RTC DS3231

Réalisé par Bruno PIQUEMAL

24 septembre 2020

Ce guide vous permettra d'acquérir des connaissances sur le module Real Time Clock (RTC).

Introduction

Un module Real Time Clock (RTC) est un composant électronique qui permet de garder l'heure. Il est utile pour des applications qui nécessitent une cadence de cycles très précise ou pour envoyer des impulsions lorsque l'alarme programmée sonne. Cette dernière caractéristique peut s'avérer utile pour réveiller un Arduino endormi (deep sleep mode).

Table des matières

⋆ ⋆ ⋆

A Matériel

Pour ce tutoriel nous allons avoir besoin d'un Arduino UNO, d'une LED, d'une résistance de polarisation de 10K Ohms et du RTC DS3231.

B Foncionnement

Le RTC est un composant électronique qui permet de garder une heure précise. Il est possible de le régler de plusieurs manières dépendant de la librairie téléchargée.

L'intérêt de ces modules vient du fait qu'ils sont souvent accompagnés d'une pile qui permet de garder l'heure même lorsqu'ils ne sont plus alimentés par le microcontrôleur.

Si l'on veut réveiller un Arduino endormi (deep sleep mode), il est nécessaire d'avoir un RTC avec une alarme qui puisse déclencher une interruption. C'est pourquoi j'ai choisi un RTC avec une puce DS3231. Malgré tout, plusieurs modèles existent dans le commerce et il faut choisir le bon.

FIGURE 1 – Différents modèles de DS3231

Pour ce tutoriel, j'ai acheté le modèle de droite qui n'a pas la patte INT/SQW de la puce DS3231 reliée à un connecteur. C'est pourquoi je conseille l'achat du modèle de gauche qui a un connecteur SQW qui permet de générer des signaux carrés et de faire sortir le signal d'interruption de la puce DS3231.

B.1 Librairie à télécharger

Afin d'avoir les mêmes librairies et de pouvoir recopier le code, il faut avant tout télécharger cette librairie : `https://github.com/rodan/ds3231`

B.2 Réglage de l'heure

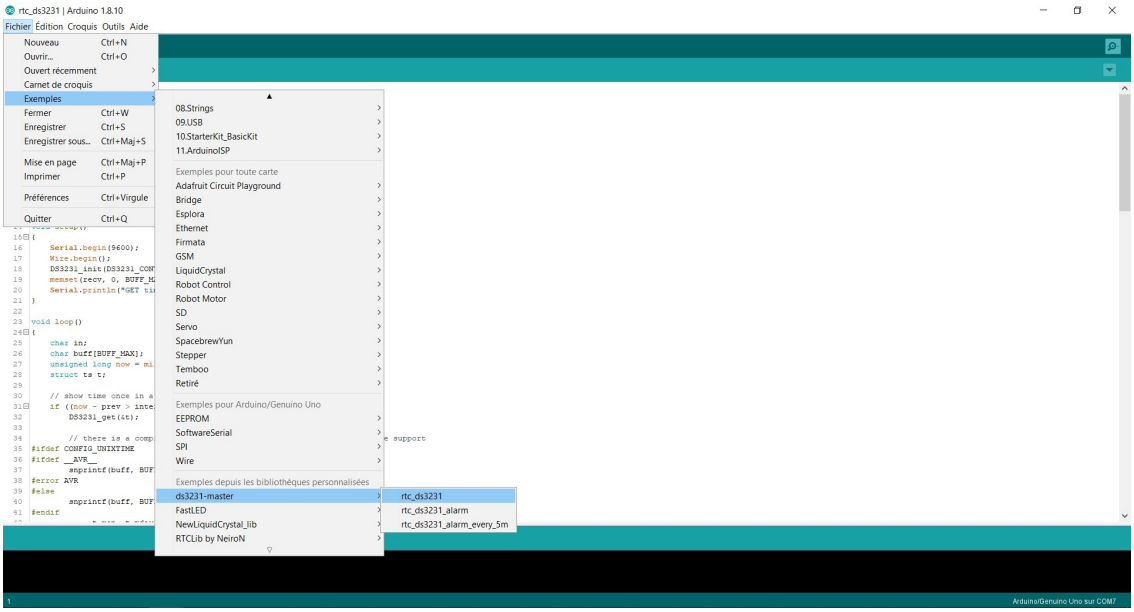

FIGURE 2 – Sketch qui permet de régler la puce DS3231

Il suffit de lancer le moniteur série et de mettre ceci : TssmmhhWDDMMYYYY Avec :

- ss : les secondes
- mm : les minutes
- hh : les heures (en mettant 08 pour 8 heures du matin par exemple)
- W : le jour de la semaine (mettre 1 pour lundi, 2 pour mardi, 3 pour mercredi…)
- DD : la date du jour
- MM : le mois
- YYYY : l'année (2020 par exemple)

B.3 Code

Codage

```
#include <Wire.h>
#include <ds3231.h>

#define wakePin 2 // when low, makes 328P wake up, must be an interrupt pin (2 or
    3 on ATMEGA328P)
#define ledPin 4 // output pin for the LED (to show it is awake)

// DS3231 alarm time
uint8_t wake_HOUR;
uint8_t wake_MINUTE;
uint8_t wake_SECOND;
#define BUFF_MAX 256

//// A struct is a structure of logical variables used as a complete unit
// struct ts {
//     uint8_t sec;         /* seconds */
//     uint8_t min;         /* minutes */
//     uint8_t hour;        /* hours */
//     uint8_t mday;        /* day of the month */
//     uint8_t mon;         /* month */
//     int16_t year;        /* year */
//     uint8_t wday;        /* day of the week */
//     uint8_t yday;        /* day in the year */
//     uint8_t isdst;       /* daylight saving time */
//     uint8_t year_s;      /* year in short notation */
//# ifdef CONFIG_UNIXTIME
//     uint32_t unixtime;   /* seconds since 01.01.1970 00:00:00 UTC*/
//#endif
// };
struct ts t;
```

On inclue la librairie <Wire.h> qui permet le dialogue en I2C avec la puce DS3231. La librairie <ds3231.h> est nécessaire pour parler à notre RTC.

Dans le code on présente la structure existante dans la deuxième librairie pour manipuler différentes données temporelles.

La LED est placée au pin 4.

Codage

```
1  void setup() {
2    Serial.begin(9600);
3
4  //Save Power by writing all Digital IO LOW
5  //— note that pins just need to be tied one way or another, do not damage devices!
6
7    for (int i = 0; i < 20; i++) {
8      if(i != 2)//just because the button is hooked up to digital pin 2
9      pinMode(i, OUTPUT);
10     digitalWrite(i,LOW);
11   }
12
13   pinMode(wakePin, INPUT_PULLUP);
14
15   // Flashing LED just to show the Controller is running
16   digitalWrite(ledPin, LOW);
17   pinMode(ledPin, OUTPUT);
18
19   // Clear the current alarm (puts DS3231 INT high)
20   Wire.begin();
21   DS3231_init(DS3231_CONTROL_INTCN);
22   DS3231_clear_a1f();
23
24   Serial.println("Setup completed.");
25 }
```

Afin d'afficher des données dans le moniteur série, il faut mettre la fonction Serial.begin() puis pour économiser de l'énergie, on met tous les pins en OUTPUT et LOW sauf le pin 2 qui va nous servir lors de l'interruption.

Le pin 2 est mis dans un mode INPUT PULLUP pour activer la résistance interne de l'ATMEGA 328P et ainsi garder ce pin en état élevé (HIGH) et détecter une interruption lorsqu'il passera en état bas (LOW).

Les dernières instructions sont là pour mettre en place le module RTC et supprimer des alarmes anciennement prédéfinies.

Codage

```
1  // The loop blinks an LED when not in sleep mode
2  void loop() {
3    char buff[BUFF_MAX]; //to print the time
4
5    // Just blink LED ONCE to show we're running
6    digitalWrite(ledPin, HIGH);
7    delay(500);
8    digitalWrite(ledPin, LOW);
9
10     // Set the DS3231 alarm to wake up in X seconds
11     setNextAlarm();
12
13     //Disable ADC - don't forget to flip back
14     // after waking up if using ADC in your application ADCSRA |= (1 << 7);
15     ADCSRA &= ~(1 << 7);
16
17     //ENABLE SLEEP - this enables the sleep mode
18       SMCR |= (1 << 2); //power down mode
19       SMCR |= 1;//enable sleep
20
21     attachInterrupt(digitalPinToInterrupt(wakePin), sleepISR,
          FALLING);
22
23     // Send a message just to show we are about to sleep
24     Serial.println("Good night!");
25     // display current time
26     snprintf(buff, BUFF_MAX, "%d.%02d.%02d %02d:%02d:%02d\n", t.
          year,
27     t.mon, t.mday, t.hour, t.min, t.sec);
28     Serial.print(buff);
29     Serial.flush();
```

Dans la boucle void loop(), on commence par déclarer une variable pour afficher l'heure à laquelle l'Arduino s'endort et se réveille. Puis, on allume la LED pendant 500 ms, on l'éteint et on met en place une alarme de 10 secondes (la fonction setNextAlarm() sera présentée après).

On configure l'Arduino avec les registres pour qu'il consomme peu et qu'il puisse s'endormir. Ensuite, on lui prévient qu'une interruption va avoir lieu sur le pin 2 (pin 2 et pin 3 sont prévus pour les interruptions).

On affiche l'heure actuelle puis on attend que l'affichage complet se fasse sur le moniteur série grâce à la commande Serial.flush().

Codage

```
    //BOD DISABLE - this must be called right before the __asm__ sleep instruction

     MCUCR |= (3 << 5); //set both BODS and BODSE at the same ti timeme
     MCUCR = (MCUCR & ~(1 << 5)) | (1 << 6); //then set the BODS bit and
         clear the BODSE bit at the same

     // And enter sleep mode as set above
      __asm__ __volatile__("sleep");//in line assembler to go to sleep

    // ------------------------------------
    //  Controller  is now asleep until woken up by an interrupt
    // ------------------------------------

    // Wakes up at this point when wakePin is brought LOW - interrupt routine is
        run first
    Serial.println("I'm awake!");

    ADCSRA &= ~(0 << 7);

    // Clear existing alarm so int pin goes high again
    DS3231_clear_a1f();
}

// When wakePin is brought LOW this interrupt is triggered FIRST (even in
    PWR_DOWN sleep)
void sleepISR() {

  // Detach the interrupt that brought us out of sleep
  detachInterrupt(digitalPinToInterrupt(wakePin));

  // Now we continue running the main Loop() just after we went to sleep
}
```

Le BOD se charge de contrôler la tension de la source ce qui est inutile pendant la phase d'endormissement, c'est pourquoi on le désactive. Puis l'Arduino s'endort.

Lorsque l'Arduino se réveille, on remet en place le régulateur de tension (ADCSRA) et on supprime l'alarme.

La fonction sleepISR() joue le rôle d'interruption lorsqu'elle est appelée par attachInterrupt(). Elle agit de manière indépendante de l'état du programme et permet d'exécuter des tâches en parallèle. Dans notre cas, une fois qu'elle tourne, on bypass le pin 2 avec la fonction detachInterrupt() afin d'éviter de créer une nouvelle interruption. *(ISR veut dire Interrupt Service Routine)*

Codage

```
1  // Set the next alarm
2  void setNextAlarm(void)
3      {
4    // flags define what calendar component to be checked against the current time in
         order
5    // to trigger the alarm - see datasheet
6    // A1M1 (seconds) (0 to enable, 1 to disable)
7    // A1M2 (minutes) (0 to enable, 1 to disable)
8    // A1M3 (hour)    (0 to enable, 1 to disable)
9    // A1M4 (day)     (0 to enable, 1 to disable)
10   // DY/DT          (dayofweek == 1/dayofmonth == 0)
11   uint8_t flags[5] = { 0, 0, 0, 1, 1 };
12
13   // get current time so we can calc the next alarm
14   DS3231_get(&t);
15
16   wake_SECOND = t.sec;
17   wake_MINUTE = t.min;
18   wake_HOUR = t.hour;
19
20   // Add a some seconds to current time. If overflow increment minutes etc.
21   wake_SECOND = wake_SECOND + 10;
22   if (wake_SECOND > 59)
23   {
24     wake_MINUTE++;
25     wake_SECOND = wake_SECOND - 60;
26
27     if (wake_MINUTE > 59)
28     {
29       wake_HOUR++;
30       wake_MINUTE -= 60;
31     }
32   }
33
34   // Set the alarm time (but not yet activated)
35   DS3231_set_a1(wake_SECOND, wake_MINUTE, wake_HOUR, 0, flags);
36
37   // Turn the alarm on
38   DS3231_set_creg(DS3231_CONTROL_INTCN | DS3231_CONTROL_A1IE);
39 }
```

La fonction setNextAlarm() sert à mettre en place une alarme. On commence par choisir le type d'alarme souhaité et on place le temps actuel dans la structure globale t.

On prend les valeurs principales (secondes, minutes, heures) puis on ajoute 10 secondes à la variable wake SECOND. Ensuite on s'assure qu'il n'y ait pas de dépassement (59 + 10 = 69, donc pas possible).

On modifie la valeur de l'alarme dans le module RTC puis on l'active. On attend que les 10 secondes s'écoulent pour créer une interruption et réveiller l'Arduino.

C Pour aller plus loin

Le module RTC DS3231 possède deux alarmes, la première a pour calibre le plus bas les secondes alors que la deuxième alarme c'est les minutes. Toutefois, elles peuvent être réglées pour être comparées avec les autres paramètres tels que les jours, les années....

Voici deux tutoriels pour de la très basse consommation et du réveil par interruption :

`https://www.youtube.com/watch?v=GvZCmH6BFHw`

`https://www.youtube.com/watch?v=-dW4XsBo3Mk`

D Bibliographie

`https://datasheets.maximintegrated.com/en/ds/DS3231.pdf`

`http://gammon.com.au/interrupts`

`https://www.kevindarrah.com/download/arduino_code/LowPowerVideo.ino`

`https://github.com/RalphBacon/192-DS3231-Wake-Up-Arduino/blob/master/YouTube%20Sketch/Arduino_DS3231_Wakeup.cpp`