Práctica 1. Medida de Temperatura con Sensor LM35

Àlex Benítez, Xavier Escribà, Pau Gabarrell

27 de Abril de 2017

220053–Avionica

1. Sensor y acondicionador

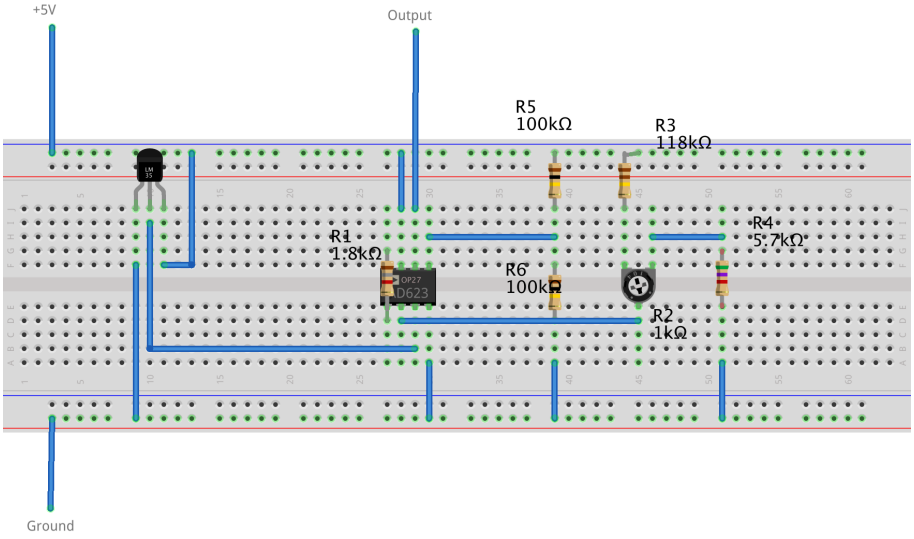
En el primer apartado de la práctica se medirá los valores suministrados por el LM35 y se montará y ajustará el acondicionador de señal en una protoboard.

1.1. Estimar el valor de la temperatura ambiente (T_{amb}) y la temperatura apretando el sensor con los dedos (T_{max}) conectando

la salida del LM35 directamente a un multímetro y aplicando la expresión de la sensibilidad dada en las características del LM35

$$\begin{cases} T_{amb} = & 23\,^oC \\ T_{max} = & 30\,^oC \end{cases}$$

1.2. Montar correctamente el acondicionador en la protoboard



1.3. Ajustar y calibrar el acondicionador para que la salida valga 2.5V a T_{amb} ¿Cuánto vale la salida a la temperatura T_{max}?

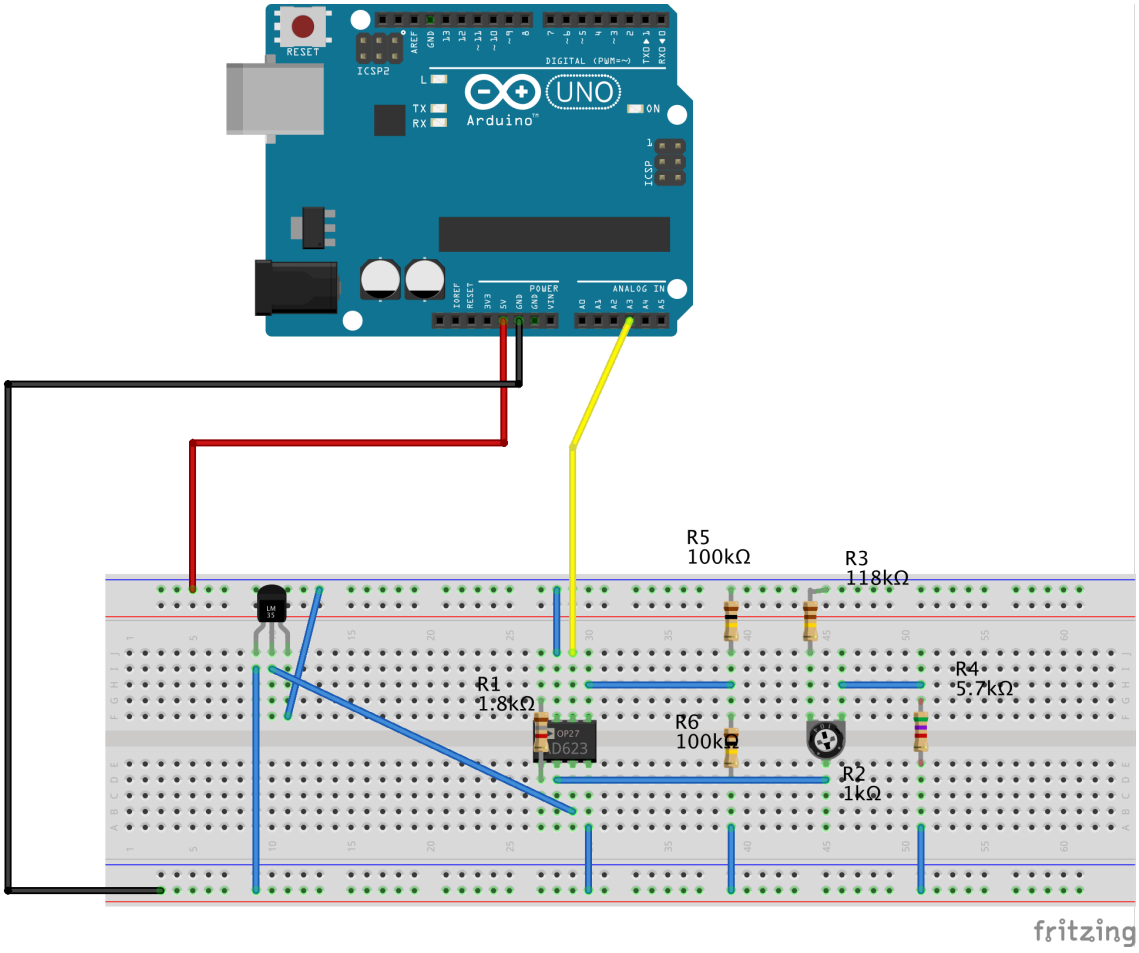
$$Salida = 4V$$

Hay que destacar que en el divisor de tensión se conseguían solamente 2.1V, suponemos que debido a la tolerancia de las resistencias, por dicho motivo no se ha alcanzado un valor superior en la salida.

2. Adquisición de datos con Arduino

En este apartado se montará el sistema completo en una protoboard y se hará un primer programa sencillo de adquisición de datos de temperatura.

2.1. Montar correctamente en la protoboard el conjunto sensor-acondicionador conectado al microcontrolador Arduino UNO.



2.2. Programar en el Arduino una adquisición continua de datos procedentes del acondicionador usando la función analogRead.

```
1  int pinanalog = 3;
2  double analog_data = 0;
3  double temp = 0;
4
5  void setup () {
6    Serial.begin(9600);
7  }
8
9  void loop () {
10   analog_data = analogRead(pinanalog);
11   temp = (analog_data*0.013)+17.027;
12   Serial.println(temp);
13 }
```

El código usado contiene en primer lugar una declaración de las variables globales que van a ser usadas por el programa, dichas variables son el pin que usaremos para el **analogRead**, el **analog_data** que será la variable en la que guardaremos el valor leído por el **analogRead** y finalmente **temp**, la variable en la que se almacenara la conversión del **analog_data** para mostrar grados centígrados. En el **setup** simplemente se ha iniciado el **serial port** para poder ver que valores de temperatura está leyendo del **analogRead**. Dentro del **loop** del programa se realiza lo antes descrito, primero se obtiene el valor leído por el **ADC** (analog to digital converter), se opera dicho valor para tenerlo en grados centígrados y finalmente se muestra por el **serial port**.

2.3. Anotar los valores enteros obtenidos de la función analogRead a T_{amb} y a T_{max}.

Para realizar este apartado se ha modificado la línea 12 del código por esta:

```
Serial.println(analog_data);
```

De esta forma se ha obtenido el valor que nos piden en este apartado.

$$\begin{cases} analogRead\,a\,T_{amb} = & 430 \\ analogRead\,a\,T_{max} = & 800 \end{cases}$$

Podemos considerar estos valores como correctos ya que si hacemos el factor de conversión del voltaje a T_{amb} y T_{max} a la lectura del **AnalogInput**, sabemos que el voltaje máximo es de $5\,V$ que corresponden a la lectura de 1023 del Arduino, por lo tanto, al voltaje de T_{max} de $4\,V$ obtenemos una lectura de $4{\cdot}1023/5 = 818.4$ y al voltaje de Tamb de $2.1\,V$ obtenemos una lectura de $2.1{\cdot}1023/5 = 429.66$, que son prácticamente iguales a los obtenidos experimentalmente.

2.4. Ampliar el programa para visualizar los valores correctos de Tamb y Tmax.

El anterior código mostrado ya tenía la implementación que nos piden en este apartado.

$$\begin{cases} T_{amb} = & 22.6\,^oC \\ T_{max} = & 27.48\,^oC \end{cases}$$

3. Temporización de la visualización

En este apartado se trata de reajustar el programa para que actualice la temperatura cada 0.5 segundos, en lugar de estar actualizando continuamente.

3.1. Utilizar la función delay del Arduino para que la temperatura se actualice cada 0.5 s.

Para implementar lo que se nos pide simplemente hay que añadir una linea de código, resultando en:

```
int pinanalog = 3;
double analog_data = 0;
double temp = 0;

void setup () {
  Serial.begin(9600);
}

void loop () {
  analog_data = analogRead(pinanalog);
  temp = (analog_data*0.013)+17.027;
  Serial.println(temp);
  delay(500);
}
```

A la función **delay** hay que decirle en milisegundos el tiempo de espera que se quiere, por esto se le ha puesto 500 de valor.

4. Alarmas de temperatura

En este último apartado se modificará el programa para que se encienda un led cuando se alcance las temperaturas extremas T_{amb} y T_{max}. Se debe visualizar claramente como se enciende al alcanzar estas temperaturas y como se apaga al salir de ellas (tener en cuenta que el paso de temperatura ambiente hasta la máxima, apretando el sensor, y viceversa, no es inmediato, por lo que debe ser posible ver el transitorio).

4.1. Buscar información para averiguar con cuál de los pines digitales del Arduino UNO se puede encender el led L que hay en la placa Arduino.

De la documentación oficial disponible en *www.Arduino.cc* se ha obtenido que el pin para controlar el LED de la placa es:

$$Pin = 13$$

4.2. Programar el Arduino para encender el led L cuando la temperatura es mayor o igual que T_{max} y cuando la temperatura es menor o igual que T_{amb}. Dejar un margen adecuado tanto en el máximo como en el mínimo para que se pueda apreciar con claridad el encendido y el apagado. Para este programa se deben usar las funciones pinMode y digitalWrite.

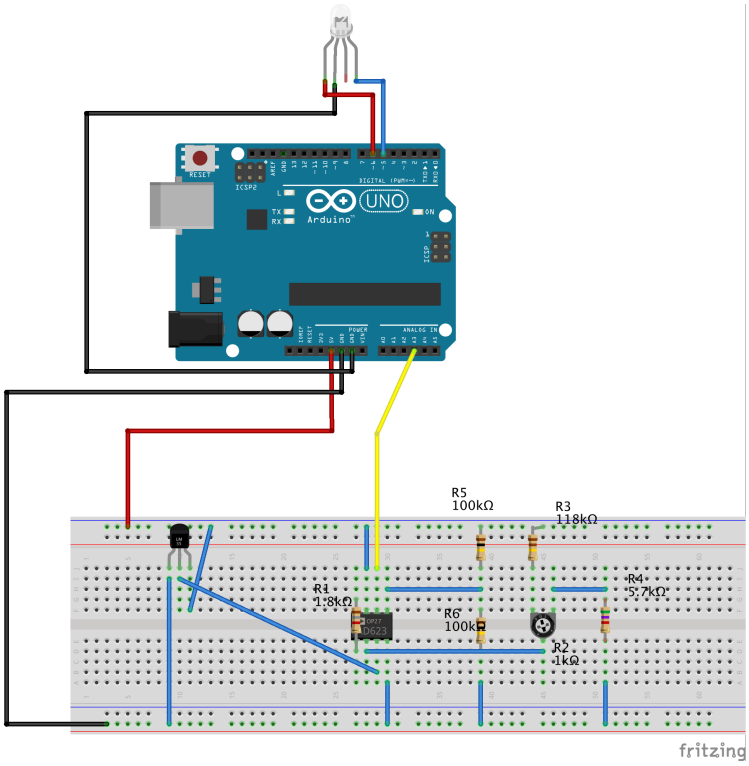
```
int led = 13;
double analog_data = 0;
double temp = 0;
double control_led;

void setup () {
```

```
7    Serial.begin(9600);
8    pinMode(led, OUTPUT);
9  }
10
11 void loop () {
12   analog_data = analogRead(3);
13   temp = (analog_data*0.013)+17.027;
14   delay(500);
15   if (temp > 27) {digitalWrite(led, 1);}
16   else if (temp < 23.5) {digitalWrite(led, 1);}
17   else {digitalWrite(led, 0);}
18   Serial.println(temp);
19 }
```

Para realizar este apartado se ha procedido igual que en los anteriores para calcular la temperatura detectada por el sensor. Para el accionado del control del LED se ha usado una combinación de **if**, **else if** y **else**, se ha hecho que a partir de cierta temperatura se encienda el led, que por debajo de cierta temperatura se encienda el led, y que cuando esta en el medio se apague. Para calibrar dicha función se han usado los valores de 23.5 i 27 grados centígrados de acuerdo con la temperatura existente al laboratorio en el momento de la realización de la práctica.

5. Ampliación

Para la ampliación de la práctica se propone el uso de un LED RGB, con dicho LED se mostrará de color rojo cuando se sobrepasa una temperatura umbral, en este caso 27 grados; dicho LED se mostrará de color azul al bajar de 23.5 grados. Para crear un efecto de transición se han usado los pins **PWM** del Arduino para poder controlar la intensidad del LED, se ha usado la función **map** para hacer que de 25 a 27 grados el LED rojo se encienda progresivamente, alcanzando el máximo a los 27 grados, se ha echo lo mismo para el LED azul, haciendo que al bajar de 25 grados progresivamente se va aumentando el brillo del LED azul, alcanzando el máximo a los 23.5 grados. Las conexiones necesarias que hay que realizar son las siguientes:



El código necesario para dicha aplicación es el siguiente:

```
int blue = 5;
int red = 6;
int red_value = 0;
int blue_value = 0;
double analog_data = 0;
double temp = 0;
double control_led;

void setup () {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}

void loop () {
  analog_data = analogRead(3);
  temp = (analog_data*0.013)+17.027;
  delay(500);
  blue_value = map(temp, 25, 23.5, 0, 255);
  red_value = map(temp, 25, 27, 0, 255);
  if (temp > 25) {analogWrite(red, red_value);}
  else if (temp < 25) {analogWrite(blue, blue_value);}
  Serial.println(temp);
}
```