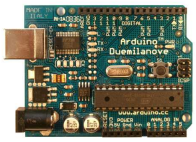
P R Á C T I C A S C O N A R D U I N O			
	**Nombres y apellidos:** …… ……	**Curso:** …… **Fecha:** ……	

SEGUNDA ACTIVIDAD

OBJETIVO: Realizar un medidor de luz con dos displays 7 segmentos (medida en porcentaje de 0 a 99%)

MATERIAL:

- 2 Displays 7 segmentos de cátodo común
- 1 LDR
- Resistencia de 1 k
- Placa arduino
- Cable.

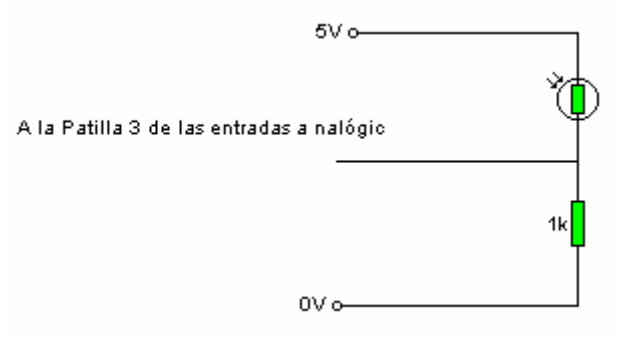
FUNDAMENTO TEÓRICO

Una LDR (Light Dependent Resistors) es una resistencia que disminuye el valor óhmico al aumentar la luz que incide sobre ella. Se emplean como sensores de luz, barreras fotoeléctricas, etc.

Su símbolo es: 

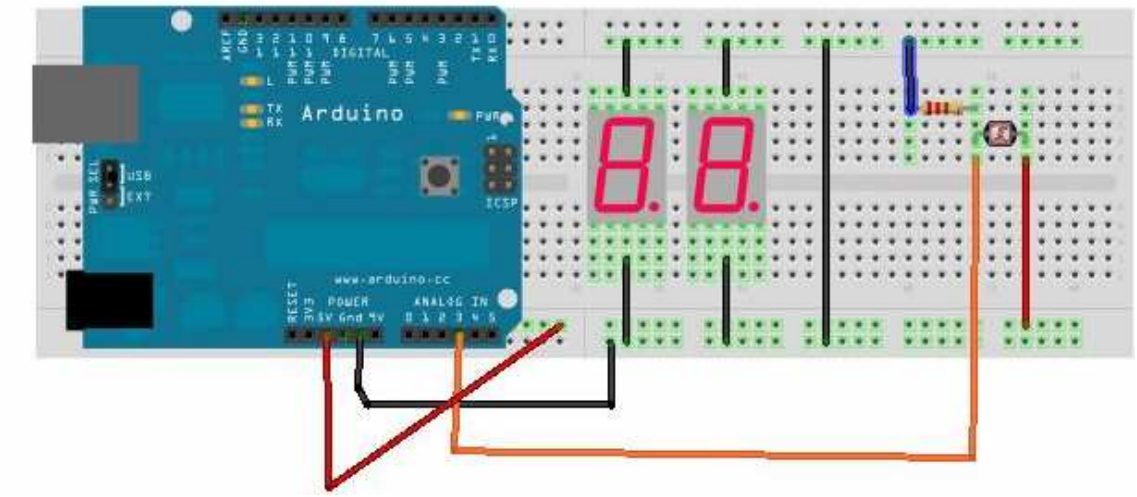
MONTAJE

Para introducir las variaciones de tensión en la entrada analógica (captura de la medida), se realiza el siguiente montaje:



Los displays se montarán exactamente igual que en la primera actividad

El montaje queda como sigue



EXPLICACIÓN DEL PROGRAMA

La etapa de los displays es la misma que en la actividad anterior

1. Definimos variables:

```
int pin_ldr = 3; // Patilla de arduino para la Medida del sensor
int lectura = 0; // Variable donde almacenamos la medida
int val = 0;  // Variable donde almacenamos el mapeo
int unidad = 0;  //variable donde se almacenará el dígito unidad de la medida mapeada
int decena = 0;     // variable donde se almacenará el dígito decena de la magnitud medida
```

2. Definimos en el setup lo mismo que en el caso de los displays

3. En el programa principal LOOP:

- Tomamos la medida del sensor y la introducimos en la variable lectura.
- Mapeamos ese valor y lo introducimos en la variable “val”
- Descomponemos el valor ”val” en dos dígitos mediante una función llamada “descompon” que ejecuta un algoritmo creado por mi.
- Según el caso se muestra el número en el display

```
            void loop()
{
lectura = analogRead(pin_ldr);// Tomamos la medida
//Serial.println(lectura);
//delay(1000);
val = map(lectura, 0, 1023, 0, 99); //Mapeamos entre esos valores
//Serial.println(val);
descompon(val); //Ejecutamos esta función para descomponer el valor en dos dígitos
//Serial.println(unidad);
//Serial.println(decena);
//delay(1000);
switch (unidad) {
   case 0:
     D1_num_0();
     break;
   case 1:
     D1_num_1();
     break;
   case 2:
     D1_num_2();
     break;
     case 3:
     D1_num_3();
     break;
     case 4:
     D1_num_4();
     break;
     case 5:
```

```
    D1_num_5();
    break;
    case 6:
    D1_num_6();
    break;
    case 7:
    D1_num_7();
    break;
    case 8:
    D1_num_8();
    break;
    case 9:
    D1_num_9();
    break;
   //default:
}
  switch (decena) {
   case 0:
    D2_num_0();
    break;
   case 1:
    D2_num_1();
    break;
   case 2:
    D2_num_2();
    break;
    case 3:
    D2_num_3();
    break;
    case 4:
    D2_num_4();
    break;
    case 5:
    D2_num_5();
    break;
    case 6:
```

```
    D2_num_6();
    break;
    case 7:
    D2_num_7();
    break;
    case 8:
    D2_num_8();
    break;
    case 9:
    D2_num_9();
    break;
   //default:
  }
 }
```

4. Función descompon(): El algoritmo va restando hasta dar con los número en dígitos

```
void descompon (int valor)
{
 if(10 - val > 0)
 {
   unidad = val;
   decena = 0;
 }
 else if(20 - val > 0)
  {
    unidad = val - 10;
    decena = 1;
  }
else if(30 - val > 0)
  {
    unidad = val - 20;
    decena = 2;
  }
else if(40 - val > 0)
  {
    unidad = val - 30;
    decena = 3;
```

```
 }
else if(50 - val > 0)
 {
   unidad = val - 40;
   decena = 4;
 }
 else if(60 - val > 0)
 {
   unidad = val - 50;
   decena = 5;
 }
else if(70 - val > 0)
 {
   unidad = val - 60;
   decena = 6;
 }
else if(80 - val > 0)
 {
   unidad = val - 70;
   decena = 7;
 }
else if(90 - val > 0)
 {
   unidad = val - 80;
   decena = 8;
 }
else
{
 unidad = val - 90;
 decena = 9;
}
```

Nota1: Algunos comandos de este programa están como comentarios, ya que interesan sólo si se quiere ver los valores por el monitor (en pantalla)