Sistemas de Tempo Real: Microcontroladores
2013/14

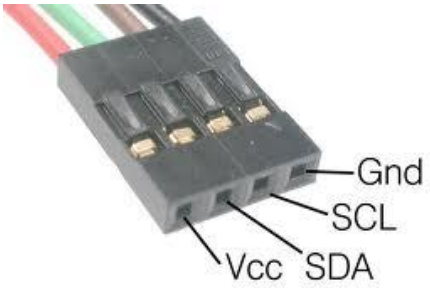
Práctica 3: RTC

Objetivo

Aprender a utilizar periféricos que hagan uso del bus I2C y gestionar el funcionamiento de un reloj no volátil de alta precisión.

RTC DS1307

El DS1307 es un RTC (*Real-Time clock*) de Dallas (datasheet) que se comunica con el Arduino a través del bus I2C. Este tipo de bus es capaz de llevar a cabo comunicaciones serie entre múltiples dispositivos conectados a un bus común (hasta 112 a la vez) a unas velocidades relativamente lentas. Estas velocidades son más que suficientes para la mayor parte de los sensores y permiten que el bus sea muy sencillo de implementar, requiriendo únicamente de cuatro líneas: VCC, GND, SCL (línea de reloj) y SDA (línea de datos).



Arduino posee una librería que soporta comunicaciones I2C (Wire), la cual facilita enormemente las comunicaciones. La librería Wire utiliza el pin analógico 4 del Arduino como SDA y el pin analógico 5 como SCL. En muchas implementaciones será necesario utilizar resistencias adicionales de pull-up para estos dos pines, pero para este RTC es suficiente con utilizar las pull-ups internas de los pines analógicos del micro[1]. Para activar las resistencias internas de pull-up, es suficiente con incluir las siguientes dos líneas en la función de *setup()*[2]:

[1] **La versión proveída del DS1307 ya posee las resistencias de pull-up soldadas en el PCB, no es necesario activar las resistencias internas.**

[2] Hay más información sobre estos pasos en este enlace.

```
void setup()
{
  // se configura el pin como entrada
  pinMode(PIN,INPUT);
  // se activa la resistencia interna de pull-up de 20k de PIN
  digitalWrite(PIN,HIGH);
}
```

A la hora de gestionar el RTC a través del I2C, nos interesan dos funciones básicamente: una que permita establecer la fecha/hora y otra que permita obtenerlas. El siguiente código expone el funcionamiento básico del bus I2C y cómo comunicarse con el DS1307:

```
#include "Wire.h"
#define DS1307_I2C_ADDRESS 0x68

// ****************************************************
// La siguiente función realiza las siguientes tareas:
//  * Configura la fecha y hora
//  * Inicia el conteo del RTC
//  * Configura la hora en modo 24h
//  * OJO! La función no verifica los rangos de entrada
// ****************************************************
void setDate(byte segundo,        // 0-59
             byte minuto,        // 0-59
             byte hora,          // 1-23
             byte diaSemana,     // 1-7
             byte diaMes,    // 1-28/29/30/31
             byte mes,         // 1-12
             byte anho)          // 0-99
{
   Wire.beginTransmission(DS1307_I2C_ADDRESS);
   Wire.write(0);
   Wire.write(decToBcd(segundo));    // 0 to bit 7 starts the clock
   Wire.write(decToBcd(minuto));
   Wire.write(decToBcd(hora));      // If you want 12 hour am/pm you need to set
                                   // bit 6 (also need to change readDateDs1307)
   Wire.write(decToBcd(diaSemana));
   Wire.write(decToBcd(diaMes));
   Wire.write(decToBcd(mes));
   Wire.write(decToBcd(anho));
   Wire.endTransmission();
}
```

```
// ************************************
// Obtiene la fecha y hora del DS1307
// ************************************
void getDate(byte *segundo,
             byte *minuto,
             byte *hora,
             byte *diaSemana,
             byte *diaMes,
             byte *mes,
             byte *anho)
{
  Wire.beginTransmission(DS1307_I2C_ADDRESS);
  Wire.write(0);
  Wire.endTransmission();

  Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

  *segundo     = bcdToDec(Wire.read() & 0x7f);
  *minuto      = bcdToDec(Wire.read());
  *hora        = bcdToDec(Wire.read() & 0x3f);
  *diaSemana   = bcdToDec(Wire.read());
  *diaMes      = bcdToDec(Wire.read());
  *mes         = bcdToDec(Wire.read());
  *anho        = bcdToDec(Wire.read());
}
```

Se requiere el uso de las funciones *bcdToDec()* y *decToBcd()* para convertir de bcd a decimal, dado que el DS1307 devuelve los datos en formato BCD. Este formato representa cada dígito decimal mediante 4 bits en binario:

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Por ejemplo, si quisiésemos representar el valor 371 en BCD, tendríamos la secuencia 0011 0111 0001.

Adicionalmente a estas funciones, hay que tener en cuenta que en la función *setup()*, antes de usar el puerto I2C, será necesario inicializarlo (únicamente hay que hacer *Wire.begin()*). Igualmente, será preciso inicializar el RTC con los valores (i.e segundos, minutos, etc…) en que se desee que se comience

a contar. Eso sí: **hay que tener en cuenta que la llamada a *setDate()* sólo será necesaria una vez para poner en hora el RTC, pero después debería de ser comentada, dado que el RTC mantiene (gracias a la pila) la hora a pesar de que se retire la alimentación procedente del Arduino.**

Enunciado de la práctica

Haciendo uso del RTC y de la estación meteorológica sencilla desarrollada en la práctica anterior, se implementará una funcionalidad adicional: cada vez que se almacene un dato de los sensores, se almacenará a continuación el *timestamp* asociado. En cuanto se envíe el comando 'D' a través del puerto serie, se devolverá la temperatura, la humedad y el instante en que fueron capturadas. Para ello, además del código fuente expuesto anteriormente, será necesario implementar las funciones *bcdToDec*() y *decToBcd*().

Con todo el trabajo desarrollado, crear una memoria explicativa y enviarla a tiago.fernandez@udc.es acompañada del código fuente comentado.