

LAB 2: Sensores y Actuadores

PARTE II: 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. (Solo simulación) Encendido de un Motor DC con Arduino | 4 |
| 1.1. Consideraciones | 4 |
| 1.2. Responda | 5 |
| 1.3. Presentar en laboratorio | 5 |
| 2. (Solo simulación) Control de un motor con Driver | 5 |
| 2.1. Consideraciones | 5 |
| 2.2. Responda | 5 |
| 2.3. Presentar: | 6 |
| 3. Uso de Relé con Motor | 6 |
| 3.1. Materiales | 7 |
| 3.2. Implemente | 7 |
| 3.2.1. Fuente y motor | 7 |
| 3.2.2. Arduino y módulo relé | 7 |
| 3.2.3. Lógica de control | 8 |
| 3.3. Nota: | 8 |
| 3.4. Presentar en laboratorio | 8 |
| 3.5. Uso de Driver Puente H L298N con Motor | 9 |
| 3.6. Materiales necesarios | 9 |
| 3.7. Conexiones | 9 |
| 3.7.1. Alimentación del L298N y motor | 9 |
| 3.7.2. Motor | 9 |
| 3.7.3. Conexión de control al Arduino | 10 |
| 3.8. Notas | 10 |
| 3.9. Se pide | 11 |
| 3.10. Responda | 11 |
| 3.11. Presentar en Laboratorio | 11 |
| 4. Sistema de riego inteligente | 12 |
| 4.1. Componentes | 12 |
| 4.2. Riego de planta | 12 |
| 4.3. IDLE | 12 |
| 4.4. Alerta | 12 |
| 4.5. Presentar: | 13 |
| 5. ANEXO 1. → Actuadores: Motores | 14 |
| 6. ANEXO 2. → PWM - Modulación por Ancho de Pulses | 15 |
| 6.1. Fundamento de funcionamiento | 15 |
| 6.2. Aplicación en el control de motores | 16 |
| 6.3. Ventajas del uso de PWM | 16 |

| | |
|--|-----------|
| 6.4. Aplicaciones comunes | 16 |
| 7. ANEXO 3. → PWM con Arduino | 17 |
| 7.1. Pines PWM disponibles en Arduino Mega | 17 |
| 7.2. Sintaxis básica para usar PWM | 18 |
| 7.3. Ejemplo de control de velocidad de motor DC con PWM. | 18 |
| 8. ANEXO 4. → Driver Puente H L293D (Circuito Integrado) | 19 |
| 8.1. Conexión típica | 20 |
| 9. ANEXO 5. → Driver Puente H L298N 2A | 21 |
| 9.1. Importante | 21 |
| 9.2. Cómo alimentarlo | 22 |
| 9.3. Cómo usarlo con Arduino | 22 |
| 9.4. Tabla de direcciones | 26 |
| 10. ANEXO 6. → Fuente para protoboard 3.3V/5V (USB-A) - MB102 | 27 |
| 10.1. Especificaciones: | 27 |
| 11. ANEXO 7. → Motor DC TT 6V/200RPM | 29 |
| 11.1. Especificaciones Técnicas | 29 |
| 12. ANEXO 8. → Relé | 31 |
| 12.1. Operación de un relé | 31 |
| 12.2. Conexión de Relé | 32 |
| 12.2.1. Ejemplo en Tinkercad | 33 |
| 12.3. Módulo Relé para Arduino | 34 |
| 12.3.1. Bloques de terminales de salida | 34 |
| 12.3.2. Control de módulos | 35 |
| 13. ANEXO 9. → Bomba de Agua | 36 |
| 14. ANEXO 10. → Lectura Análoga con Arduino | 37 |
| 15. ANEXO 11. → Sensor de Humedad | 39 |
| 16. ANEXO 12. → Sensor de Temperatura | 40 |
| 17. ANEXO 13. → El uso de botones con Arduino. | 42 |
| 18. ANEXO 14. → Uso del monitor Serial. | 44 |
| 19. ANEXO 15. → Uso del teclado matricial | 46 |
| 20. ANEXO 16. → Diagramas de flujo | 48 |
| 20.1. Simbología básica | 48 |
| 20.2. Ejemplo de Diagrama de Flujo para un código simple. | 49 |
| 21. ANEXO 17. → Diagrama de Bloques | 52 |
| 21.1. Objetivo de su Uso en el Diseño Electrónico | 52 |
| 21.2. Componentes Típicos | 52 |
| 21.3. Ejemplo | 53 |

PARTE II: Actuadores

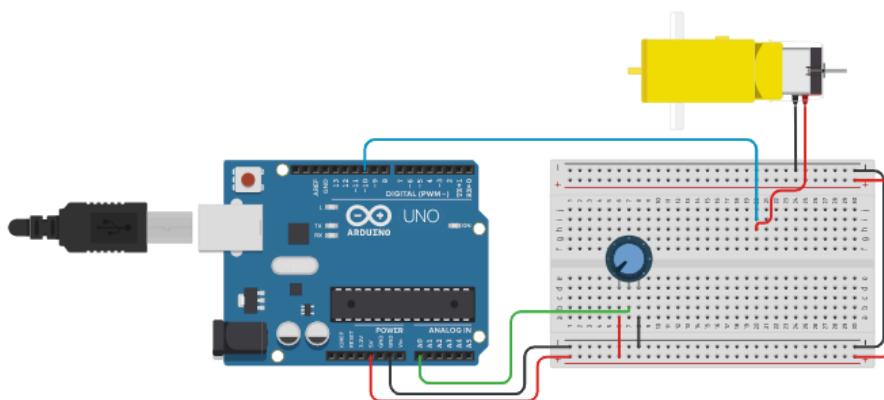
En esta parte de la experiencia realizará implementaciones de sistemas con sensores y actuadores. Los actuadores serán empleados principalmente con alimentación externa. Es decir, con una fuente de alimentación directa ya que los actuadores generalmente consumen más corriente al ser dispositivos inductivos.

1. (Solo simulación) 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. ¿Qué sucede en su simulación con el Arduino?
2. ¿Cuál es la corriente que consume el motor durante su operación?
3. Si usted quisiera regular la velocidad del motor (intensidad de corriente), ¿qué añadiría a su esquema de conexión?

1.3. Presentar en laboratorio

1. **Funcionamiento** del sistema en simulación.
2. **(Informe) Diagrama de Bloques** del sistema implementado.
3. **Respuesta** de las preguntas.

[Checkpoint 6]

2. (Solo simulación) Control de un motor con Driver

Implementar el control de un motorreductor CC con driver L293D (**Anexo 4**), 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.

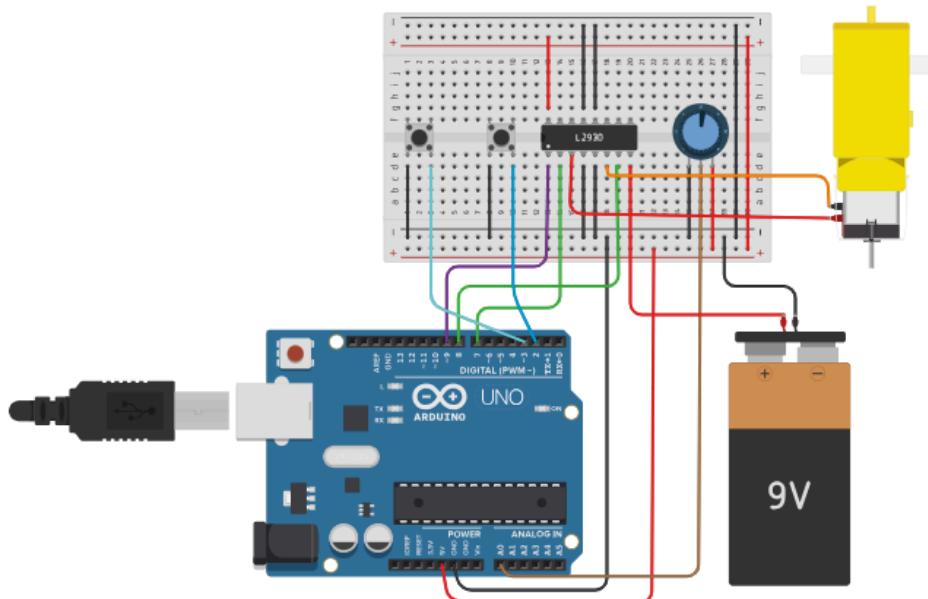
2.2. Responda

- ¿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:

1. **Funcionamiento del sistema** en simulación.
2. **(Informe) Diagrama de Bloques** del sistema implementado.
3. **Respuesta** de las preguntas.



[Checkpoint 7]

3. Uso de Relé con Motor

En esta experiencia de laboratorio se explorarán dos métodos fundamentales para el control de motores de corriente directa (DC) utilizando el microcontrolador Arduino MEGA como unidad de control.

En primer lugar, se emplea un módulo de relé (**Anexo 8**) para controlar el encendido y apagado de un motor DC utilizando una fuente de alimentación externa (la fuente para protoboard). Este enfoque permite aislar eléctricamente la etapa de potencia del circuito de control, siendo útil cuando se requiere accionar cargas de mayor corriente o tensión que las soportadas por el microcontrolador.

3.1. Materiales

- 1 Arduino (UNO, Nano, etc.)
- 1 Módulo de relé de 1 canal (de 5V)
- 1 Motor DC de 6V
- 1 Fuente de alimentación para protoboard con entrada de 12V y salida de 5V/3.3V
- 1 Fuente de 12V (puede ser adaptador o batería)
- Protoboard y cables
- Diodo 1N4007 o similar (protección contra corriente inversa)

3.2. Implemente

3.2.1. Fuente y motor

- Conecta la fuente de 12V a la entrada de la **fuente de protoboard (Anexo 6)**.
- Conecta el **positivo de la fuente de protoboard 5V** al **común (COM)** del relé.
- Conecta el **normalmente abierto (NO)** del relé a uno de los terminales del motor.
- El otro terminal del motor va al **GND** de la fuente de protoboard.
- Coloca un **diodo 1N4007** en paralelo con el motor (ánodo al GND, cátodo al terminal positivo) para evitar picos de voltaje cuando se apague el motor.

3.2.2. Arduino y módulo relé

Nota: Estas instrucciones son para la conexión del módulo PCB con 2 relés. Para la simulación en Tinkercad referirse al **Anexo 8**, Figure 8 y Figure 9. Adicionalmente se le brinda un ejemplo en Tinkercad.

- VCC del relé → 5V del Arduino
- GND del relé → GND del Arduino
- IN1 del relé → pin digital del Arduino (por ejemplo, pin 7)

3.2.3. Lógica de control

- Cuando el Arduino pone el pin 7 en **LOW** (en algunos módulos de relé), el relé se activa y se cierra el circuito entre COM y NO, encendiendo el motor.
- Al poner el pin 7 en **HIGH**, el relé se desactiva y el motor se apaga.

3.3. Nota:

- El motor recibirá 5V, lo cual es seguro aunque no esté operando a su máximo rendimiento.

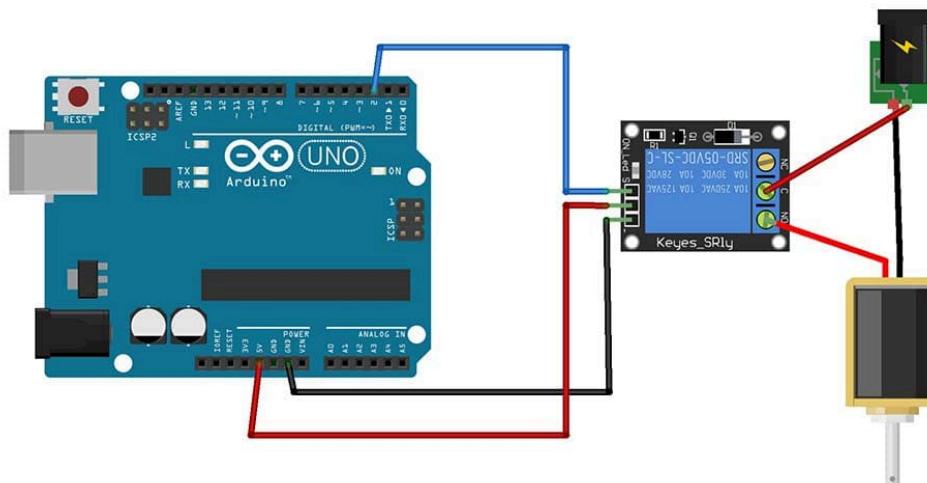


Figure 1. Diagrama de conexión con relé.

3.4. Presentar en laboratorio

1. **Simulación** del sistema. (Llame al docente cuando tenga la simulación)

[Checkpoint 8]

- **Implementación** del sistema.
- **(Informe)** Diagrama de Flujo del sistema implementado.
- **(Informe)** Diagrama de Bloques del sistema implementado.

[Checkpoint 9]

3.5. Uso de Driver Puente H L298N con Motor

En segundo lugar, se va a emplear un **driver de motor basado en el integrado L298N**, que implementa una topología de **puente H**. Este método permite no solo encender o apagar el motor, sino también invertir su dirección de giro y controlar su velocidad mediante señales PWM. El uso del L298N representa una solución más versátil y eficiente para aplicaciones donde se requiere un control dinámico del motor.

3.6. Materiales necesarios

- 1 Arduino (UNO, Nano, etc.)
- 1 Módulo driver L298N
- 1 Motor DC de 6V
- 1 Fuente de alimentación de 12V conectada a la fuente de protoboard
- 1 Módulo fuente para protoboard (salida 5V)
- Cables y protoboard

3.7. Conexiones

3.7.1. Alimentación del L298N y motor

1. **VCC del L298N → 5V de la fuente para protoboard.**
2. **GND del L298N → GND común** (puede ser el GND del Arduino o de la fuente).
3. **Pin 12V del L298N** (si está disponible como entrada independiente) **NO se conecta**, ya que usaremos solo 5V.
4. **Puente ENA activado (jumper colocado)** para habilitar el canal A.

3.7.2. Motor

5. Conecta el **motor DC** entre los pines **OUT1** y **OUT2** del L298N.

3.7.3. Conexión de control al Arduino

6. **IN1** del L298N → pin digital 8 del Arduino
7. **IN2** del L298N → pin digital 9 del Arduino
8. **ENA** del L298N → pin 10 (PWM) del Arduino
9. Retira el jumper de ENA si está colocado.

3.8. Notas

- El pin ENA controla la velocidad mediante señales PWM. Mientras más alto el valor (hasta 255), mayor será la velocidad.
- IN1 e IN2 definen la dirección de giro del motor.
- Esta configuración solo usa 1 canal del L298N, dejando libre el segundo para otro motor si se desea.

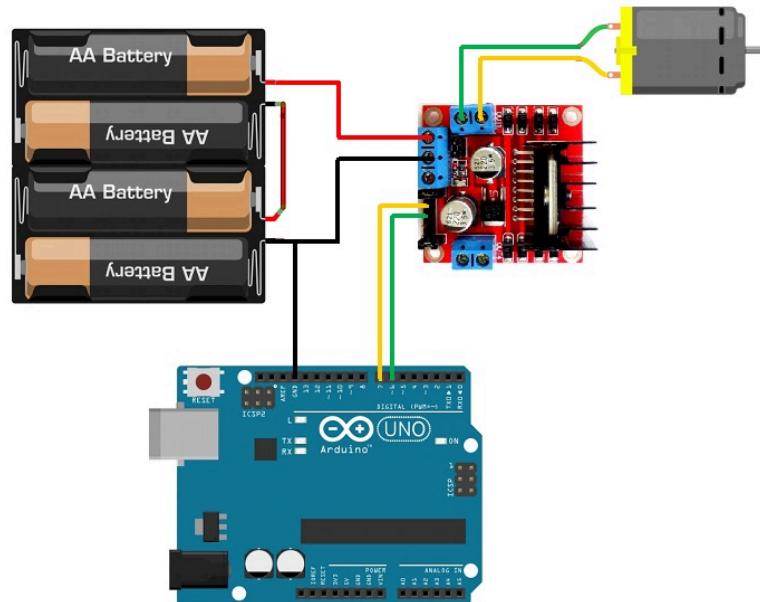


Figure 2. Diagrama de conexión. En lugar de las baterías considere la fuente de 5V de su Fuente para Protoboard (**Anexo 6**).

3.9. Se pide

- a. Prueba alimentar el módulo L298N directamente desde el pin de 5V del Arduino (en lugar de la fuente para protoboard). ¿Notas alguna diferencia en el rendimiento del motor? Justifica técnicamente tu observación.

Graba un video para mostrarlo al docente.

- b. Restablezca la alimentación del motor. Modifica el circuito para invertir el sentido de giro automáticamente cada 3 segundos.

3.10. Responda

- c. ¿Qué función cumple el pin ENA en el módulo L298N y por qué debe conectarse a un pin PWM del Arduino?
- d. ¿Qué sucede si se intercambian las conexiones de IN1 e IN2? ¿Cómo afecta esto a la dirección del giro del motor?

3.11. Presentar en Laboratorio

1. **Implementación** del sistema con cambio de sentido de giro.
2. **Respuesta** a las preguntas.
3. **(Informe)** Diagrama de Flujo del sistema implementado.
4. **(Informe)** Diagrama de Bloques del sistema implementado.

[Checkpoint 10]

4. Sistema de riego inteligente

Diseñe un sistema de riego inteligente el debe obtener el nivel de humedad en la tierra mediante el uso del **sensor de Humedad de tierra**.

4.1. Componentes

- Sensor de humedad de tierra (**Anexo 11**)
- Buzzer
- **Bomba de agua (Anexo 9)**
- **Relé (Anexo 8)**
- Leds
- Resistencias

El sistema tendrá los siguientes funcionamientos:

4.2. Riego de planta

Si la humedad detectada está por debajo del nivel umbral (20% de humedad), el sistema deberá entrar a un modo de riego de planta:

- Deberá encenderse un LED AZUL, indicando que se ha ingresado al estado de riego.
- Así mismo, el sistema deberá de accionar una bomba de agua durante un tiempo t_{riego} . (Para la experiencia de laboratorio, este tiempo será de 10 segundos).

Tomar en consideración el uso de relés (Más información el Anexo 8).

- Pasado el tiempo indicado, el sistema entrará en modo **IDLE**.
- A través del Monitor Serial debe indicar el nivel de humedad detectado por el sensor

4.3. IDLE

Este estado es de espera. Específicamente tras regar la planta.

- La bomba debe apagarse, y no realizar mediciones durante un tiempo de 15 segundos.
- El Monitor Serial deberá mostrar un mensaje: “Modo IDLE”.

4.4. Alerta

Si la humedad detectada está por encima de un nivel máximo (75%), el sistema deberá:

- Encender una alarma (mediante un LED ROJO y el buzzer) para indicar que la planta se encuentra sobre-hidratada, y es necesaria la asistencia de una persona.
- Tanto el LED como el buzzer deben funcionar de manera intermitente.

4.5. Presentar:

1. **Implementación y funcionamiento** del sistema (explicado).
2. **Diagrama de Flujo** del sistema implementado.
3. **Diagrama de Bloques** del sistema implementado.

[Checkpoint 11]

ANEXOS TÉCNICOS

5. ANEXO 1. → 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.



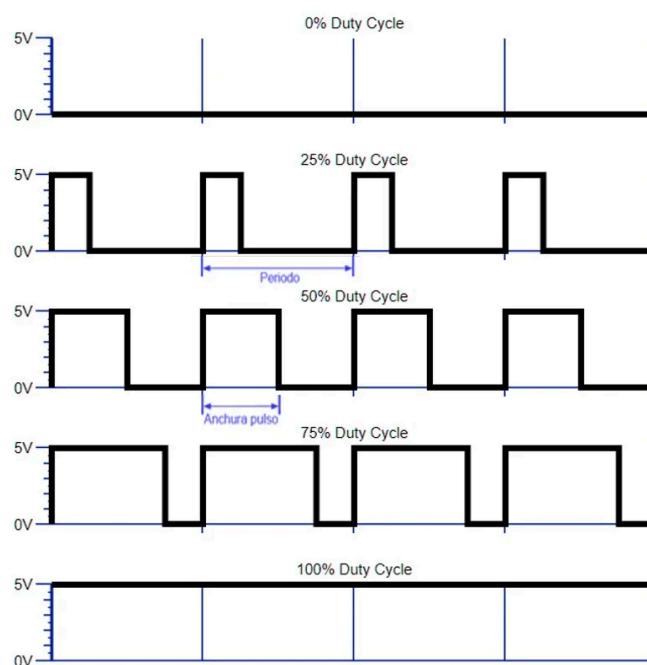
Para más información de actuadores visitar el enlace:

[Actuadores Lineales con Arduino](#)

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

6. ANEXO 2. → PWM - Modulación por Ancho de Pulso

La **Modulación por Ancho de Pulso**, conocida por sus siglas en inglés como **PWM (Pulse Width Modulation)**, es una técnica ampliamente utilizada en electrónica para controlar la entrega de energía a dispositivos eléctricos, especialmente motores, LEDs y actuadores. Consiste en la generación de una señal digital que oscila entre dos niveles (alto y bajo) a una frecuencia determinada, variando el **ciclo de trabajo (duty cycle)** para modular la cantidad de energía media suministrada.

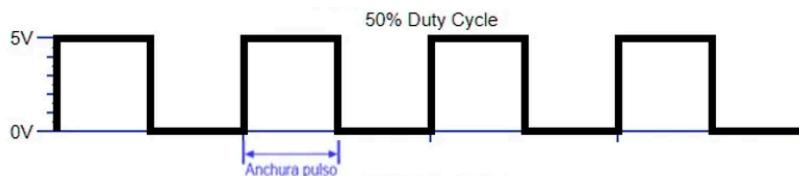


6.1. Fundamento de funcionamiento

Una señal PWM se caracteriza por dos parámetros fundamentales:

- **Frecuencia (f):** Es el número de ciclos por segundo (medido en Hz). Define la rapidez con la que se repite el pulso.
- **Ciclo de trabajo (duty cycle):** Es el porcentaje del tiempo en que la señal permanece en nivel alto durante un ciclo completo. Se expresa generalmente en porcentaje (%).

Por ejemplo, una señal PWM con un ciclo de trabajo del 50 % implica que la señal está en nivel alto la mitad del tiempo y en nivel bajo la otra mitad.



6.2. Aplicación en el control de motores

Cuando se utiliza PWM para controlar un motor de corriente directa, el valor del ciclo de trabajo determina la **velocidad de rotación** del motor:

- Un ciclo de trabajo bajo (por ejemplo, 10 %) entrega menos energía promedio, resultando en una velocidad reducida.
- Un ciclo de trabajo alto (por ejemplo, 90 %) proporciona una mayor energía promedio, permitiendo mayor velocidad.

La ventaja de PWM frente a técnicas analógicas es su **alta eficiencia energética**, ya que el componente de conmutación (por ejemplo, un transistor o un driver como el L293D) disipa muy poca potencia en estado conmutado (on/off).

6.3. Ventajas del uso de PWM

- Control preciso de la energía entregada.
- Alta eficiencia energética.
- Fácil implementación mediante microcontroladores.
- Menor generación de calor en comparación con reguladores lineales.

6.4. Aplicaciones comunes

- Control de velocidad en motores DC.
- Regulación de brillo en pantallas o iluminación LED.
- Conversores de potencia (fuentes conmutadas).
- Generación de señales analógicas simuladas (mediante filtrado).

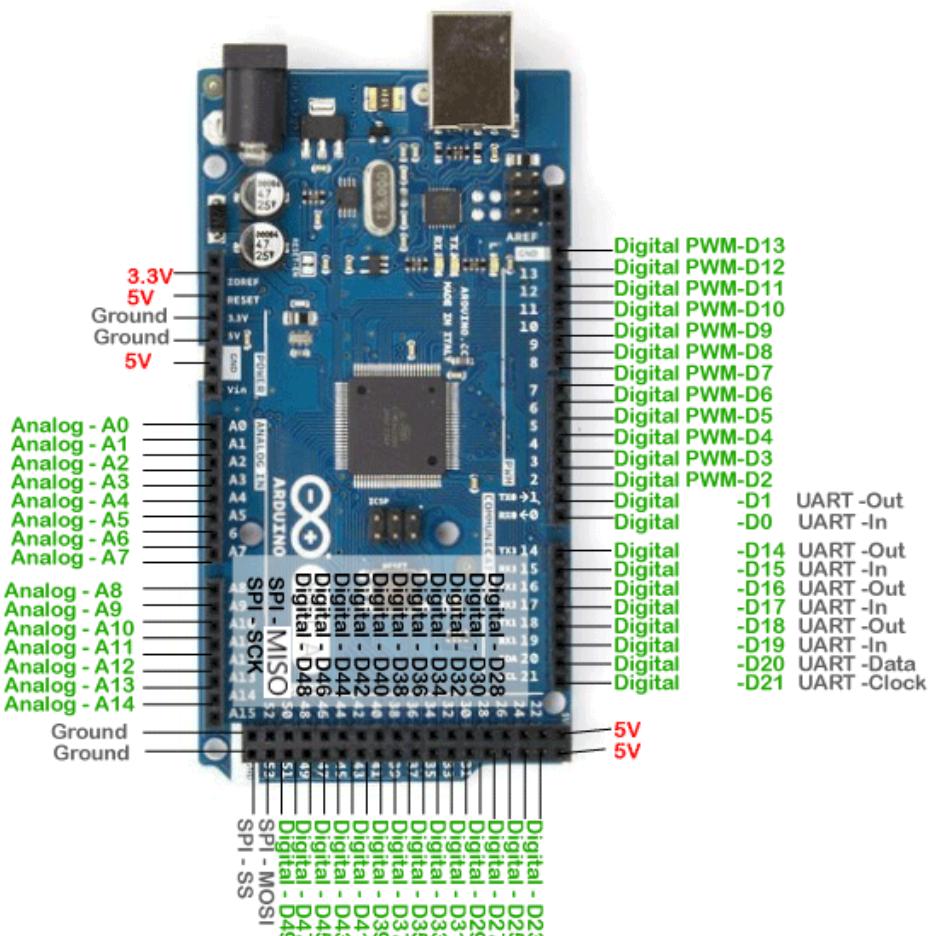
7. ANEXO 3. → PWM con Arduino

La placa **Arduino Mega 2560** permite la generación de señales PWM mediante sus pines digitales, lo cual es útil para controlar dispositivos como motores DC, servomotores, LEDs y otros actuadores que requieren una modulación de potencia.

7.1. Pines PWM disponibles en Arduino Mega

A diferencia del Arduino Uno, el Arduino Mega cuenta con un mayor número de pines capaces de generar señales PWM. Los pines habilitados para PWM están marcados con el símbolo “~” en la serigrafía de la placa:

- **Pines PWM del Mega 2560:** 2 a 13 y 44 a 46
- **Frecuencia por defecto:** ~490 Hz (algunos pines operan a ~980 Hz)



Pinout de pines Digitales, PWM y Analógicos en Arduino MEGA.

7.2. Sintaxis básica para usar PWM

El Arduino utiliza la función analogWrite() para generar una señal PWM:

```
analogWrite(pin, valor);
```

- pin: Número del pin digital con capacidad PWM.
- valor: Valor de ciclo de trabajo entre 0 (0 %) y 255 (100 %).

Por ejemplo, para enviar un PWM del 50 % (equivalente a 2.5 V en un sistema de 5 V):

```
analogWrite(9, 127); // Ciclo de trabajo del 50 % en el pin 9
```

7.3. Ejemplo de control de velocidad de motor DC con PWM.

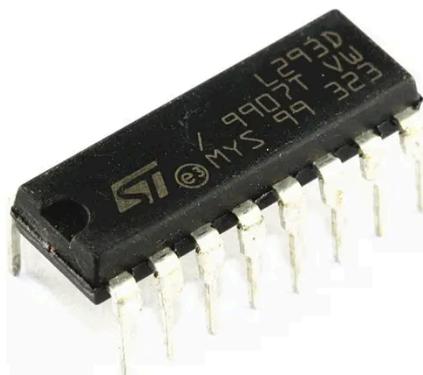
```
const int motorEnable = 10; // Pin PWM
const int motorIn1 = 8;
const int motorIn2 = 9;

void setup() {
    pinMode(motorEnable, OUTPUT);
    pinMode(motorIn1, OUTPUT);
    pinMode(motorIn2, OUTPUT);
}

void loop() {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW); // Gira en una dirección
    analogWrite(motorEnable, 150); // Velocidad ~59 %
    delay(2000);
    analogWrite(motorEnable, 0); // Motor detenido
    delay(1000);
}
```

8. ANEXO 4. → Driver Puente H L293D (Circuito Integrado)

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.



Circuito Integrado L293D.

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(modo más común) ó también podría controlar un motor a pasos de bipolar o unipolar.

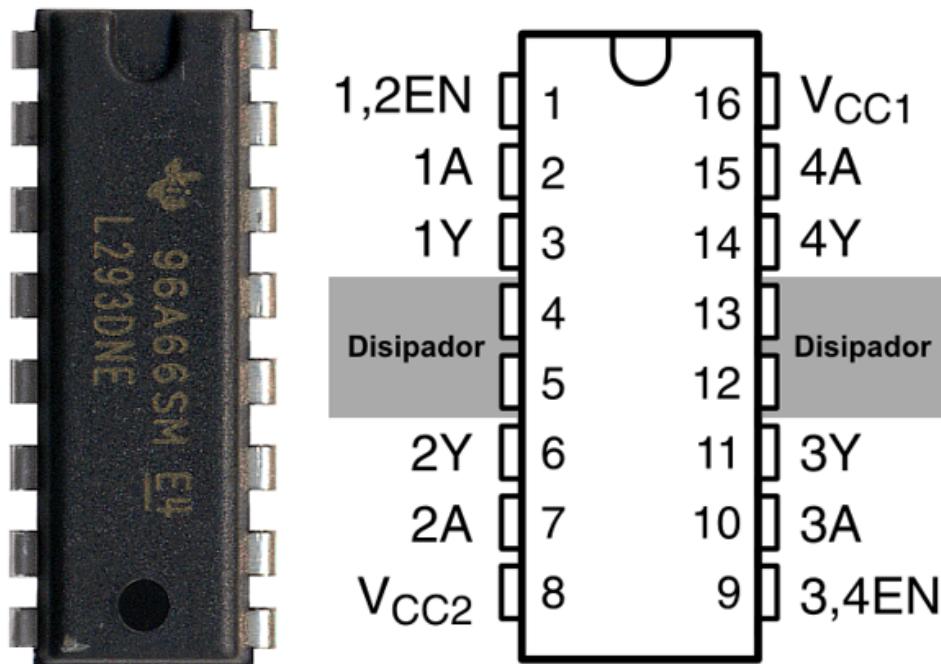
Una de las principales ventajas del controlador L293D es que permite una alimentación independiente para los motores (voltaje de potencia). Por ejemplo, se pueden controlar motores desde los 4.5 VDC hasta 36 VDC. Es importante mencionar que cuando se manejan potencias mayores a 5W ($P = V \cdot I$) es 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.

8.1. 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://naylampmechatronics.com/drivers/223-driver-puente-h-l293d-1a-dip-16.html>

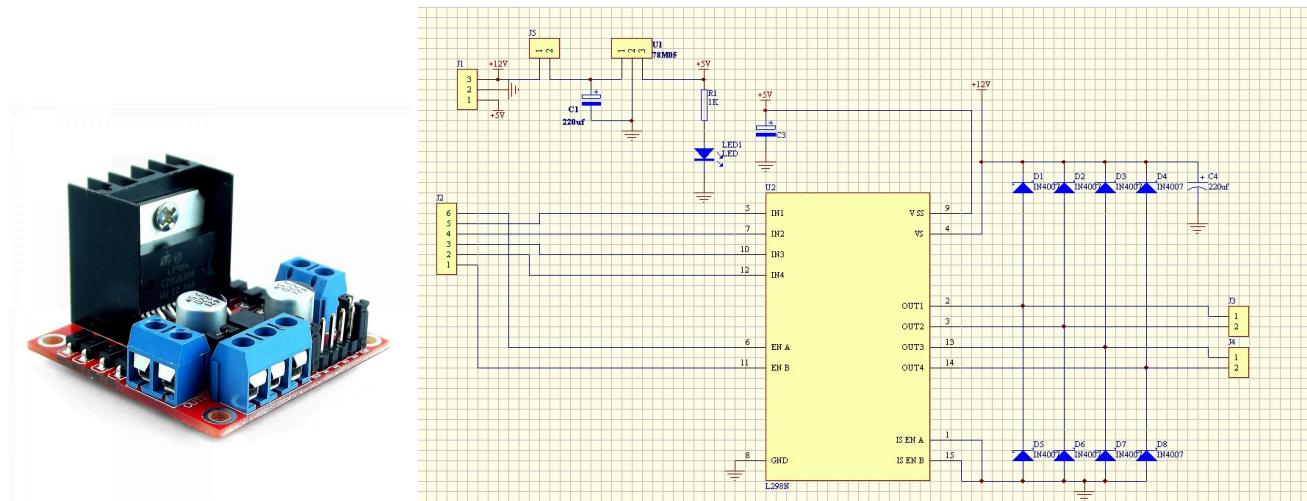
9. ANEXO 5. → Driver Puente H L298N 2A

El driver puente H L298N es el modulo más utilizado para manejar motores DC de hasta 2 amperios.

El chip L298N internamente posee dos puentes H completos que permiten controlar 2 motores DC o un motor paso a paso bipolar/unipolar.

El módulo permite controlar el sentido y velocidad de giro de motores mediante señales TTL que se pueden obtener de microcontroladores y tarjetas de desarrollo como Arduino, Raspberry Pi o Launchpads de Texas Instruments. El control del sentido de giro se realiza mediante dos pines para cada motor, la velocidad de giro se puede regular haciendo uso de modulación por ancho de pulso (PWM por sus siglas en inglés).

Tiene integrado un regulador de voltaje LM7805 de 5V encargado de alimentar la parte lógica del L298N, el uso de este regulador se hace a través de un Jumper y se puede usar para alimentar la etapa de control.



Fotografía del Driver Puente H junto a su esquema electrónico.

9.1. Importante

- Voltaje de alimentación, mínimo de 5V. Posee dos entradas, una de 5V para controlar la parte lógica y otra para alimentar las salidas al motor, que pueden ser de 5V o más.
- Posee un regulador de voltaje de 5V 78M05, para alimentar la etapa lógica del L298N, sin embargo, cuando la alimentación supera los 12V, se recomienda, utilizar una fuente de 5V externa como fuente de alimentación.
- Admite entradas de señal PWM para el control de velocidad.
- Posee 8 diodos de protección contra corriente inversas.

9.2. Cómo alimentarlo

Este módulo se puede alimentar de 2 maneras gracias al regulador integrado LM7805. Cuando el jumper de selección de 5V se encuentra activo, el módulo permite una alimentación de entre 6V a 12V DC. Como el regulador se encuentra activo, la bornera marcada como +5V tendrá un voltaje de salida de 5V DC. Este voltaje se puede usar para alimentar la parte de control del módulo ya sea un microcontrolador o un Arduino, pero recomendamos que el consumo no sea mayor a 500mA.

Cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC. Como el regulador de 5 voltios no está funcionando, tendremos que conectar la bornera de +5V a una fuente de 5V para alimentar la parte lógica del L298N, podemos usar como fuente la salida de 5 voltios de nuestro Arduino.

9.3. Cómo usarlo con Arduino

El módulo L298N posee dos canales de Puente H, pudiéndolos utilizar para controlar dos motores DC o un motor Pasó a Paso, controlando el sentido de giro y velocidad.

Básicamente está conformado por un driver L298N sus diodos de protección y un regulador de voltaje de 5V(78M05)

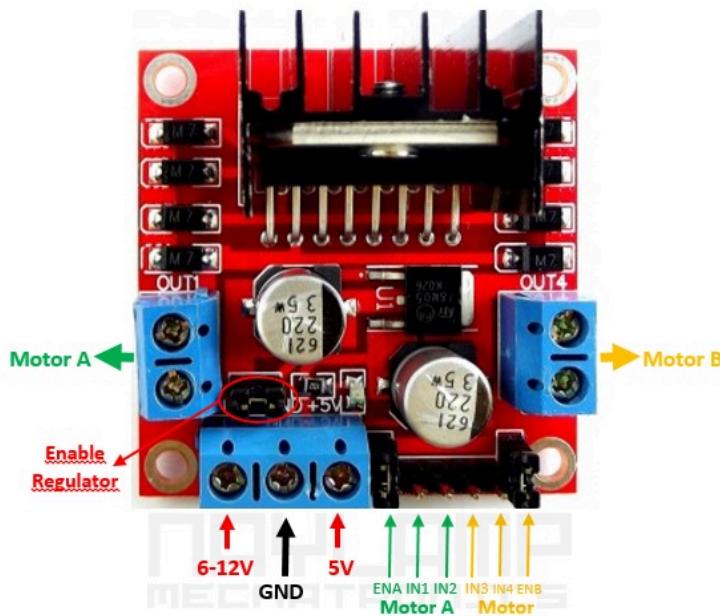


Diagrama de conexión con motores.

Empecemos explicando la forma de alimentar el módulo, hay dos formas de hacer esto:

1. Utilizando una sola fuente, conectada a la entrada de 12V y con el Jumper para habilitar el regulador, aclarando que el voltaje de la fuente es el que soporta el motor. De esta forma la entrada de 5V no debe estar conectada a ninguna fuente, ya que en este pin están presentes 5V a través del regulador interno; pero puedes utilizar este pin como una salida de 5V, pero sin exceder los 500mA de consumo. Se recomienda hacer esta conexión para voltajes menores de 12V para no sobrecalentar el regulador
2. Utilizando dos fuentes, una de 5V conectada a la entrada de 5V (puede ser los 5V de un Arduino) y otra fuente con el valor del voltaje que trabaja el motor, conectada al pin de 12V. Para esto se tiene que desconectar el Jumper lo que deshabilitará al regulador.

Para el control del módulo:

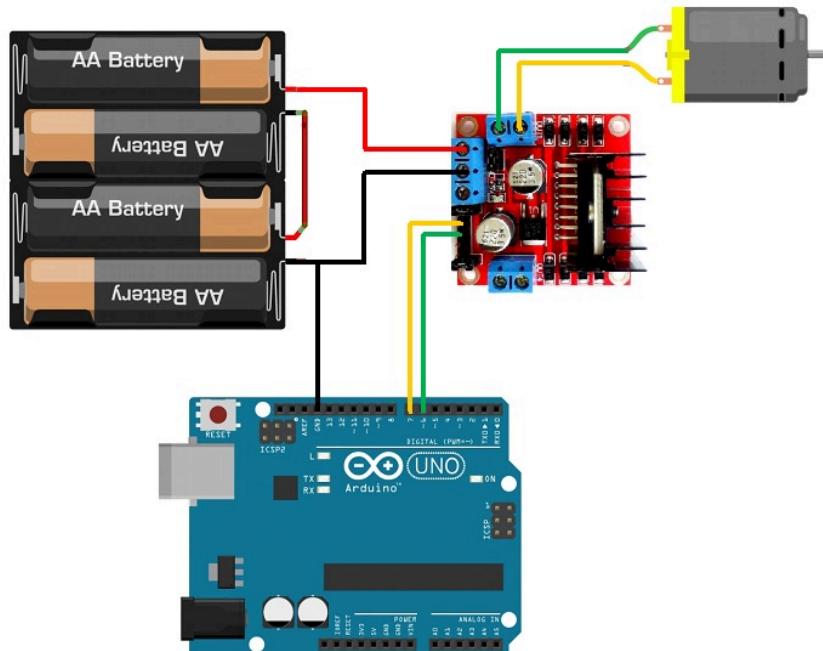
Los pines ENA, IN1, IN2 corresponden a las entradas para controlar el MOTOR A (OUT1 y OUT2)

De igual manera ENB, IN3, IN4 permiten controlar el MOTOR B (OUT3 y OUT4)

ENA y ENB, sirven para habilitar o deshabilitar sus respectivos motores, generalmente se utilizan para controlar la velocidad, ingresando una señal de PWM por estos pines. Si no se usan se deben de conectar los Jumper para que siempre estén habilitados.

Ahora realicemos un ejemplo básico para controlar un motor DC

En el ejemplo utilizaremos un motor de 6V y lo conectaremos de la primera forma antes explicado.



Ejemplo de conexión. **Nota.** En lugar del banco de baterías, usar la salida de la fuente para protoboard.

Como vemos en la imagen para controlar el motor solo utilizaremos dos pines del Arduino.

Si enviamos un 1 lógico por la entrada IN1 del driver, saldrán 6V por la salida OUT1 (cable de color

Amarillo en la imagen) y si enviamos un 0 lógico por IN1, saldrá GND (0V) por OUT1

De igual manera sucede con el pin IN2.

Por ejemplo si queremos hacer girar el motor en una dirección, enviamos:

IN1=1 → OUT1= 6V

IN2=0 → OUT2= GND

Y para invertir el sentido de giro:

IN1=0 → OUT1=GND

IN2=1 → OUT2=6V

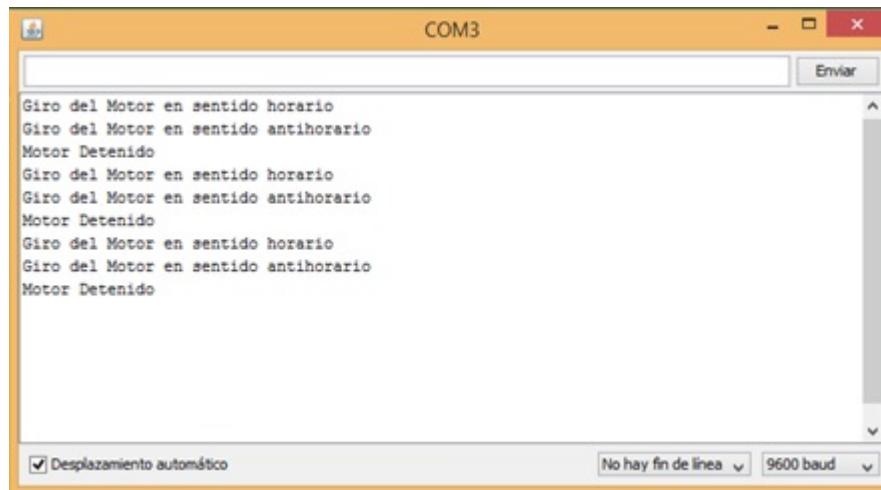
EL código en Arduino sería el siguiente:

```
const int PinIN1 = 7;  
const int PinIN2 = 6;  
  
void setup() {  
    // inicializar la comunicación serial a 9600 bits por segundo:  
    Serial.begin(9600);  
    // configuramos los pines como salida  
    pinMode(PinIN1, OUTPUT);  
    pinMode(PinIN2, OUTPUT);  
}  
  
void loop() {  
  
    MotorHorario();  
    Serial.println("Giro del Motor en sentido horario");  
    delay(5000);  
  
    MotorAntihorario();  
    Serial.println("Giro del Motor en sentido antihorario");  
    delay(5000);
```

```
MotorStop();  
Serial.println("Motor Detenido");  
delay(3000);  
  
}  
  
//función para girar el motor en sentido horario  
void MotorHorario()  
{  
    digitalWrite (PinIN1, HIGH);  
    digitalWrite (PinIN2, LOW);  
}  
//función para girar el motor en sentido antihorario  
void MotorAntihorario()  
{  
    digitalWrite (PinIN1, LOW);  
    digitalWrite (PinIN2, HIGH);  
}  
  
//función para apagar el motor  
void MotorStop()  
{  
    digitalWrite (PinIN1, LOW);  
    digitalWrite (PinIN2, LOW);  
}
```

Como se observa en el programa se han implementado funciones para cada movimiento del motor, las cuales llamamos desde el void loop(),

Inicialmente hacemos girar el motor en sentido horario, luego en antihorario y después lo detenemos, esto se repite constantemente, cada estado lo enviamos por comunicación serial a la PC.



Si el sentido de giro no corresponde al que se muestra en el monitor serial, simplemente invertir la polaridad de la conexión del motor, o también cambiar las funciones en el programa.

9.4. Tabla de direcciones

| IN1 | IN2 | Salida | Comportamiento del motor |
|------|------|--------------|---------------------------|
| LOW | HIGH | OUT1 < OUT2 | Gira en una dirección |
| HIGH | LOW | OUT1 > OUT2 | Gira en la otra dirección |
| LOW | LOW | 0 V en ambos | Motor apagado (libre) |
| HIGH | HIGH | 5 V en ambos | Motor frenado activamente |

10. ANEXO 6. → Fuente para protoboard 3.3V/5V (USB-A) - MB102

Fuente para protoboard con salidas de voltaje seleccionable de 3.3V y/o 5V, posee un chip regulador para cada voltaje. Puede ser alimentado por el conector Jack-DC o por el conector USB-A (el mismo USB de las PCs). Si alimentamos la placa por el Jack-DC, el conector USB se puede utilizar como salida de 5V. Ideal para alimentar circuitos en protoboard utilizando una fuente o cargador DC o un Powerbank/Cargador portátil de celular.

10.1. Especificaciones:

- Voltaje de entrada 1: 6.5V-12V DC (Jack-DC)
- Voltaje de entrada 2: 5V DC (USB-A hembra)
- Voltaje de salida: 3.3V o 5V (seleccionable por jumpers)
- **Corriente de salida: 700mA máx (esto nos permitirá alimentar motores).**
- Dos canales independientes de salida, uno para cada fila de alimentación del Protoboard
- Compatible con Protoboard de 400 puntos y 830 puntos
- Posee switch on/off
- Led indicador on/off
- Modelo: MB102/MB-102
- Dimensiones: 52*32*24 mm
- Peso: 5 gramos

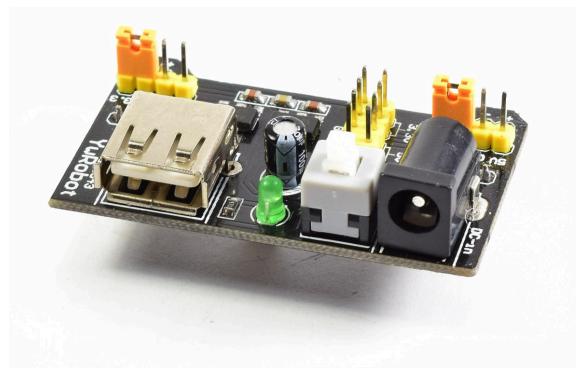
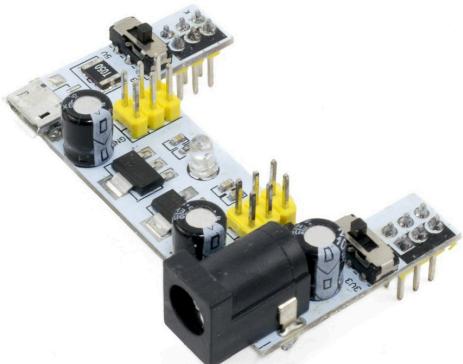


Figure 3. Imagen referencial. Versión de alimentación Micro-USB y USB, respectivamente.

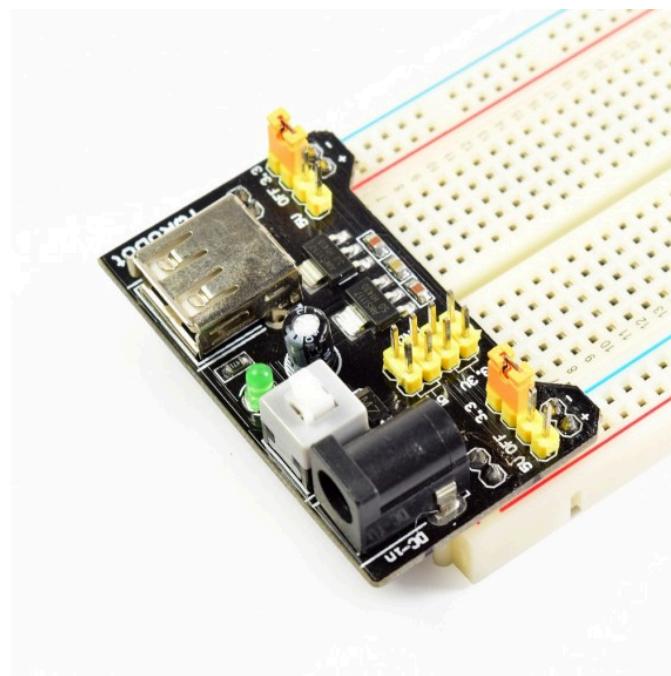


Figure 4.Conexión con Protoboard.



Figure 5. Pinout

11. ANEXO 7. → Motor DC TT 6V/200RPM

El Motor DC TT con caja reductora 6V/200RPM posee una caja reductora integrada que le permite entregar un buen torque en un pequeño tamaño y bajo voltaje. La carcasa del motor es de plástico resistente, no toxico y de color amarillo. Es ideal para proyectos de robótica móvil como robots seguidores de línea, robots zumo, robots velocistas.



Imagen referencial del motor DC TT 6V

11.1. Especificaciones Técnicas

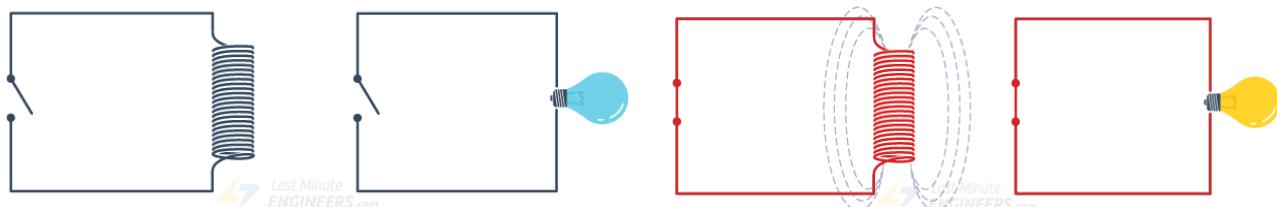
- Voltaje de Operación: 3V – 6V DC
- Velocidad Angular nominal: 200 RPM
- Reducción: 48:1
- Consumo de corriente sin carga: 100mA (a 5V)
- Consumo de corriente nominal: 140mA (a 5V)
- Consumo de corriente eje detenido: 500mA (a 5V)
- Material carcasa: plástico ABS amarillo
- Material caja reductora (engranes): plástico ABS
- Material eje: plástico ABS
- Diámetro eje para rueda: 55 mm
- Peso: 30g

| Voltaje de Operación | Parámetros | DC 3V | DC 5V | DC 6V |
|----------------------------------|---|--------------------|-----------|-----------|
| Parámetros Caja Reductora | Reducción | 48:1 | | |
| | Velocidad sin carga | 125 RPM | 200 RPM | 230 RPM |
| | Velocidad con carga | 95 RPM | 152 RPM | 175 RPM |
| | Torque de salida | 0.8kg.cm | 1.0kg.cm | 1.1kg.cm |
| | Velocidad del robot sin carga (metros/minuto) | 25.9 | 41.4 | 47.7 |
| | Corriente | 110-130mA | 120-140mA | 130-150mA |
| | Diámetro máximo de llanta | 6.5cm | | |
| | Dimensiones | 70mm x 22mm x 18mm | | |
| | Peso | 50g | | |
| | Ruido | <65dB | | |

Tabla de parámetros de operación.

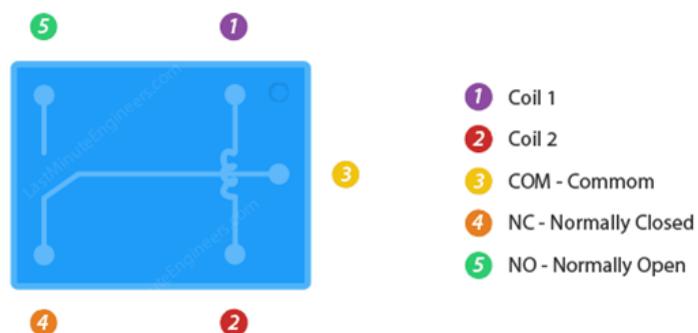
12. ANEXO 8. → Relé

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.



12.1. 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.

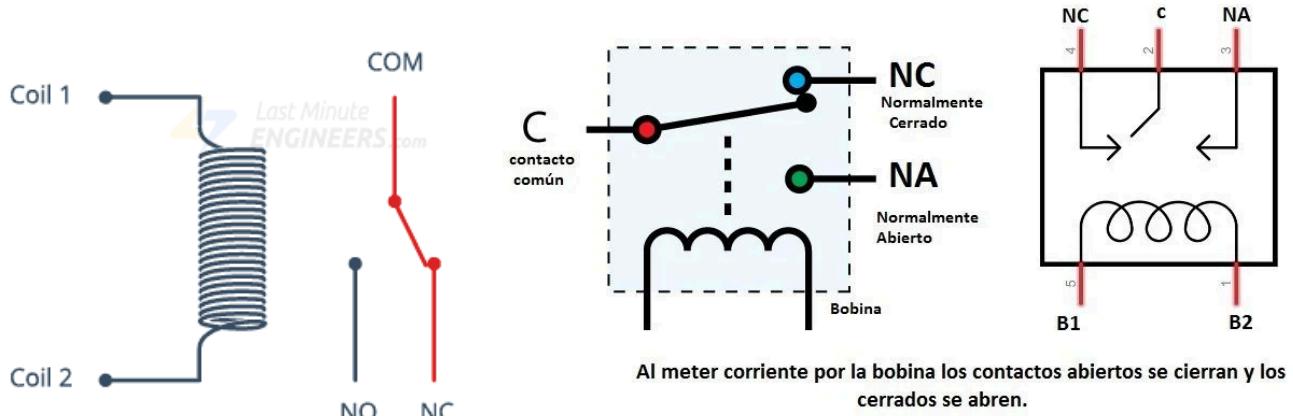


Figure 6. Entre los dos pines restantes (bobina 1 y bobina 2) hay una bobina que actúa como un electroimán.

12.2. Conexión de Relé

En la figura se muestra el esquema de conexión de un relé.

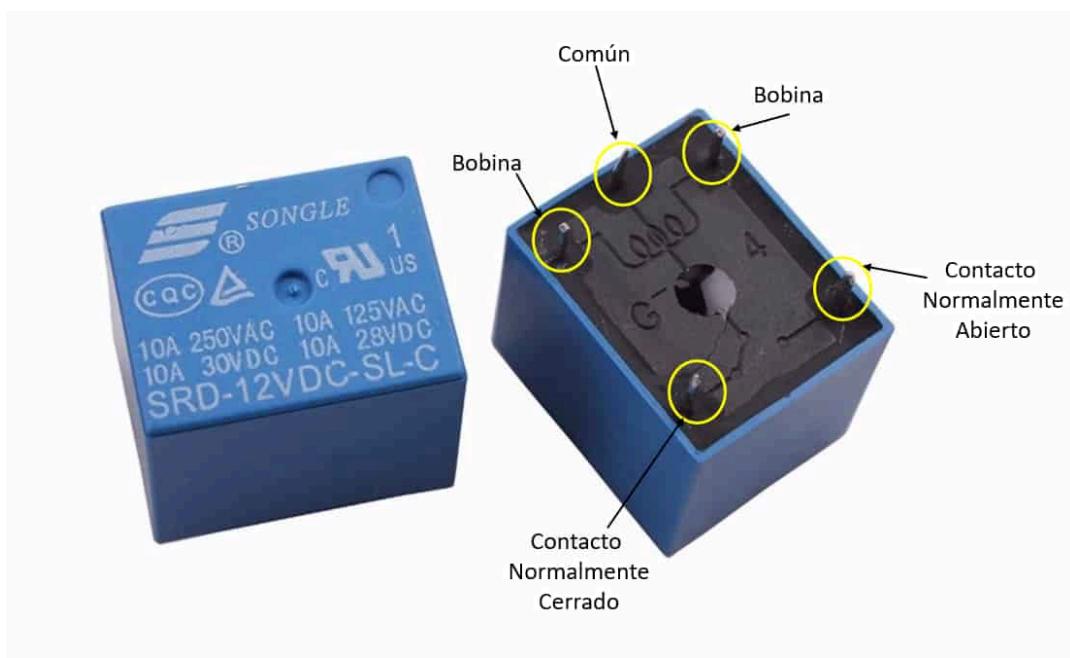


Figure 7. Pines de un relé.

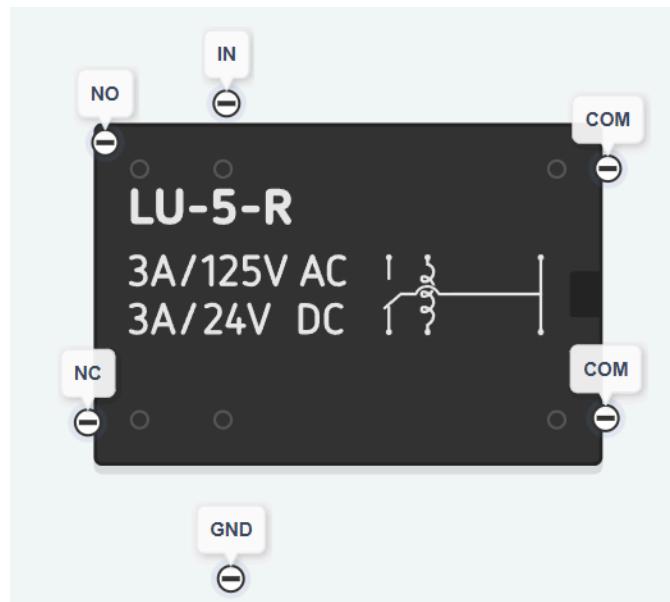


Figure 8. Pines en Tinkercad. Los pines del inductor (IN y GND) pueden ser conectados en ese sentido o inverso, pero debe existir una diferencia de potencial entre estos pines para activarlo.

12.2.1. Ejemplo en Tinkercad

Se muestra el siguiente esquema de conexión en Tinkercad.

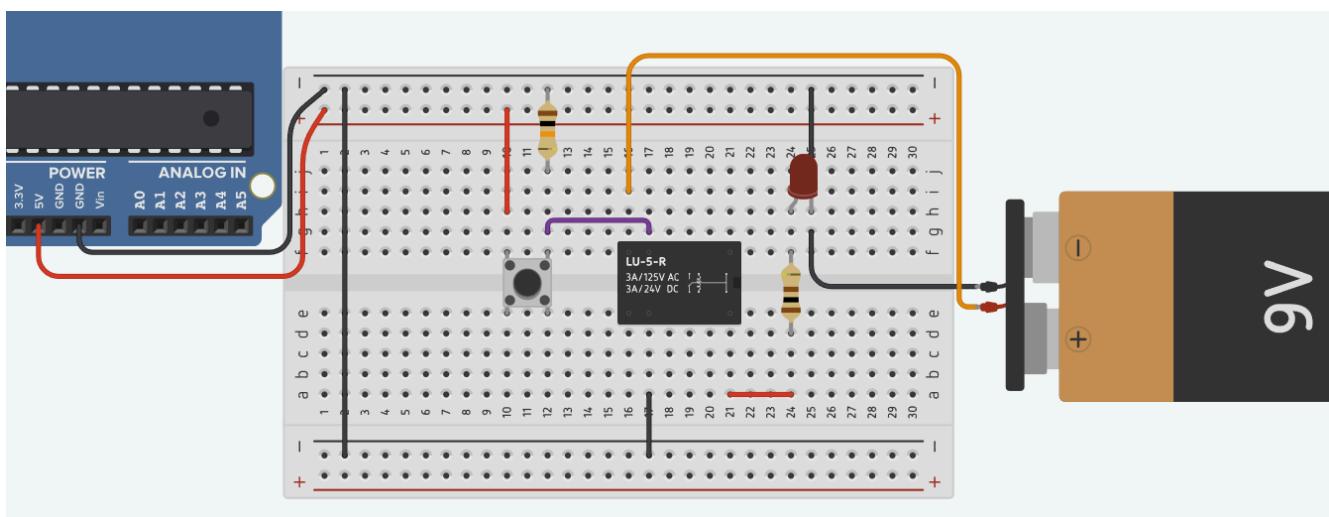
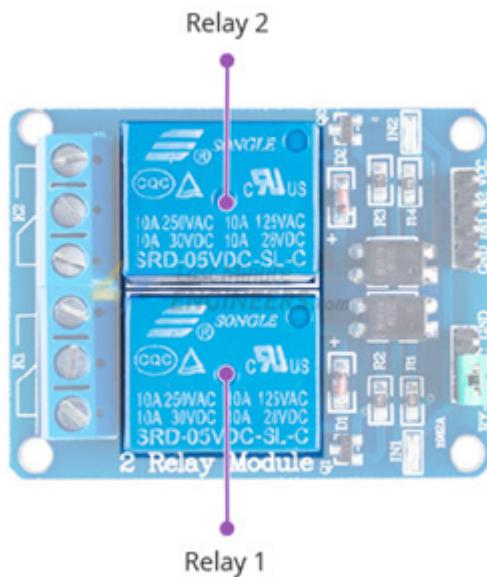


Figure 9. Ejemplo de conexión en Tinkercad ([Enlace](#))

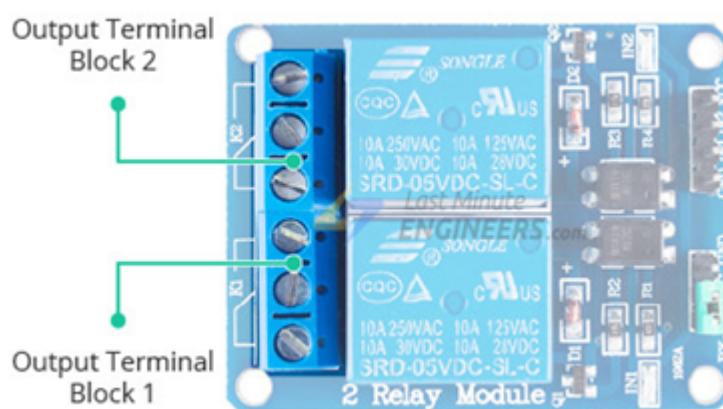
12.3. Módulo Relé para Arduino

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.



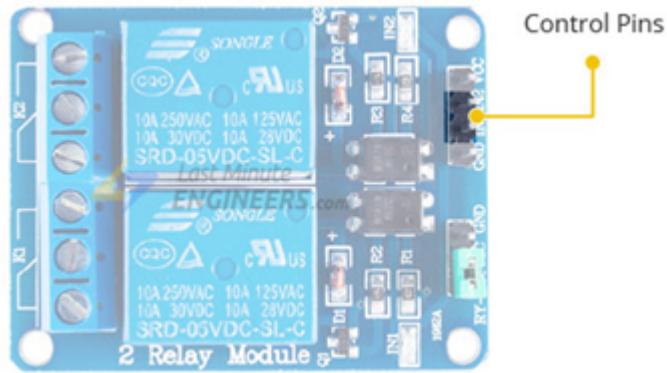
12.3.1. Bloques de terminales de salida

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.



12.3.2. 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.

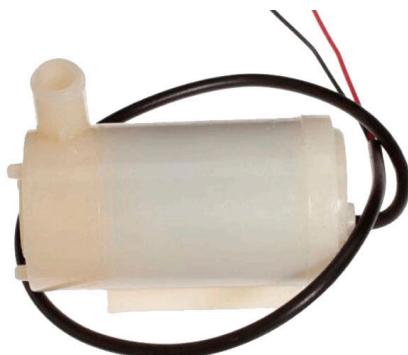
El módulo de relé tiene dos LED que indican el estado del relé. Cuando se activa un relé, se enciende el LED correspondiente.

Más información en

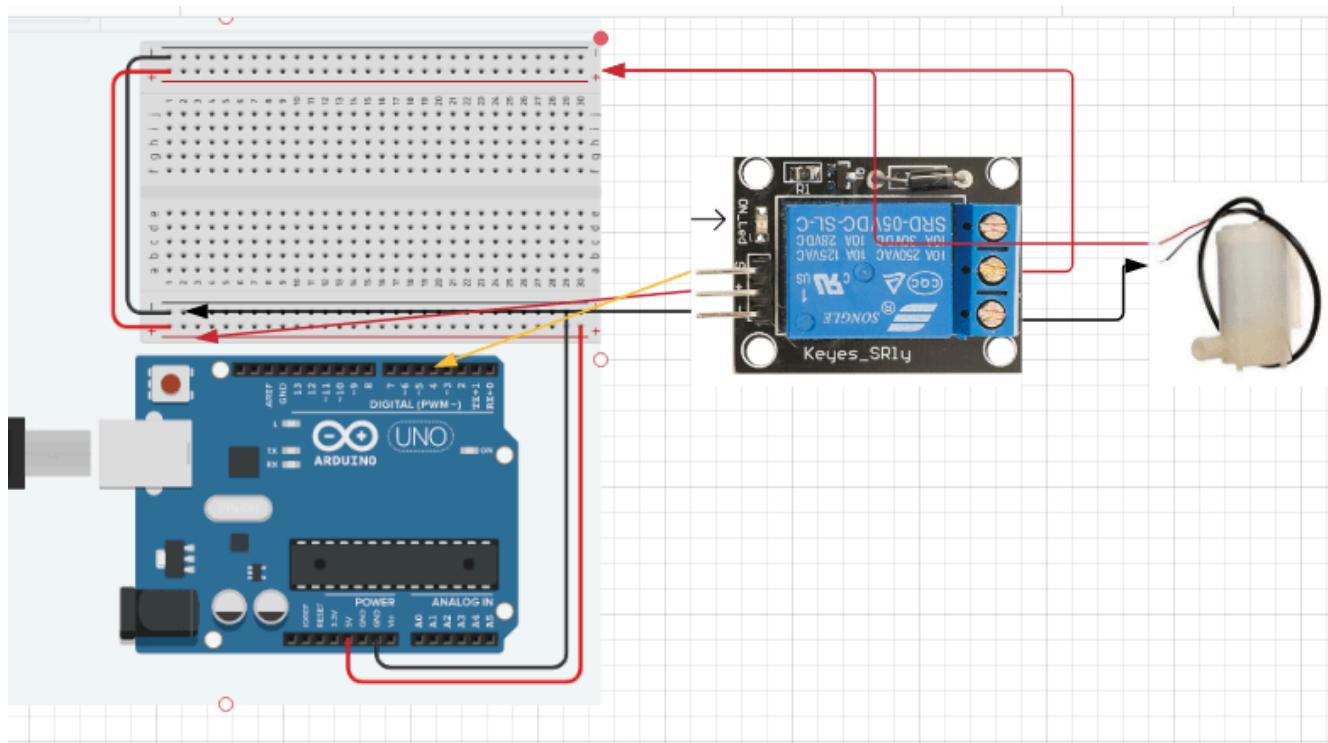
[Módulo de relé de dos canales con Arduino](#)

13. ANEXO 9. → 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 a 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**.



Bomba de agua.



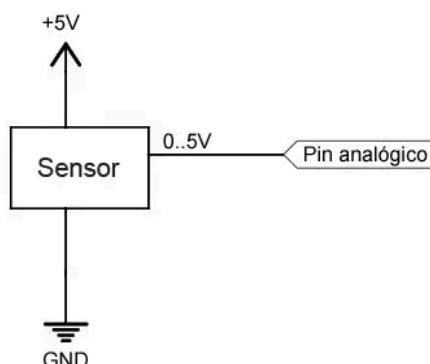
Esquema de conexión con Arduino y Relé.

14. ANEXO 10. → 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 -Vcc y + Vcc. 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).

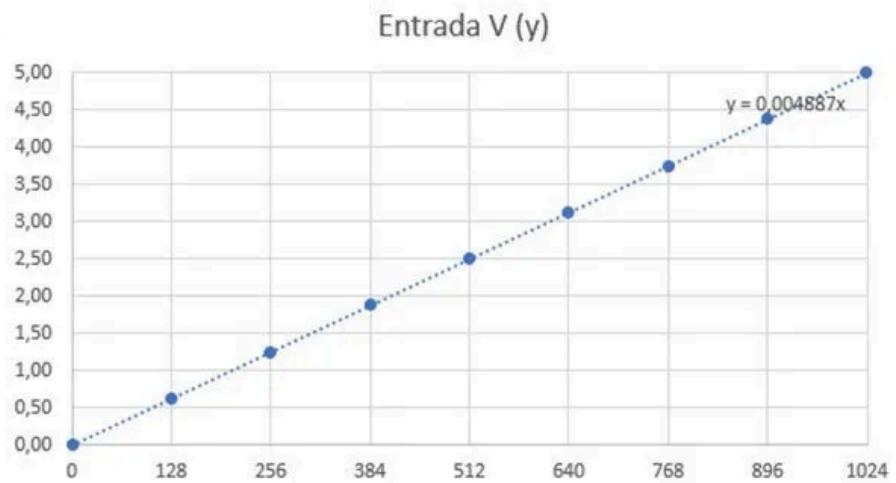


Se utiliza el comando `analogRead()`

```
sensorValue = analogRead(sensorPin);
```

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

| $y = 0,004887x$ | | |
|-----------------|---------|---------------|
| Nº | 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.

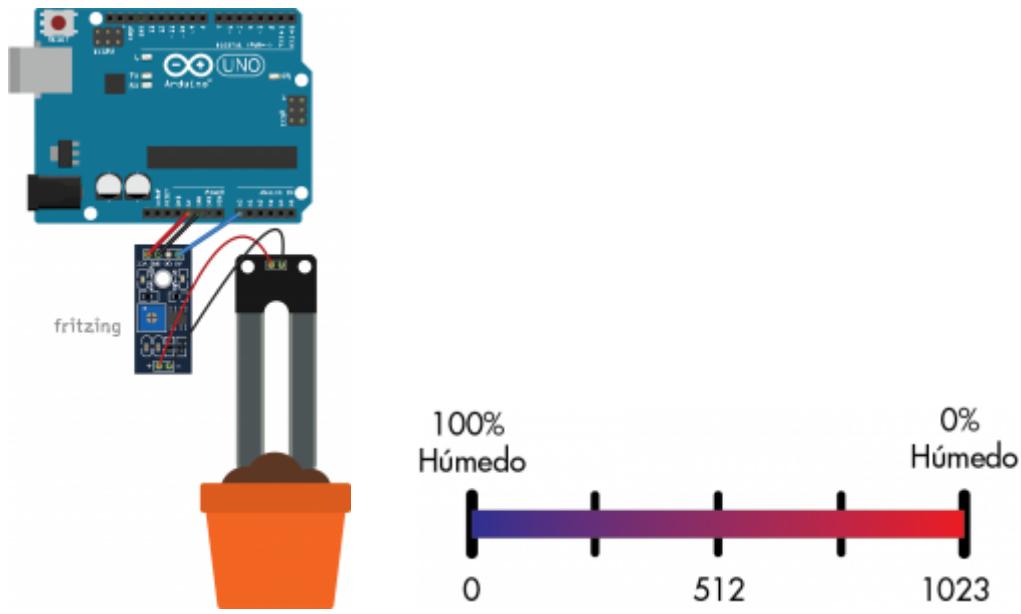
[Entradas Analógicas en Arduino](#)

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

15. ANEXO 11. → 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);
}
```

16. ANEXO 12. → Sensor de Temperatura

Referencia:

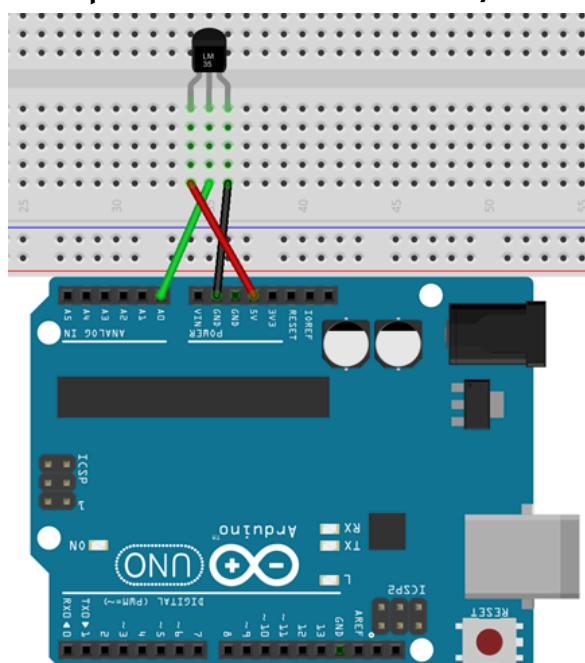
[Sensor de Temperatura con Arduino](#)

[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);

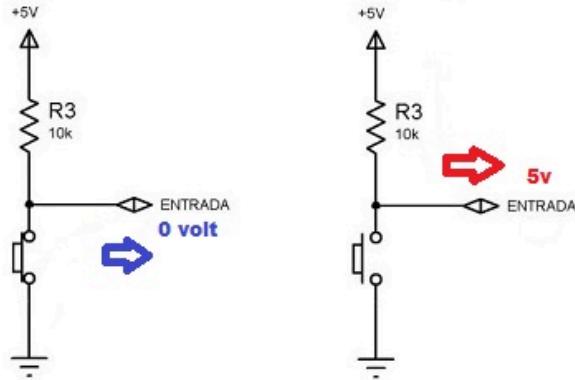
// Calculamos la temperatura con la fórmula
tempC = (5.0 * tempC * 100.0)/1024.0;

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

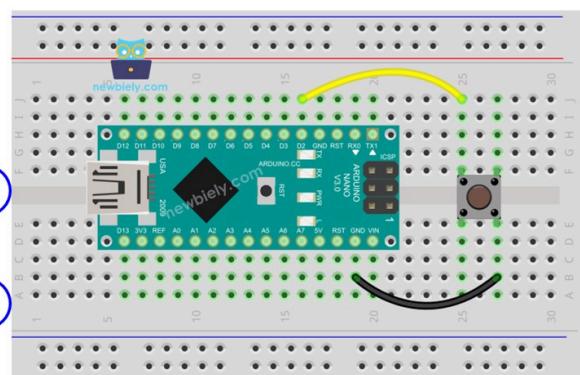
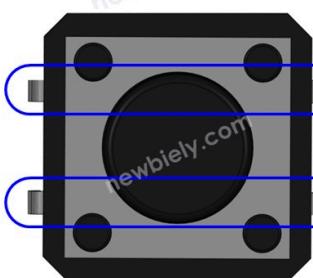
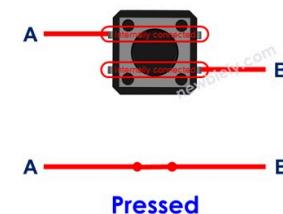
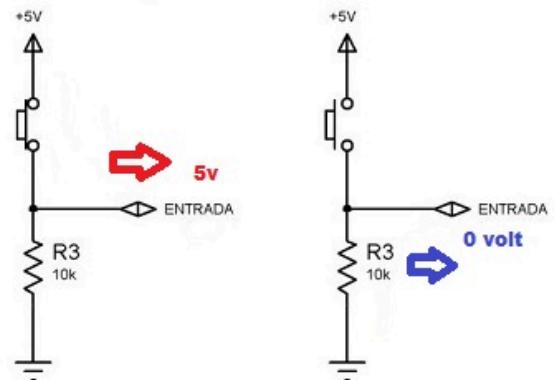
// Esperamos un tiempo para repetir el loop
delay(1000);
}
```

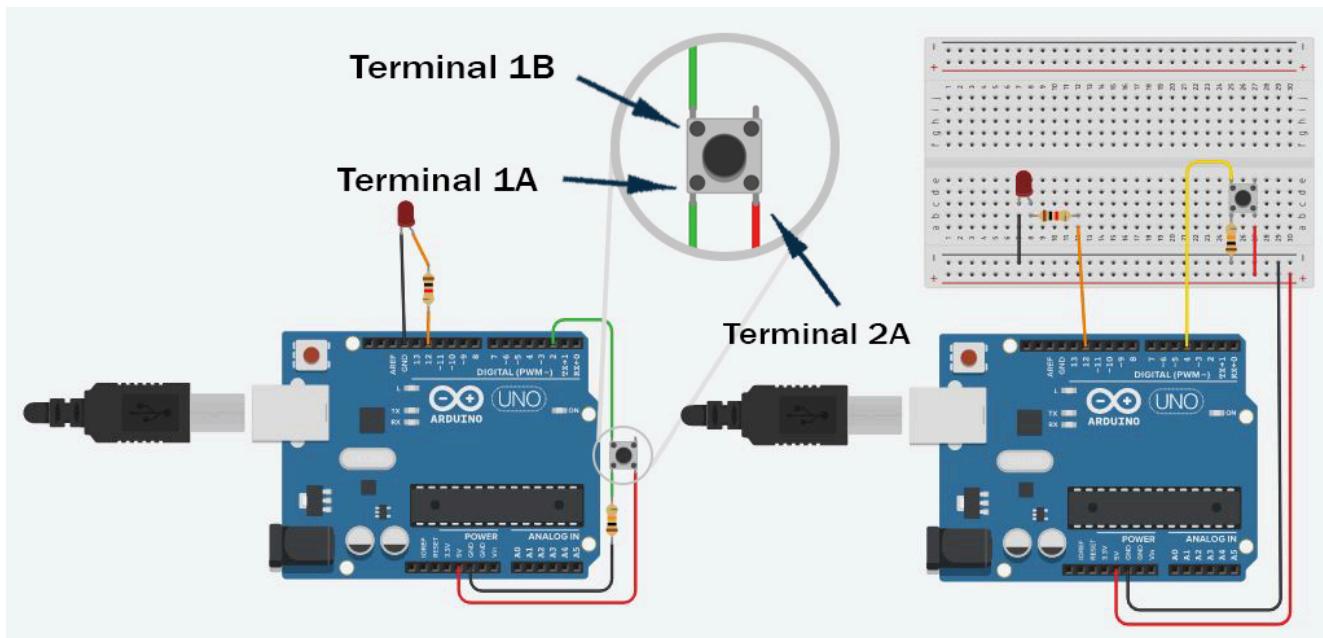
17. ANEXO 13. → El uso de botones con Arduino.

Configuración Pull Up



Configuración Pull Down





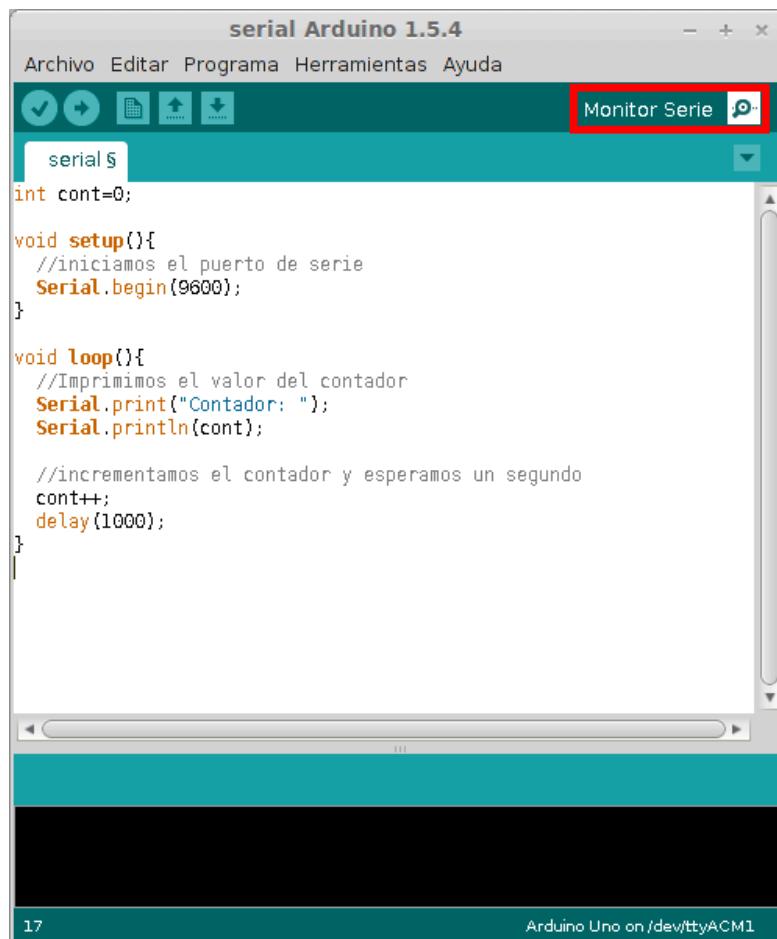
18. ANEXO 14. → Uso del monitor Serial.

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

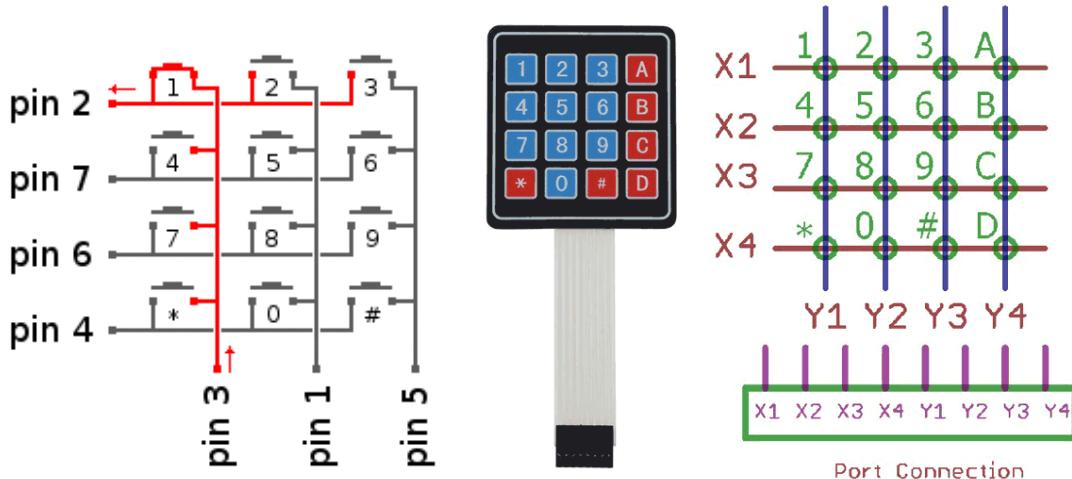
Encender o apagar el LED integrado en la placa Arduino. Para ello enviamos un carácter a la placa Arduino, empleando el monitor serial. En caso de enviar 'a' la placa Arduino apaga el LED, y en caso de enviar 'b' lo enciende.

```
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 }
```

19. ANEXO 15. → Uso del teclado matricial

Usar referencia:

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



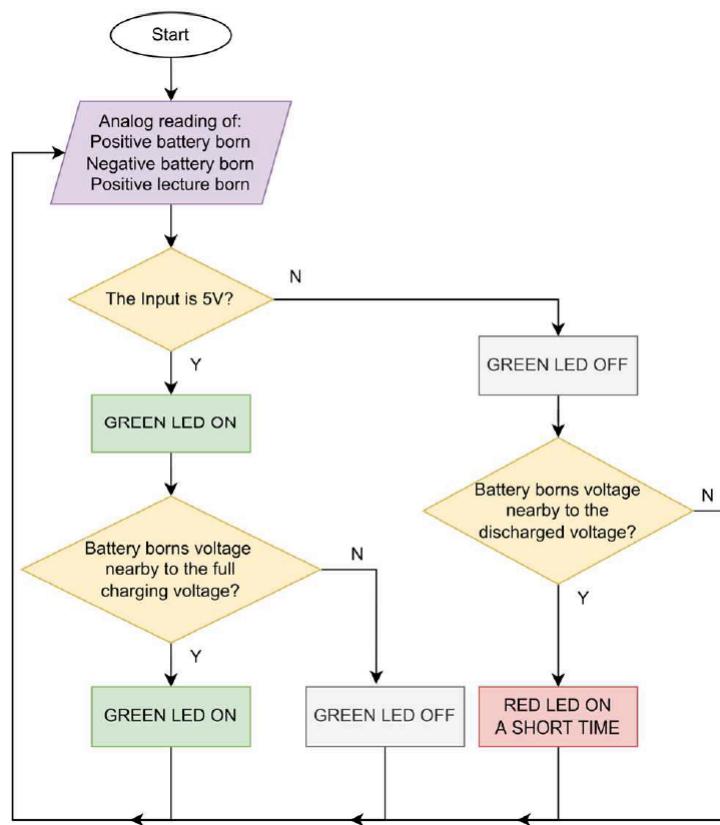
Una forma rápida de usar un Teclado con Arduino, es valernos de su librería Keypad (<https://www.arduino.cc/reference/en/libraries/keypad/>), bastante sencilla de entender, para el caso de un teclado matricial 4x4 (KEYPAD 4x4) tenemos:

```
#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);
}
```

20. ANEXO 16. → 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.



20.1. 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 |

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

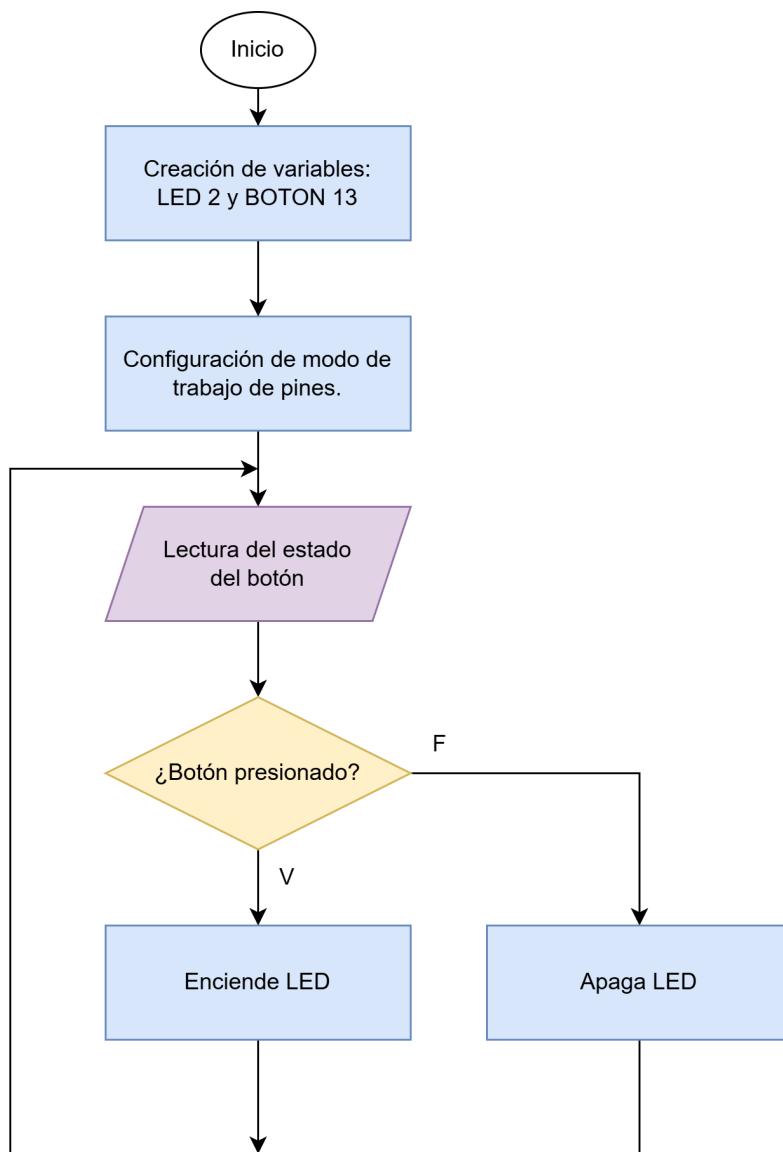
Ejemplo sencillo de código para Arduino que enciende un LED cuando se presiona un botón. Este ejemplo asume que tienes un botón conectado al pin digital 2 y un LED al pin digital 13.

```
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
    }
}
```



21. ANEXO 17. → 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.

21.1. 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.

21.2. 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).

21.3. Ejemplo

