ARDUINO Y LOS RELOJES RTC

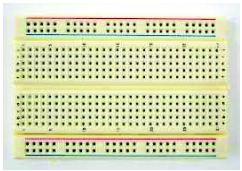
Usando un reloj integrado y batería de respaldo

OBJETIVOS

.

- Presentar los RTC Real Time Clocks.
- Montar un circuito con un Tiny RTC.
- Presentar las librerías complemento de Time, llamadas DS1307RTC y TimeAlarms.
- Argumentos __DATE__ y __TIME__

MATERIAL REQUERIDO.

	Arduino UNO o similar.
 	Protoboard y algunos cables macho hembra.
	Un reloj digital TinyRTC con respaldo de batería

LOS RELOJES DE TIEMPO REAL O RTCS

En la sesión anterior, presentamos la nueva librería Time para Arduino. Empezamos utilizando el reloj interno para los ejemplos, porque es una opción que siempre está disponible y además es de precio imbatible.

Pero dijimos que tenía varios problemas evidentes, como eran el hecho de que el reloj interno se reinicia cada 50 días aproximadamente, y además si cortamos la corriente a nuestro Arduino, pone el reloj a 00 y vuelve a empezar desde el 1 de enero de 1970, lo que no mucho.

Necesitamos una solución, que nos permita usar un **reloj fiable** que se mantenga incluso cuando apagamos Arduino. ¿Y qué ocurre cuando tenemos que resolver un problema electrónico? Pues que hay un fabricante, o varios, que nos suministran un chip para resolverlo.

Uno muy extendido para reloj digital es el **DS1307** de Maxim. Por muy poco dinero realiza todas las funciones para llevar un reloj con calendario y solo hay que añadirle una batería externa para que no pierda la hora al perder la corriente. Sus características son:

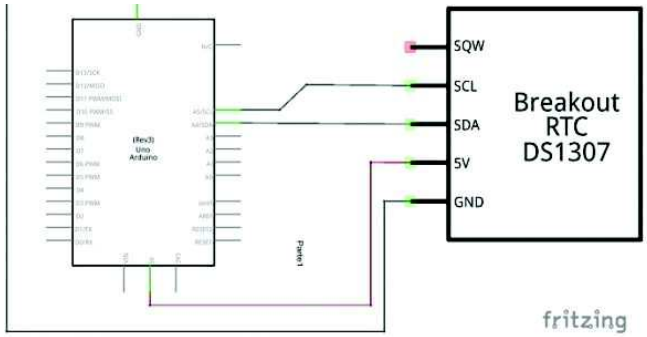
- RTC: Real Time Clock, o reloj de tiempo real. Que lleva la cuenta de segundos minutos y horas además de día mes y año automáticamente, válido hasta el año 2100.
- 56 byte de memoria RAM respaldada por una batería exterior, que mantiene la fecha y hora cuando no hay corriente.
- Detección automática de corte de corriente y cambio a modo batería.
- Muy bajo consumo, lo que ayuda a que la batería dure entre 5 y 10 años.
- I2C integrado en el mismo chip.

Además, es bastante fácil conseguir un reloj con batería montado en una placa y listo para usarse en nuestros proyectos, por unos pocos euros.

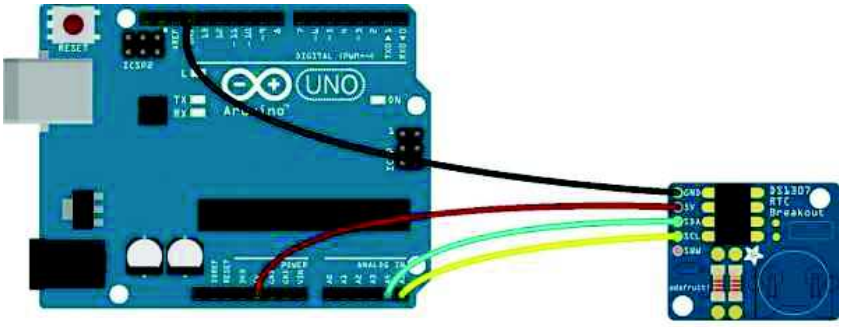
En esta sesión vamos a usar uno de estas placas, la Tiny **RTC**, con el **DS1307**, montado en un soporte de conexión, con su cristal, una batería de respaldo, y bus I2C, con lo que solo tendremos que conectar unos pocos cables y podremos empezar a jugar con el tema.

DIAGRAMA DE CONEXIÓN

De nuevo la conexión es trivial. Simplemente conectar SDA a Arduino A4 y SCL a A5.



Y para la protoboard:



El pin SQW, es una salida digital del chip DS1307, que nos permite generar una señal cuadrada de frecuencia programable(De1Hz, 4kHz, 8kHz,32kHz) que podemos emplear como base de tiempos para el reloj de cualquier circuito sin necesidad de un cristal adicional de cuarzo y circuito oscilador (Que ya lleva el propio Tiny RTC)..

Por eso, en este caso, como no vamos a usarlo, lo dejamos sin conectar tranquilamente, después de comprobar en la hoja de normas que podemos hacerlo sin problemas.

EL PROGRAMA DE CONTROL

En la sesión anterior descargamos e instalamos la nueva librería Time en nuestro IDE. La ventaja de utilizar una librería coherente es que todas las funciones que vimos en la sesión previa siguen siendo válidas y se usan igual aunque tengamos ahora un reloj diferente.

Basta con que instalemos el modulo correspondiente al nuevo reloj DS1307. Y para ello descargamos e instalamos una nueva librería aquí Libreria DS1307RTC,

Como lo normal cuando instalas un RTC, es que no tenga la hora ajustada, vamos a empezar por ahí. Vamos a ver si el chip ha sido puesto en hora:

```
#include <Time.h>
#include <Wire.h>
#include <DS1307RTC.h>  // La libreria que gestiona el DS1307 RTC para Time

void setup()
   {    Serial.begin(9600);
        while (!Serial) ;  // Solo si usas un Arduino Leonardo
        setSyncProvider(RTC.get);     // Sincronizar con el RTC
        if(timeStatus()!= timeSet)
             Serial.println("Unable to sync with the RTC");
        else
             Serial.println("RTC has set the system time");
   }
```

Las tres primeras líneas cargan las librerías necesarias. Las dos primeras ya las usamos en la sesión anterior y la correspondiente al DS1307 se debe al RTC que vamos a usar. Fíjate en la línea:

```
    setSyncProvider(RTC.get);     // Sincronizar con el RTC
```

Las librerías están escritas de forma que usando setSyncProvider() podemos definir el modelo de reloj que vamos a usar y que todo el resto del programa sea el mismo, independientemente de la base del reloj. No esta mal, es la ventaja de usar unas librerías coherentes.

En el setup hacemos uso de timeStatus () para saber si el reloj está o no en hora. En caso de que no esté en hora podemos añadir al setup una línea más con setTime () para ajustar el reloj. En mi caso he añadido:

```
setTime(21,46,00,8,11,2014); // Las 21:45:00 del dia 8 de Nov de 2014
```

Reemplaza la hora por el valor que corresponda y corre el programa para sincronizar el reloj.

- *Asegurate de que corres el setTime una única vez, porque si lo haces repetidamente volverás una y otra vez a poner la misma hora en el reloj interno del DS1307.*
- *Cuando hayas ejecutado el programa y la hora sea correcta, simplemente convierte en comentario la línea del setTime.*

Podemos ya escribir un programa completo que nos de la fecha y la hora en la puerta serie. **PROG_55_1**

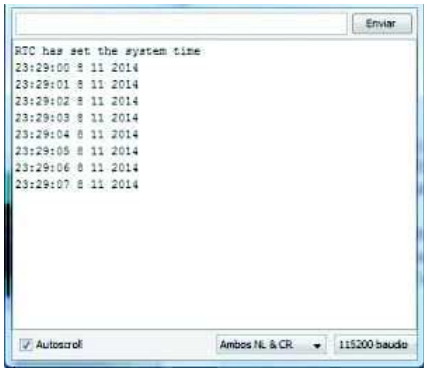
```
#include <Time.h>
#include <Wire.h>
#include <DS1307RTC.h>  // a basic DS1307 library that returns time as a time_t

void setup()
   {   Serial.begin(115200);
       while (!Serial) ;              // Solo para el Leonardo
       setSyncProvider(RTC.get);      // Vamos a usar el RTC

       setTime(21,46,00,8,11,2014); // Las 21:45:00 del dia 8 de Nov de 2014

       if (timeStatus() != timeSet)
           Serial.println("Unable to sync with the RTC");
       else
           Serial.println("RTC has set the system time");
   }
void loop()
   {     digitalClockDisplay();
         delay(1000);
   }
void digitalClockDisplay()
   {     Serial.print(hour());
         printDigits(minute());
         printDigits(second());
         Serial.print(" ");
         Serial.print(day());
         Serial.print(" ");
         Serial.print(month());
         Serial.print(" ");
         Serial.print(year());
         Serial.println();
   }
void printDigits(int digits)
   {     Serial.print(":");
         if(digits < 10)
            Serial.print('0');
         Serial.print(digits);
   }
```

La función printDigits, se asegura de que siempre haya dos dígitos en los números correspondientes, añadiendo un cero por delante cuando el número es menor que 10, con lo que tendremos un bonito calendario en la puerta serie.



USANDO LA FECHA Y HORA DEL COMPILADOR.

Si estás haciendo pruebas con la librería a deshoras, es frecuente que tengas que recompilar una y otra vez tu código, y si cada vez tienes que poner en hora a mano el reloj, lo normal es que acabes dando alaridos de frustración con el asunto (Si fueras alguien tranquilo es poco probable que estuvierasleyendo esto, tendrías un oficio sensato como contable, o abogado de un ayuntamiento).

Pensando en tus vecinos, el compilador de Arduino, dispone de un par de argumentos que nos informan de la fecha y hora de la última compilación: __DATE__ y __TIME__.

Podemos aprovechar esto para poner en hora el reloj interno corriendo un programita que convierta estos valores a formato fecha time_t y asignarlo al valor del reloj interno.

De este modo, aunque arrastraremos un par de segundos de error, asignaremos una hora razonablemente buena al reloj, sin necesidad de estar recompilando continuamente para ponerlo en hora.

Basta con que corráis este programa y la magia se encargara del resto. Los que no creáis en la magia podeis echar una ojeada al programa, porque aunque contiene algunas instrucciones de C++ que no hemos visto no tendréis mucho problema en seguirlo.

Aquí tenéis el programa de Ajuste automatico de fecha y hora

LIBRERÍA TIMEALARMS

Existe una librería más, relacionada con la **librería Time**, llamada **TimeAlarms** que nos permite definir alarmas en nuestro programa, muy al estilo de la que podemos fijar en un reloj despertador.Cuando una alarma dispara, llama a una función específica que se ejecuta en el momento preestablecido.

Podemos descargar la librería TimeAlarms de aquí TimeAlarms.zip

Dispone de varias funciones que vamos a enumerar aquí:

FUNCIÓN	DESCRIPCIÓN
Alarm.alarmRepeat(hours, minutes, seconds, function);	Crea una alarma que se repite cada día a la hora especificada y ejecuta la función
Alarm.alarmRepeat(dayofweek, hours, minutes, seconds, function);	Crea una alarma que dispara solo el día de la semana especificada a la hora prevista
Alarm.alarmOnce(hours, minutes, seconds, function);	Dispara mañana una única vez a la hora fijada y ejecuta la función
	Dispara una única alarma el próximo día de la semana que le digamos y

Alarm.alarmOnce(dayofweek, hours, minutes, seconds, function); .	ejecuta la función
Alarm.timerRepeat(seconds, function);	Llamará a la función especificada cada tantos segundos
.timerOnce(seconds, function);	Llamará una única vez a la función al cabo de los segundos especificados

Es una librería interesante si tenéis que repetir tareas periódicamente (Como enviar datos a algún sitio) y además nos ofrece uno timers cómodos que se ejecutan de forma programada, sin necesidad de definir interrupciones que podrían interferir con otros funciones.

- La librería TimeAlarm no emplea interrupciones, sino que se apoya en la gestión interna de Time,
- TimeAlarm necesita ningún hardware o RTC especifico, solo requiere que la librería Time esté cargada.
- Se pueden definir hasta 6 alarmas, cambiando el header de TiemeAlarm.h
- No se pueden definir intervalos menores de 1 segundo. Para periodos inferiores se requiere usar una interrupción programada o Timer (que veremos en una sesión posterior)

RESUMEN DE LA SESIÓN

- Hemos visto cómo usar un RTC con la librería Time. No hay diferencias con usar el reloj interno de Arduino más allá de definir el setSyncProvider ().
- Vimos algún ejemplo de cómo poner el reloj en hora con setTime y usando los argumentos del compilador __TIME__ y __DATE__.
- Presentamos además la librería de TimeAlrms para disparar funciones en alarmas programadas.

ANTERIOR SIGUIENTE

(107) COMMENTS

Luis *18 Ene 2015*

Hola, he visto tu post y me ha servido para corregir los 3 minutos que se atrasaba mi juguete arduino.
Viendo lo de la alarma me ha surgido una pregunta, si yo esa alarma la activara con un rele y por ejemplo el lunes hiciera se activara a las 15:00 y se apagara a las 16:00, que codigo tendria que utilizar para que ese rele se active cada dia un minuto antes y se apague un minuto despues, hasta estar activada 60 minutos ams que el primer dia en 30 dias, no se si me explique bien. Saludos y muchas gracias

Reply

admin *18 Ene 2015*

Hola Luis
Me temo que no te he entendido. Si puedes definir el problema con mas claridad te ayudare encantado

Un saludo

Reply

Bryan *04 Mar 2015*