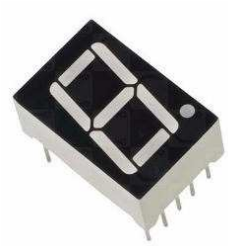
ACCESORIOS - Grupo 01

En esta Guía, explicaremos de Varios dispositivos, que habiendo resuelto las guías anteriores, resultaran muy simples y serán de interés para completar nuestros proyectos, o incentivar nuestra imaginación.

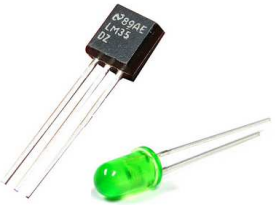
			
Sensor de Temperatura	Display Numérico (7 segmentos)	Ventilar (Cooler)	Láser (Diodo Láser)

Sensor de Temperatura LM35

El Sensor LM35, es un dispositivo diseñado para medir la temperatura digitalmente. A diferencia de otros dispositivos, el LM35 es un integrado con su propio circuito de control, que proporciona una salida de voltaje proporcional a la temperatura.

La salida es lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. El rango estimado de mediciones puede oscilar entre los -55ºC (-550mV) a 150ºC (1500 mV). Su precisión a temperatura ambiente es de 0,5ºC.

Sensor de Temperatura LM35


Relación de Tamaños

Podemos decir que este Sensor es muy pequeño, y para que tenga un idea, se lo puede comprar con el LED.

$$\text{Temp } ^oC = \frac{\text{valor sensor} \times 5 \times 100}{1023}$$

Como usar el Valor entregado por del sensor

En cuanto a la conexión, el LM35 es muy simple, y es el que se muestra en la imagen. Si vemos la cara plana de frente, el Pin (Pata izquierda) es el Positivo (3 a 5,5V), el pin central es donde realizaremos la medición, a razón de 10mV/ºC y debemos conectarlo a un Pin Analógicos. Y finalmente el pin o pata derecha la conectamos a GND común de nuestro dispositivo o al GND de nuestra placa Arduino.

Los sensores LM35 son muy simples de usar y baratos, lo que nos permitirá usarlos en infinidad de proyectos, desde un simple termómetro, a sistemas de control de ambiente.

1- Reconocimiento de un Sensor de Temperatura LM35. Lectura de Valores Entregados.

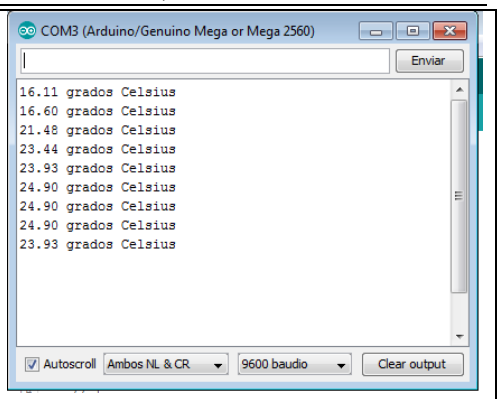
Lecturas de los valores entregados por un Sensor de Temperatura LM35. Visualización de Resultados en el monitor del Puerto Serie.

(Programa "250_Sensor_de_Temperatura_LM-35_01")

```
1  int PinLM35 = A0; // Definimos la entrada en pin A0
2  float tempC;
-
3  void setup(){
4     Serial.begin(9600);
```

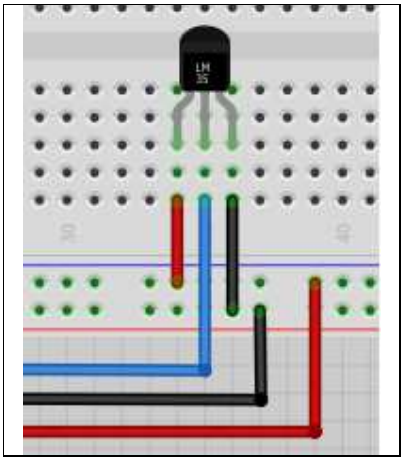
En el monitor serie, podremos ver:

```
5     while (!Serial) {
6       ; // Esperamos que el puerto serie este abierto.
7     }
8     pinMode( PinLM35, INPUT );
9   }
-
10  void loop(){
11    // Lee el valor desde el sensor
12    tempC = analogRead(PinLM35);
-
13    // Convierte el valor a temperatura
14    tempC = (5.0 * tempC * 100.0)/1024.0;
-
15    // Envia el dato al puerto serial
16    Serial.print(tempC);
17    Serial.print(" grados Celsius\n");
-
18    // Espera cinco segundo para repetir el loop
19    delay(5000);
20  }
```



Estaba fresco el lugar donde funcionaba el Sensor!

CIRCUITO PARA NUESTRO PROYECTO



Para Lograr que el Sensor perciba una temperatura diferente a la que hay en el ambiente, solo toca suavemente con tus dedos el sensor y observa el monitor, como subirá rápidamente

Lista de Materiales: 1 Sensor LM35 – 3 Cables Macho/Macho – 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: Cable Azul a A0, Nego a GND y cable Rojo a 5V,.

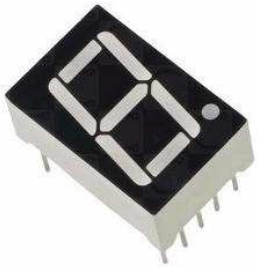
Como veras, es muy simple usar este sensor. Ahora imagina.. puedes mostrar la temperatura con una secuencia de Led, o prender un Ventilador si la temperatura se eleva, grabarlo en un Archivo de temperaturas o enviar el dato por Bluetooth. etc.

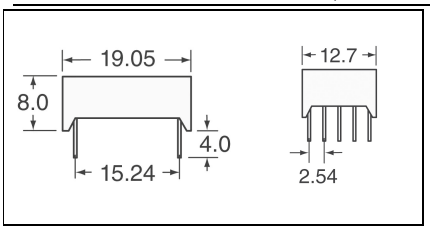
Display de 7 Segmentos (1 dígito)

Un display de segmentos (o visualizador de dígitos) es un componente electrónico que se utiliza para representar números.

Como su propio nombre indica, el display está compuesto por 7 segmentos, los cuales se encenderán y/o apagarán independientemente según sea el número que se quiera mostrar.

Internamente, podemos decir que son siete LEDs conectados estratégicamente formando el número 8.


Display 7 Segmentos (1 dígito)



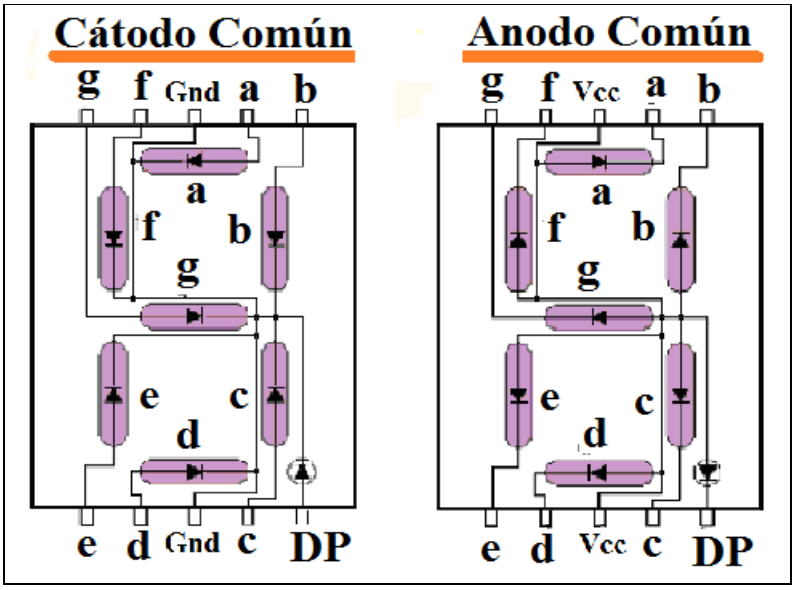
Como referencia, A la izquierda, puede visualizar las medidas generales de los display que usaremos en esta guía, aunque en el mercado se comercializan diversos tamaños.

Pero siembre debe recordar, que todos los modelos, en general se pueden clasificar en dos tipos:

1. Display de segmentos de **cátodo común**, en la que todos los cátodos de los LEDs están internamente unidos a una patilla común, la cual está conectada al potencial negativo de la placa.
2. Display de segmentos de **ánodo común**, en la que todos los ánodos se encuentran al positivo.

A los segmentos que componen estos display se denominan **a, b, c, d, e, f** y **g,** y **DP** simboliza el punto decimal (Decimal Piont), tal y como se muestra en la imagen de la derecha.

==========>



Anodo: Es el Contacto o Polo **POSITIVO**.
Cátodo: Es el Contacto o Polo **NEGATIVO**.

Para visualizar un numero solo deberemos prender los led o segmentos necesarios que formen el numero que queremos representar.

Entonces, por ejemplo:
- Para mostrar el número 0, tendremos que encender a, b, c, d, e y f.
- Para el número 2, tendríamos a, b, g, e y d.
- Y de la misma forma para cualquier otro número.

A continuación encontrarán las combinaciones para generar cualquier número.

	Catodo Comun							
	Numero	A	B	C	D	E	F	G
Enable	0	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0
0	2	1	1	0	1	1	0	1
0	3	1	1	1	1	0	0	1
0	4	0	1	1	0	0	1	1
0	5	1	0	1	1	0	1	1
0	6	1	0	1	1	1	1	1
0	7	1	1	1	0	0	0	0
0	8	1	1	1	1	1	1	1
0	9	1	1	1	1	0	1	1

	Anodo Comun							
	Numero	A	B	C	D	E	F	G
Enable	0	0	0	0	0	0	0	1
1	1	1	0	0	1	1	1	1
1	2	0	0	1	0	0	1	0
1	3	0	0	0	0	1	1	0
1	4	1	0	0	1	1	0	0
1	5	0	1	0	0	1	0	0
1	6	0	1	0	0	0	0	0
1	7	0	0	0	1	1	1	1
1	8	0	0	0	0	0	0	0
1	9	0	0	0	0	1	0	0

2- Usando un Display de un dígito (Display 7 segmentos), implementar un contador numérico de 0 (cero) a 9 (nueve).

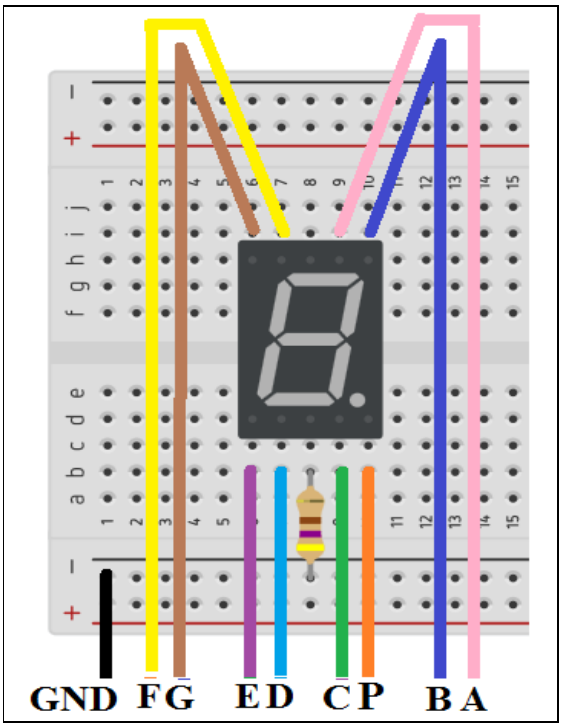
Usando un display de 7 segmentos implementar un contador que en forma automática cuente desde el cero al 9, recomenzando en cero cada vez que llega al nueve.

(Programa "400_1_Digito_Display_7_Segmentos_Catodo_Comun_01")

```
1  int pausa=2000; // intervalo de tiempo entre cada digito
2  const int Pin_A = 2;
3  const int Pin_B = 3;
4  const int Pin_C = 4;
5  const int Pin_D = 5;
6  const int Pin_E = 6;
7  const int Pin_F = 7;
8  const int Pin_G = 8;
9  const int Pin_Punto = 9;
10
11 void setup(){
12   pinMode(Pin_Punto, OUTPUT);
13   pinMode(Pin_A, OUTPUT);
14   pinMode(Pin_B, OUTPUT);
15   pinMode(Pin_C, OUTPUT);
16   pinMode(Pin_D, OUTPUT);
17   pinMode(Pin_E, OUTPUT);
18   pinMode(Pin_F, OUTPUT);
19   pinMode(Pin_G, OUTPUT);
20 }
21
22 void display (int a, int b, int c, int d, int e, int f, int g, int Punto){
23   digitalWrite (Pin_A,a);   //Se reciben cada uno de los 7 segmentos
24   digitalWrite (Pin_B,b);   //y Adicionalmente el Punto
25   digitalWrite (Pin_C,c);   //Formando cada numero o Punto
26   digitalWrite (Pin_D,d);   // Según se quiera.
27   digitalWrite (Pin_E,e);
28   digitalWrite (Pin_F,f);
29   digitalWrite (Pin_G,g);
30   digitalWrite (Pin_Punto,Punto);
31 }
32
33 void loop(){ //Funcion principal
34 // Dependiendo de cada dígito, se envía a la función display
35 // los estados (0 y 1) a cada uno de los segmentos
36   display (1,1,1,1,1,1,0,0); //Muestra un 0
37   delay(pausa);
38   display (0,1,1,0,0,0,0,0); //Muestra un 1
39   delay(pausa);
40   display (1,1,0,1,1,0,1,0); //Muestra un 2
41   delay(pausa);
42   display (1,1,1,1,0,0,1,0); //Muestra un 3
43   delay(pausa);
44   display (0,1,1,0,0,1,1,0); //Muestra un 4
45   delay(pausa);
46   display (1,0,1,1,0,1,1,0); //Muestra un 5
47   delay(pausa);
```

```
48    display (1,0,1,1,1,1,1,0); //Muestra un 6
49    delay(pausa);
50    display (1,1,1,0,0,0,0,0); //Muestra un 7
51    delay(pausa);
52    display (1,1,1,1,1,1,1,0); //Muestra un 8
53    delay(pausa);
54    display (1,1,1,0,0,1,1,0); //Muestra un 9
55    delay(pausa);
56    display (0,0,0,0,0,0,0,1); //Muestra un Punto
57    delay(pausa);
58  }
```

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Display 7 Segmentos de 0.56" Cátodo Común (No importa color), 1 Resistencia de 470Ω, 9 cables conectores macho/macho, 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: abajo, de izquierda a derecha: Cable Negro a GND, Cable Amarillo al pin digital configurado para el segmento F (En nuestro ejemplo será el pin 7), Cable Marrón para el segmento G (pin 8), Cable Morado para el segmento E (el pin 6), Cable Celeste segmento D (pin 5), Cable Verde segmento C (pin 4), Cable Naranja al pin digital configurado para el segmento P (Este es el Punto y usaremos el pin 9), Cable Azul segmento B (el pin 3), Cable Rosado el segmento A (el pin 2).

RECORDAR que el Punto, siempre debe ubicarse abajo y a la derecha

Mantenga este circuito conectado, ya que lo usaremos tres veces mas, La primera tal cual está, y la segunda y tercera agregaremos algunas cositas.

3- Repetir el Programa Anterior, pero esta vez usando Arreglos MultiDimensiónenles.

Usando un display de 7 segmentos implementar un contador que en forma automática cuente desde el cero al 9, recomenzando en cero cada vez que llega al nueve.

(Programa "400_1_Digito__Display_7_Segmentos_Catodo_Comun_02")

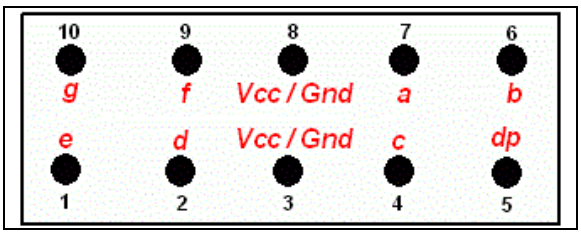
```
1   int pausa=1000; // intervalo de tiempo entre cada digito
2
3   const int Cant_Pines = 8;
4
5   //Array con los números de pin de cada segmento
6   //Usar para Arduino UNO y similares
7   //const int Pin_Display_01[Cant_Pines] = { 2, 3, 4, 5, 6, 7, 8, 9};
8
9   //Usar para Arduino MEGA
```

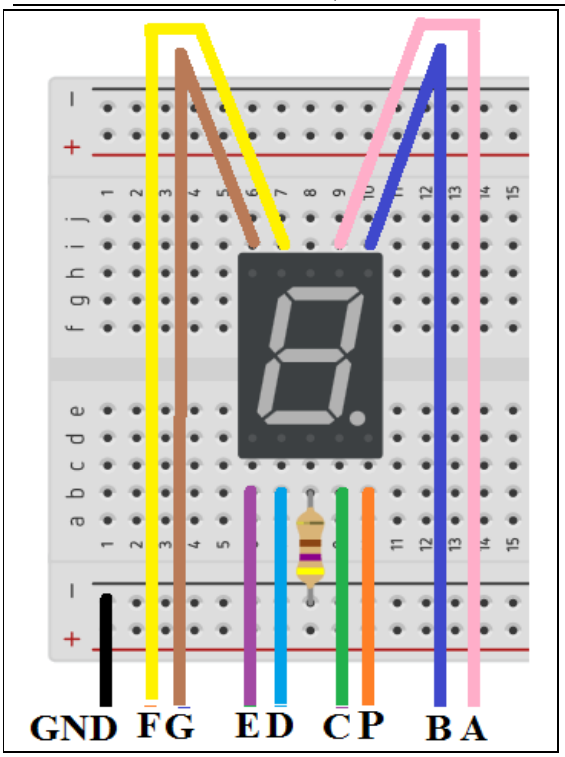
```
  const int Pin_Display_01[Cant_Pines] = {22,23,24,25,26,27,28,29};
/*                                          A  B  C  D  E  F  G  Punto*/
// Array multidimensional, donde cada fila indica el estado
// (prendido o apagado) de los segmentos de un número
int  Segmentos[11][Cant_Pines] = {
  { 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
  { 0, 1, 1, 0, 0, 0, 0, 0 }, // 1
  { 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
  { 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
  { 0, 1, 1, 0, 0, 1, 1, 0 }, // 4
  { 1, 0, 1, 1, 0, 1, 1, 0 }, // 5
  { 1, 0, 1, 1, 1, 1, 1, 0 }, // 6
  { 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
  { 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
  { 1, 1, 1, 0, 0, 1, 1, 0 }, // 9
  { 0, 0, 0, 0, 0, 0, 0, 1 }  // Punto
};
int i;

void setup(){
// Configuro los pines destinados a cada segmento
   for(i=0; i<Cant_Pines; i++){
      pinMode(Pin_Display_01[i], OUTPUT);
   }
}

void display( int v[ ] ){ // recibe 1 fila (vector) de la Matriz (Arreglo multidimencional)
  int c;
  for(c=0; c<Cant_Pines; c++){ // Cada Pin tiene un estado (prendido o apagado)
     digitalWrite ( Pin_Display_01[c], v[c] );
  }
}

void loop(){ //Función principal
// Dependiendo de cada numero, se envía la fila correspondiente
// Recordar que cada fila indica que segmentos prender
   for(i=0; i<11; i++){
      display(Segmentos[i]); // envío una fila de la matriz de segmentos
      delay(pausa);
   }
}
```

CIRCUITO PARA NUESTRO PROYECTO

Acá puede encontrar, como ayuda de memoria, donde puede ver el pin que corresponde a cada segmento.

Como puede apreciar, el GND, es el mismo sea el de arriba o el de abajo.





Lista de Materiales: 1 Display 7 Segmentos de 0.56" Cátodo Común (No importa color), 1 Resistencia de 470Ω, 9 cables conectores macho/macho, 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: abajo, de izquierda a derecha: Cable Negro a GND, Cable Amarillo al pin digital configurado para el segmento F (En nuestro ejemplo será el pin 27), Cable Marrón para el segmento G (pin 28), Cable Morado para el segmento E (el pin 26), Cable Celeste segmento D (pin 25), Cable Verde segmento C (pin 24), Cable Naranja al pin digital configurado para el segmento P (Este es el Punto y usaremos el pin 29), Cable Azul segmento B (el pin 23), Cable Rosado el segmento A (el pin 22).

RECORDAR que el Punto, siempre debe ubicarse abajo y a la derecha

Mantenga este circuito conectado, ya que lo usaremos dos veces más.

4- Construir un contador de cero a 9. Cada vez que se presione un boton, el contador debe incrementar en uno.

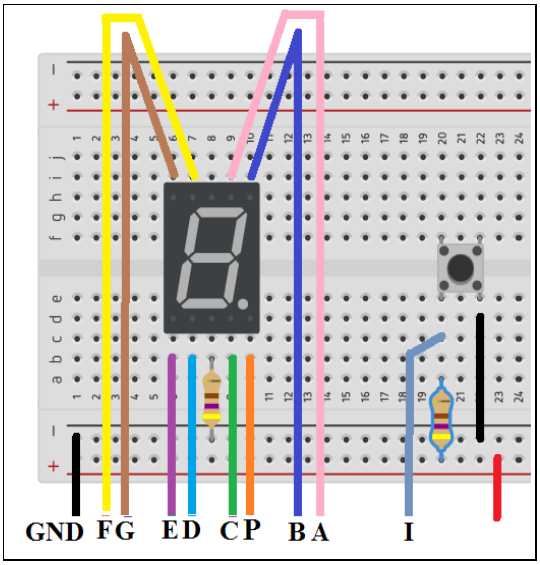
Usando un display de 7 segmentos implementar un contador que cada vez que se presione el botón se incremente en uno el valor mostrado, recomenzando en cero cada vez que llega al nueve.

(Programa "400_1_Digito__Display_7_Segmentos_Catodo_Comun_03_con_Boton")

```
1  int pausa=3; // intervalo de tiempo entre cada digito
2
3  const int Pin_Boton = 12;
4  int EstadoBoton, Presionado;
5
6  const int Cant_Segmentos = 8;
7
8  //Array con los numeros de pin de cada segmento
9  //Usar para Arduino UNO y similares
10 //const int Pin_Display_01[Cant_Segmentos] = { 2, 3, 4, 5, 6, 7, 8, 9};
11
12 //Usar para Arduino MEGA
13   const int Pin_Display_01[Cant_Segmentos] = {22,23,24,25,26,27,28,29};
14 /*                                          A  B  C  D  E  F  G  Punto*/
15
16 // Array multidimensional
17 // cada fila son los segmentos de un numero
18 int  Segmentos[11][Cant_Segmentos] = {
19   { 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
20   { 0, 1, 1, 0, 0, 0, 0, 0 }, // 1
21   { 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
22   { 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
23   { 0, 1, 1, 0, 0, 1, 1, 0 }, // 4
24   { 1, 0, 1, 1, 0, 1, 1, 0 }, // 5
25   { 1, 0, 1, 1, 1, 1, 1, 0 }, // 6
```

```
  { 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
  { 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
  { 1, 1, 1, 0, 0, 1, 1, 0 }, // 9
  { 0, 0, 0, 0, 0, 0, 0, 1 }  // Punto
};

int i;

void setup(){
  Serial.begin(9600);
  while (!Serial) {
    ; // Esperamos que el puerto serie este abierto.
  }
  pinMode(Pin_Boton, INPUT);
  Presionado=0;

// Conguro los pines destinados a cada segmento
  for(i=0; i<Cant_Segmentos; i++){
    pinMode(Pin_Display_01[i], OUTPUT);
  }
  i=0;
  Serial.println("Programa Botón Contador Iniciado...");
}

void display(int v[]){
 int c;
 for(c=0; c<Cant_Segmentos; c++){  // Cada Pin tiene un estado (prendido o apagado)
   digitalWrite( Pin_Display_01[c], v[c] );
 }
}

void loop(){
  EstadoBoton = digitalRead(Pin_Boton); // Leemos el PinDigital.
  if (EstadoBoton == LOW) { //Pregunta si el pulsador está presionado
     Presionado = 1;     //La variable cambia de valor
     Serial.println("Precionado - ");
  }
  delay(pausa);
  EstadoBoton = digitalRead(Pin_Boton); // Leemos el PinDigital.
  if(EstadoBoton == HIGH && Presionado == 1) {
    Presionado = 0;    //La variable vuelve a su valor original
    i++;
    if(i>9){  // si el contador supera el 9, regresa a Cero
      i=0;
    }
    Serial.print("Valor de I: ");
    Serial.println(i);
  }
// Dependiendo del numero, se envía la fila correspondiente del número a mostrar
// Recordar que cada fila indica que segmentos prender
  display(Segmentos[i]); // envío una fila de la matriz de segmentos
}
```

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 1 Display 7 Segmentos de 0.56" Cátodo Común (No importa color), 2 Resistencia de 470Ω, 1 Botón Pulsador de 2 Patas, 12 cables conectores macho/macho, 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: abajo, de izquierda a derecha: Cable Negro a GND, Cable Amarillo al pin digital configurado para el segmento F (En nuestro ejemplo será el pin 27), Cable Marrón para el segmento G (pin 28), Cable Morado para el segmento E (el pin 26), Cable Celeste segmento D (pin 25), Cable Verde segmento C (pin 24), Cable Naranja al pin digital configurado para el segmento P (Este es el Punto y usaremos el pin 29), Cable Azul segmento B (el pin 23), Cable Rosado el segmento A (el pin 22), Cable Gris, debe conectarse al botón digital (el pin 12 para nuestro ejemplo) y finalmente el Cable Rojo, que conectaremos a 5V de nuestra placa Arduino.

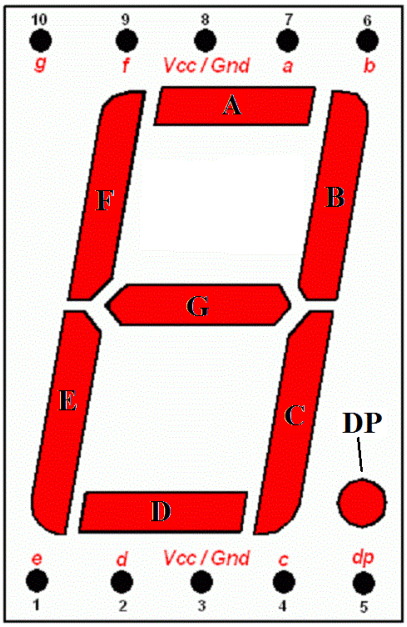
RECORDAR: en el Display, el Punto siempre se ubica abajo y a la derecha.

5- Construir un contador de 2 dígitos. El dispositivo debe ser capas de contar automáticamente desde cero a 99 y comenzar nuevamente.

Usando dos display de un dígitos, siendo ambos de 7 segmentos, implementar un contador automático que comenzando en cero llegue a 99 y recomience.

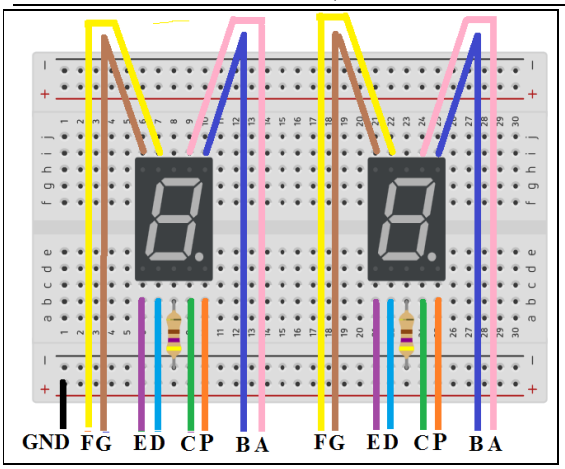
(Programa "400_2_Digitos__Display_7_Segmentos_Catodo_Comun_01")

```
1  int pausa= 500; // intervalo de tiempo entre cada digito
2
3  const int Cant_Pines = 8;
4
5  //Para Arduino MEGA  Array multidimensional
6  // cada fila son los Pines, segmentos de un Digito
7  const int Pin_Display [2][Cant_Pines] = {
8     {22,23,24,25,26,27,28,29},
9     {40,41,42,43,44,45,46,47}
10 }; /* A  B  C  D  E  F  G   Punto*/
11
12 // Array multidimensional
13 // cada fila son los segmentos de un numero
14 const int  Segmentos[11][Cant_Pines] = {
15    { 1, 1, 1, 1, 1, 1, 0, 0 }, // 0
16    { 0, 1, 1, 0, 0, 0, 0, 0 }, // 1
17    { 1, 1, 0, 1, 1, 0, 1, 0 }, // 2
18    { 1, 1, 1, 1, 0, 0, 1, 0 }, // 3
19    { 0, 1, 1, 0, 0, 1, 1, 0 }, // 4
20    { 1, 0, 1, 1, 0, 1, 1, 0 }, // 5
21    { 1, 0, 1, 1, 1, 1, 1, 0 }, // 6
22    { 1, 1, 1, 0, 0, 0, 0, 0 }, // 7
23    { 1, 1, 1, 1, 1, 1, 1, 0 }, // 8
24    { 1, 1, 1, 0, 0, 1, 1, 0 }, // 9
25    { 0, 0, 0, 0, 0, 0, 0, 1 }  // Punto
26 };
```



```
27  int i,          // Variable usada en for( )
28      d1, d2;  // Para contener el primer y segundo digito del numero a mostrar
29
30  void setup(){
31    Serial.begin(9600);
32    while (!Serial) {
33      ; // Esperamos puerto serie este abierto.
34    }
35
36  // Configuro los pines destinados a cada segmento
37    for(i=0;i<Cant_Pines;i++){
38      pinMode(Pin_Display[0][i], OUTPUT);
39      pinMode(Pin_Display[1][i], OUTPUT);
40    }
41    Serial.println("Programa Contador de 2 Dígitos Iniciado...");
42  }
43
44  void Display( int d, int v[ ] ){
45    int c;
46    for(c=0;c<Cant_Pines;c++){
47        digitalWrite ( Pin_Display[d-1][c], v[c]);
48    }
49  }
50
51  void loop(){ //Función principal
52  // Dependiendo de cada numero, se envía la fila correspondiente
53  // Recordar que cada fila indica que segmentos prender
54
55    for( i=0; i<99; i++ ){
56      d1=(int) ( i /10 );   // Guardo primer dígito truncando el segundo
57      d2= i - ( d1*10 );   // le resto al valor de I, el primer digito
58
59      Serial.print("Número: ");
60      Serial.print( i );
61      Serial.print(" - Primer Digito: ");
62      Serial.print(  d1 );
63      Serial.print(" - Segundo Digito: ");
64      Serial.println( d2 );
65
66      Display( 1, Segmentos[d1] );
67      Display( 2, Segmentos[d2] );
68
69      delay(pausa);
70
71    }// Fin del for()
72  }// Fin del Loop()
```

CIRCUITO PARA NUESTRO PROYECTO



Lista de Materiales: 2 Display 7 Segmentos de 0.56" Cátodo Común (No importa color), 2 Resistencia de 470Ω, 17 cables conectores macho/macho, 1 Placa Protoboard - Placa Arduino y 1 Cable USB.

Los cables deberán ser conectados: abajo, de izquierda a derecha: Cable Negro a GND, y los cables restantes corresponden a los dos display. **Comenzando por el primero** Cable Amarillo al pin digital configurado para el segmento F (En nuestro ejemplo será el pin 27), Cable Marrón para el segmento G (pin 28), Cable Morado para el segmento E (el pin 26), Cable Celeste segmento D (pin 25), Cable Verde segmento C (pin 24), Cable Naranja al pin digital configurado para el segmento P (Este es el Punto y usaremos el pin 29), Cable Azul segmento B (el pin 23), Cable Rosado el segmento A (el pin 22). **Ahora el segundo Display**, Cable Amarillo segmento F (En nuestro ejemplo será el pin 45), Cable Marrón para el segmento G (pin 46), Cable Morado para el segmento E (el pin 44), Cable Celeste segmento D (pin 43), Cable Verde segmento C (pin 42), Cable Naranja al pin digital configurado para el segmento P (Este es el Punto y usaremos el pin 47), Cable Azul segmento B (el pin 41), Cable Rosado el segmento A (el pin 40).

RECORDAR: en el Display, el Punto siempre se ubica abajo y a la derecha.

Ver Decodificador cd4511

http://www.learnerswings.com/2014/04/seven-segment-display-using-arduino.html

y

https://blog.ars-electronica.com.ar/2017/08/cd4511-decodificador-para-display-7.html

COOLER

(cooler, fan, cúler). Ventilador que se utiliza en los gabinetes de computadoras y otros dispositivos electrónicos para refrigerarlos. Por lo general el aire caliente es sacado desde el interior del dispositivo con los coolers. También puede cumplir una función de limpieza, dado que puede retirar aire sucio (con pelusas) del interior del dispositivo.

Dónde se utilizan los coolers en hardware

Los coolers se utilizan especialmente en las fuentes de energía, generalmente en la parte trasera del gabinete de la computadora. Actualmente también se incluyen coolers adicionales para el microprocesador y placas que pueden sobrecalentarse. Incluso a veces son usados en distintas partes del gabinete para una refrigeración general.

Por lo general los coolers en las PCs de escritorio están continuamente encendidos, en cambio en las computadoras portátiles suelen prenderse y apagarse automáticamente dependiendo de las necesidades de refrigeración (por una cuestión de ahorro energético).

Buen funcionamiento del cooler

La falta, obstrucción o rotura de un cooler, puede terminar con un dispositivo inutilizado por el exceso de temperatura.

Los coolers son uno de los elementos que, en funcionamiento, suelen ser de los más ruidosos en una computadora. Por esta razón, deben mantenerse limpios, aceitados y ser de buena calidad. Los viejos ventiladores podían producir sonidos de hasta 50 decibeles, en cambio, los actuales están en los 20 decibeles.

Consejos para el buen funcionamiento de los cooler:

- No obstruir la salida de aire del ventilador.

- Alejarlo de paredes y superficies que se calienten.

- Los coolers suelen ser relativamente silenciosos, por lo que un exceso de ruido puede significar algún problema (falta de aceite, deformidad de las aspas o las aspas tocan algún cable o material cuando se mueven).

- Controlar los coolers regularmente para saber que funcionan.

Actualmente también las computadoras incluyen detección y aviso de funcionamiento de coolers. Antiguamente los coolers podían estropearse y dejar de funcionar sin que el usuario lo note, ocasionando que la computadora aumente su temperatura y produciendo errores de todo tipo.

Los coolers nunca deben ser obstruidos con ningún objeto, pues esto puede causar un sobrecalentamiento en la computadora.

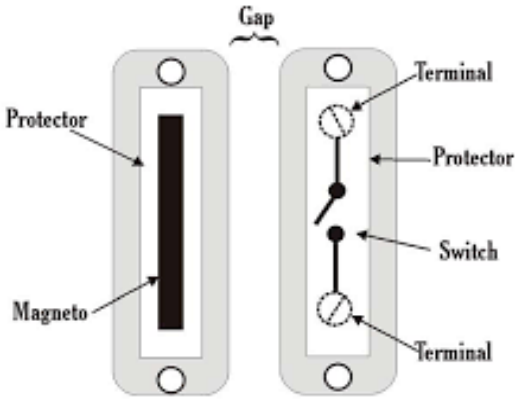
ACCESORIOS - Grupo 02

			
Sensor Magnético			

Sensor Magnético

Los sensores magnéticos, comúnmente conocidos como contactos magnéticos, se usan para detectar aperturas no deseadas y activar alarmas.

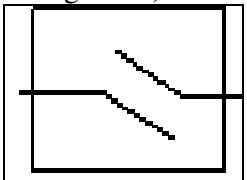

Sensor Magnético



Descripción del Dispositivo

Estos sensores constan de un imán y un reedswitch (interruptor magnético).

Este sensor funciona como un switch normalmente abierto (mientras hay campo magnético).


Simbología Usada

Cuando la puerta o ventana se abre, el circuito eléctrico se cierra y en consecuencia es posible detectar la apertura de la misma.
Generalmente estos sensores vienen completamente sellado en plástico (resistentes al agua y a los agentes corrosivos del ambiente) lo que lo hace extremadamente resistentes y duraderos.

Recordar: es indistinto a que polo se conecten los cables o terminales del sensor, ya que solo es un interruptor.

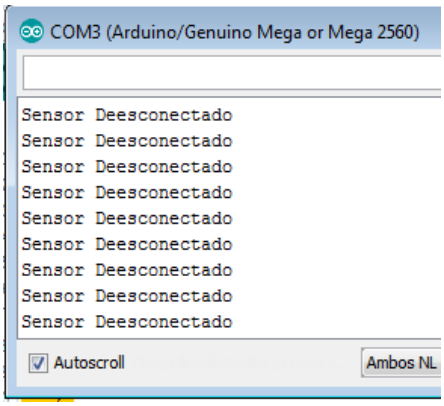
6- Reconocimiento de un Sensor Magnético. Detectar la desconexión del Sensor.

Detectar la desconexión del Sensor, informando por medio del monitor del Puerto Serie la desconexión. Simultáneamente, prender y apagar un Led indicando la desconexión.

(Programa "300_Sensor_Magnetico_001")

```
1   int ledAlarmaPin = 5;         // Pin de salida para el LED
2   int SensorMagnetico =  10; // Pin Sensor Magnético
-
3   void setup( ) {
4     pinMode(ledAlarmaPin, OUTPUT);
5     pinMode(SensorMagnetico, INPUT_PULLUP);
6     Serial.begin(9600);
-     while (!Serial) {
7       ;    //Esperamos que el puerto serie este abierto.
8     }
9   }
-
10  void loop( ) {
11    if (digitalRead( SensorMagnetico ) == HIGH){
12      Serial.println("Sensor Deesconectado");
13      digitalWrite(ledAlarmaPin, HIGH); // Prende  LED
14      delay(250);                  // Pause de ¼ de segundo
15      digitalWrite(ledAlarmaPin, LOW);  // Apaga el LED
16      delay(250);                  // Pausa de ¼ de segundo
17    }
18  }
```

En el monitor serie, podremos ver:



Cada vez que el Sensor es desconectado.

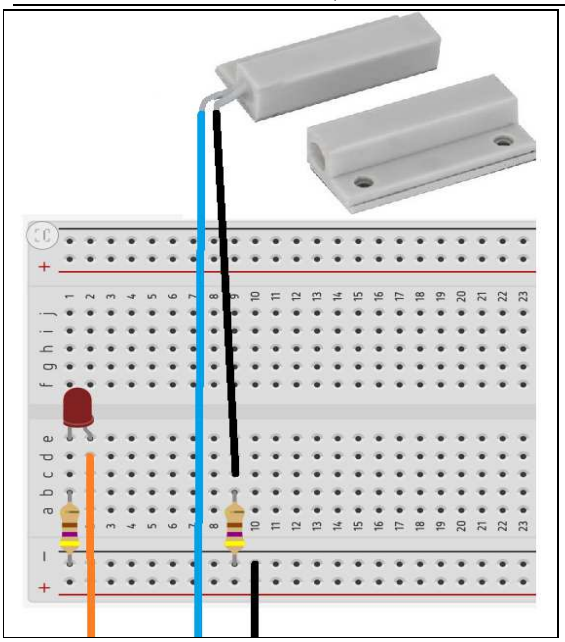
IMPORTANTE: usamos **INPUT_PULLUP** para asegurar que no haya fluctuaciones entre **LOW** y **HIGH**

El código muestra un ejemplo sencillo. Usamos una entrada digital para leer el estado del sensor magnético (magnetic reed). Si el sensor está activado, la entrada leerá LOW, pero si se desconecta (las partes están separadas) entonces registraremos HIGH y prenderemos y apagaremos a intervalos regulares un led de alarma. Por supuesto, en un proyecto real, en lugar de encender un LED ejecutaríamos las acciones que quisiéramos, algo más elaborado.

CIRCUITO PARA NUESTRO PROYECTO

Lo que hay que tener en cuenta y siempre recordar, es que cuando se activa el sensor, deja pasar toda la corriente que el pin es capas de suministrar, y el máximo que soporta es 500mA, si sobrepasamos ese valor podemos dañar permanentemente el pin o la placa completa, es por esto que debemos usar una resistencia apropiada, siendo esta de un mínimo de 470Ω, la que usamos en proyectos anteriores (para conectar led). Aunque recomiendo siempre realizar el calculo apropiado para el tipo de placa Arduino y sensor que usemos.

Otra opción disponible para conectar, aunque solo para una prueba de funcionamiento, es usar las resistencias internas de **pull-up** de Arduino, por lo que simplemente conectamos el sensor (magnetic reed) a GND y la entrada digital que queramos emplear.



Lista de Materiales: 1 Sensor magnético para puertas y ventanas, 1 Placa Protoboard, 1 Led rojo, 2 Resistencias de 470Ω, cables conectores.

Los cables deberán ser conectados, abajo, de izquierda a derecha: Cable Naranja al pin digital 5, cable Azul al pin digital 10, y el cable negro a GND.

Recordar: es indistinto a que polo se conecten los cables o terminales del sensor, ya que solo es un interruptor.

01010100 01101111 01100100 01101111 00100000 01101100 01101111 00100000 01110001
01110101 01100101 00100000 01110101 01101110 01100001 00100000 01110000 01100101
01110010 01110011 01101111 01101110 01100001 00100000 01110000 01110101 01100101
01100100 01100101 00100000 01101001 01101101 01100001 01100111 01101001 01101110
01100001 01110010 00101100 00100000 01000011 01101111 01101110 00100000 01100101
01110011 01110100 01110101 01100100 01101001 01101111 00101100 00100000 01101110
01101111 01110011 01101111 01110100 01110010 01101111 01110011 00100000 01010001
01010101 01001001 01011010 11000011 10000001 01010011 00100000 01010000 01001111
01000100 01000001 01001101 01001111 01010011 00100000 01101000 01100001 01100011
01100101 01110010 01101100 01101111 00100000 01110010 01100101 01100001 01101100
01101001 01100100 01100001 01100100 00101110

Si tienes algunas Correcciones y/o Sugerencias, por favor contáctame.