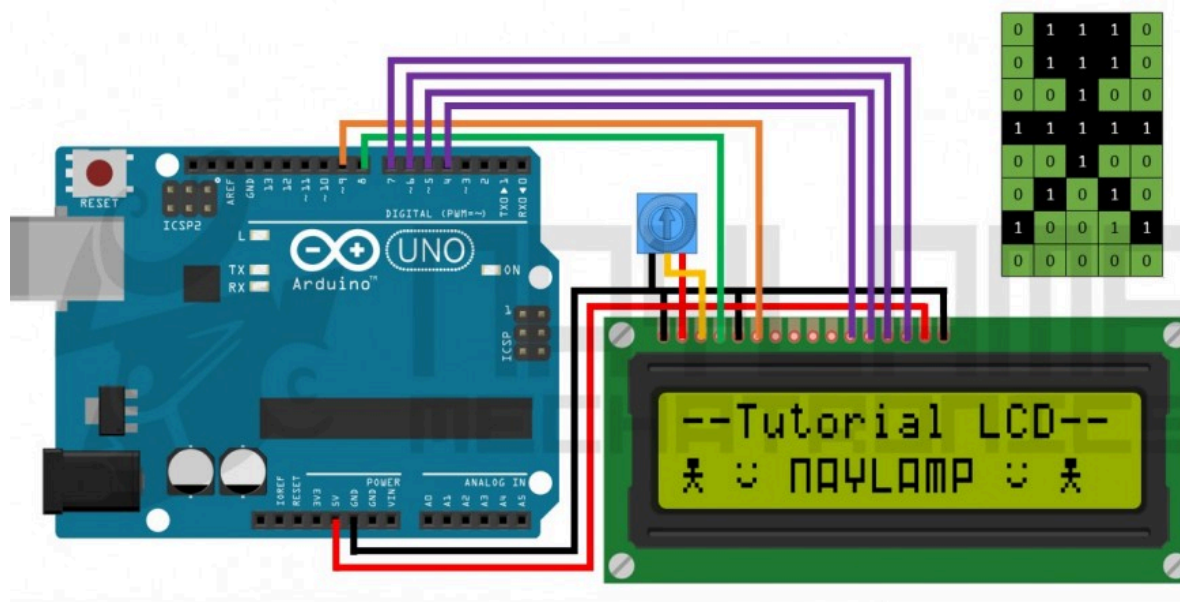
Se desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.

- **scrollDisplayRight()**

Se desplaza el contenido de la pantalla (texto y el cursor) un espacio a la derecha.

- **createChar (num, datos)**

Crea un carácter personalizado para su uso en la pantalla LCD. Se admiten hasta ocho caracteres de 5x8 píxeles (numeradas del 0 al 7). Donde: num es el número de carácter y datos es una matriz que contienen los pixeles del carácter. Se verá un ejemplo de esto mas adelante.



Recomendación.

Utilizar la menor cantidad de pines disponibles.

```
#include <LiquidCrystal.h>
```

```
//Crear el objeto LCD con los números correspondientes (rs, en, d4, d5, d6, d7) SOLO  
USANDO 6 PINES.
```

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

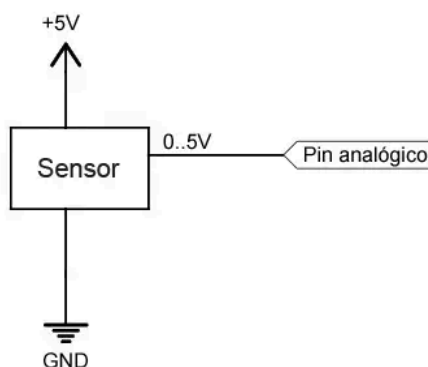
8. Lectura Análoga con Arduino

En entradas anteriores hemos visto cómo emplear las entradas digitales de nuestro Arduino. En esta entrada vamos a ver las entradas analógicas, su funcionamiento y características.

Las entradas analógicas funcionan de una forma similar a las entradas digitales, por lo que en la práctica el montaje y código final son muy similares.

Una señal analógica es una magnitud que puede tomar cualquier valor dentro de un intervalo $-V_{cc}$ y $+V_{cc}$. Por ejemplo, una señal analógica de tensión entre 0V y 5V podría valer 2,72V, o 3.41V (o cualquier otro valor con cualquier número de decimales). Por otro lado, recordemos que una señal

digital de tensión teórica únicamente podía registrar dos valores, que denominamos LOW y HIGH (en el ejemplo, 0V o 5V).

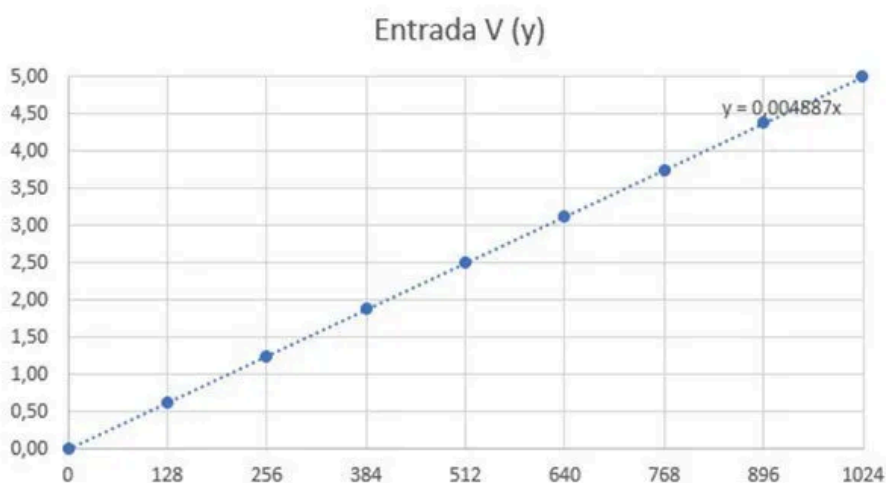


Se utiliza el comando `analogRead()`

`sensorValue = analogRead(sensorPin);`

Donde el valor retornado por `sensorValue` es un número entre 0 y 1024 (dependiendo de la resolución de bits del ADC del MCU usado).

$y = 0,004887x$		
N°	ADC (x)	Entrada V (y)
0	0	0,00
1	128	0,63
2	256	1,25
3	384	1,88
4	512	2,50
5	640	3,13
6	768	3,75
7	896	4,38
8	1023	5,00



Referencias.

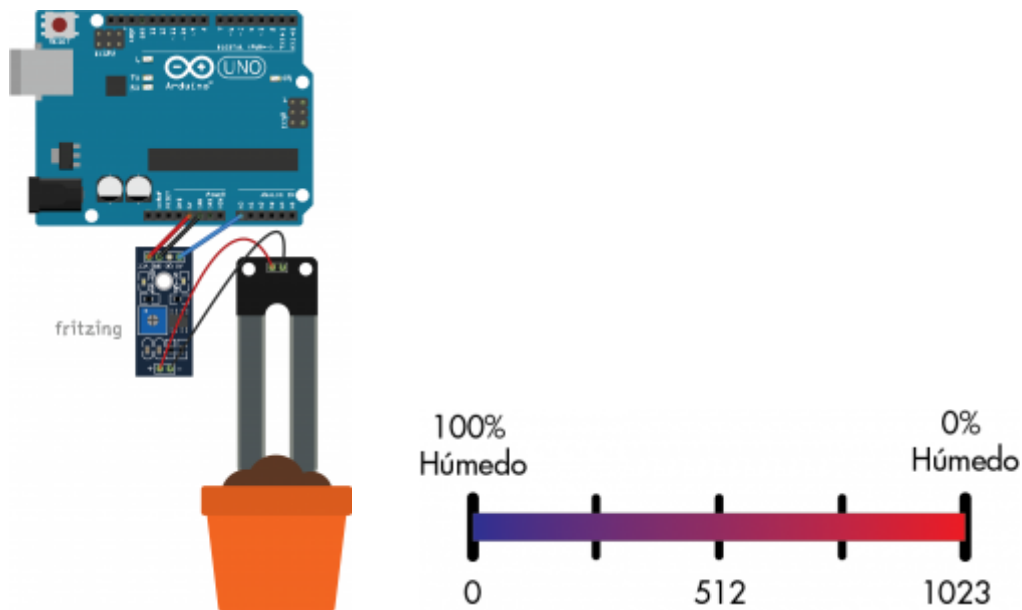
[Entradas Analógicas en Arduino](#)

[analogRead\(\) - Arduino](#)

9. Sensor de Humedad

Referencia.

[Sensor de Humedad con Arduino](#)



Escala del sensor de humedad.

Código Ejemplo

```
int SensorPin = A0;
void setup() {
  pinMode(7,OUTPUT);
  Serial.begin(9600);
}
void loop() {
  int humedad = analogRead(SensorPin);
  Serial.println(humedad);
  if(humedad>=460)
  {
    digitalWrite(7,LOW);
  }
  else
  {
    digitalWrite(7,HIGH);
  }
}
```

```
}  
delay(1000);  
}
```

10. Sensor de Temperatura

Referencia:

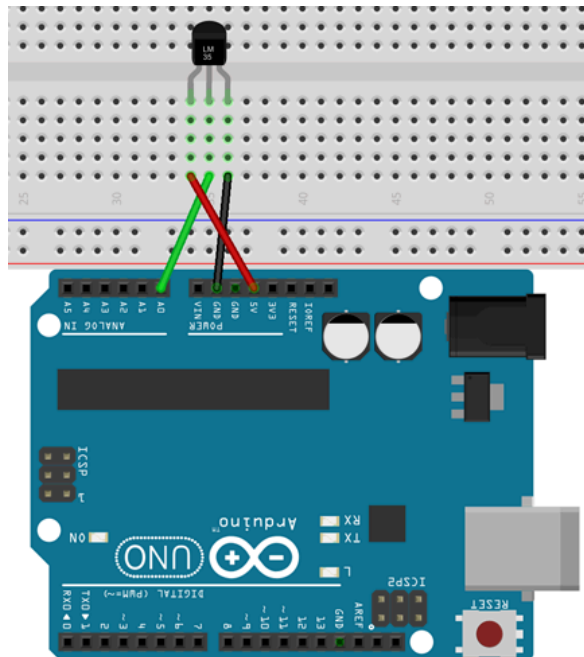
[Sensor de Temperatura con Arduino](#)

[Guia PDF de Sensor de Temperatura](#)

Nota.

Este sensor utiliza una fórmula para convertir el valor analógico medido a temperatura.

$$\text{Temperatura} = \text{Valor} * 5 * 100 / 1024$$



Código Ejemplo

```
// Declaracion de variables globales  
float tempC; // Variable para almacenar el valor obtenido del sensor (0 a 1023)  
int pinLM35 = 0; // Variable del pin de entrada del sensor (A0)  
  
void setup() {
```

```
// Configuramos el puerto serial a 9600 bps
Serial.begin(9600);

}

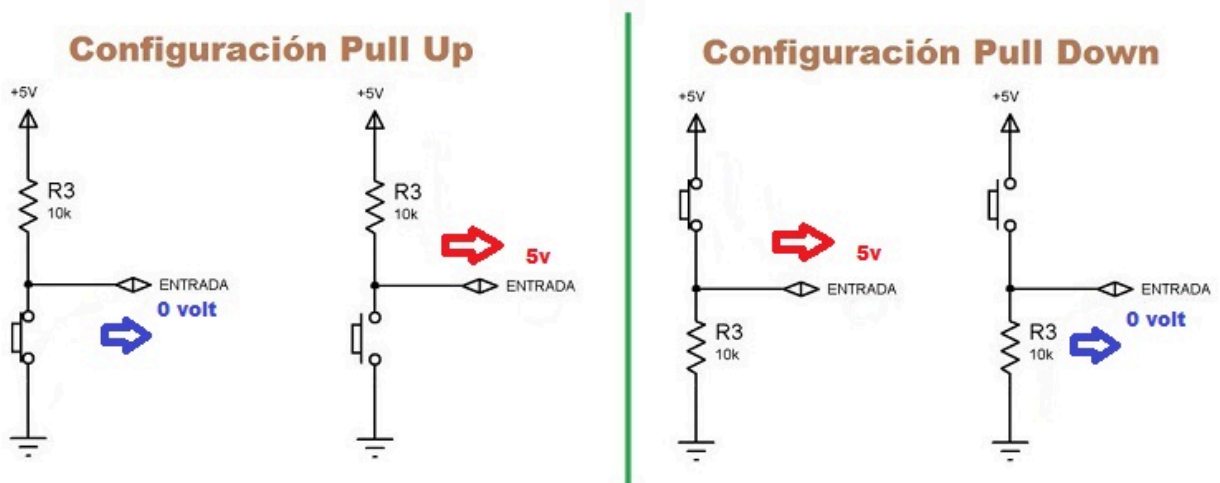
void loop() {
  // Con analogRead leemos el sensor, recuerda que es un valor de 0 a 1023
  tempC = analogRead(pinLM35);

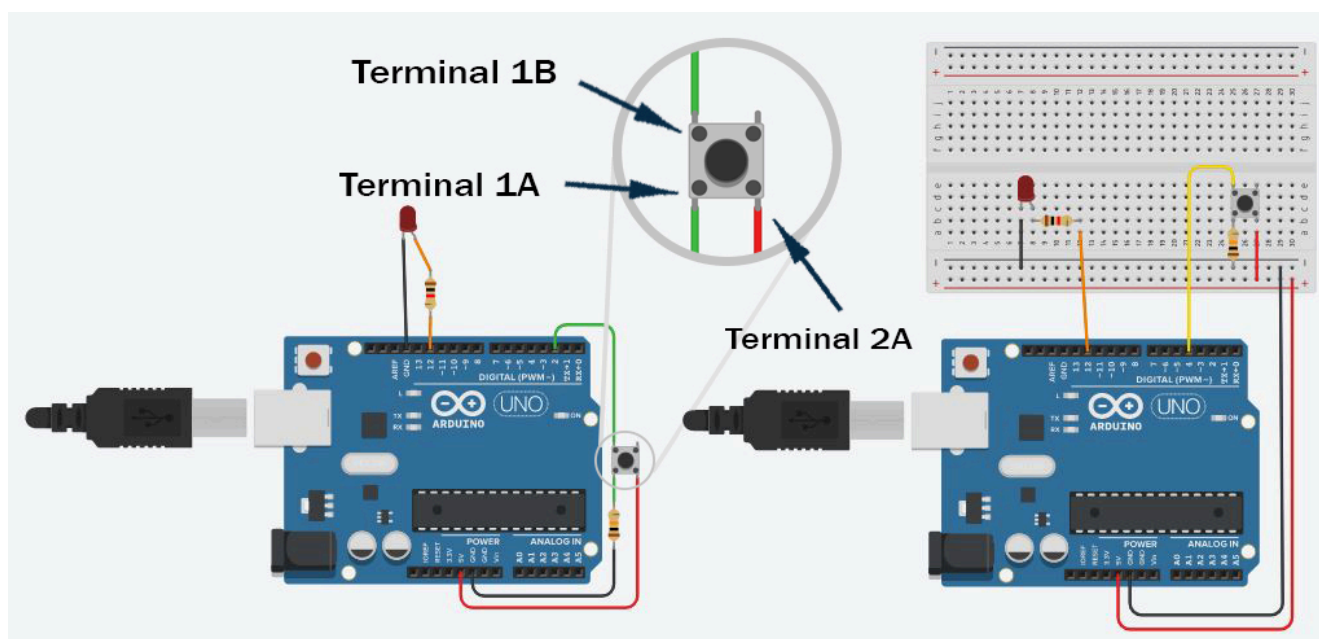
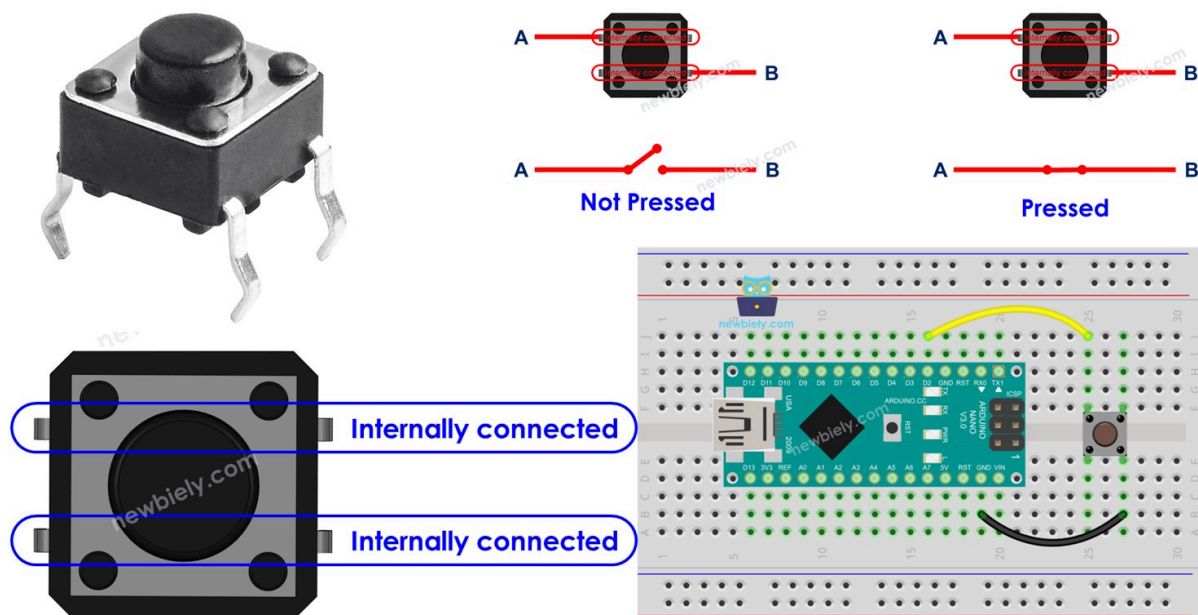
  // Calculamos la temperatura con la fórmula
  tempC = (5.0 * tempC * 100.0)/1024.0;

  // Envía el dato al puerto serial
  Serial.print(tempC);
  // Salto de línea
  Serial.print("\n");

  // Esperamos un tiempo para repetir el loop
  delay(1000);
}
```

11. El uso de botones con Arduino.





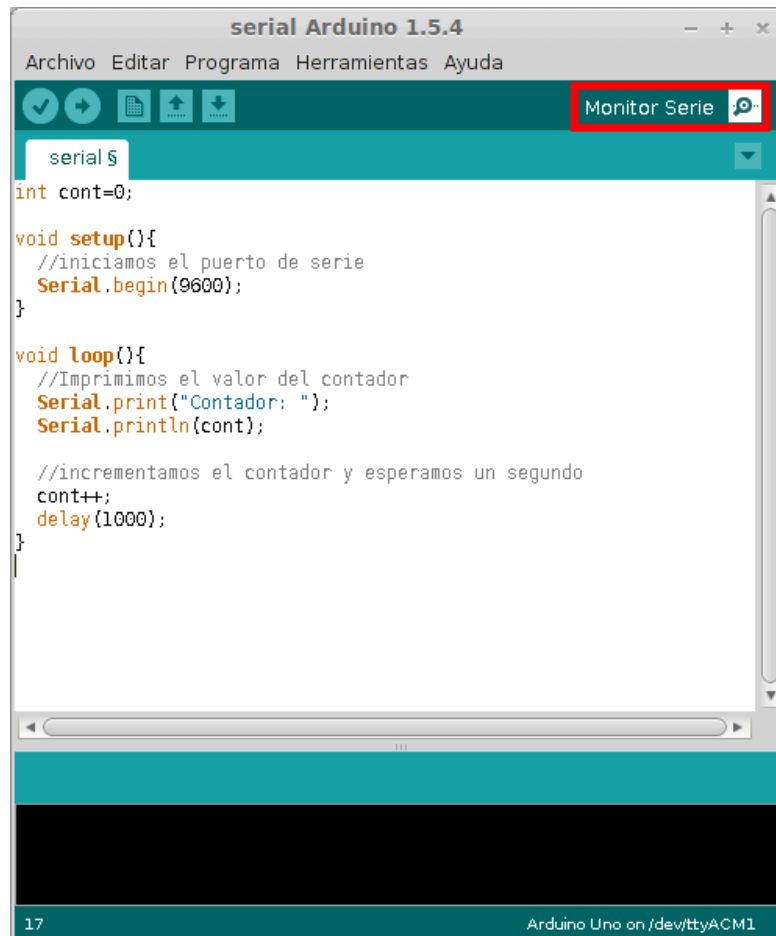
12. Uso del monitor Serial.

Usar referencia:

<https://aprendiendoarduino.wordpress.com/category/monitor-serie/>

El monitor serial es el 'cable' entre el ordenador y el Arduino UNO. Permite enviar y recibir mensajes de texto, útiles para la depuración y también control de Arduino. Por ejemplo, es posible enviar comandos desde el ordenador para encender LEDs.

Después de que han subido el sketch sobre el Arduino UNO, haga clic en el botón derecho en la barra de herramientas en el IDE de Arduino.



Ejemplo

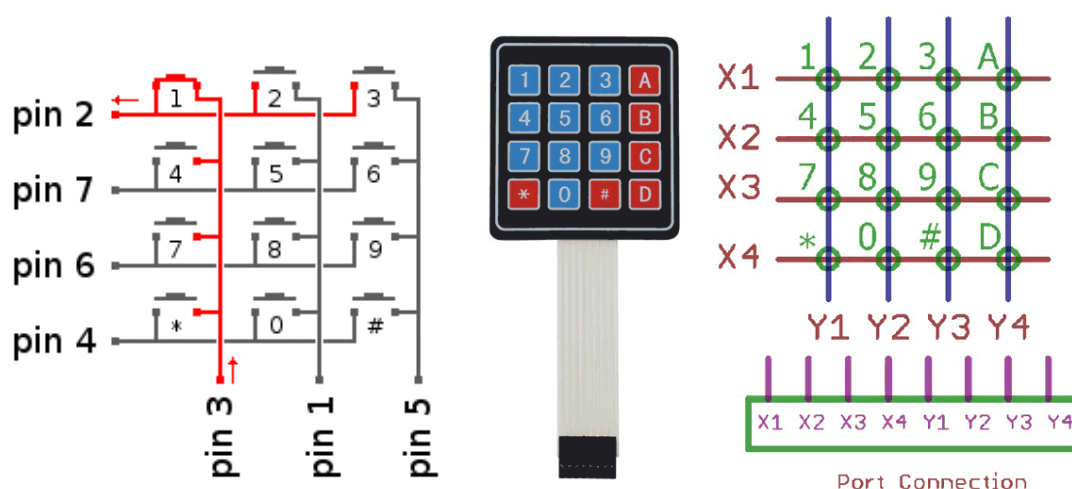
Encender o apagar el LED integrado en la placa Arduino. Para ello enviamos un carácter a la placa Arduino, empleando el monitor serial. En caso de enviar 'a' la placa Arduino apaga el LED, y en caso de enviar 'b' lo enciende.

```
1  int option;  
2  int led = 13;  
3  
4  void setup(){  
5    Serial.begin(9600);  
6    pinMode(led, OUTPUT);  
7  }  
8  
9  void loop(){  
10   //si existe datos disponibles los leemos  
11   if (Serial.available() > 0){  
12     //leemos la opcion enviada  
13     option = Serial.read();  
14     if(option == 'a') {  
15       digitalWrite(led, LOW);  
16       Serial.println("OFF");  
17     }  
18     if(option == 'b') {  
19       digitalWrite(led, HIGH);  
20       Serial.println("ON");  
21     }  
22   }  
23 }
```

13. Uso del teclado matricial

Usar referencia:

<https://controlautomaticoeducacion.com/sistemas-embebidos/arduino/teclado-matricial-keypad/>



Una forma rápida de usar un Teclado con Arduino, es valernos de su librería Keypad (<https://www.arduino.cc/reference/en/libraries/keypad/>), bastante sencilla de entender, para el caso de un teclado matricial 4×4 (KEYPAD 4×4) tenemos:

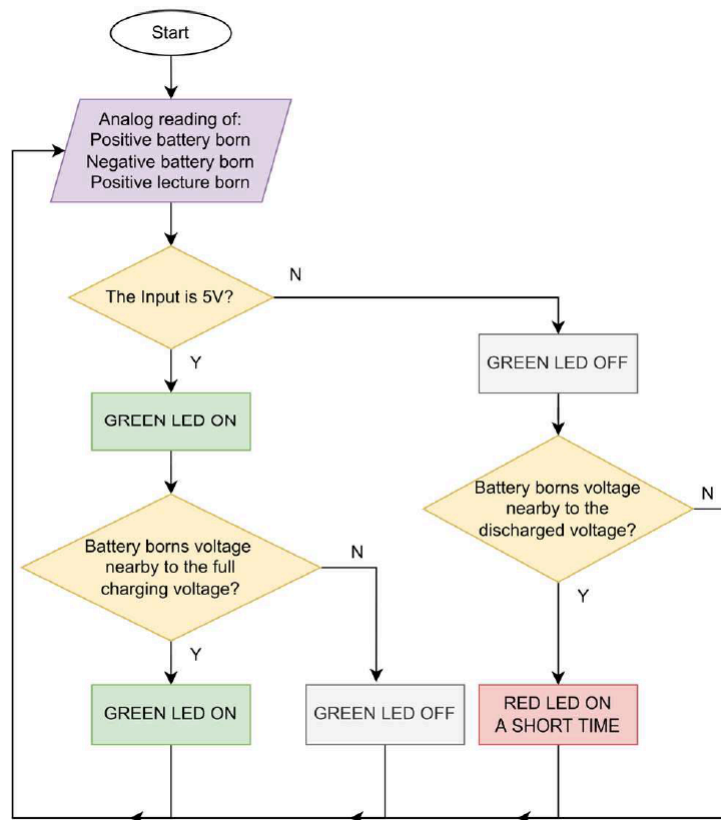
```
#include <Keypad.h>
const byte filas = 4;
const byte columnas = 4;
byte pinesFilas[] = {9,8,7,6};
byte pinesColumnas[] = {5,4,3,2};
char teclas[4][4] = {{ '1','2','3','A'},
                     { '4','5','6','B'},
                     { '7','8','9','C'},
                     { '*', '0', '#', 'D' } };

Keypad teclado1 = Keypad( makeKeymap(teclas), pinesFilas, pinesColumnas, filas, columnas);
void setup() {
  Serial.begin(9600);
  Serial.println("Teclado 4x4 con Biblioteca Keypad");
  Serial.println();
}
void loop() {
  //Verifica si alguna tecla fue presionada
  char tecla_presionada = teclado1.getKey();

  //Monitor Serial
  if (tecla_presionada)
  {
    Serial.print("Tecla: ");
    Serial.println(tecla_presionada);
  }
}
```

14. Diagramas de flujo




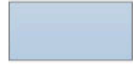
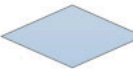
Los **diagramas de flujo** son representaciones gráficas que describen de manera secuencial los pasos o procesos necesarios para llevar a cabo una tarea específica. Su principal objetivo es facilitar la comprensión, el análisis y la comunicación de procesos complejos mediante el uso de **símbolos normalizados** y conectores que indican el flujo lógico de las operaciones.



14.1. Simbología básica

A continuación, se describen los símbolos más comunes utilizados en los diagramas de flujo:

- **Inicio / Fin:** Representado por un óvalo. Indica el punto de entrada o salida del proceso.
- **Proceso:** Representado por un rectángulo. Indica una acción, tarea o conjunto de operaciones.
- **Decisión:** Representado por un rombo. Indica un punto de bifurcación basado en una condición lógica (sí/no).
- **Entrada / Salida:** Representado por un paralelogramo. Indica operaciones de ingreso de datos o presentación de resultados.
- **Conector:** Círculo pequeño que permite enlazar diferentes partes del diagrama cuando este no puede ser representado de forma continua.

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

14.2. Ejemplo de Diagrama de Flujo para un código simple.

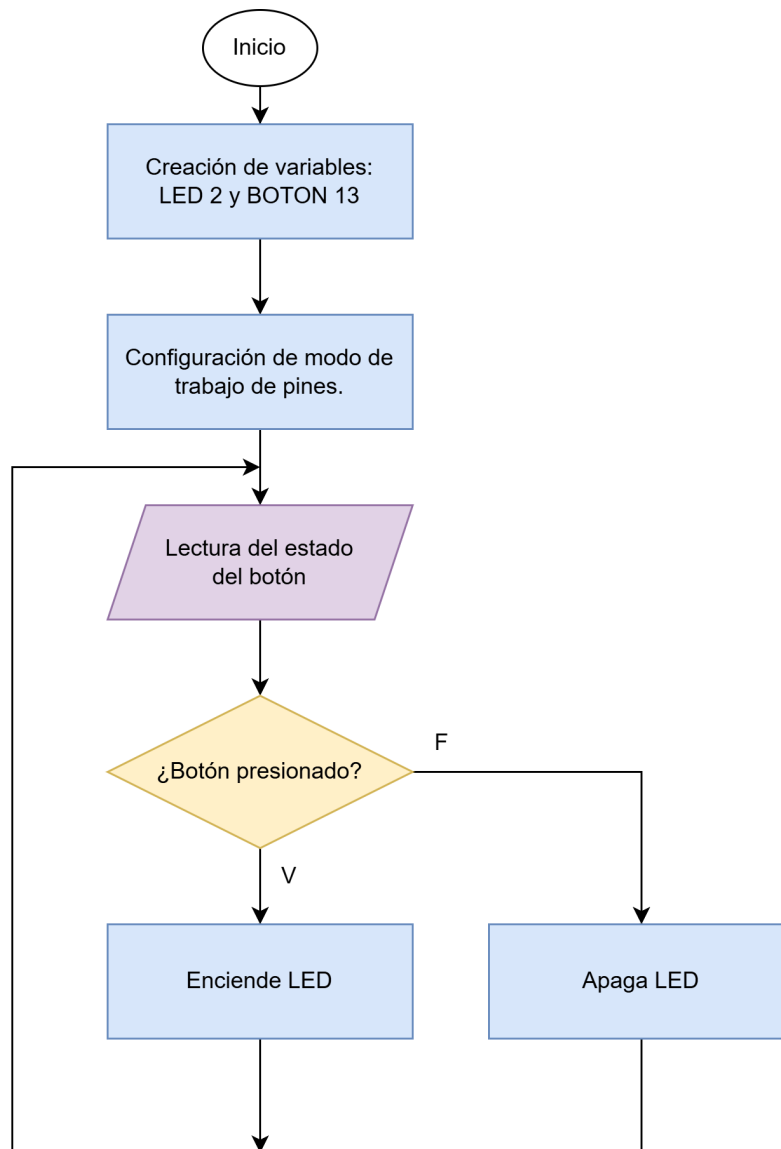
Ejemplo sencillo de código para Arduino que enciende un LED cuando se presiona un botón. Este ejemplo asume que tienes un botón conectado al pin digital 2 y un LED al pin digital 13.

```
const int botonPin = 2;    // Pin donde está conectado el botón
const int ledPin = 13;     // Pin donde está conectado el LED
int estadoBoton = 0;       // Variable para leer el estado del botón

void setup() {
  pinMode(botonPin, INPUT); // Configura el pin del botón como entrada
  pinMode(ledPin, OUTPUT);  // Configura el pin del LED como salida
}

void loop() {
  estadoBoton = digitalRead(botonPin); // Lee el estado del botón

  if (estadoBoton == HIGH) {
    digitalWrite(ledPin, HIGH); // Enciende el LED si el botón está presionado
  } else {
    digitalWrite(ledPin, LOW);  // Apaga el LED si el botón no está presionado
  }
}
```



15. Diagrama de Bloques

Un **diagrama de bloques** es una representación gráfica de un sistema electrónico en la que se describen sus principales funciones o subsistemas mediante bloques rectangulares interconectados. Cada bloque representa una función específica, como amplificación, filtrado, conversión o control, sin detallar su implementación interna. Esta herramienta se utiliza ampliamente en la etapa de diseño conceptual para visualizar el flujo de señales, la jerarquía funcional del sistema y la interacción entre los distintos módulos.

15.1. Objetivo de su Uso en el Diseño Electrónico

El propósito de los diagramas de bloques es proporcionar una visión estructurada y abstracta del sistema a diseñar. Facilitan:

- La **planificación del diseño** a nivel macro.
- La **comunicación entre diseñadores**, al ofrecer un lenguaje común.
- La **identificación de módulos reutilizables** o estándar.
- La **definición de interfaces** entre subsistemas.

15.2. Componentes Típicos

Un diagrama de bloques está compuesto por:

- **Bloques funcionales:** Representan unidades lógicas del sistema (p. ej., fuente de alimentación, amplificador, ADC, microcontrolador).
- **Líneas de conexión:** Indican el flujo de señales o energía entre los bloques.
- **Etiquetas o señales:** Denotan entradas, salidas o parámetros críticos (p. ej., Vcc, Vin, GND, CLK, UART).

15.3. Ejemplo

