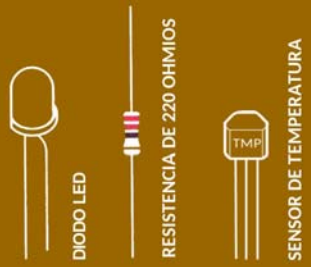
03



INGREDIENTES

MEDIDOR DE ENAMORAMIENTO

CONVIERTA SU PLACA ARDUINO EN UNA MÁQUINA QUE MIDE LO ENAMORADO QUE ESTÁ. USANDO UNA ENTRADA ANALÓGICA, VA A PODER REGISTRAR SU NIVEL DE "ENAMORAMIENTO"

Descubra: entrada analógica, usando el monitor serie

Tiempo: **45 MINUTOS**

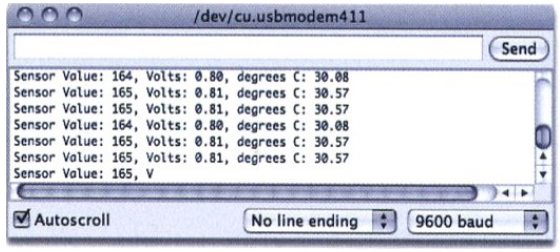
Nivel: **bajo**

Proyectos en los que se basa: **1,2**

Aunque los pulsadores e interruptores son útiles, existen muchas más cosas en el mundo físico que solo encender y apagar algo. Aunque Arduino es una herramienta digital, es posible adquirir con él información de sensores analógicos para medir parámetros físicos como la temperatura o el nivel de iluminación. Para poder hacerlo, Arduino dispone en su interior de un Convertidor Analógico – Digital (ADC), el cual transforma una señal analógica presente en su entrada en una señal digital. Las entradas analógicas de Arduino son los pins A0 – A5 las cuales pueden proporcionar un valor entre 0 y 1023, que equivale a un rango de 0 voltios a 5 voltios, por ejemplo, si la tensión de una de las entradas analógicas vale 2.5V el valor que proporciona el convertidor ADC vale 512.

Va a usar un *medidor de temperatura* para medir la temperatura de la piel. Este componente varía su tensión de salida dependiendo de la temperatura que detecta. Dispone de tres terminales: uno se conecta a masa, otro se conecta a la alimentación y el tercero produce una tensión de salida variable que se aplica a Arduino. En el sketch de este proyecto, se va a leer la tensión de salida del sensor y usarla para encender o apagar unos diodos LEDs, como indicadores de la temperatura de su piel (lo enamorado que está). Existen varios modelos de sensores de temperatura. Este modelo, el TMP36, es utilizado por que los cambios de la tensión de salida son directamente proporcionales a la temperatura en grados Celsius.

El IDE de Arduino incorpora una herramienta llamada *monitor serie* el cual le proporciona información de lo que el microcontrolador está haciendo. Utilizando el monitor serie, se puede conseguir información acerca del estado de sensores, y así tener una idea de lo que sucede en un circuito y en el código cuando se esta ejecutando.



Monitor serie

Figura 1

MONTANDO EL CIRCUITO

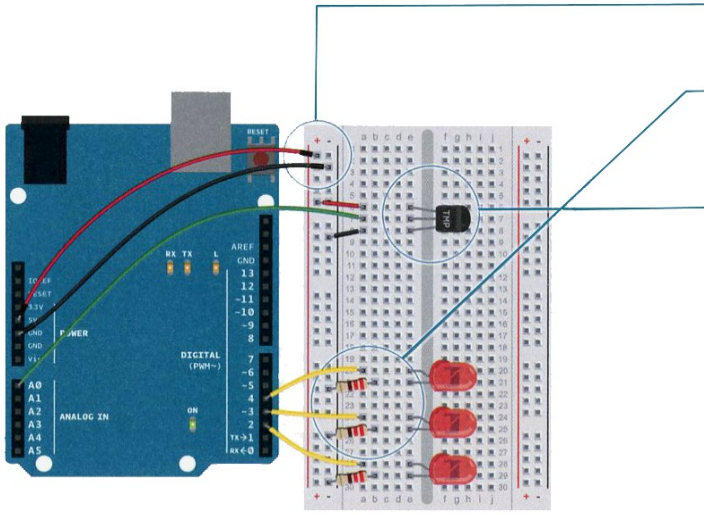


Figura 2

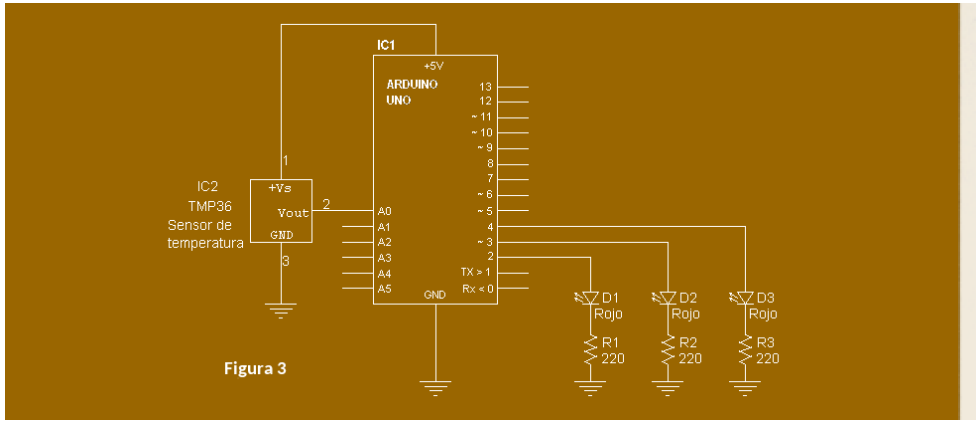


Figura 3

En este proyecto, necesita chequear la temperatura ambiente de la habitación antes de ejecutar el programa. Esto se puede comprobar ahora mismo manualmente, pero también se puede lleva a cabo a través de la calibración que proporciona Arduino, así no será tan complicado. Se puede usar un botón para establecer la temperatura mínima, o establecer dentro del programa este valor antes de que se ejecute el bucle loop() y usarlo como punto de referencia. El proyecto 6 entra en detalle sobre esto, o puede mirar el ejemplo de calibración que se muestra en la siguiente página de Arduino:

arduino.cc/calibration

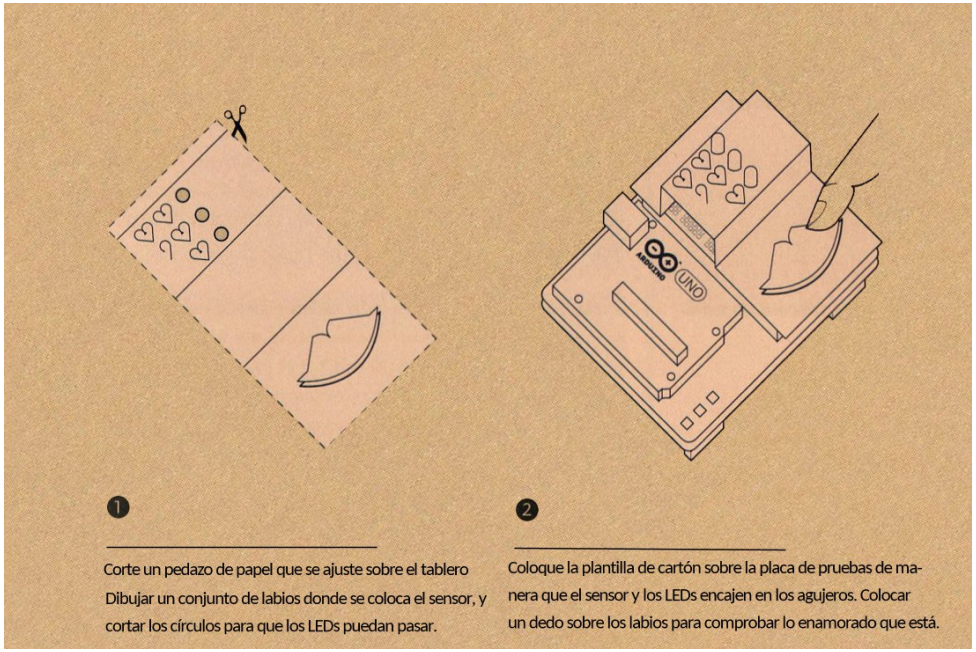
También es posible acceder a otro ejemplo parecido desde el IDE de Arduino dentro de Archivos > Ejemplos > 03Analog > Calibration

1. Tal como lo ha estado haciendo en proyectos anteriores, conectar la placa de pruebas a la alimentación y a masa.

2. Conectar el cátodo (terminal corto) de cada diodo LED a masa a través de una resistencia en serie de 220 ohmios. Conectar los ánodos de los LEDs a los terminales 2, 3 y 4 respectivamente. Estos diodos serán los indicadores de este proyecto.

3. Colocar el sensor de temperatura TMP36 sobre la placa de pruebas con la parte curvada de su cuerpo mirando hacía el lado contrario de la placa Arduino (la colocación de los terminales es importante) como se muestra en la figura 2. Conectar el terminal de la izquierda de la cara plana del cuerpo a la alimentación y el terminal derecho a masa. Conectar el pin central al pin A0 de la placa Arduino. Este pin es la entrada analógica 0.

Crear una interface para el sensor de manera que la gente pueda interactuar con él. Un cartón cortado adecuadamente puede servir perfectamente para esta aplicación. Para realizar un diseño original puede dibujar unos labios en la zona del sensor. También puede etiquetar los LEDs para darles algún significado. Por ejemplo, si solo se enciende un led puede indicar que no está enamorado, dos LEDs encendidos que le gusta alguien, y tres LEDS encendidos querrá decir que está muy enamorado.



EL CÓDIGO

Un par de constantes útiles

Las constantes son similares a las variables (guardar información) las cuales permiten usar un único nombre con los datos del programa, pero a diferencia de las variables su contenido no se puede cambiar. Definir una constante con un nombre para la entrada analógica que haga referencia al tipo de pin que se trata (**Pin_del_Sensor**), y definir otra constante con un nombre para la temperatura de referencia (**Temperatura_de_Referencia**). Para cada dos grados sobre esta temperatura de referencia un led se enciende, por ejemplo, si la temperatura de referencia es de 20 grados, con 22 grados se enciende el primer led, con 24 grados el segundo y así sucesivamente. En la lección anterior vimos el tipo de datos INT (página 36 segundo párrafo), usado en este programa para identificar cual es el pin analógico de entrada de Arduino en donde se conecta la salida del sensor de temperatura. Por otro lado el valor de la temperatura de referencia se guarda como una constante en coma flotante. Es tipo de número tiene un punto decimal, y es usado para números que se pueden ser expresados como fracciones o para expresar números con decimales.

```
1  const int Pin_del_Sensor = A0;
2  const float Temperatura_de_Referencia = 20.0;
```

Inicializar el puerto serie para establecer la velocidad de comunicación con el ordenador

Dentro del bloque de la configuración "setup()" vamos a usar un nuevo comando, **Serial.begin()**. Este comando permite una comunicación entre la tarjeta Arduino y el ordenador, de manera que pueda ver los valores que se leen en la entrada analógica en la pantalla de un ordenador.
El argumento 9600 de la instrucción Serial.begin establece la velocidad a la que Arduino se comunica con el ordenador, 9600 bits por segundo. Usará el monitor serie del IDE de Arduino para ver los datos que almacena una variable que ha escogido y que es enviada desde el microcontrolador de Arduino. Cuando abra el monitor serie del IDE verificar que la tasa de baudios (bits por segundo) es de 9600.

```
3  void setup() {
4     Serial.begin(9600); // Abrir el puerto serie
```

Definir cuales son las salidas de los pins digitales y ponerlos a cero

La siguiente instrucción **for()** define algunos de los pins como salidas digitales dentro de un bucle. Algunos de estos pins ya se han usado con los LEDs en el proyecto anterior. Ahora en lugar de escribir cada instrucción **pinMode()** para cada uno de estos pins, se utiliza la instrucción **for()**, la cual ejecuta un bucle definiendo las características de cada pin rápidamente. Se trata de una forma de ahorrar tiempo y líneas de código si es necesario definir las propiedades de muchos pins a la vez. La instrucción **for()** se ejecuta en un bucle donde se definen las características de los pins 2 a 4 consecutivamente, definiendo cada pin como una salida digital (**pinMode(NumeroPin, OUTPUT);**) y además en estado bajo (**digitalWrite(NumeroPin; LOW)**).

```
5     for(int NumeroPin=2; NumeroPin<5; NumeroPin++){
6       pinMode(NumeroPin,OUTPUT);
7       digitalWrite(NumeroPin,LOW);
8    }
9  }
```

Para ver un tutorial sobre la instrucción for() abrir este enlace:
arduino.cc/for

El siguiente bloque de código es el programa en sí, el cual comienza con la función **loop()**. Aquí se usa una variable llamada **Valor_del_Sensor** en donde se guarda la lectura del sensor. Para poder leer la información del sensor, se utiliza la instrucción **analogRead()** la cual toma un argumento: de que pin se va a tomar la información de lectura del sensor; siendo este argumento la constante **Pin_del_Sensor** que se define al comienzo del programa. El valor leído, el cual esta comprendido entre 0 y 1023, representa indirectamente la tensión que existe en ese pin (para "0" son 0 voltios y para "1023" son +5 voltios).

```
10 void loop(){
11    int Valor_del_Sensor = analogRead(Pin_del_Sensor);
```

Enviar los valores del sensor de temperatura al ordenador

La función **Serial.print()** enviá información desde Arduino al ordenador al que está conectado. Puede ver esta información en el monitor serie. Si dentro del argumento de **Serial.print()** escribe una palabra entre comillas, mostrará este texto en el monitor serie tal y como esta escrito, en este caso "**Valor del sensor:** ". Si le da una variable como argumento, el monitor serie mostrará el valor de esa variable, como se ve a continuación: **Serial.print(Valor_del_Sensor)**, el monitor serie podrá mostrar cualquier valor comprendido entre 0 y 1023.

```
12    Serial.print("Valor del sensor: ");
13    Serial.print(Valor_del_Sensor);
```

Convertir la lectura del sensor a tensión

Usando las matemáticas, es posible averiguar la tensión real que existe en el pin. La tensión será una valor entre 0 y 5 voltios, y podrá tener una parte fraccionaria (por ejemplo, podría ser 2.5 voltios), por eso será necesario almacenar este valor de tensión dentro de una variable con coma **flotante**. Se crea una variable llamada **Tension** en donde se guarda este número. Se divide el valor que almacena la variable **Valor_del_Sensor** por 1024 y se multiplica por 5. El resultado obtenido representa la tensión que existen en el pin.

```
// Convertir la lectura ADC a tensión
float Tension = (Valor_del_Sensor/1024.0) * 5.0 ;
```

Al igual que se hizo con el valor del sensor, se mostrará el valor de esta tensión dentro de la ventana del monitor serie.

```
Serial.print(", Voltios: ");
Serial.print(Tension);
```

Convertir la tensión a temperatura y enviar este valor al ordenador

Si examina la hoja de datos del sensor, se muestra información sobre el rango de la tensión de salida. Las hojas de datos son como los manuales de los componentes electrónicos. Son escritas por ingenieros, para otros ingenieros. En la hoja de datos de este sensor se indica que por cada 10 mili voltios de cambio en la tensión del sensor equivale a un cambio de 1 grado Celsiu de temperatura. También indica que el sensor puede leer temperaturas por debajo de los 0 grados. Debido a esto, es necesario crear un offset para valores por debajo de los cero grados. Si se obtiene la tensión, se le resta 0.5 y se multiplica por 100, y así se obtiene la temperatura exacta en grados Celsius. Se almacena este valor dentro de una variable con coma flotante llamada **Temperatura**.

Una vez leída la temperatura real del sensor también se puede mostrar este valor en la ventana del monitor serie. Puesto que la variable de temperatura es lo último que se va a "imprimir" en el monitor serie dentro de este bucle, se va a usar un comando ligeramente diferente: **Serial.println()**. Este comando crea una nueva línea en el monitor serie después de enviar el valor que se va a mostrar. De esta manera se pueden ver mejor los resultados de las medidas al mostrarse los datos en líneas independientes.

```
Serial.print(", grados C: ");
// Convertir la tensión en temperatura en valores en grados
float Temperatura = (Tension - 0.5) * 100;
Serial.println(Temperatura);
```

Apagar los LEDs para una temperatura baja

Con la temperatura real, ahora se puede configurar la instrucción **if()...else** para encender los LEDs. Utilizando como punto de partida la temperatura de referencia, se encenderá un LED por cada dos grados que se incremente la temperatura sobre la temperatura de referencia. Se va a buscar un rango de valores mientras se mueve a través de la escala de temperatura, esto quiere decir que el segundo LED deberá de encenderse si la temperatura está cuatro grados por encima de la temperatura de referencia, y el tercer LED se encenderá si está sobre seis grados.

```
if(Temperatura < Temperatura_de_Referencia){
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
```


Hojas de datos de los componentes electrónicos del Kit de Iniciación

arduino.cc/kitdatasheets

Encender un LED para una temperatura baja

El operador && significa "Y" en una sentencia lógica, equivale a la multiplicación. Se utiliza para verificar múltiples condiciones: "Si la temperatura es 2 grados mayor que la temperatura de referencia y si es menor que 4 grados por encima de dicha temperatura de referencia."

Encender dos LEDs para una temperatura media

Si la temperatura está entre dos y cuatro grados sobre la temperatura de referencia, este bloque de código también enciende el LED conectado al pin 3.

Encender tres LEDs para una temperatura alta

Apagar los LEDs para una temperatura baja

El convertidor analógico digital puede leer demasiado rápido, así que hay que introducir una pequeña pausa al final del programa (lazo loop()). Si no existiese esta pausa se producirán errores en los valores que se leen.

USARLO

Una vez cargado el programa dentro de la tarjeta Arduino, hacer click sobre el icono del monitor serie. Debe de ver que aparece un listado de valores en el siguiente formato: **Valor del sensor: 200, Voltios; .70, grados C:17**

Coloque sus dedos sobre el sensor mientras está colocado sobre la placa de pruebas y ver que valores aparecen en el monitor serie. Tomar nota de la temperatura que el sensor detecta cuando está al aire.

Cerrar el monitor serie y cambiar el valor de la constante de la temperatura de referencia en el programa al mismo valor que se muestra en el monitor serie cuando el sensor está al aire. Cargar el código de nuevo a la tarjeta Arduino y a continuación poner los dedos sobre el sensor. Como la temperatura aumenta al colocar los dedos, debe de ver como los LEDs se van encendiendo uno a uno. ¡Felicitaciones, el proyecto funciona!,


```
26    }else if(Temperatura >= Temperatura_de_Referencia+2 &&
       Temperatura < Temperatura_de_Referencia+4){
27        digitalWrite(2, HIGH);
28        digitalWrite(3, LOW);
29        digitalWrite(4, LOW);


30    }else if(Temperatura >= Temperatura_de_Referencia+4 &&
       Temperatura < Temperatura_de_Referencia+6){
31        digitalWrite(2, HIGH);
32        digitalWrite(3, HIGH);
33        digitalWrite(4, LOW);


34    }else if(Temperatura >= Temperatura_de_Referencia+6){
35        digitalWrite(2, HIGH);
36        digitalWrite(3, HIGH);
37        digitalWrite(4, HIGH);


38    }
39        delay(100);
40  }
```

Crear un interface para que dos personas puedan comprobar la compatibilidad entre ambos. Debe de decidir de que tipo de compatibilidad se trata, y como actuarán sobre el sensor. ¿Quizás deberán de juntar sus manos para generar el suficiente calor? ¿Tal vez tengan que abrazarse? ¿Que piensa?

Aumentando los tipos de entradas que puede leer y usando la instrucción analogRead() junto con el monitor serie ha podido ver lo que ocurre dentro de su Arduino. Ahora ya puede usar un mayor número de sensores analógicos y entradas.