LA NUEVA LIBRERÍA TIME EN ARDUINO

Midiendo y sincronizando el tiempo

OBJETIVOS

.

- Mostrar la necesidad de gestionar un calendario con fecha y hora.
- Presentar la nueva **librería Arduino Time**.
- Funciones básicas y tipos propios.
- La necesidad de la sincronización.

MATERIAL REQUERIDO.

	**Arduino UNO o equivalente**.

LA MEDICIÓN DEL TIEMPO

La medida del tiempo exacto es una de esas maravillas de la tecnología moderna a la que no prestamos atención, y que si le dedicáis una búsqueda en Internet os sorprenderá por el nivel de precisión y precio que se ha alcanzado.

Pero disponer de un reloj razonablemente preciso es un lujo, que solamente hace alrededor de 150 años que disponemos (Y solo para ricos). La democratización de la medida del tiempo es bastante más reciente.

Desde la antigüedad se han utilizado clepsidras de agua y arena para medir el tiempo, pero la verdad es que eran un asco (*Hacía falta un esclavo para rellenar el reloj de agua o arena periódicamente*). Ya en la edad media aparecen los primeros relojes mecánicos en Europa para campanarios, nada de llevar uno en el bolsillo, pues daban las horas en campanas de 2 toneladas.

El primer sistema fiable, para medir el tiempo nació con el reloj de escape en el siglo XVIII, por un concurso de la marina inglesa, que necesitaba un

El primer sistema fiable, para medir el tiempo fiable con el reloj se acaba en el siglo XVIII, por un concurso de la marina inglesa, que necesitaba un sistema de conocer la longitud, a la que se encontraban sus barcos al expandir el imperio británico por el Pacifico. Su solución fue un factor

y se resolvía con un sextante, pero la longitud era un tema peliagudo para los marinos de todos los tiempos previos.

- *¿Cuántos sabrías determinar la longitud aproximada a la que os encontráis usando vuestro reloj de pulsera (O el iPhone, pero sin internet, claro)?*

El siglo XIX trajo relojes razonablemente precisos y de bolsillo, mecánicos claro y probablemente aun quede alguno por vuestras casas, del abuelo por ejemplo, entre las joyas y reliquias familiares.

Pero cuando de verdad empezaron los relojes a ser precisos y al alcance de todos, fue con el reloj digital de cuarzo y eso fueron los años 70, y eran como lo definió un relojero de los de antes, relojes en los que lo único que se movían eran los números.

El cuarzo fue la base de osciladores con una precisión desconocida hasta entonces y aun hoy cualquier reloj de pulsera normal llevara uno de estos pequeños cristales de cuarzo como corazón.

Como ya dijimos en la sesión sobre los zumbadores, Arduino incorpora un cristal piezoeléctrico como base de tiempos, que bate a 16Mhz, es decir 16 millones de pulsos por segundo. Y la precisión qué podemos esperar es poco más o menos una deriva menor de 0,5 segundos por día y con una estabilidad excelente frente a los cambios de temperatura (El coco de los relojes mecánicos)

Todo este pegajoso prolegómeno (Uno se va haciendo mayor), viene a cuenta de que más antes que después, necesitaras controlar el tiempo en tu Arduino.

Hasta ahora hemos visto algunas instrucciones muy básicas como delay (), millis () y micros () etc. Estas instrucciones son muy útiles para controlar tiempos muy cortos, pero pronto sentirás la necesidad de medir el tiempo en minutos, horas, días y probablemente meses y años.

¿Qué no? Bueno depende mucho del tipo de proyectos que hagas. Pero imagínate por ejemplo un data logger (O registro de eventos) en el que tengas un Arduino conectado a una maquina cualquiera supervisando sus variables, que puede ser las revoluciones por minuto, sus picos de tensión o intensidad o simplemente si está encendida o parada.

No hablamos (Aun) de cómo medir estas cosas, sino sencillamente guardar registro de los valores, para su estudio posterior. Imagínate que quieres tu factura energética y saber exactamente cuándo consumes y cuanto….

Hay infinidad de casos en los que quieres monitorizar un sistema cualquiera y querrás guardar registro de los valores y acompañarlos con un sello de tiempo, para después hacer graficas o pasarlo por un programa tuyo.

Colocar el sello temporal a tus datos es un imperativo.

Y por eso, porque no somos los primeros en tener este problema, ya existen soluciones (Gracias a Dios). En esta sesión vamos a presentar la nueva **librería de Arduino Time**, ver cómo se maneja y como emplearla cuando la necesitemos y que opciones tenemos.

LA LIBRERÍA TIME DE ARDUINO

Time es una librería estándar en Arduino pero requiere que la instales antes de usarla. Esto es porque consumo recursos valiosos especialmente en un UNO.

La puedes descargar de aquí: Librería Arduino Time , y la instalas siguiendo el procedimiento habitual.

¿Por qué usar una librería para medir el tiempo? ¿Por qué no hacer cuatro cálculos y ya está? Ahora somos ya unos expertos programadores (o casi) . Podemos hacerlo.

Santa inocencia. Principalmente porque operar con el tiempo es bastante molesto como vimos cuando hacíamos el reloj en la sesión **Display con interface.**

El principal inconveniente es que las horas y minutos se cuentan en un sistema sexagesimal y no centesimal normal, y por si esto no fuera

suficiente, los días tiene 24 horas o dos ciclos de 12, unos meses 30 días y otros 31 y uno 28 0 29, depende. Un año requiere 12 meses y no 10 como debería si fuera decimal (Y luego nos creemos tranquilamente que solo usamos el sistema decimal).

Y espera, que aún no hemos hablado de las correcciones del calendario gregoriano, del año bisiesto y de las excepciones a las reglas. Es como un infierno informático.

Haced caso al abuelo. Usad la librería. Ganareis mucho en salud

La **librería Time** define en primer lugar un tipo especial de variable que llamamos time_t de 32 bits, que es la base de todo.

La idea es que time_t almacena valores de tiempo en un formato especial calculado como el número de segundos transcurridos desde una fecha dad hasta el 1 de enero de 1970.

¿ Que guarda qué? Tranquilos, para los que vengáis del mundo de la computación os diré que es el método estándar de Unix/Linux. Y para los que seáis nuevos, deciros que las conversiones son automáticas y la librería te hace el trabajo así que sin miedo.

La ventaja de una cosa tan anti intuitiva es que al convertir una fecha y hora en un numero de segundos desde una fecha de referencia, podemos restar fechas y horas tranquilamente fácilmente y el resultado sigue siendo una fecha u hora que mide la diferencia entre ambas en forma de años, meses, días, minutos y segundos.

Limpio y eficaz. Además la librería hace todo el trabajo de convertirnos fechas y horas en time_T para que podamos operar tranquilamente.

> *El inconveniente está en la letra pequeña, la librería no funciona con fechas previas a 1970, lo que no suele ser grave siempre que lo tengáis claro*

La forma práctica de manejar el tiempo es empezar definiendo una variable time_t

```
time_t T = now();
```

Los lectores astutos se habrán dado cuenta de que now() es una función, que dispone del valor actual de tiempo interno en Arduino. ¿Y qué hora tiene? Pues de momento un asco, porque cada vez que apagas tu Arduino o lo reseteas, el tiempo se reinicia desde 0.

¿Podemos hacer algo para ponerlo en hora? . Nada más fácil:

```
setTime(19,58,00,6,11,2014); // Las 19:58:00 del dia 6 de Nov de 2014
```

Para hacer un reloj con calendario podríamos usar un programa como este:

```
#include <Time.h>

void setup()
  {   Serial.begin(115200);
      setTime(19,58,00,6,11,2014);
  }
void loop()
  {   time_t t = now();

      Serial.print(day(t));
      Serial.print(+ "/") ;
      Serial.print(month(t));
      Serial.print(+ "/") ;
      Serial.print(year(t));
      Serial.print( " ") ;
      Serial.print(hour(t));
      Serial.print(+ ":") ;
      Serial.print(minute(t));
      Serial.print(":") ;
      Serial.println(second(t));
      delay(1000);
  }
```

Como veis, este programa usa las funciones primarias de manejo de fechas. Como poner en hora el reloj primero y como sacar los valores discretos de día, mes año, hora minuto y segundo.

Imagínate ahora, que tenemos dos fechas como “6 nov 2014 20:17” y “13 nov 2014 16:45” y queremos restarlas para saber cosas vulgares, como cuantos días u horas ha trabajado alguien (La gente es muy quisquillosa con lo que cobrar según el número de horas y días que ha trabajado).

> *Os propongo que lo calculéis a mano y luego lo comprobáis con vuestro Arduino. A ver cuantos acertáis. No, no es una broma aunque yo me estoy riendo.*

Para calcular esta diferencia, tenemos que dar un rodeo y hablar de una estructura importante en la nueva librería Time, que ha puesto patas arriba la antigua.

Cuando queramos generar fechas time_t a partir de números de días meses (u horas y minutos) la nueva librería define una estructura de tipo tmElements_t (No os asustéis), que tiene los siguientes miembros:

```
tmElements_t  tm ;
   tm.Second  Segundos   0 to 59
   tm.Minute  Minutos    0 to 59
   tm.Hour    Horas      0 to 23
   tm.Wday    Dia semana 0 to 6  (No se usa en  mktime)
   tm.Day     Dia        1 to 31
   tm.Month   Mes        1 to 12
   tm.Year    Año        0 to 99 (Diferencia desde 1970)
```

Así, para la primera fecha de arriba tendríamos:

```
tmElements_t Fecha ;
   Fecha.Second = 0;
   Fecha.Minute = 17;
   Fecha.Hour = 20;
   Fecha.Day =6 ;
   Fecha.Month = 11 ;
   Fecha.Year = 2014 -1970 ;

T0 = makeTime(Fecha);
```

Donde asignamos valores a los miembros de Fecha del tipo tmElements, con las propiedades que veis. De ese modo T0 es “6 nov 2014 – 20:17:00” en segundos. Y makeTime es una función que nos devuelve la hora en formato time_t, de una estructura tmElements_t.

Para mostrar su uso, vamos a modificar un poco el programa anterior: **PROG_53_1**

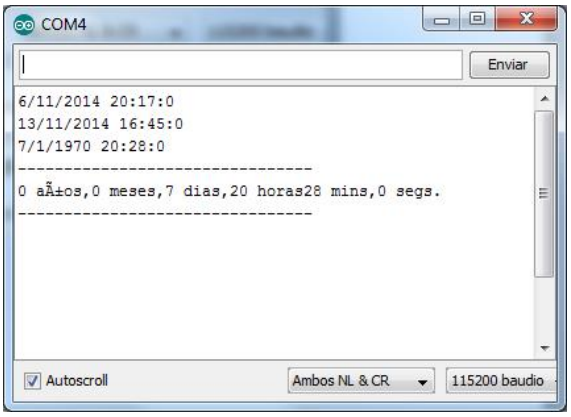
```
#include <Time.h>
time_t  T0, T1 ;         // Contenedores de fechas

void setup()
   {    Serial.begin(115200);  }

void loop()
   {    T0 = SetFecha(2014, 11, 6, 20, 17, 0);  // 6 nov 2014  20:17
        printFecha(T0) ;
        T1 = SetFecha(2014, 11, 13, 16, 45, 0);  // 13 nov 2014 16:45
        printFecha(T1) ;
        printFecha(T1 - T0);

        Serial.println("--------------------------------------------------");
        time_t H =  T1 - T0 ;

        Serial.print(String(year(H )- 1970)+" años,"  + String(month(H)-1 )+" meses,");
        Serial.println(String(day(H))+ " dias," + String(hour(H))+ " horas");
        Serial.println("--------------------------------------------------");
   }


void printFecha(time_t  t)
   {
       Serial.print(day(t)) ;    Serial.print(+ "/") ;   Serial.print(month(t));   Serial.print(+ "/") ;
```

```
        Serial.print(year(t));    Serial.print( " ") ;
        Serial.print(hour(t));   Serial.print(+ ":") ;   Serial.print(minute(t));   Serial.print(":") ;
        Serial.println(second(t));
        delay(1000);
    }

time_t SetFecha(int y, int m, int d, int h, int mi, int s  )
   {  tmElements_t Fecha ;
      Fecha.Second = s;
      Fecha.Minute = mi;
      Fecha.Hour = h;
      Fecha.Day = d ;
      Fecha.Month = m ;
      Fecha.Year = y -1970 ;

      return makeTime(Fecha);
   }
```

La función printFecha() formatea la salida de una variable time_t en sesundos, y más interesante es la función SetFecha() que utiliza una estructura tm_Elements_t para asignar los valores que le pasamos como paramentros de la fecha, para devolver una fecha estándar convertida.

Ya podemos calcular la diferencia entre ambas pero el resultado sería en segundos, lo que iba a ser muy poco descriptivo. Por eso calculamos H como su diferencia y lo imprimimos con un poco de formateo.

La salida del programa es la siguiente:



La tercera línea contiene la respuesta correcta a la diferencia entre fechas, pero hay que interpretarlo. “7/1/1970” significa 7 días, 0 meses 0 años, porque enero es el mes 1 (No existe el mes 0). Del mismo modo 1970 es el primer año del calendario interno y por tanto significa 0 años.

La respuesta es válida pero por alguna razón, no hay mucha gente dispuesta a aceptarla así. Por eso formateamos la respuesta en la siguiente línea.

LA SINCRONIZACIÓN DE LOS RELOJES EN ARDUINO

Aún tenemos otra bomba de relojería en ciernes. La hora interna de Arduino se guarda en una variable que cuenta los milisegundos transcurridos a partir del momento en que lo encendimos o que lo reseteamos. Si lo dejamos funcionando suficiente tiempo otro problema a tener en cuenta es el overflow(Desbordamiento) del reloj interno de Arduino.

Dado que almacena el número de milisegundos transcurridos desde el encendido. Si usa una variable unsigned long de 32 bits para guardarlo, el

máximo valor que puede contener es:

2^{32} = 4.294.967.296

Y como el número de milisegundos por día es de

```
1000 ms * 60 segs * 60 mins * 24 horas =  86.400.000
```

El reloj desbordará inevitablemente en

```
4.294.967.296 / 86400000 = 49, 7102696 días
```

O sea, algo menos de 50 días, al cabo de los cuales el reloj se pondrá a cero y empezará de nuevo.

La cuestión de la **sincronización de los relojes**, no es un tema baladí, y buena parte de la razón de ser de la nueva librería Time tiene que ver con esta idea (*Por increíble que ahora os pueda parecer*).

- *Cualquiera que tenga varios relojes en su casa sabe lo desesperante que resulta ponerlos en hora tras un apagón y además a medida que los meses pasan conseguir que todos marquen la misma hora es un curro a horario completo.*
- *¿Se os ha ocurrido pensar alguna vez, que lo lógico es que se pusieran solos en hora por algún procedimiento digamos WIFI que consultara en Internet la hora?*
- *Es otras de las muchas maneras en que la Internet de las cosas IOT puede ayudar a al humanidad..*

Hay varios métodos que se pueden usar para poner en hora el reloj interno de Arduino cada vez que lo reseteamos:

- A mano, como hemos hecho en los ejemplos de este capítulo. Para los ejemplos puede valer pero poco más. Necesitamos otra solución
- Con un programa serie desde nuestro PC. Funciona pero exige que nuestro PC este encendido y corra un cierto programa para el servicio sincronización.
- Con un chip de reloj y una pila que manténgala fecha y hora, aunque apaguemos. Este será el tema de nuestra próxima sesión.
- Sincronizándose a través de internet con un servicio horario (Si señor, eso existe).Se llaman servidores NTP (Network Time Protocol o protocolo de fecha en red).
- Usando el reloj de un GPS, que por si no lo sabéis son los relojes más descabelladamente precisos que podéis comprar a un precio asequible (Hablaremos del asunto cuando lleguemos a los GPSs).

Tenéis ejemplos de todos estos métodos en los ejemplos de la librería Time y no creo que valga la pena entrar en cada uno ahora. Cuando instalasteis la librería también se instalaron los ejemplos. Echadles un vistazo.

Para ayudarte sincronizar tu Arduino, la nueva librería Time, dispone de funciones lógicas que nos indican si la hora ha sido sincronizadda y cunato hace de eso

FUNCION	OBJETIVO
timeStatus();	Puede devolver, una de 3 situaciones: • timeNotSet No se ha ajusatdo/sincronizado el reloj. • timeSet El reloj ha sido ajustado. • timeNeedsSync El reloj se ajustó pero fallo la sincronización con un servcio time externo.
setSyncProvider	Define una función ala que se llama periódicamente para sincronizar con un servicio exterior
setSyncInterval()	Define cada cuanto tiempo se llama a la función anterior.

RESUMEN DE LA SESIÓN

.

- Hemos visto como instalar la nueva librería Time.
- Hemos hecho un par de programas para mostrar el uso de los nuevos tipos time_t.
- Presentamos los conceptos básicos de la sincronía con servicios externos.

ANTERIOR SIGUIENTE

CATEGORIAS DE LOS PRODUCTOS

Selecciona una categoría