UE INF-IoT
« Internet of Things »

Sujet de TP : Interaction entre un MCU et une horloge RTC via le standard de bus I^2C

L'objectif de cette séance de travaux pratiques est d'étudier comment peut s'effectuer l'interaction entre un micro-contrôleur (MCU) ou micro-processeur (MPU) et un équipement périphérique via le standard de bus I^2C *(Inter-Integrated Circuit)*.

Dans le cas présent, vous allez utiliser un MCU (le modèle exact vous sera indiqué en cours de séance), et allez le connecter à une horloge temps-réel (RTC : *Real-Time Clock)* basée sur un circuit DS3231 [1]. Vous allez ensuite développer un petit programme permettant de régler cette horloge, ou de lire son état.

1 Aperçu du circuit imprimé ZS-042 et du circuit intégré DS3231

Le circuit imprimé ZS-042 (voir figure 1) porte un circuit intégré DS3231, et permet d'alimenter ce circuit soit à partir d'une alimentation externe connectée aux broches VCC et VND, soit à partir d'une batterie LIR2032 (batterie Li-Ion rechargeable [2]) clipsée au dos du circuit imprimé. Lorsque ce circuit imprimé est intégré dans un montage, il est alimenté par les broches VCC/GND tant que le montage fonctionne. La batterie LIR2032, si elle est présente, se recharge alors lentement pendant cette période. Lorsque le montage ne fonctionne pas, c'est la batterie LIR2032 qui assure l'alimentation du circuit intégré DS3231. Celui-ci est donc constamment alimenté, ce qui est bien sûr un pré-requis pour qu'une horloge puisse fonctionner en continu.

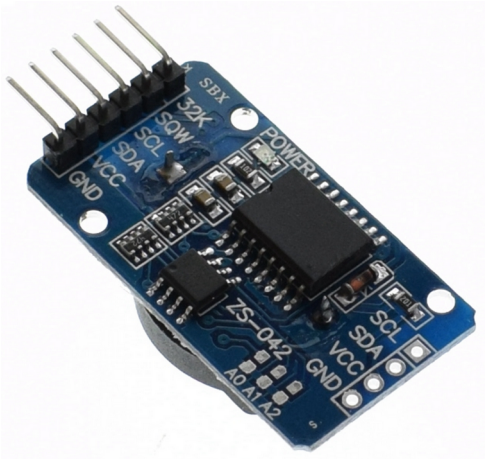

FIGURE 1 – Circuit imprimé ZS-042 supportant un circuit intégré DS3231.

Le circuit intégré DS3231 constitue une horloge temps-réel de précision, avec compensation de la température. En conditions normales d'utilisation (i.e., à température ambiante) la dérive de cette horloge ne devrait pas dépasser 2 ppm, soit environ 0.17 seconde par jour. Outre sa fonction principale d'horloge, le circuit DS3231 fournit également deux alarmes journalières programmables, et peut produire un signal carré pour cadencer le fonctionnement d'autres circuits.

Le circuit DS3231 peut communiquer avec d'autres équipements (et notamment un MCU ou MPU) via le standard de bus I^2C. Pour ce faire le circuit imprimé ZS-042 offre les deux broches SDA (circuit de données) et SCL (circuit d'horloge) caractéristiques de la communication I^2C. Par construction, le circuit DS3231 est accessible à l'adresse 0x68.

1. Circuit DS3231 : `https://datasheets.maximintegrated.com/en/ds/DS3231.pdf`
2. Attention à ne pas confondre une batterie LIR2032, qui est rechargeable, avec une pile CR2032, qui ne l'est pas ! Une pile CR2032 placée dans le support du circuit imprimé ZS-042 risquerait de chauffer, voire d'exploser, lorsque ce circuit est alimenté par les broches VCC/GND.

2 Montage

Vous allez réaliser un montage simple pour connecter le circuit imprimé ZS-042 à un micro-contrôleur. Il vous suffit de connecter les broches VCC, GND, SDA, et SCL du ZS-042 aux broches correspondantes du micro-contrôleur (voir figure2). Veillez à bien alimenter le ZS-042 en 3.3 V, et pas en 5 V !

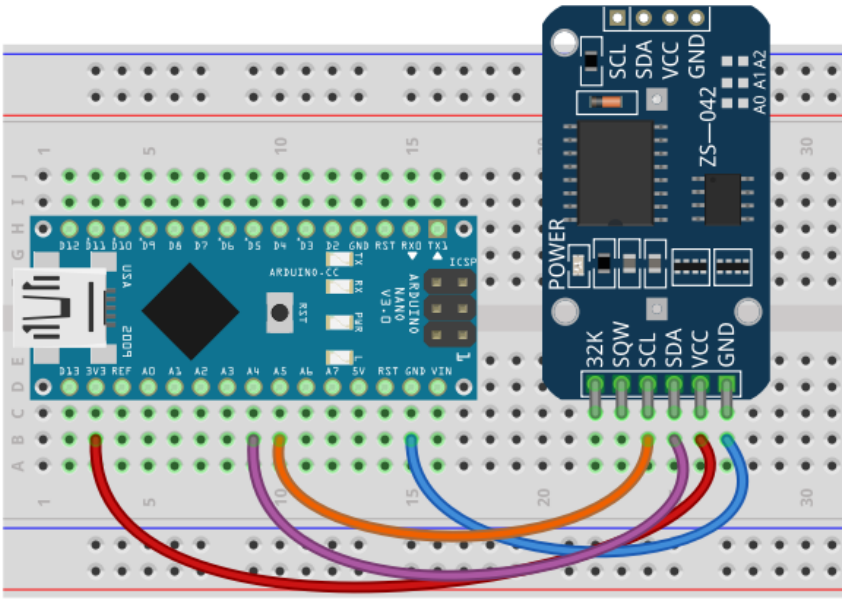

FIGURE 2 – Schéma du montage sur plaque d'expérimentation *(breadboard)*

3 Développement d'un programme de test

3.1 Objectif

En utilisant l'éditeur Visual Studio Code avec l'extension PlatformIO, vous allez développer un petit programme permettant :

1. de consulter les registres du DS3231 et d'afficher les valeurs de ces registres en format binaire ;
2. de consulter les registres du DS3231 et d'afficher la date et l'heure en conséquence ;
3. de saisir via la console l'heure actuelle (exprimée en nombre de secondes écoulées depuis le 01/01/1970 00:00:00 [3]) et de fixer en conséquence les valeurs des registres du DS3231.

Ce programme sera développé en utilisant le Framework Arduino, et notamment la librairie `Wire` [4] qui fournit les fonctions élémentaires de communication I^2C. Les accès aux registres de l'horloge DS3231 devront s'effectuer **uniquement** grâce aux fonctions de la librairie `Wire` : vous n'êtes pas censé(e) utiliser une autre librairie spécialisée permettant le contrôle d'une horloge RTC : de telles librairies existent, mais l'objectif est justement ici d'apprendre à vous en passer.

3.2 Démarche

Pour vous faire gagner du temps, le squelette du programme que vous devez réaliser est mis à votre disposition. Dans ce squelette les fonctions `setup()` et `loop()` sont déjà définies, ainsi que quelques fonctions annexes dont vous aurez besoin.

La fonction `loop()` gère principalement l'interaction avec un utilisateur via la console : l'utilisateur peut consulter l'état des registres de l'horloge DS3231 et en afficher l'état sous forme binaire ou développée, et il peut également modifier l'état de ces registres en indiquant l'heure actuelle. La figure 3 illustre le comportement attendu au niveau de la console.

3. Le 01/01/1970 00:00:00 constitue la date de référence, appelée date EPOCH, à partir de laquelle est décompté le temps dans les systèmes Unix. D'autres systèmes (dont le framework Arduino) utilisent d'autres dates de référence.
4. `https://www.arduino.cc/en/Reference/Wire`

```
Possible commands:
?       : Read the RTC's registers and display the current time
          in human-readable form
@       : Read the RTC's registers and display each value in binary form
!<value>: Set the RTC's registers based on the specified value,
          where <value> is the number of seconds elapsed
          since 1970/01/01 00:00:00 (EPOCH)

?
Current RTC time is Fri Mar 22 23:00:25 2013

!1530015159
Setting RTC to Tue Jun 26 12:12:39 2018

?
Current RTC time is Tue Jun 26 12:12:41 2018

@
Displaying RTC's registers:
sec : 01000011
min : 00010010
hour: 00010010
wday: 00000011
mday: 00100110
mon : 00000110
year: 00011000
```

FIGURE 3 – Illustration de l'interaction avec le programme via la console

Les fonctions `showRegisters()`, `getTime()`, et `setTime()` sont incomplètes. À vous de les compléter pour atteindre l'objectif visé. Poue ce faire vous aurez besoin de connaître les adresses et formats des différents registres de l'horloge DS3231. Ces informations sont disponibles, parmi bien d'autres, dans la fiche technique [5] *(datasheet)* de ce circuit.

3.3 Quelques remarques

Structure tm

Les fonctions `getTime()` et `setTime()` prennent en paramètre un pointeur sur une structure `tm`. Cette structure est la structure C standard pour représenter le temps sous une forme décomposée en champs tels que l'année, le mois, le jour, etc. Elle est notamment utilisée par les fonctions C `asctime()` et `mktime()`. Une description de la structure `tm` est disponible au début du squelette de programme qui vous est fourni.
Notez que dans cette structure le champ `tm_mon` est censé porter des valeurs prises dans l'intervalle [0..11], alors que le registre correspondant sur l'horloge DS3231 doit porter des valeurs prises dans l'intervalle [1..12]. Des nuances semblables peuvent exister pour d'autres champs.

Nombre de secondes écoulées depuis EPOCH

Pour fixer l'heure depuis la console (avec l'option préfixée '!'), vous aurez besoin de pouvoir déterminer combien de secondes se sont écoulées depuis la date de référence EPOCH. Cette information peut vous être fournie par la commande Unix `date`, avec l'option adéquate :

```
% date +%s
1530012392
```

Par défaut cette commande s'appuie sur l'heure UTC, et non sur l'heure locale. Pour tenir compte de l'heure locale en France il faut ajouter 3600 ou 7200 secondes, selon qu'on est en heure d'hiver ou en heure d'été :

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

```
% echo $(($(date +%s) + 7200))
1530019900
```

Autres informations utiles

— La date de référence est le 01/01/1970 00:00:00 sur les systèmes Unix. Dans le framework Arduino, c'est le 01/01/2000 00:00:00.

— La plupart des registres de l'horloge DS3231 contiennent des valeurs au format BCD *(Binary Coded Digital)*. Ce format de représentation, très utilisé en électronique numérique, permet de coder une valeur entière décimale sur un ou plusieurs octets, chaque chiffre de la valeur décimale étant lui-même codé sur quatre bits (un *nibble*) dans la représentation binaire. Ainsi l'entier 53 (en représentation décimale) est codé 01010011 en représentation BCD : le chiffre '5' de la représentation décimale est codé sur les quatre bits de poids fort (soit '0101'), et le chiffre '3' est codé sur les quatre bits de poids faible (soit '0011').

Squelette du programme

Les fonctions `showRegisters()`, `getTime()`, et `setTime()` doivent être complétées.

```
#include <Arduino.h>
#include <Wire.h>
#include <time.h>

// struct tm {
//                 int tm_sec;     /* Seconds (0-60) */
//                 int tm_min;     /* Minutes (0-59) */
//                 int tm_hour;    /* Hours (0-23) */
//                 int tm_mday;    /* Day of the month (1-31) */
//                 int tm_mon;     /* Month (0-11) */
//                 int tm_year;    /* Year - 1900 */
//                 int tm_wday;    /* Day of the week (0-6, Sunday = 0) */
//                 int tm_yday;    /* Day in the year (0-365, 1 Jan = 0) */
//                 int tm_isdst;   /* Daylight saving time */
//             };

#define DS3231_ADDR  0x68   // I2C address of the DS3231 RTC
#define SECONDS_FROM_1970_TO_2000 946684800

// ------------------------------------------------------------------------
// Get a command from the console and return this command as a string
//
// WARNING: this function is blocking until a command has been read
// (i.e., until a '\n' character has been received).
String getCommand() {

  Serial.print("> );

  static char buffer[64];
  int line_idx = 0;

  while (true) {
    while (Serial.available() > 0) {
      char c = Serial.read();
      Serial.print(c);

      if (c == '\b') {
        if (line_idx > 0) {
          Serial.print(' '); Serial.print('\b');
          line_idx--;
        }
      }
      else if (c == '\n') {
        buffer[line_idx-1] = 0;
        return String(buffer);
      }
      else
        buffer[line_idx++] = c;
    }
  }
}

// ------------------------------------------------------------------------
// Convert an integer value (encoded on 8 bits) to BCD format (i.e. binary
// Coded Decimal)
static uint8_t int2bcd(uint8_t val) { return ((val/10*16) + (val%10)); }

// ------------------------------------------------------------------------
// Convert a BCD (i.e., Binary Coded Decimal) value to an integer value
```

```
60 // (encoded on 8 bits)
61 static uint8_t bcd2int(uint8_t val) { return ((val/16*10) + (val%16)); }
62
63 // ------------------------------------------------------------------------
64 // Display v in binary form
65 void showBits(uint8_t v) {
66
67   for (int i=0; i<8; i++) {
68     Serial.print((v & 0x80) != 0);
69     v = v << 1;
70   }
71 }
72
73 // ------------------------------------------------------------------------
74 // Read the DS3231's registers and display their values in binary form
75 void showRegisters() {
76
77   Serial.println("TO BE COMPLETED");
78 }
79
80 // ------------------------------------------------------------------------
81 // Read the date and time from the DS3231 and set dt's structure accordingly
82 void getTime(tm *dt) {
83
84   Serial.println("TO BE COMPLETED");
85 }
86
87 // ------------------------------------------------------------------------
88 // Set the date and time in the DS3231 based on the values in dt's structure
89 void setTime(tm *dt) {
90
91   Serial.println("TO BE COMPLETED");
92 }
93
94 // ------------------------------------------------------------------------
95 void process_command() {
96
97   String cmd = getCommand();
98   cmd.trim();
99
100  if (cmd.startsWith("!")) {
101    String str = cmd.substring(1);
102    time_t val = (time_t)(str.toInt() - SECONDS_FROM_1970_TO_2000);
103    struct tm *dt;
104    dt = localtime(&val);
105    setTime(dt);
106    Serial.print("Setting RTC to ");
107    Serial.println(asctime(dt));
108    Serial.println();
109    return;
110  }
111
112  if (cmd.startsWith("?")) {
113    struct tm dt;
114    getTime(&dt);
115    Serial.print("Current RTC time is ");
116    Serial.println(asctime(&dt));
117    Serial.println();
118    return;
119  }
120
121  if (cmd.startsWith("@")) {
122    showRegisters();
```

```
123     Serial.println();
124     return;
125   }
126
127   Serial.println("Invalid command");
128   Serial.println();
129 }
130
131 // -----------------------------------------------------------------------
132 void setup_RTC () {
133
134   // Enable the I2C bus
135   Wire.begin();
136
137   Serial.println();
138   Serial.println("F(Possible commands:"));
139   Serial.println();
140   Serial.println(F("?       : Read the RTC's registers and display the current time in human-
141   Serial.println(F("@       : Read the RTC's registers and display each value in binary form"
142   Serial.println(F("!<value>: Set the RTC's registers based on the specified value,"));
143   Serial.println(F("          where <value> is the number of seconds elapsed"));
144   Serial.println(F("          since 1970/01/01 00:00:00 (EPOCH)"));
145   Serial.println();
146 }
147
148 // -----------------------------------------------------------------------
149 void setup() {
150
151   // Enable the serial console
152   Serial.begin(9600);
153
154   setup_RTC();
155 }
156
157 // -----------------------------------------------------------------------
158 void loop () {
159
160   process_command();
161 }
```