

# Le temps avec Arduino : utiliser un module « Temps-réel » (ou RTC – Real Time Clock) (l'exemple du DS 1307) pour utiliser l'heure et la date avec Arduino.



## Ateliers Arduino

par X. HINAULT

[www.mon-club-elec.fr](http://www.mon-club-elec.fr)



Tous droits réservés – 2012.

**Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.**

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

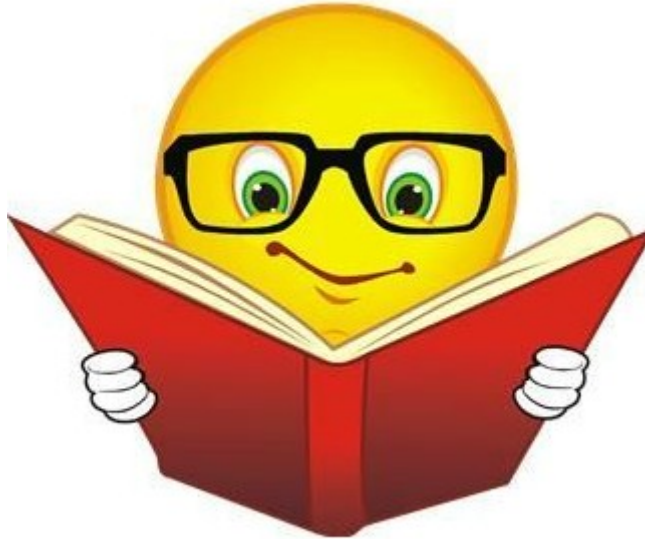
Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

## 1. *Intro*

L'objectif ici est de montrer comment utiliser le temps réel avec Arduino, à l'aide d'un module RTC non volatile (le DS 1307), afin d'être en mesure de mettre en place des applications utilisant l'heure et la date.



**Prêt ? C'est parti !**

## 2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

### De l'espace de développement Arduino

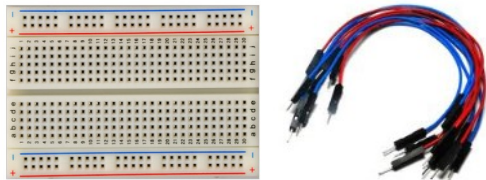


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

### Du nécessaire pour réaliser des montages sans soudure

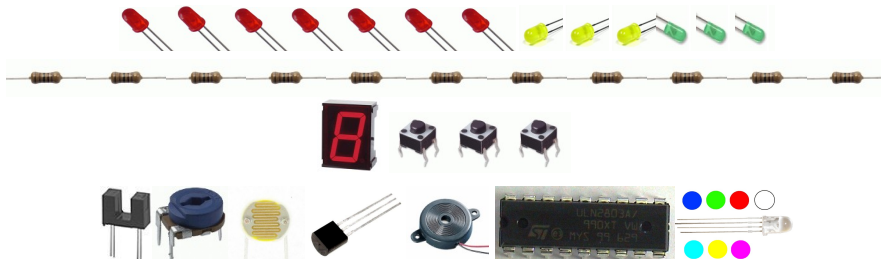


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

### De quelques composants de base



**Pour vous simplifier la vie, nous avons négocié ce kit pour vous !**

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire

<http://www.gotronic.fr/> avec le code express **701710**

**GO TRONIC**  
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

### 3. Matériel spécifique nécessaire pour cet atelier

Pour cet atelier vous aurez besoin également :

**Soit d'une carte d'extension (shield) mémoire + temps réel (Snootlab)**



La carte d'extension (ou shield) mémoire SD + « temps réel » est une carte électronique enfichable broche à broche sur la carte Arduino et qui dispose :

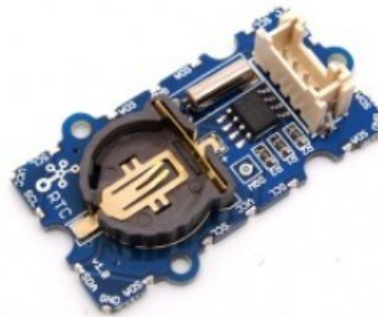
- d'un étage « carte mémoire SD » d'utiliser une carte mémoire micro SD, SD et SDHC. Cet étage utilise la **communication SPI** (broches 13,12,11, et 10 ) pour communiquer avec Arduino.
- d'un étage « temps-réel » basé sur un **DS1307**. Cet étage utilise la **communication I2C** (broches A4 et A5 ) pour communiquer avec Arduino.

disponible chez <http://snootlab.com/> Prix constaté : 18€ environ

**Et d'une manière générale, de n'importe quel module utilisant le circuit intégré DS1307 :**



Module DFR0151 - Gotronic



Module Grove SEN12671P - Gotronic



Le DS1307 « brut » - Gotronic  
(nécessite quelques composants en +)

## 4. Le « temps-réel » : introduction

### Le « temps-réel », c'est quoi ?

- Pour faire simple, le « temps-réel », c'est l'heure et la date ! Le module « temps-réel » que vous connaissez tous, c'est la montre !



### Générer le « temps-réel » avec Arduino

- Comme vous le savez, le langage Arduino dispose de plusieurs instructions permettant de gérer le temps :
  - `millis()` qui renvoie le nombre de millisecondes depuis le lancement du programme,
  - `micros()` qui renvoie le nombre de microsecondes depuis le lancement du programme.
- Ces instructions peuvent servir de base pour mettre en place une « émulation » de temps réel : par une incrémentation de variables toutes les secondes, minutes, heure, jour voire mois, etc... il est possible d'obtenir une application « temps réel » basique...

### Les limites de cette façon de faire

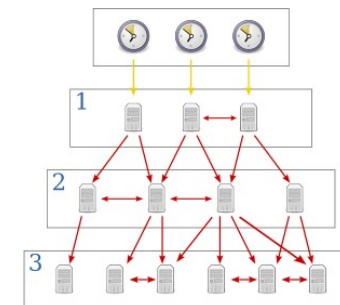
- Bien que très pratique sur de petits programmes d'essai, cette façon de faire présente de nombreuses limites.
- Tout d'abord, le temps est volatile : si Arduino s'éteint, l'heure est perdue. On pourrait certes mettre les données en Eeprom, mais un arrêt prolongé ne serait pas pris en compte.
- D'autre part, l'instruction `millis()` est au format long et ne pourra pas compter au-delà de 2 147 483 647 soit 24 jours environ...
- Enfin, l'horloge de l'arduino « glisse » un peu et manque de précision au fil du temps (décalage des secondes...)

### La bonne solution serait...

- Ce qu'il faudrait, c'est une horloge « temps-réel » :
  - que l'on puisse régler une fois pour toute et non volatile : même si mise hors tension, l'heure n'est pas perdue,
  - qui ne glisse pas dans le temps...
  - qui n'aie pas de limite de durée...
  - et bien sûr assez simple à utiliser...
- La bonne nouvelle, c'est que c'est possible à réaliser avec Arduino à l'aide d'un circuit spécialisé et une pile bouton de 3V !
- La solution passe par l'utilisation d'un circuit à communication I2C (voir le tuto dédié si vous ne savez pas ce que c'est...!) spécialisé pour fournir le « temps-réel » : le plus utilisé est le DS1307 et il existe une librairie Arduino pour l'utiliser facilement.

### Pour info : Le temps en informatique

- Pour la plupart des systèmes informatiques, la référence du comptage du temps est le 1er janvier 1970 à 00:00:00 UTC (Temps Universel Coordonné).
- On appelle « temps Unix » le nombre de secondes écoulées depuis le 1er janvier 1970 00:00:00 UTC jusqu'à l'événement à dater (en dehors de quelques secondes intercalaires...)
- Cette formulation du temps est souvent intéressante pour réaliser facilement des calculs entre des dates ou des événements, plutôt que de gérer une comparaison entre les jours, heures, secondes, etc...
- Sur les réseaux, la gestion du temps et de sa synchronisation est un problème crucial : il existe donc des serveurs de temps, dits NTP (Network Time Protocol) qui permettent une telle synchronisation. Ceci ne nous sera pas directement utile ici, mais c'est bon à savoir.



## 5. Temps réel : Première approche : Emuler une horloge « temps-réel » avec **millis()**

### Ce qu'on va faire ici...

- A présent nous allons donc ici aborder le « temps-réel » avec Arduino : c'est le comptage de l'heure « vraie », au format heure:minute:seconde
- La première façon de le faire assez simplement est de se baser sur la fonction **millis()** qui, je le rappelle une nouvelle fois, renvoie une valeur de type **long** correspondant au nombre de millisecondes écoulées depuis la mise sous tension de l'Arduino.
- Le principe que nous allons utiliser ici a été présenté dans le tuto dédié aux techniques de temporisation avec Arduino : une variable de mémorisation de la dernière valeur de **millis()** prise en compte et une variable de délai. Sur cette base, nous allons gérer des variables de comptage des minutes et des heures :
  - les minutes seront incrémentées toutes les 60 secondes...
  - les heures seront incrémentées toutes les 60 minutes...
  - etc...
- Le code est disponible ici : <https://gist.github.com/sensor56/e3d92a9b933f8e2a3d1b>

#### Remarques

La fonction **millis()** a une limite importante : elle ne pourra pas compter au delà de 2 147 483 647 ms (type **long**) soit 24 jours environ... Au-delà, la remise à zéro de **millis()** entraînera une erreur de comptage. Donc, cette solution n'est utilisable que pour des durées limitées, pas pour du « vrai » temps réel permanent.

D'autre part, **millis()** glissera au fil du temps, avec un décalage de quelques secondes possible au bout d'un certain délai,

et d'autre part, cette « horloge » sera remise à zéro si Arduino est mis hors tension...

Ses réserves étant posées, utiliser **millis()** comme base « temps réel » est très pratique et suffisant dans de nombreuses situations.

### Entête déclarative

#### Variables utiles

- On déclare une variable de mémorisation de **millis()** et une variable de délai
- On déclare 3 variables pour la gestion de l'heure, en utilisant l'heure courante.

```
//-- variables pour base temps
long millis0=0; // variable pour mémoriser dernier millis() pris en compte
int delai=1000; // intervalle à utiliser en millisecondes

//-- variables de mémorisation de l'heure
int secondes=0;
int minutes=0;
int heures=0;
```

## Fonction **setup()**

### Initialisation série

- On initialise la fonction série à 115200 bauds

### Initialisation des variables de l'heure en se basant sur l'heure système

- Lors de la compilation, la variable `__TIME__` donne accès à l'heure système sous la forme d'un tableau de char au format HH:MM:SS
- A partir de là :
  - on stocke cette chaîne dans un String
  - à l'aide de la fonction `substring` de l'objet String, on extrait les sous-chaînes de l'heure, des minutes et des secondes
  - puis on convertit les caractères en valeur numérique correspondante,
  - pour enfin initialiser les variables de comptage de l'heure...
- **Ce truc, très pratique, permet de mettre à jour l'heure au moment de la programmation de la carte Arduino !**

### Initialisation du comptage de l'heure

- On mémorise `millis()` courant

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    Serial.begin(115200); // initialise port série
    Serial.println(__TIME__);
    millis0=millis(); // mémorise millis()

    String heureCourante=String(__TIME__); // __TIME__ : heure compilation au format HH:MM:SS

    String strHeure=heureCourante.substring(0,2); // extrait HH
    Serial.println(strHeure);
    heures=((strHeure.charAt(0)-48)*10)+(strHeure.charAt(1)-48); // conversion du String en int
    Serial.println(heures);

    String strMinute=heureCourante.substring(3,5); // extrait MM
    Serial.println(strMinute);
    minutes=((strMinute.charAt(0)-48)*10)+(strMinute.charAt(1)-48); // conversion du String en int
    Serial.println(minutes);

    String strSeconde=heureCourante.substring(6); // extrait SS
    Serial.println(strSeconde);
    secondes=((strSeconde.charAt(0)-48)*10)+(strSeconde.charAt(1)-48); // conversion du String en int
    Serial.println(secondes);

} // fin de la fonction setup()
```



## Fonction `loop()`

- On commence par tester si le délai de 1 seconde est initialisé :
  - ensuite par un jeu de condition, on assure l'incrémentation des variables des heures, minutes et secondes,
  - ainsi que la remise à zéro des variables lorsque les valeurs maximales sont atteintes.
- On affiche ensuite l'heure au format HH:MM:SS : remarquer la façon d'enchaîner plusieurs `Serial.print()` sur la même ligne...

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    if (millis()-millis0>=delai) { // si le délai est écoulé
        millis0=millis(); // mémorise millis()

        //----- gestion variables de l'heure ---
        secondes=secondes+1; // incrémente secondes
        if (secondes>59) {
            secondes=0; // RAZ secondes
            minutes=minutes+1; // incrémente minutes
        } // fin if secondes

        if (minutes>59) {
            minutes=0; // RAZ minutes
            heures=heures+1; // incrémente heures
        } // fin if minutes

        if (heures>23) heures=0; // RAZ heures

        //--- affiche heure ---
        if (heures<10)Serial.print("0"),Serial.print(heures), Serial.print(":"); else Serial.print(heures), Serial.print(":");
        if (minutes<10)Serial.print("0"),Serial.print(minutes), Serial.print(":"); else Serial.print(minutes), Serial.print(":");
        if (secondes<10)Serial.print("0"),Serial.print(secondes), Serial.println(); else Serial.print(secondes), Serial.println();

    } // fin si délai écoulé

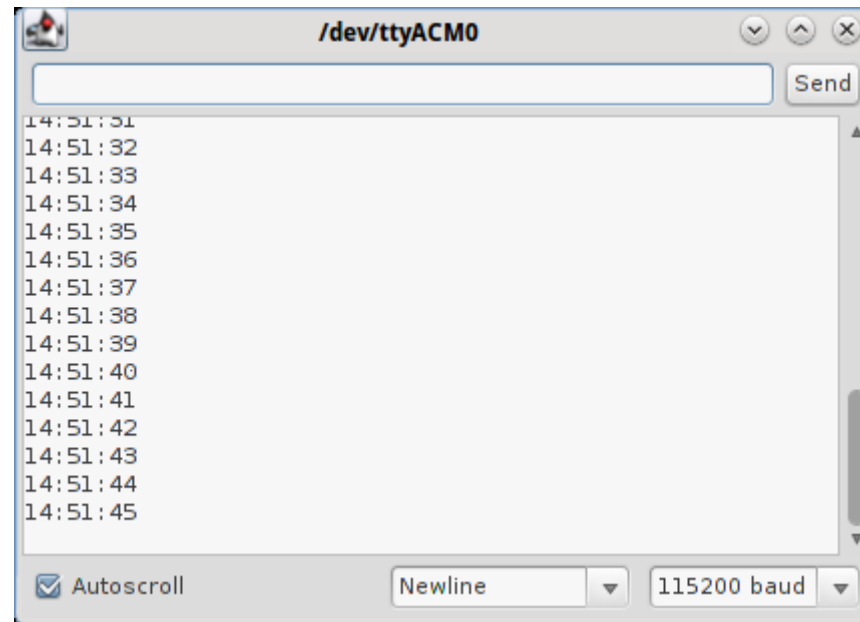
    // les autres instructions ici
    // elles seront exécutées meme si le délai n'est pas écoulé

} // fin de la fonction loop()
```



## Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 », ce qui donne ...



### Quelques trucs utiles...

Truc n°1 : pour s'assurer du bon comptage des minutes et des heures, utiliser un délai court (100ms, 10ms et même 1ms) pour tester...

Truc n°2 : il est possible de mettre à l'heure automatiquement lors de la programmation à l'aide de la variable `__TIME__` ...

## 6. Arduino : Temps-réel : la librairie RTCLib

### La librairie Arduino RTCLib pour DS1307

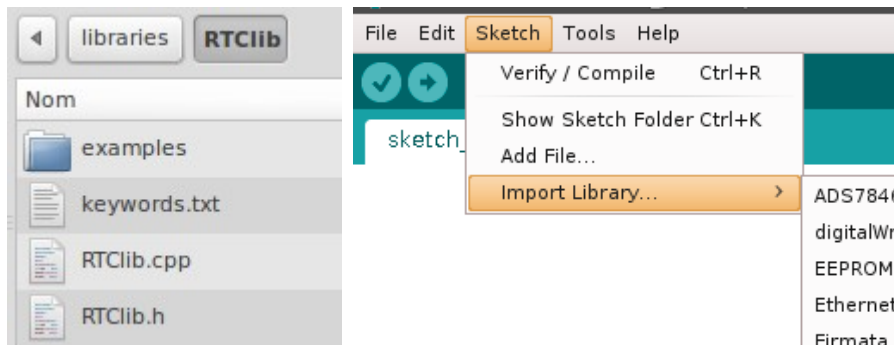
- La librairie RTCLib implémente toutes les fonctions utiles pour l'utilisation du « temps-réel » :
  - soit avec le circuit DS1307 (I2C) (nombreux modules compatibles)
  - soit à partir de la base temps `millis()` de l'Arduino (ce qui permet de démarrer même sans disposer d'un module RTC DS1307)

### Télécharger la librairie

- La librairie RTCLib est disponible ici : <https://github.com/adafruit/RTCLib>

### Installation

- Télécharger l'archive. au format zip ou autre. L'extraire
- Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier \*.h ou \*.cpp principal. Corriger au besoin. Ici le nom est **RTCLib**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary**.



### Le constructeur principal

- Trois constructeurs pour cette librairie :

```
DateTime now ; // représente une date/heure
RTC_DS1307 RTC ; // représente le DS1307
RTC_Millis RTC; // représente une base temps basée sur millis() !
```

### Fonctions de la librairie

#### Des objets RTC

- `begin()` : initialise la base temps réel
- `adjust()` : règle la base temps réel
- `isrunning()` : test si la base temps réel est active
- `now()` : renvoie la date/heure courante

#### De l'objet DateTime

- `year()`, `month()`, `day()` : renvoie l'année, mois, jour
- `hour()`, `minute()`, `second()` : renvoie heure, minute, secondes
- `dayOfWeek()` : renvoie le jour de la semaine
- `secondstime()` : renvoie le temps absolu en secondes **depuis le 1/1/2000**
- `unixtime()` : renvoie le temps unix, le nombre de secondes **depuis le 1/1/1970**

### Code d'exemple

```
#include <Wire.h>
#include "RTCLib.h"

RTC_DS1307 RTC; // déclare objet représentant le DS1307

void setup () {
    Serial.begin(115200); // initialise la communication série
    Wire.begin(); // initialise I2C
    RTC.begin(); // initialise le DS1307
}

void loop () {
    DateTime now = RTC.now(); // récupère l'heure courante

    Serial.print(now.hour(), DEC); // affiche heure
    Serial.print(':');
    Serial.print(now.minute(), DEC); // affiche minutes
    Serial.print(':');
    Serial.print(now.second(), DEC); // affiche secondes
    Serial.println();
    delay(3000); // pause
}
```

## 7. Temps réel : Utiliser *millis()* comme base « temps-réel » avec la librairie RTLib

### Ce qu'on va faire ici...

- Ici, nous allons tout simplement tester l'affichage de l'heure et de la date en se basant sur.... *millis()* et la librairie RTCLib !
- Bon, je sais, je viens de vous expliquer toutes les limites de *millis()*... mais il se peut que vous n'ayez pas de circuit ou de module RTC sous la main... et que vous ayez envie ou besoin d'utiliser le temps réel...
- *Je vous montre ici comment utiliser simplement la date et l'heure « temps-réel », avec uniquement la carte Arduino.* Ceci n'est pas forcément très connu... mais la lib' RTCLib le permet, et personnellement, je trouve ça bien pratique !
- Ce code est disponible ici : <https://gist.github.com/sensor56/09966c590264cf320396>

### Entête déclarative

#### Inclusion des librairies utiles

- On commence par inclure les librairies
  - la librairie *Wire.h* pour la communication I2C, *bien que nous n'utilisions ici aucun module I2C... mais comme la librairie RTCLib est prévue pour le circuit I2C DS1307 à la base, il faut intégrer la librairie Wire ici.* Nous verrons ceci par la suite...
  - la librairie *RTCLib.h* pour la gestion simplifiée du temps réel

#### Objet utiles

- On déclare un objet RTC\_Millis représentant la base temps basée sur la fonction *millis()* :

```
#include <Wire.h>
#include "RTCLib.h"

RTC_Millis RTC; // déclare objet représentant base temps Millis
```

## Fonction **setup()**

### Initialisation Série

- On initialise la communication série à 115200 Bauds avec l'instruction **Serial.begin()**

### Initialisation « temps-réel »

- On initialise la librairie à l'aide de la fonction **begin()** de l'objet RTC\_Millis précédemment déclaré

```
void setup () {  
  Serial.begin(115200); // initialise la communicatin série  
  RTC.begin(DateTime(__DATE__, __TIME__)); // initialise base temps millis  
}
```

## Fonction **loop()**

- On commence par créer un objet **DateTime** représentant la date/heure actuelle, appelé ici now (=maintenant)
- Puis, par un jeu de **Serial.print()**, on affiche successivement l'année, le mois, le jour, l'heure, les minutes, les secondes...
- le code boucle après une pause d'une seconde

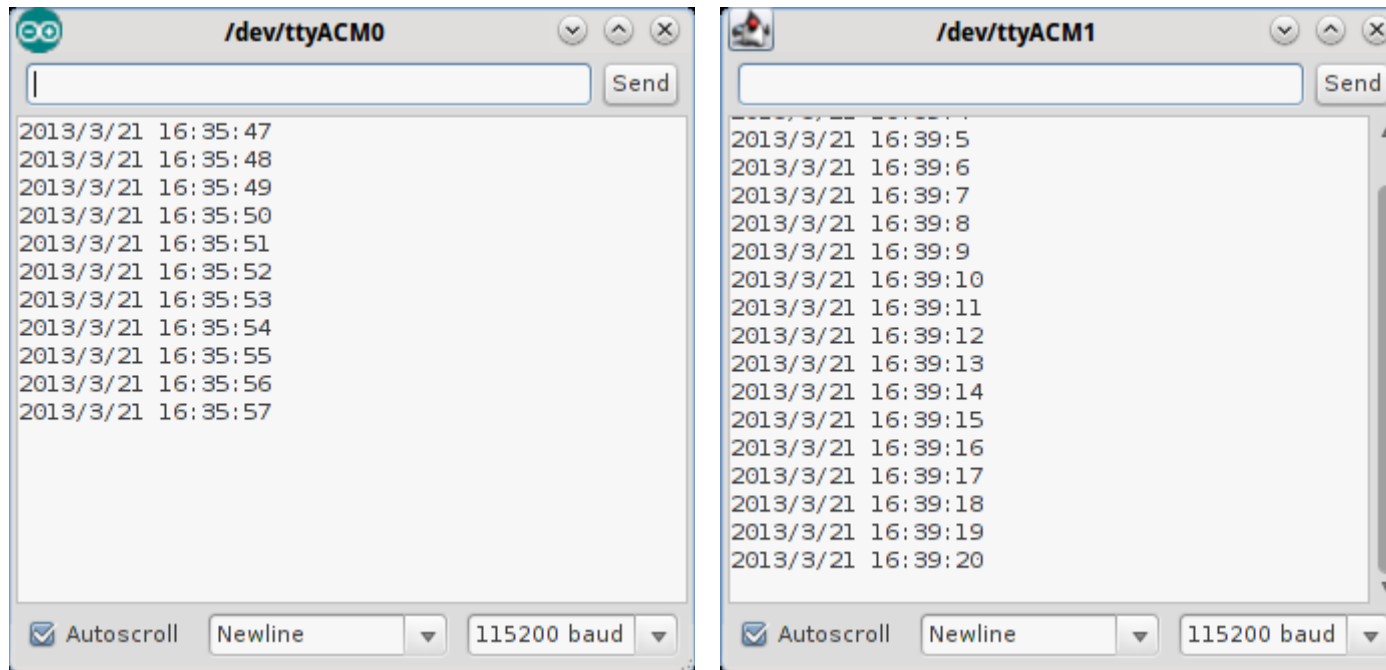
```
void loop () {  
    DateTime now = RTC.now(); // récupère l'heure courante  
  
    Serial.print(now.year(), DEC); // affiche année  
    Serial.print('/');  
    Serial.print(now.month(), DEC); // affiche mois  
    Serial.print('/');  
    Serial.print(now.day(), DEC); // affiche jour  
    Serial.print(' ');  
    Serial.print(now.hour(), DEC); // affiche heure  
    Serial.print(':');  
    Serial.print(now.minute(), DEC); // affiche minutes  
    Serial.print(':');  
    Serial.print(now.second(), DEC); // affiche secondes  
  
    Serial.println();  
    delay(1000); // pause  
}
```

Vous allez me demander : « Mais comment Arduino se met-il à l'heure ? »

Et bien, très astucieusement, la librairie **RTCLib** va s'initialiser avec l'heure du système de votre ordinateur au moment de la programmation, grâce à l'utilisation des variables `__TIME__` et `__DATE__`. Simple et efficace !

## Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 », ce qui donne ...



Pour la suite, si vous n'avez pas de dispositif temps réel (RTC) à base de DS1307, vous pourrez le faire de cette même façon : pratique !



## 8. Le DS1307 : circuit intégré « Temps-réel » RTC (Real Time Clock) à communication I2C

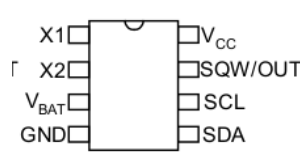
### Présentation

- Le circuit DS1307 est un circuit intégré à **communication I2C**, qui assure le calcul automatique de la date, de l'heure, du jour et de l'année de façon fiable (utilise un quartz horloger).
- Typiquement, l'heure est fixée avec l'heure courante au moment de la programmation initiale.
- Ce circuit est typiquement alimenté par une pile bouton 3V qui assure son bon fonctionnement et la mémorisation de l'heure même lorsque le circuit est hors-tension.
- Un tel circuit « temps-réel » est très pratique pour réaliser notamment de l'enregistrement de données « horodatées ».

Le DS1703 utilisé sur de nombreux shields est facile à trouver et est très utilisé. Il dispose d'une librairie Arduino opérationnelle.

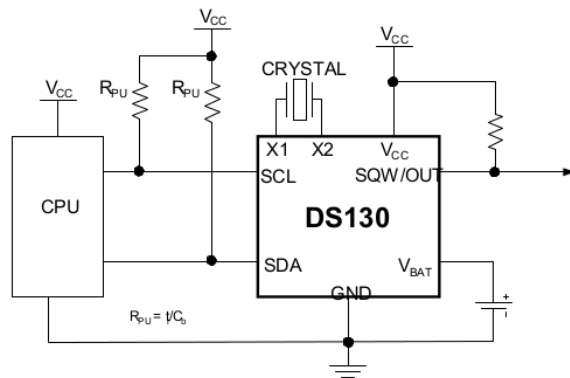
### Le brochage

- Le DS1307 se présente sous la forme d'un boîtier DIL 8 broches :



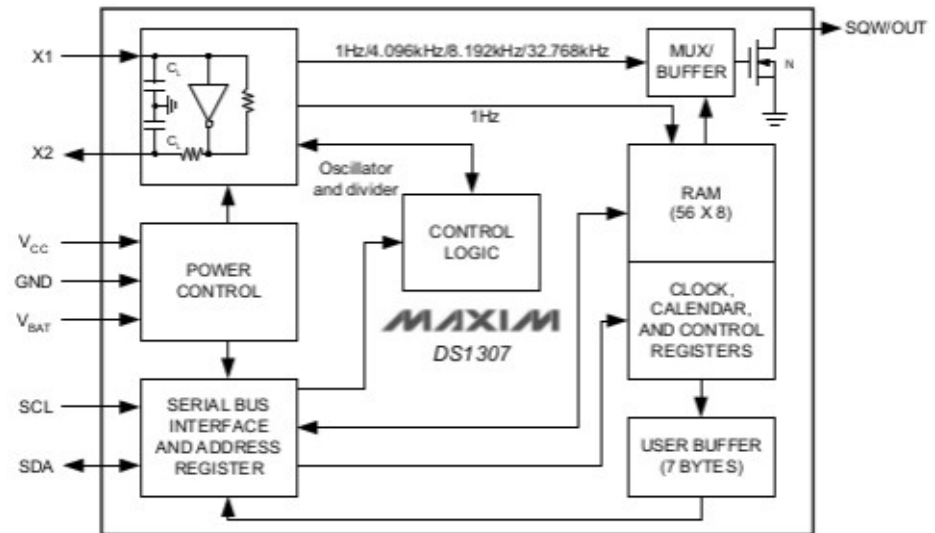
- Il dispose de 2 broches pour la communication I2C (SCL et SDA), de 2 broches pour le quartz (X1 et X2) ainsi qu'une pour la pile (VBat).

### Le montage type



### Détails internes du DS1307, circuit I2C

- Ce circuit assure le calcul automatique de l'heure, de la date, du jour, des secondes... Logiquement, en interne, il dispose :
  - d'un étage oscillateur,
  - de plusieurs registres qui contiennent les données d'heure et de date,
  - de l'étage de communication I2C...



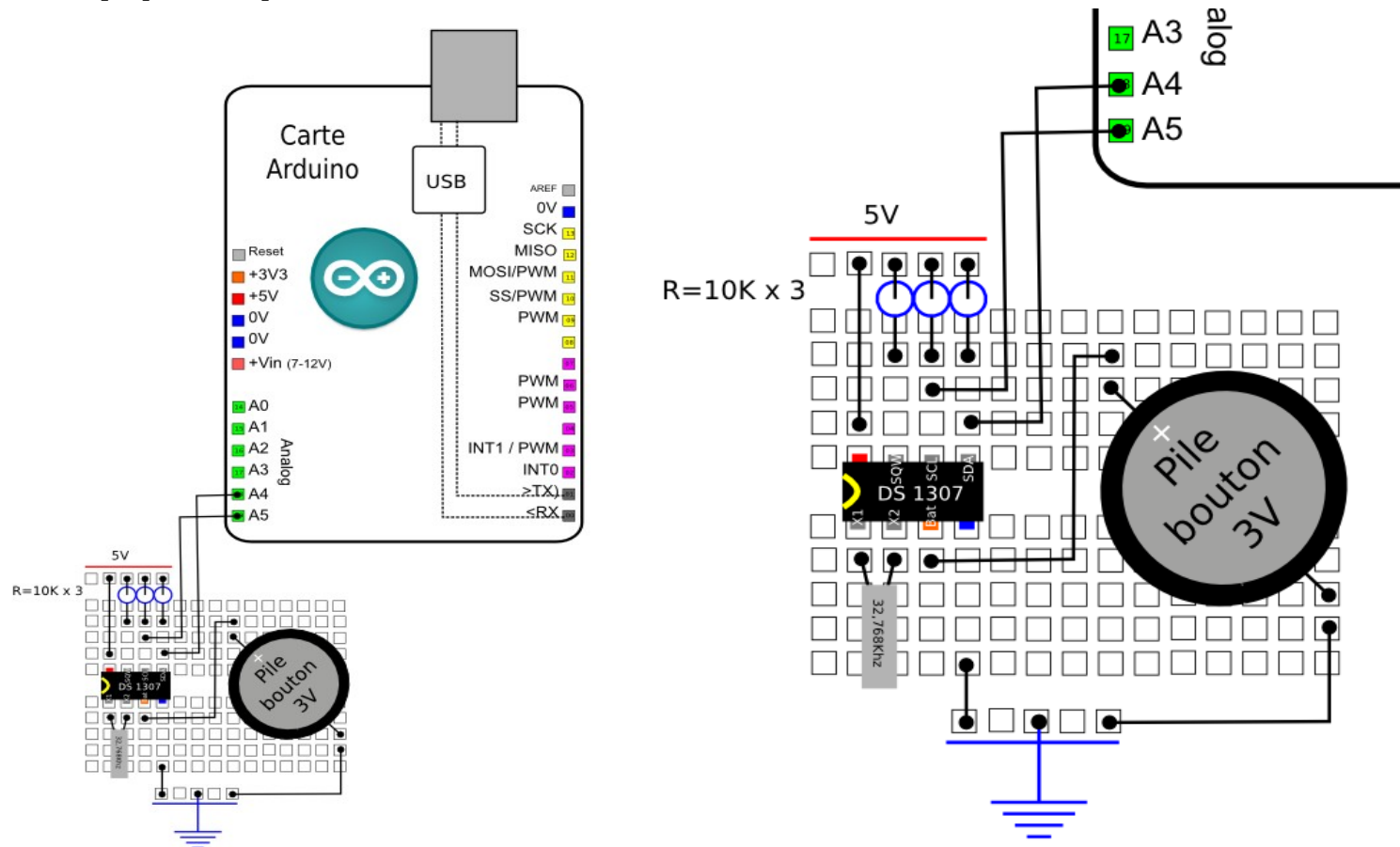
ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds			Seconds	Seconds	00-59
01h	0	10 Minutes			Minutes			Minutes	Minutes	00-59
02h	0	12	10 Hour	10 Hour	Hours			Hours	Hours	1-12 +AM/PM 00-23
		24	PM/AM							
03h	0	0	0	0	0	DAY			Day	01-07
04h	0	0	10 Date			Date			Date	01-31
05h	0	0	0	10 Month	Month			Month	Month	01-12
06h	10 Year			Year			Year			00-99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h-3Fh									RAM 56 x 8	00h-FFh

- Je ne détaille pas ici le protocole de communication I2C qui sera géré par la librairie utilisée (voir ci-dessous).
- Pour plus de détails, se reporter au datasheet du DS1307.



## 9. Utiliser le DS1307 sur une plaque d'essai

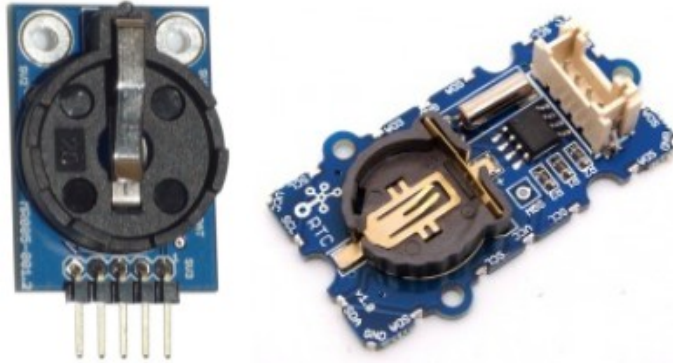
- Comme on vient de le voir, l'utilisation d'un DS1307 nécessite :
  - un quartz horloger à 32,768 KHz
  - de 3 résistances de rappel au plus, de 10KOhms typiquement
  - d'un support pour pile bouton et d'une pile bouton 3V
  - et... c'est tout.
- Le montage sur une plaque d'essai pour une utilisation avec Arduino sera le suivant :



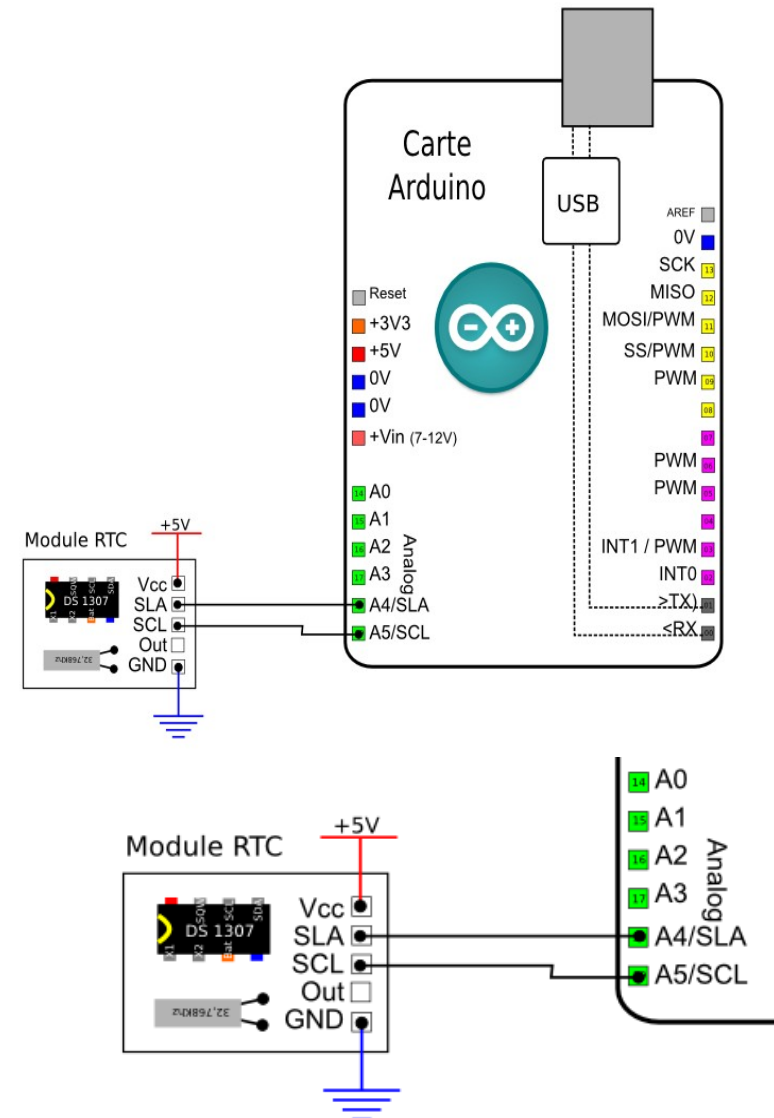
**Il est à priori nécessaire que la pile 3V soit connectée au DS1307, sinon, l'oscillateur ne fonctionnera pas.**

## 10. Exemple d'utilisation d'un module RTC utilisant le DS1307 (communication I2C)

- Les modules RTC basé sur un DS1307 sont nombreux et faciles à utiliser avec Arduino :



- Typiquement, ces modules vont présenter 4 voire 5 broches de connexion :
  - les 2 broches d'alimentation 0V et +5V
  - les 2 broches de communication I2C SLA et SCL
  - une broche (optionnelle) fournissant une impulsion carrée de fréquence voulue
- L'utilisation d'un tel module est simple :
  - connecter le 0V et le +5V au 0V et au +5V de la carte Arduino
  - connecter les 2 broches I2C SLA et SCL du modules au broches SLA et SCL de la carte Arduino
- Encore une fois, il est à priori nécessaire que la pile 3V du module soit en place sinon, l'oscillateur ne fonctionnera pas**



Exemple avec une Arduino UNO, à adapter à votre situation.

## 11. Exemple de shield Arduino utilisant le DS1307 : L'étage RTC du shield « Mémoire » de chez Snootlab

### Présentation

La carte d'extension (ou shield) mémoire SD + « temps réel » est une carte électronique enfichable broche à broche sur la carte Arduino et qui dispose :

- d'un étage « carte mémoire SD » d'utiliser une carte mémoire micro SD, SD et SDHC. Cet étage utilise la **communication SPI** (broches 13,12,11, et 10 ) pour communiquer avec Arduino.
- d'un étage « temps-réel » basé sur un DS1307. Cet étage utilise la **communication I2C** (broches A4 et A5 ) pour communiquer avec Arduino.

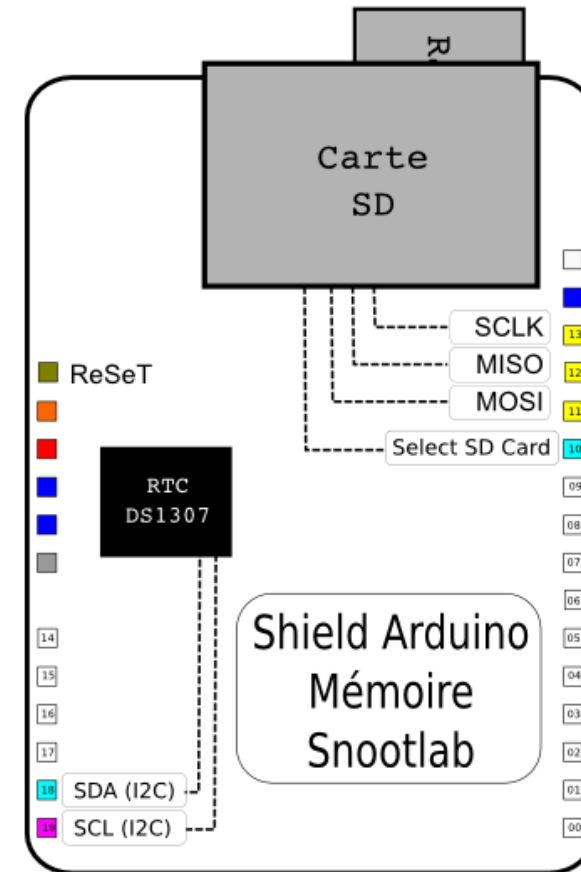
disponible chez <http://snootlab.com/> Prix constaté : 18€ environ



### L'étage « temps-réel »

- L'étage de ce shield qui nous intéresse ici est l'étage « temps réel » à communication I2C.
- Cet étage « temps-réel » est basé sur un composant I2C, le DS1307, qui assure le calcul automatique de la date, de l'heure, du jour et de l'année de façon fiable (utilise un quartz horloger). Ce circuit est alimenté par une pile bouton qui assure son bon fonctionnement même lorsque le shield est hors-tension.
- Un tel étage « temps-réel » est très pratique pour réaliser notamment de l'enregistrement de données « horodatées », et il est donc logique de l'associer à un étage de stockage sur carte SD comme ici.

Le DS1703 utilisé sur ce shield est facile à trouver et est très utilisé, dispose d'une librairie Arduino opérationnelle. Plusieurs autres shields ou modules utilisent également ce circuit très polyvalent.



Quelque soit le shield ou module RTC que vous utiliserez, le principe sera le même : les broches SDA et SCL de votre shield / module seront connectées aux broches A4 (SDA) et A5 (SCL) de la carte Uno.

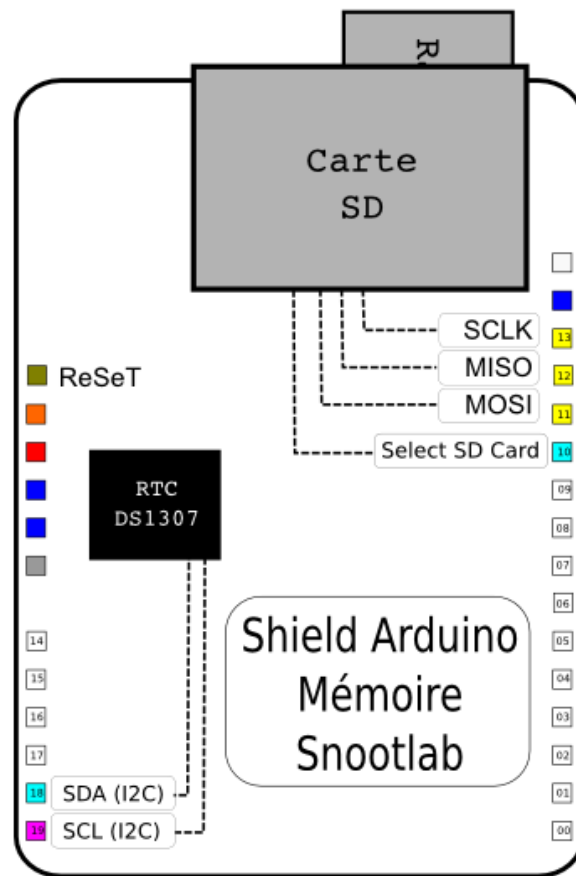
Sur la Mega et la Due, aux broches 21 (SCL) et 20 (SDA)

Sur la Leonardo, aux broches 2 (SDA) et 3 (SCL)

## 12. Temps-réel avec un DS1307 : le montage

Dans ce premier exemple, nous allons voir comment une librairie dédiée permet de simplifier grandement l'utilisation du DS1307, composant I2C : toute la communication I2C « directe » est gérée par la librairie et les fonctions de la librairie permettent d'accéder directement aux fonctionnalités utiles.

- Le montage se résume à sa plus simple expression : il suffit de réaliser l'un des montages présentés précédemment. Avec le shield « Mémoire », cela donne :



Ici, la carte mémoire SD n'est pas utilisée.

## 13. Temps-réel avec un DS1307 : test simple : le programme

### Ce qu'on va faire ici...

- Ici, nous allons tout simplement tester l'affichage de l'heure et de la date avec le DS1307 en utilisant la communication I2C et la librairie RTCLib.
- Le code est disponible ici : <https://gist.github.com/sensor56/4c837ac24ed5a12cbbde>

### Entête déclarative

#### Inclusion des librairies utiles

- On commence par inclure les librairies
  - la librairie **Wire.h** pour la communication I2C
  - la librairie **RTCLib.h** pour la gestion simplifiée du temps réel avec le DS1307

#### Objet utiles

- On déclare un objet RTC\_DS1307 représentant la base temps basée sur le DS1307 :

```
#include <Wire.h>
#include "RTCLib.h"

RTC_DS1307 RTC; // déclare objet représentant le DS1307
```

## Fonction **setup()**

### *Initialisation Série*

- On initialise la communication série à 115200 Bauds avec l'instruction **Serial.begin()**

### *Initialisation I2C*

- On initialise la communication I2C en mode « Arduino maître » avec l'instruction **Wire.begin()**

### *Initialisation « temps-réel »*

- On initialise le DS1307 à l'aide de la fonction **begin()** de l'objet RTC\_DS1307 précédemment déclaré

```
void setup () {  
  Serial.begin(115200); // initialise la communication série  
  Wire.begin(); // initialise I2C  
  RTC.begin(); // initialise le DS1307  
  
} // fin setup()
```

## Fonction **loop()**

- On commence par créer un objet **DateTime** représentant la date/heure actuelle, appelé ici now (=maintenant)
- Puis, par un jeu de **Serial.print()**, on affiche successivement l'année, le mois, le jour, l'heure, les minutes, les secondes...
- le code boucle après une pause d'une seconde

```
void loop () {  
    DateTime now = RTC.now(); // récupère l'heure courante  
  
    if (now.day()<10)Serial.print("0"),Serial.print(now.day()), Serial.print("/"); else Serial.print(now.day()), Serial.print("/");  
    if (now.month()<10)Serial.print("0"),Serial.print(now.month()), Serial.print("/"); else Serial.print(now.month()), Serial.print(" ");  
    Serial.print(now.year()), Serial.print(" ");  
  
    if (now.hour()<10)Serial.print("0"),Serial.print(now.hour()), Serial.print(":"); else Serial.print(now.hour()), Serial.print(":");  
    if (now.minute()<10)Serial.print("0"),Serial.print(now.minute()), Serial.print(":"); else Serial.print(now.minute()), Serial.print(":");  
    if (now.second()<10)Serial.print("0"),Serial.print(now.second()); else Serial.print(now.second());  
  
    Serial.println();  
    delay(1000); // pause  
}  
// fin loop
```

Vous allez me demander : « Mais comment le DS1307 se met-il à l'heure ? »

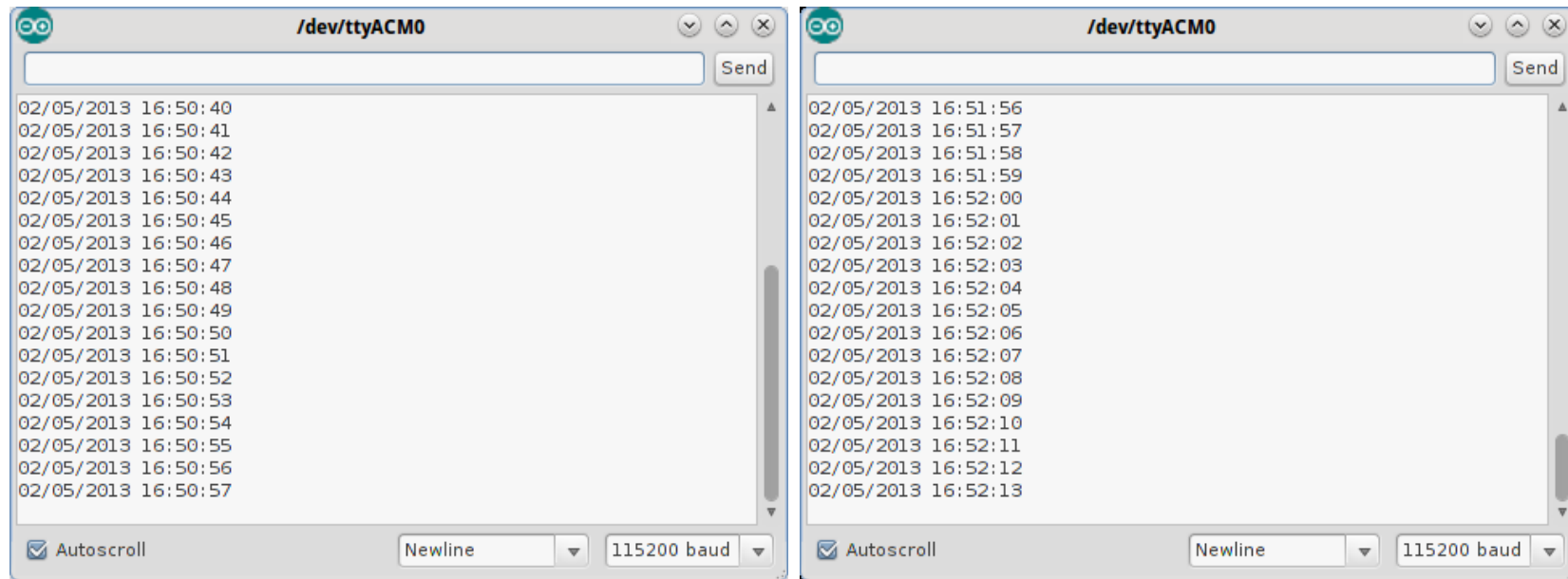
Et bien, très astucieusement, la librairie **RTCLib** va initialiser le **DS1307** avec l'heure du système de votre ordinateur au moment de la programmation.  
Simple et efficace !

La fonction **adjust()** permet au besoin de régler l'heure en cours d'exécution.



## Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 », ce qui donne ...



Débranchez votre Arduino, patientez un moment puis rebranchez-là : Arduino est toujours à l'heure !! Ceci grâce au DS1307 et à sa pile...

Simple et efficace non ?

Une solution polyvalente qui pourra être utilisée dans toutes les situations où il est nécessaire de gérer l'heure et la date réelle de façon précise et non volatile.



## 14. Temps-réel avec un DS1307 : Afficher le comptage des secondes

### Ce qu'on va faire ici...

- Bien, maintenant que vous nous avez testé notre librairie RTCLib et notre DS1307, nous allons commencer par quelque chose de très simple : compter et afficher les secondes. Ceci nous servira de base ensuite pour gérer la survenue d'événements à intervalles réguliers. Rien de sorcier, juste pour se mettre en jambe..
- Le principe va consister ici à se baser sur la fonction `unixtime()` de la librairie RTCLib qui donne le nombre de secondes écoulées depuis le 01/01/1970. Cette fonction s'apparente à la fonction `millis()` si l'on veut, sauf que le comptage se fait en seconde et est basé sur une date réelle. On va ici mémoriser la valeur renvoyée par `unixtime()` et on va attendre qu'une seconde se soit écoulée.
- Ce code est disponible ici : <https://gist.github.com/sensor56/572c9579ad100c2e1297>

### Entête déclarative

#### Inclusion des bibliothèques utiles

- On commence par inclure les bibliothèques
  - la bibliothèque `Wire.h` pour la communication I2C
  - la bibliothèque `RTCLib.h` pour la gestion simplifiée du temps réel avec le DS1307

#### Variables globales utiles

- On déclare :
  - une variable de mémorisation de la dernière seconde prise en compte
  - une variable de délai, ici 1 seconde
  - une variable de comptage des secondes

#### Objet utiles

- On déclare un objet `RTC_DS1307` représentant la base temps basée sur le DS1307 :

```
#include <Wire.h>
#include "RTCLib.h"

long seconde0=0; // variable pour mémoriser la dernière seconde
int delai=1; // delai de comptage en secondes
long comptSec=0; // variable de comptage des secondes écoulées

RTC_DS1307 RTC; // déclare objet représentant le DS1307
```

## Fonction **setup()**

### *Initialisation Série*

- On initialise la communication série à 115200 Bauds avec l'instruction **Serial.begin()**

### *Initialisation I2C*

- On initialise la communication I2C en mode « Arduino maître » avec l'instruction **Wire.begin()**

### *Initialisation « temps-réel »*

- On initialise le DS1307 à l'aide de la fonction **begin()** de l'objet RTC\_DS1307 précédemment déclaré

```
void setup () {  
  Serial.begin(115200); // initialise la communicatin série  
  Wire.begin(); // initialise I2C  
  RTC.begin(); // initialise le DS1307  
}
```

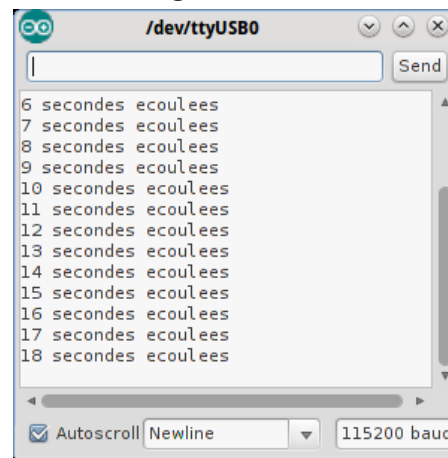
## Fonction `loop()`

- On commence par créer un objet `DateTime` représentant la date/heure actuelle, appelé ici `now` (=maintenant)
- A l'aide d'une condition `if ()` on teste si la différence entre la valeur courante de `unixtime()` et la dernière seconde prise en compte est supérieure au délai voulu : on teste en clair si le délai est écoulé.
- Si c'est le cas, on affiche un message, on mémorise la valeur courante de `unixtime()` et on incrémente la variable de comptage.
- Le programme boucle sans fin :

```
void loop () {  
    DateTime now = RTC.now(); // récupère l'heure courante  
  
    if (now.unixtime()-seconde0>=delai) { // si le délai est écoulé ou dépassé  
  
        Serial.print(comptSec);  
        Serial.println(" secondes ecoules");  
  
        seconde0=now.unixtime(); // mémorise nouvelle seconde courante  
        comptSec=comptSec+1; // incrémente variable comptage  
  
    } // fin if  
}  
// fin loop
```

## Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 », ce qui donne ...



Facile... !

## 15. Temps-réel avec un DS1307 : Exécuter une fonction à intervalle régulier : le programme

### Ce qu'on va faire ici...

- A présent, nous allons compliquer légèrement les choses : sur le même principe, nous allons appeler une fonction à intervalle régulier. Le délai d'appel de la fonction sera facilement définissable. Nous nous appuierons toujours ici sur la valeur de la fonction `unixtime()`.
- L'intérêt ici est de poser les bases pour beaucoup mieux : la gestion d'événement à effectuer à intervalles régulier, notamment des mesures. Nous verrons cela juste après...
- Ce code est disponible ici : <https://gist.github.com/sensor56/f9276d549832b846a271>

### Entête déclarative

#### Inclusion des bibliothèques utiles

- On commence par inclure les bibliothèques
  - la bibliothèque `Wire.h` pour la communication I2C
  - la bibliothèque `RTCLib.h` pour la gestion simplifiée du temps réel avec le DS1307

#### Variables globales utiles

- On déclare :
  - une variable de mémorisation de la dernière seconde prise en compte
  - une variable de délai, ici 1 seconde
  - une variable de comptage des secondes
- De la même façon, pour la gestion de l'événement à intervalle régulier, on déclare :
  - une variable de mémorisation de la dernière seconde prise en compte
  - une variable de délai, ici plusieurs secondes.

#### Objet utiles

- On déclare un objet `RTC_DS1307` représentant la base temps basée sur le DS1307 :

```
#include <Wire.h>
#include "RTCLib.h"

long seconde0=0; // variable pour mémoriser la dernière seconde
int delai=1; // delai de comptage en secondes
long comptSec=0; // variable de comptage des secondes écoulées

long seconde0Event=0; // variable pour mémoriser le dernier évènement
int delaiEvent=5; // delai en secondes pour appel fonction

RTC_DS1307 RTC; // déclare objet représentant le DS1307
```

## Fonction **setup()**

### *Initialisation Série*

- On initialise la communication série à 115200 Bauds avec l'instruction **Serial.begin()**

### *Initialisation I2C*

- On initialise la communication I2C en mode « Arduino maître » avec l'instruction **Wire.begin()**

### *Initialisation « temps-réel »*

- On initialise le DS1307 à l'aide de la fonction **begin()** de l'objet RTC\_DS1307 précédemment déclaré

```
void setup () {  
  Serial.begin(115200); // initialise la communicatin série  
  Wire.begin(); // initialise I2C  
  RTC.begin(); // initialise le DS1307  
}
```

## Fonction **loop()**

### **Affichage des secondes**

- On commence par créer un objet **DateTime** représentant la date/heure actuelle, appelé ici now (=maintenant)
- A l'aide d'une condition if () on teste si la différence entre la valeur courante de **unixtime()** et la dernière seconde prise en compte est supérieure au délai voulu : on teste en clair si le délai est écoulé.
- Si c'est le cas, on affiche un message, on mémorise la valeur courante de **unixtime()** et on incrémente la variable de comptage.

### **Gestion fonction à appeler à intervalle régulier**

- De la même façon que pour le comptage des secondes, à l'aide d'une condition if () on teste si la différence entre la valeur courante de **unixtime()** et la dernière seconde prise en compte est supérieure au délai voulu : on teste en clair si le délai est écoulé.
- Si c'est le cas, on appelle la fonction voulue, on mémorise la valeur courante de **unixtime()** et on incrémente la variable de comptage.
- Le programme boucle sans fin :

```
void loop () {  
    DateTime now = RTC.now(); // récupère l'heure courante  
  
    //--- affiche les secondes ---  
    if (now.unixtime()-seconde0>=delai) { // si le délai est écoulé ou dépassé  
  
        Serial.println(comptSec);  
  
        seconde0=now.unixtime(); // mémorise nouvelle seconde courante  
        comptSec=comptSec+1; // incrémente variable comptage  
  
    } // fin if  
  
    //----- appel fonction à intervalle régulier  
    if (now.unixtime()-seconde0Event>=delaiEvent) { // si le délai est écoulé ou dépassé  
  
        fonctionEvent(); // fonction à appeler  
  
        seconde0Event=now.unixtime(); // mémorise nouvelle seconde courante  
  
    } // fin if  
  
} // fin loop
```



## Fonction **fonctionEvent()**

- On définit enfin la fonction à appeler à intervalle régulier, que l'on pourra appeler comme on veut, ici `fonctionEvent()`. A ce niveau on se contente d'afficher un message mais on pourra prévoir de faire ce que l'on veut à ce niveau.

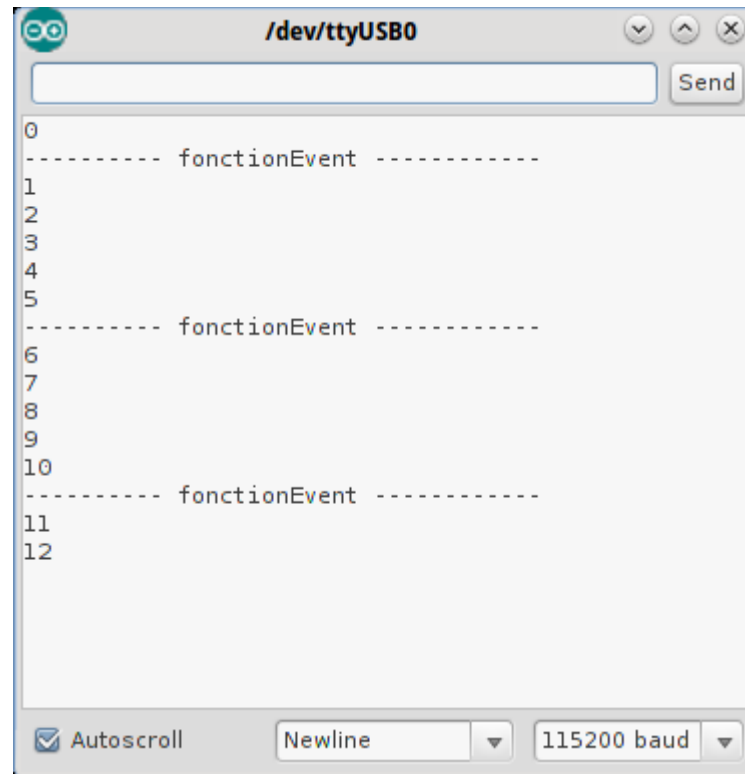
```
// fonction appelée à intervalle régulier
void fonctionEvent() {

    Serial.println("----- fonctionEvent -----");

} // fin fonctionEvent
```

## Fonctionnement du programme

- Une fois la carte Arduino programmée, ouvrir le Terminal Série en réglant sur « newline » et « 115200 », ce qui donne ...



La fonction est appelée toutes les 5 secondes... Plusieurs minutes d'intervalle sont possibles si on le souhaite.

## 16. Les éléments du langage Arduino étudiés dans cet atelier

- La librairie RTCLib

La documentation complète du langage Arduino en français est disponible ici :  
[http://www.mon-club-elec.fr/pmwiki\\_reference\\_arduino/pmwiki.php?n=Main.ReferenceMaxi](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi)

## **17. A présent, vous devriez être capable :**

- de mettre en place des applications Arduino utilisant l'heure et la date.

## Table des matières

Le temps avec Arduino : utiliser un module « Temps-réel » (ou RTC – Real Time Clock) (l'exemple du DS 1307) pour utiliser l'heure et la date avec Arduino.

[Intro](#) |

[Matériel nécessaire pour les ateliers Arduino](#) |

[Matériel spécifique nécessaire pour cet atelier](#) |

[Le « temps-réel » : introduction](#) |

[Temps réel : Première approche : Emuler une horloge « temps-réel » avec millis\(\)](#) |

[Arduino : Temps-réel : la librairie RTCLib](#) |

[Temps réel : Utiliser millis\(\) comme base « temps-réel » avec la librairie RTLib](#) |

[Le DS1307 : circuit intégré « Temps-réel » RTC \(Real Time Clock\) à communication I2C](#) |

[Utiliser le DS1307 sur une plaque d'essai](#) |

[Exemple d'utilisation d'un module RTC utilisant le DS1307 \(communication I2C\)](#) |

[Exemple de shield Arduino utilisant le DS1307 : L'étage RTC du shield « Mémoire » de chez Snootlab](#) |

[Temps-réel avec un DS1307 : le montage](#) |

[Temps-réel avec un DS1307 : test simple : le programme](#) |

[Temps-réel avec un DS1307 : Afficher le comptage des secondes](#) |

[Temps-réel avec un DS1307 : Exécuter une fonction à intervalle régulier : le programme](#) |

[Les éléments du langage Arduino étudiés dans cet atelier](#) |

[A présent, vous devriez être capable :](#) |

**Bravo !**  
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

[http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)