

## LAB 2: Sensores y Actuadores

Realizar un informe conteniendo los siguientes ítems:

- Introducción.
  - Marco Teórico (incluir descripción de todos los pines).
  - Estado del Arte
- Metodología (diagrama de flujo de cada código).
- Desarrollo (de cada una de las experiencias).
- Conclusiones
- Referencias

*Subir únicamente el PDF del informe a Canvas, usar un link para el repositorio, incluir un vídeo mostrando el experimento. Considere una placa de Arduino MEGA2560 tal cual existe en el laboratorio. Sin embargo, para simulaciones, considere la placa Arduino UNO disponible en Tinkercad.*

## ÍNDICE

<b>1. Sensor de luz ambiental</b>	<b>3</b>
<b>2. Cerradura Inteligente</b>	<b>4</b>
2.1. Presencia de persona	4
2.2. Mensaje en LCD.	4
2.3. Autenticación.	5
2.4. Ahorro energético (Standby).	5
<b>3. Sistema de control de aforo.</b>	<b>6</b>
3.1. Elementos	6
3.2. Funcionamiento.	6
3.3. Presentar	7
<b>4. ANEXO I: Fococelda o Resistor Dependiente de Luz (LDR)</b>	<b>8</b>
4.1. Características eléctricas típicas	9
4.2. Principio de funcionamiento	9
4.3. Uso de un LDR con Arduino	9
<b>5. ANEXO II: Sensor de Proximidad</b>	<b>11</b>
<b>6. ANEXO III. LCD 2x16</b>	<b>13</b>
<b>7. ANEXO IV. Lectura Análoga con Arduino</b>	<b>14</b>
<b>8. ANEXO V. Sensor de Movimiento PIR</b>	<b>16</b>
<b>9. ANEXO VI. Sensor de Humedad</b>	<b>17</b>
<b>10. ANEXO VII. Sensor de Temperatura</b>	<b>19</b>
<b>11. ANEXO V. El uso de botones con Arduino.</b>	<b>20</b>
<b>12. ANEXO VI. Uso del monitor Serial.</b>	<b>21</b>

<b>13. ANEXO VII. Uso del teclado matricial</b>	<b>23</b>
<b>14. ANEXO VIII. Sensor de Movimiento PIR</b>	<b>24</b>
<b>15. ANEXO IX. Diagramas de flujo</b>	<b>26</b>
15.1. Simbología básica	26
15.2. Ejemplo de Diagrama de Flujo para un código simple.	27
<b>16. ANEXO X. Diagrama de Bloques</b>	<b>29</b>
16.1. Objetivo de su Uso en el Diseño Electrónico	30
16.2. Componentes Típicos	30
16.3. Ejemplo	30

# PARTE I: Sensores - I

## 1. Sensor de luz ambiental

Deberá utilizar para esta experiencia:

- 1 LDR
- Diodos LED
- Buzzer (alarma)
- Botón o switch.

Debe realizar un sistema de medición ambiental que le permita poder medir la cantidad de luz en un determinado ambiente que acompañará a un sistema de cámaras de seguridad, junto con un control inteligente de focos . El sistema cumple con las siguientes funciones:

1. Si el nivel de **iluminación es bajo**, debe prender las luminarias de la habitación. Esto se representará mediante el encendido de un **LED ROJO**.
2. Si el nivel de **iluminación es alto**, no hay necesidad de prender las luminarias, por lo tanto el **LED ROJO** debe estar **APAGADO**.
3. Las cámaras de seguridad deben estar en funcionamiento en todo momento. Esto es debido a la entrada de energía (considere una entrada digital) que tiene el sistema de cámaras. Mientras esté en funcionamiento, se representa mediante el encendido de un **LED AZUL**.
4. En caso que no se detecte energía eléctrica para el sistema de cámaras (**entrada digital detecta GND**), su sistema debe de empezar a encender y apagar el **LED Azul** de manera **intermitente**, y así mismo encender un **BUZZER** que representa una alarma de error. (Puede utilizar un botón, un switch o una desconexión de cable para este fin).

Presente:

- a. Simulación en Tinkercad.

**CHECKPOINT 1**

- b. Implementación presencial en laboratorio.

**CHECKPOINT 2**

## 2. Cerradura Inteligente

Deberá utilizar para esta experiencia:

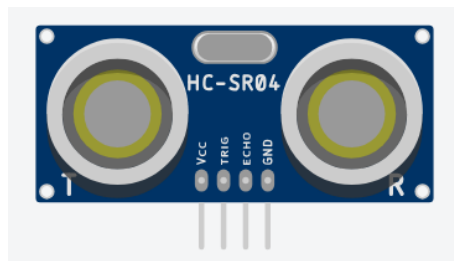
- 01 sensor de proximidad.
- 01 teclado matricial.
- 01 LCD 2x16.
- Diodos LED.
- Resistencias variadas.
- Jumpers.

Realizar un sistema de seguridad para el control de acceso de personas a un área restringida. Este sistema funciona de la siguiente manera:

### 2.1. Presencia de persona

Cuando una persona se acerca a la entrada del área restringida (a menos de 20 cm del sistema colocado en la entrada), se debe prender un LED rojo que se prende y se apaga, lo cual representa la activación de un sistema de grabación.

**Nota.** En tinkercad utilizar el siguiente modelo de sensor ultrasónico.



Para más información de cómo usarlo, referirse a la guía anexada al final del documento. **Realice la implementación en simulación.**

### 2.2. Mensaje en LCD.

Inmediatamente se debe solicitar una contraseña a través del LCD (2x16).

**NOTA: Utilice la librería para el LCD 2x16.**

**Realice la implementación en simulación.**

### CHECKPOINT 3

## 2.3. Autenticación.

Utilice ahora un teclado matricial para poder ingresar una clave al sistema.

Establezca una contraseña de 4 dígitos para la autenticación respectiva.

1. En caso la persona ingrese la contraseña correspondiente, se mostrará en el LCD un mensaje de retroalimentación indicando que la **contraseña es correcta**. Se debe encender un **LED verde** que indique que el acceso por la puerta ha sido desbloqueado.
2. En caso la persona ingrese una contraseña errónea, se mostrará en el LCD un mensaje de error (contraseña incorrecta) y se activará un **LED Rojo** de retroalimentación. Solicitando otra vez el ingreso de la contraseña.

Se pide:

- a) **Realizar y presentar el diagrama de flujo correspondiente.**
- b) Realizar la simulación (puede realizar dos simulaciones con funcionamientos separados en caso no le alcancen los pines en simulación).

## 2.4. Ahorro energético (Standby).

Si la persona se aleja de la puerta de ingreso (debe obtener esta información utilizando el sensor de proximidad) el sistema debe:

- Apagar el sistema de grabación (representado por el **led rojo**).
- Borrar cualquier mensaje que se mostraba en el LCD o apagarlo.
- Cuando el sensor de proximidad vuelva a detectar a una persona, el sistema vuelve a iniciar mostrando el mensaje correspondiente.

Se pide:

- a) **Realizar y presentar el diagrama de flujo correspondiente.**
- b) Realizar la simulación (puede realizar dos simulaciones con funcionamientos separados en caso no le alcancen los pines en simulación).
- c) Implementar el circuito y demostrar el funcionamiento mencionado.

### CHECKPOINT 4

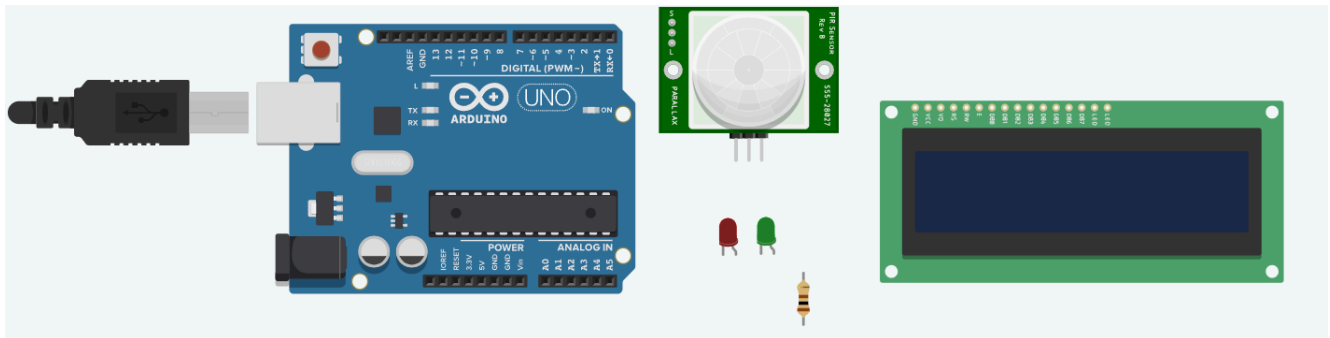
### 3. Sistema de control de aforo.

Deberá usar un **sensor de movimiento PIR** junto a la placa de desarrollo de **Arduino**. El sistema se emplea para indicar si hay movimiento de personas en la entrada de una oficina. El sistema deberá cumplir los siguientes requisitos.

#### 3.1. Elementos

Como mínimo su sistema deberá incluir:

- Arduino
- Sensor de Movimiento PIR.
- Diodo LED Rojo
- Diodo LED Verde
- LCD 2x16



#### 3.2. Funcionamiento.

El sensor de movimiento PIR se encargará de detectar el movimiento (entrada) de personas a una oficina. Tiene los siguientes modos de comportamiento.

##### I. Modo de Reposo

Si el sistema no detecta a ninguna persona, el diodo LED Verde se encenderá y se apagará de manera intermitente en ventanas de tiempo de 200 ms. Esto indica que el sistema se encuentra activo.

##### II. Detección de movimiento.

Al detectar movimiento en la región donde se instaló el sistema, el sistema debe apagar el diodo LED Verde y encender el diodo LED Rojo. Así mismo, en el LCD se debe indicar la cantidad de incidencias de detección de movimiento que hay.

- Si el sistema detecta el movimiento de una persona, el contador en el LCD debe mostrar: "INCIDENCIAS 001".

- El sistema debe volver a evaluar si hay movimiento. Si sigue detectando movimiento a los 500 ms, el contador no debe incrementar dado que puede tratarse de la misma persona.
- Si al cabo de **5 segundos** el sistema vuelve a detectar movimiento tras el primer incremento, el contador recién deberá incrementar. “INCIDENCIAS 002”.
- Si el sistema no detecta movimiento alguno, deberá volver al modo de Reposo.

### 3.3. Presentar

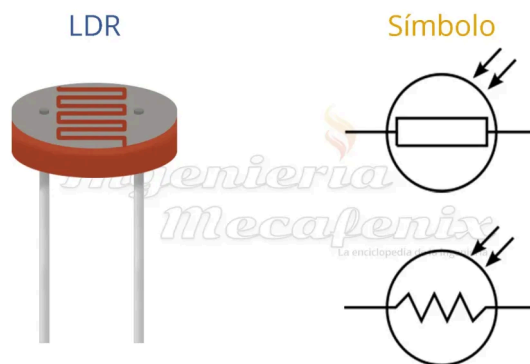
- Funcionamiento del sistema (**solo implementación**).
- Diagrama de flujo del funcionamiento de su sistema.
- (En el informe) Diagrama de **bloques**.

#### CHECKPOINT 5

## ANEXOS

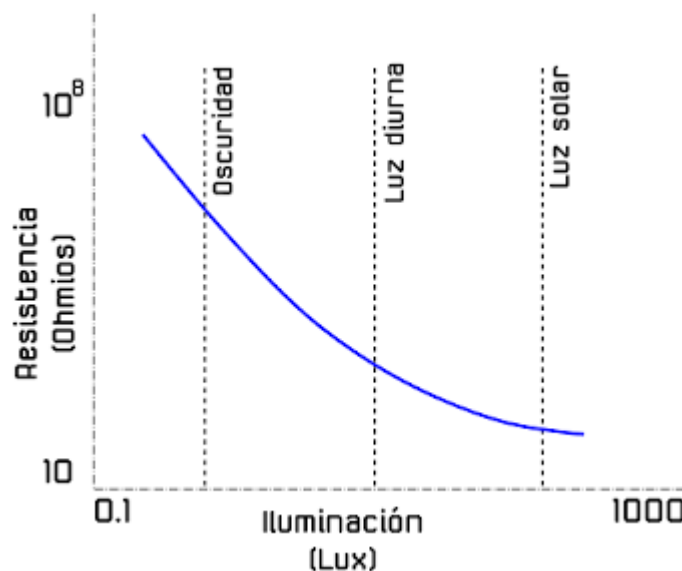
### 4. ANEXO I: Fotocelda o Resistor Dependiente de Luz (LDR)

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.



## 4.1. Características eléctricas típicas

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

## 4.2. 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.

## 4.3. 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.

### 4.3.1. 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 k $\Omega$** , 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.

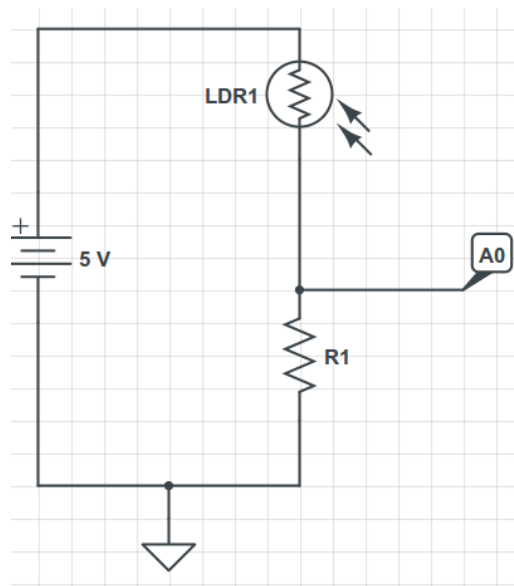


Diagrama de conexión de un LDR en configuración de divisor de voltaje.

#### 4.3.2. 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
}
```

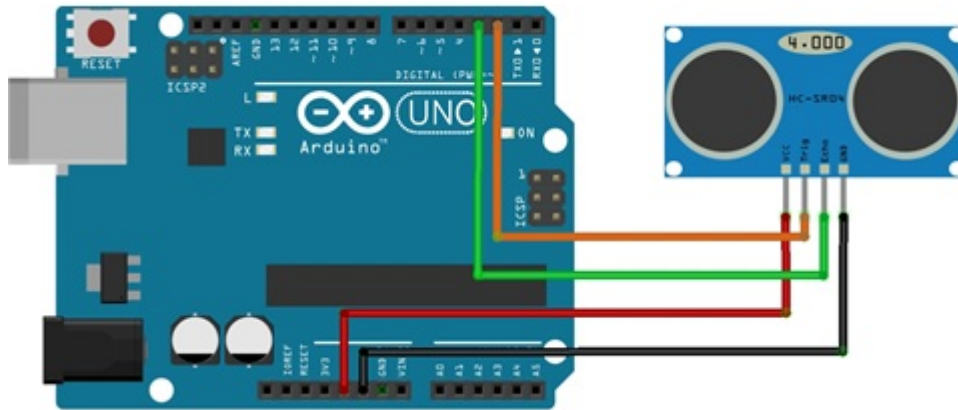
#### 4.3.3. Funcionamiento

- **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.

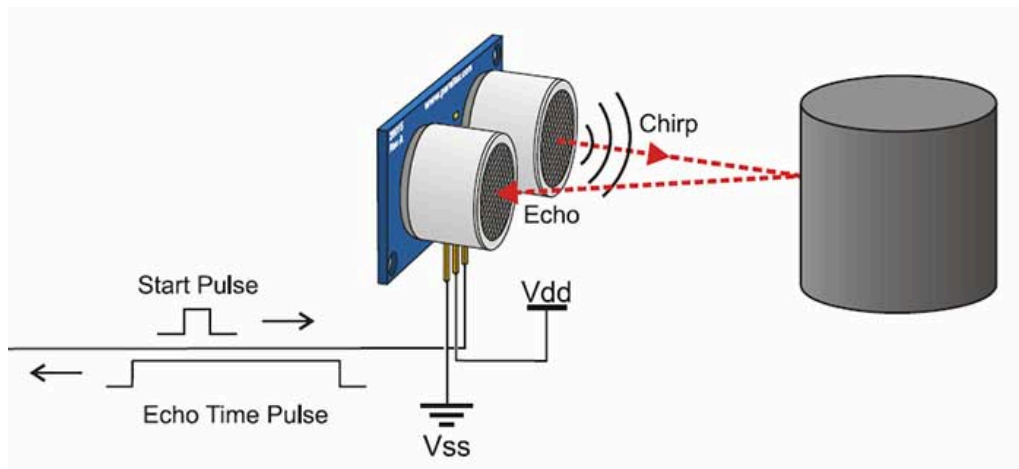
## 5. ANEXO II: Sensor de Proximidad

Para el correcto uso del sensor de proximidad se recomienda la guía disponible en línea de Naylamp.

[Enlace a tutorial de Sensor de Proximidad Ultrasónico](#)



Tenga en cuenta que la información es referencial. Nuestro trabajo utiliza como microcontrolador principal el Arduino MEGA.



Partimos de la siguiente formula:

$$Velocidad = \frac{distancia\ recorrida}{tiempo}$$

Donde **Velocidad** es la velocidad del sonido 340m/s, pero usaremos las unidades en cm/us pues trabajaremos en centímetros y microsegundos, **tiempo** es el tiempo que demora en llegar el ultrasonido al objeto y regresar al sensor, y la **distancia recorrida** es dos veces la distancia hacia el objeto, reemplazando en la formula tenemos:

$$\frac{340m}{s} \times \frac{1s}{1000000us} \times \frac{100cm}{1m} = \frac{2d}{t}$$
$$d(cm) = \frac{t(us)}{59}$$

Ejemplo de código:

```
const int Trigger = 2; //Pin digital 2 para el Trigger del sensor
const int Echo = 3; //Pin digital 3 para el Echo del sensor

void setup() {
  Serial.begin(9600); //iniciailzamos la comunicación
  pinMode(Trigger, OUTPUT); //pin como salida
  pinMode(Echo, INPUT); //pin como entrada
  digitalWrite(Trigger, LOW); //Inicializamos el pin con 0
}

void loop()
{
  long t; //timepo que demora en llegar el eco
  long d; //distancia en centímetros

  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10); //Enviamos un pulso de 10us
  digitalWrite(Trigger, LOW);

  t = pulseIn(Echo, HIGH); //obtenemos el ancho del pulso
  d = t/59; //escalamos el tiempo a una distancia en cm

  Serial.print("Distancia: ");
  Serial.print(d); //Enviamos serialmente el valor de la distancia
  Serial.print("cm");
  Serial.println();
  delay(100); //Hacemos una pausa de 100ms
}
```

## 6. ANEXO III. 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()**

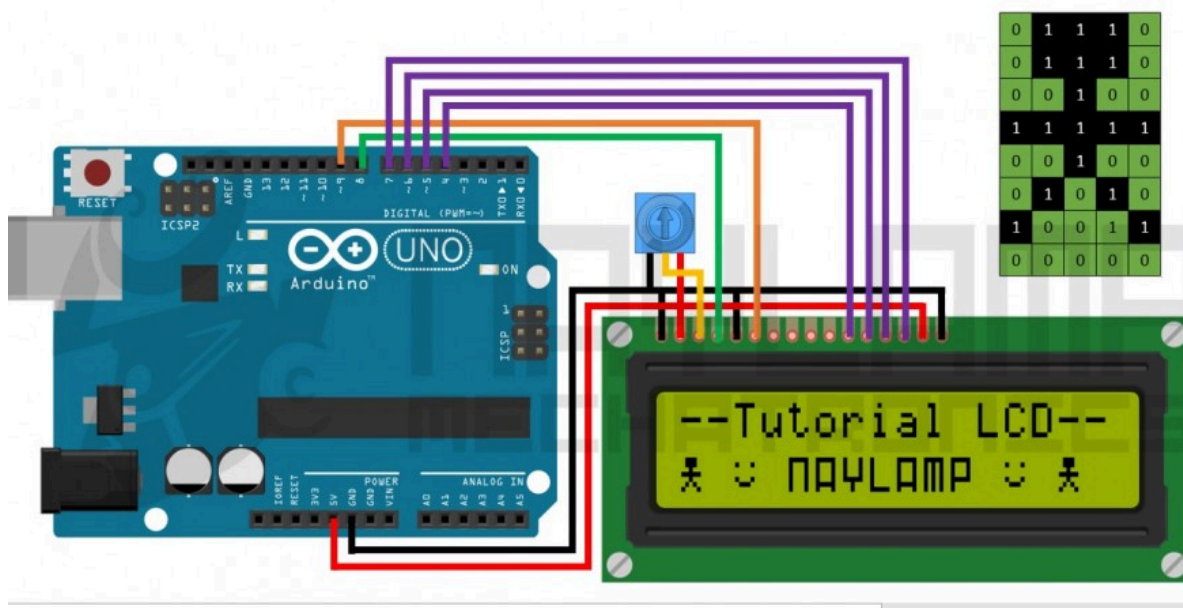
Se desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.

- **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 píxeles 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.
```

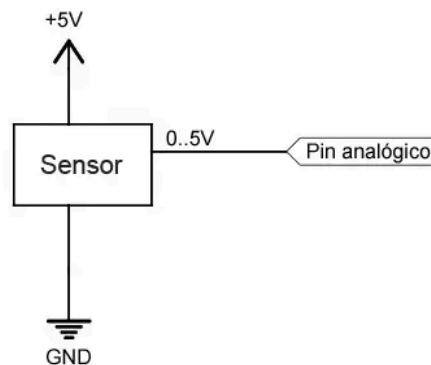
```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

## 7. ANEXO IV. 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).

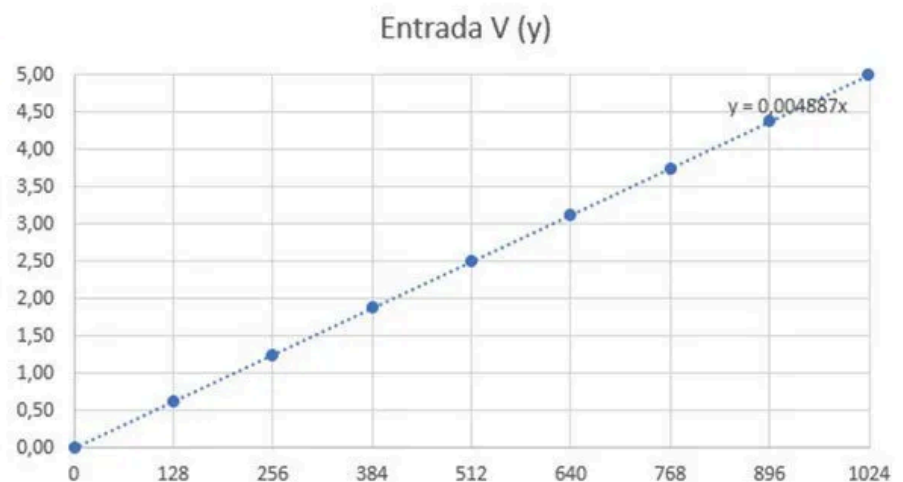


Se utiliza el comando `analogRead()`

**`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).

$y = 0,004887x$		
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](#)

## 8. ANEXO V. Sensor de Movimiento PIR

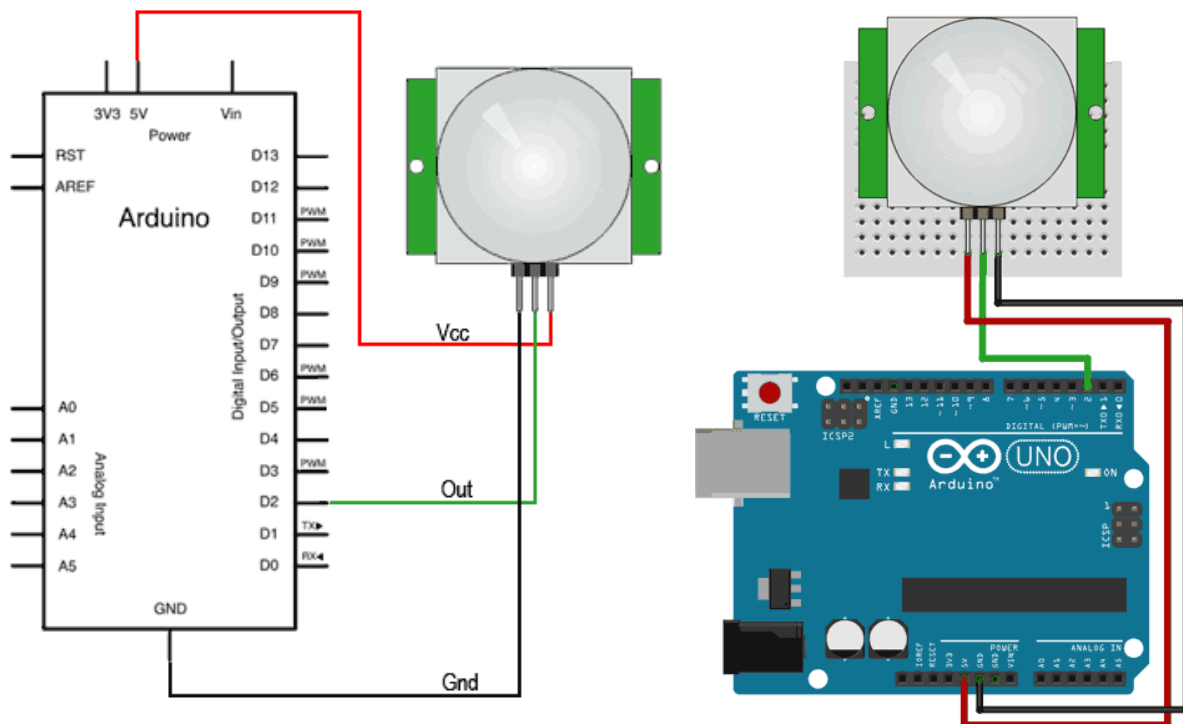
UN PIR (Infrarrojos pasivos) sensor de movimiento es un dispositivo que detecta el movimiento midiendo los cambios en radiación infrarroja. Detecta específicamente los niveles de radiación infrarroja emitida por cuerpos y objetos calientes, incluido el cuerpo humano.



Sensor de movimiento PIR.

Referencia:

[Enlace a tutorial de Sensor PIR + Arduino](#)



Ejemplo de conexionado.

Ejemplo de código.



```
const int LEDPin= 13;
const int PIRPin= 2;

void setup()
{
  pinMode(LEDPin, OUTPUT);
  pinMode(PIRPin, INPUT);
}

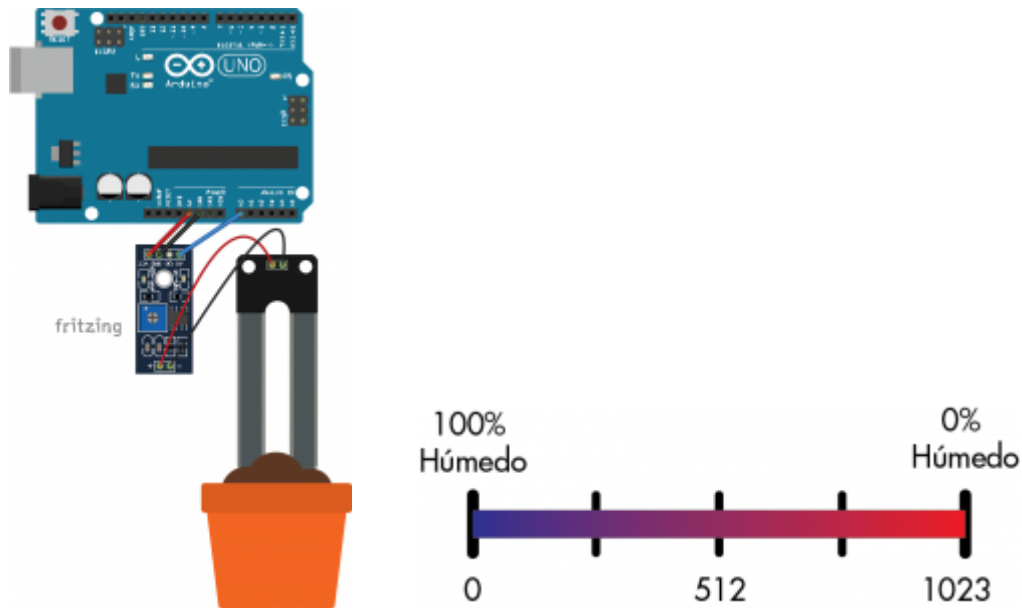
void loop()
{
  int value= digitalRead(PIRPin);

  if (value == HIGH)
  {
    digitalWrite(LEDPin, HIGH);
    delay(50);
    digitalWrite(LEDPin, LOW);
    delay(50);
  }
  else
  {
    digitalWrite(LEDPin, LOW);
  }
}
```

## 9. ANEXO VI. 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. ANEXO VII. Sensor de Temperatura

Referencia:

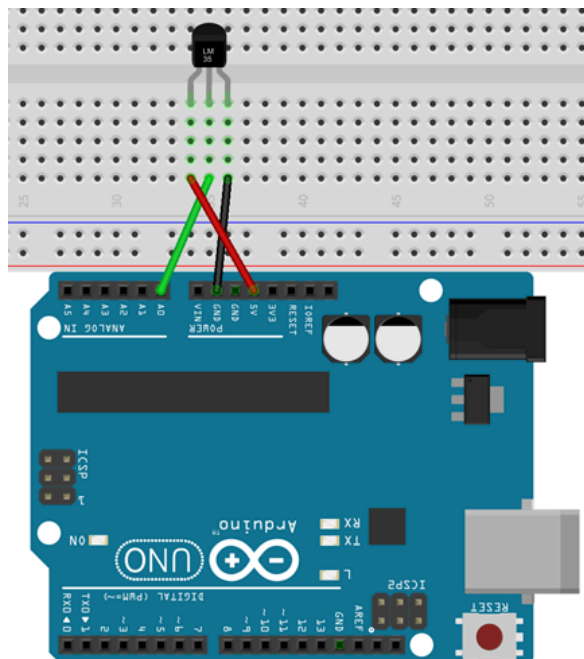
[Sensor de Temperatura con Arduino](#)

[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() {
```

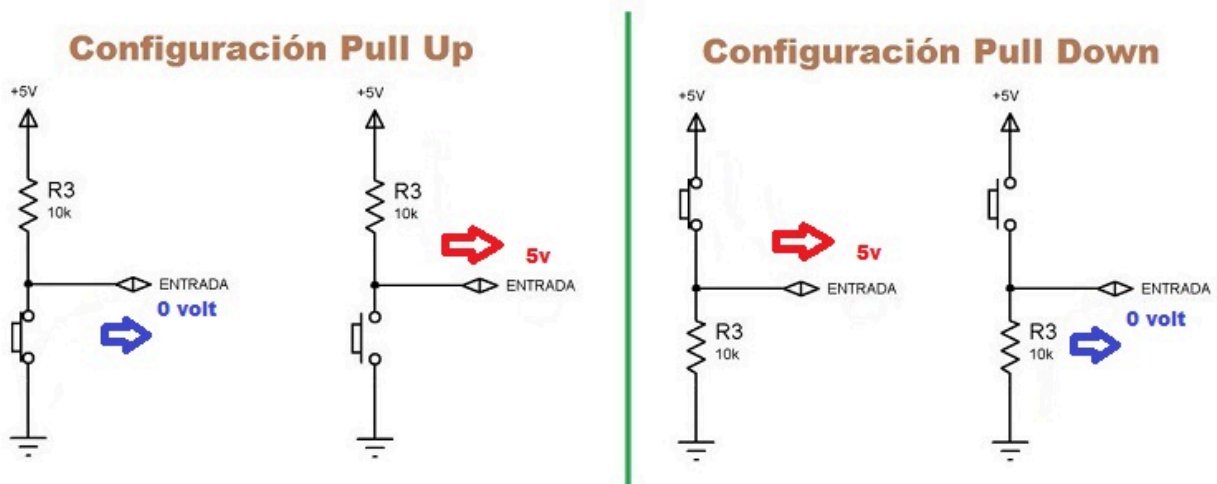
```
// Con analogRead leemos el sensor, recuerda que es un valor
de 0 a 1023
tempC = analogRead(pinLM35);

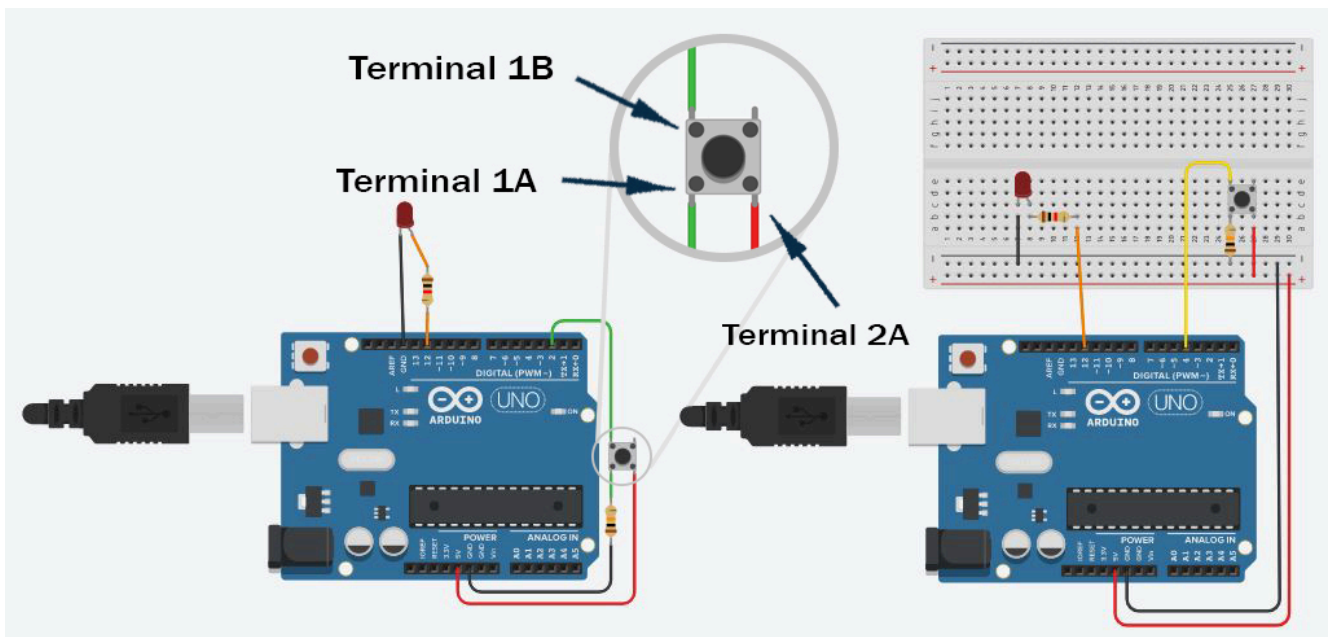
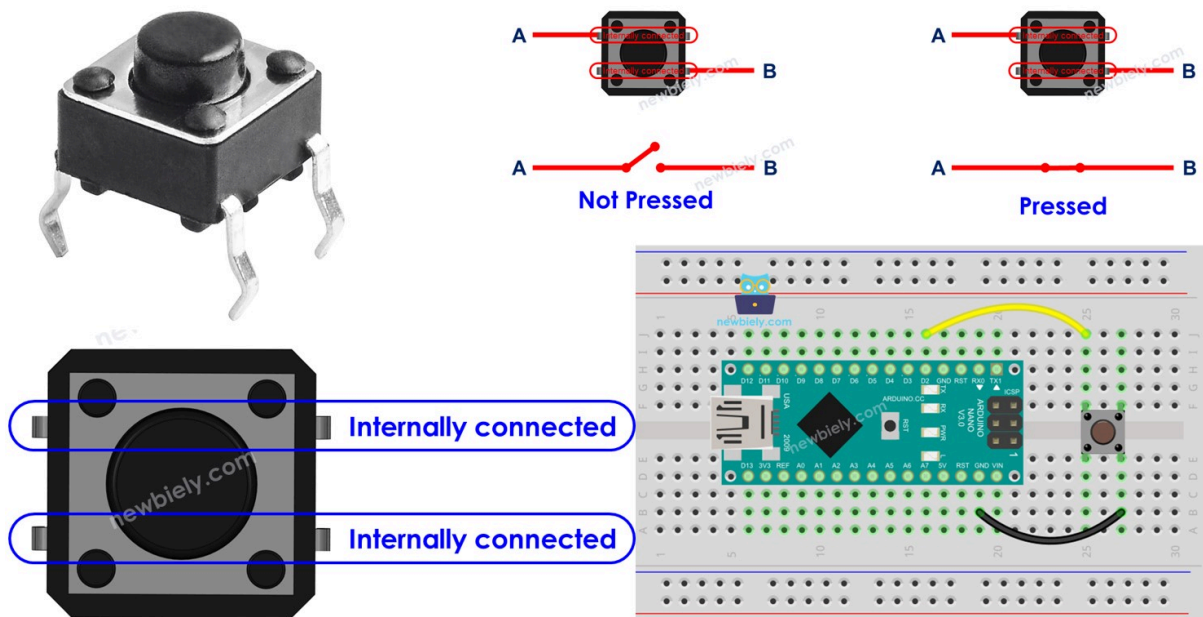
// Calculamos la temperatura con la fórmula
tempC = (5.0 * tempC * 100.0)/1024.0;

// 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. ANEXO V. El uso de botones con Arduino.





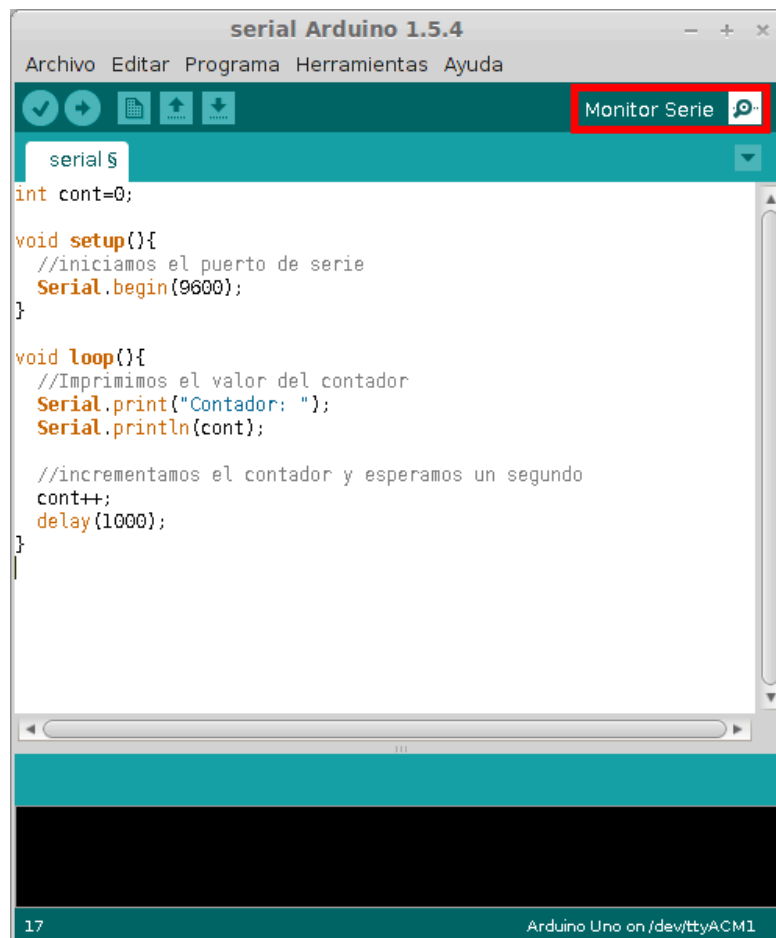
## 12. ANEXO VI. Uso del monitor Serial.

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

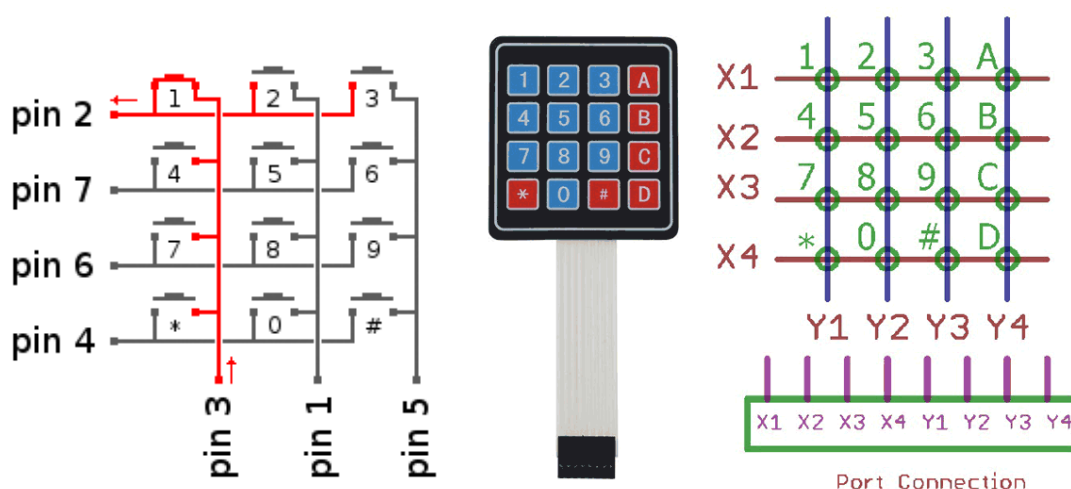
Encender o apagar el LED integrado en la placa Arduino. Para ello enviamos un carácter a la placa Arduino, empleando el monitor serial. En caso de enviar 'a' la placa Arduino apaga el LED, y en caso de enviar 'b' lo enciende.

```
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. ANEXO VII. 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. ANEXO VIII. Sensor de Movimiento PIR

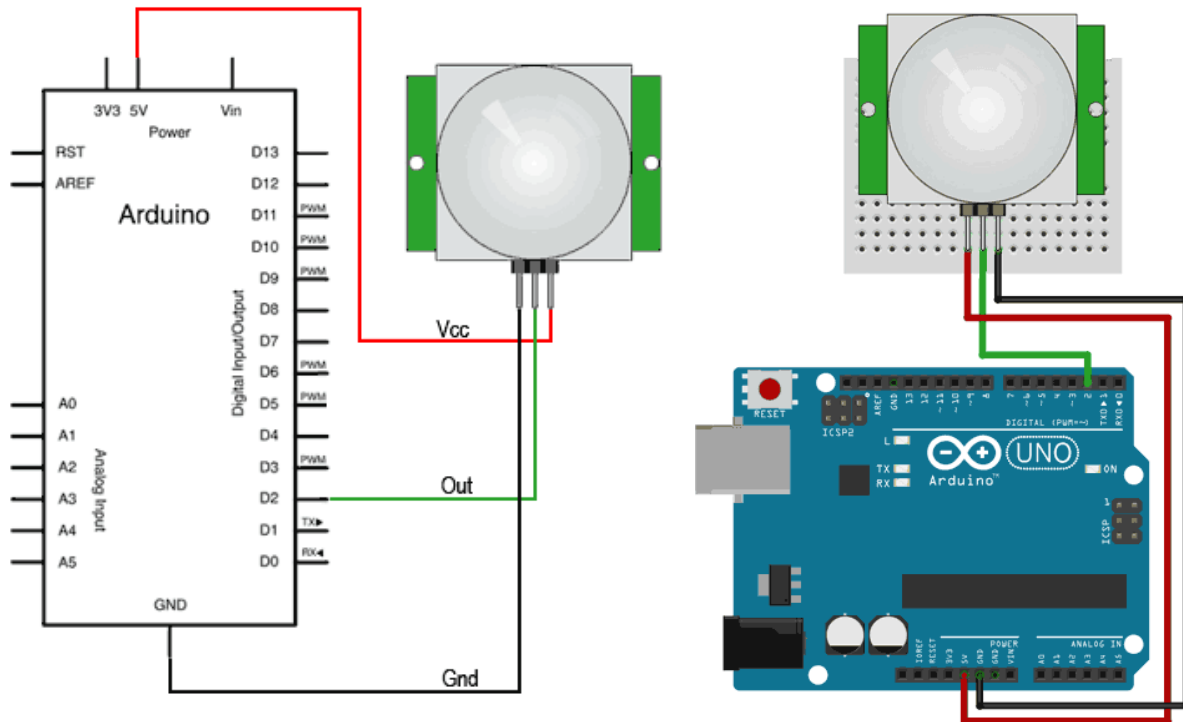
UN PIR (Infrarrojos pasivos) sensor de movimiento es un dispositivo que detecta el movimiento midiendo los cambios en radiación infrarroja. Detecta específicamente los niveles de radiación infrarroja emitida por cuerpos y objetos calientes, incluido el cuerpo humano.





Referencia:

[Enlace a tutorial de Sensor PIR + Arduino](#)



Ejemplo de conexionado.

### Ejemplo de código.

```
const int LEDPin= 13;
const int PIRPin= 2;

void setup()
{
  pinMode(LEDPin, OUTPUT);
  pinMode(PIRPin, INPUT);
}

void loop()
{
  int value= digitalRead(PIRPin);

  if (value == HIGH)
  {
    digitalWrite(LEDPin, HIGH);
    delay(50);
  }
}
```

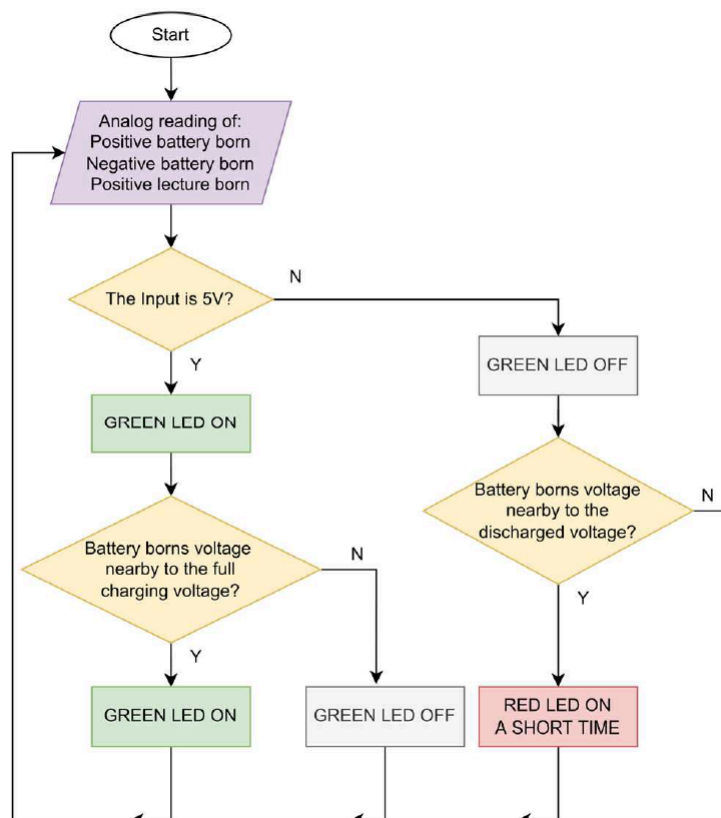
```

    digitalWrite(LEDPin, LOW);
    delay(50);
  }
  else
  {
    digitalWrite(LEDPin, LOW);
  }
}

```

## 15. ANEXO IX. Diagramas de flujo



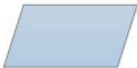

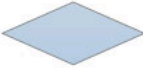
Los **diagramas de flujo** son representaciones gráficas que describen de manera secuencial los pasos o procesos necesarios para llevar a cabo una tarea específica. Su principal objetivo es facilitar la comprensión, el análisis y la comunicación de procesos complejos mediante el uso de **símbolos normalizados**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



### 15.1. 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

## 15.2. Ejemplo de Diagrama de Flujo para un código simple.

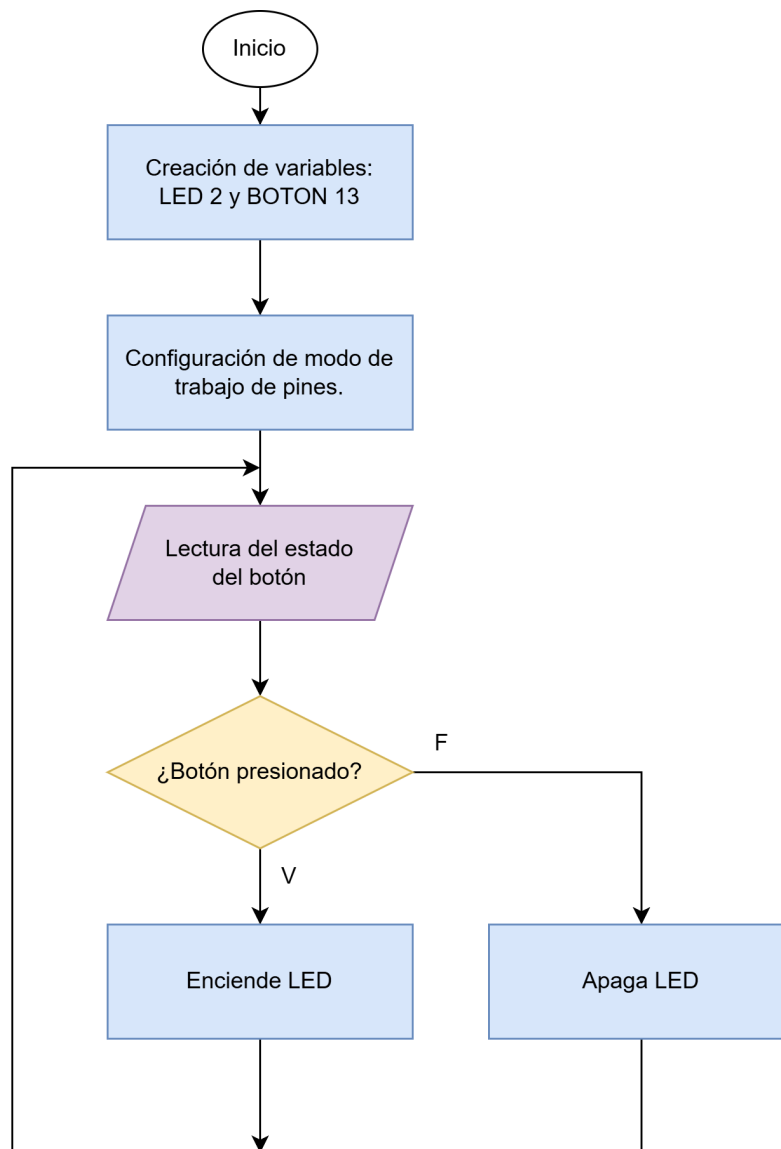
Ejemplo sencillo de código para Arduino que enciende un LED cuando se presiona un botón. Este ejemplo asume que tienes un botón conectado al pin digital 2 y un LED al pin digital 13.

```
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
    }
}
```



## 16. ANEXO X. 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.

## 16.1. 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.

## 16.2. 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).

## 16.3. Ejemplo

