```
/*  ex_i2cB - Noeud B
 *  Ce programme est un exemple de communication I2C
 *  entre un coordonnateur (Pi) et un neoud (Arduino).
 *  Ce noeud est capable de transférer vers le coordonnateur:
 *
 *    - la valeur de la température interne;
 *    - le numéro de l'échantillon.
 *
 *  De plus, le noeud est capable de recevoir les commandes suivantes
 *  du coordonnateur:
 *    - STOP: arrêter l'échantillonnage;
 *    - GO:   démarrer l'échantillonnage.
 *
 *  Dans cet exemple, l'arrêt de l'échantillonnage remet à zéro le numéro
 *  de l'échantillon.
 *
 *  GPA788 Conception et intégration des objets connectés
 *  T. Wong
 *  06/2018
 *  08/2020
 *  11/2021
 */
#include <Wire.h>                       // Pour la communication I2C
#include <util/atomic.h>                // Pour la section critique
#include "ChipTemp.h"                   // Pour lire la température du microcontrôleur
#include "calculateur_leq.h"            // Pour lire le son

/* ----------------------------------------------------------------
   Globales pour la classe ChipTemp
   ---------------------------------------------------------------- */
const float DECALAGE{324.31};           // Choisir la bonne: 316.0, 324.31, 332.70, 335.2
const float GAIN{1.22};                 // Choisir la bonne: 1.06154, 1.22
ChipTemp ct(GAIN, DECALAGE);     // Instancier un objet de classe ChipTemp

/* ----------------------------------------------------------------
   Globales pour la classe calculateurLeq
   ---------------------------------------------------------------- */
const uint8_t PIN{A0};                      // Broche du Capteur sonore
const uint32_t TS = 62;                     // Péruide d'échantillionnage (ms)
const uint16_t NB_SAMPLE = 32;              // 32 x 62 ms ~ 2 secondes
const uint32_t NB_LI = 3;                  // 150 * 2 sec = 5 min
uint32_t countMillis;                       // Compter les minutes (pour debug seulement)


/* ----------------------------------------------------------------
   Globales pour la communication I2C
   ---------------------------------------------------------------- */
const uint8_t ADR_NOEUD{0x45};        // Adresse I2C du noeud
const uint8_t NB_REGISTRES{7};        // Nombre de registres de ce noeud



/* La carte des registres ------------------------------------------ */
union CarteRegistres {
  // Cette structure: Utilisée par le noeud pour lire et écrire
  //                  des données.
```

```
  struct {
    // Taux d'échantillonnage (1 octet)
    volatile uint8_t Ts;
    // Nombre d'échantillons (2 octets)
    volatile int16_t nb_echantillons;
    // Température interne du processeur ATMEGA en celsius
    // (4 octets)
    volatile float son;
  } champs;
  // Ce tableau: Utilisé par le coordonnateur pour lire et écrire
  //             des données.
  uint8_t regs[NB_REGISTRES];
};

union CarteRegistres cr;               // Une carte des registres
float intensiteSon;                     // Variable intermédiaire pour mémoriser les
db
uint8_t adrReg;                        // Adresse du registre reçue du coordonnateur
enum class CMD { Stop = 0, Go, pause};        // Commandes venant du coordonnateur
volatile CMD cmd;                      // Go -> échantilloner, Stop -> arrêter
const uint8_t MIN_Ts = 5;              // Période d'échantillonnage min (sec)
const uint8_t MAX_Ts = 200;            // Période d'échantillonnage max (sec)

//Création de l'objet
Calculateur_Leq leq(TS, NB_SAMPLE, NB_LI);

/* ----------------------------------------------------------------
   Initialisation
   ---------------------------------------------------------------- */
void setup()
{
  // Pour la communication série
  Serial.begin(115200);

  // Sur le VS Code, l'ouverture du port série prend du temps et on
  // peut perdre des caractères. Ce problème n'existe pas sur l'IDE
  // de l'Arduino.
  waitUntil(2000);

  // Pour l'ADC du microcontrôleur...
  analogReference(EXTERNAL);                // utiliser VREF externe pour l'ADC
  pinMode(PIN, INPUT);

  // Initialiser les champs de la carte des registres
  cr.champs.Ts = MIN_Ts;
  cr.champs.nb_echantillons = 0;
  cr.champs.son = -1;
  intensiteSon = -1;
  // Initialiser les variables de contrôle de la
  // communication I2C
  cmd = CMD::Stop;
  adrReg = -1;
  // Réglage de la bibliothèque Wire pour le I2C
  Wire.begin(ADR_NOEUD);
  // Fonction pour traiter la réception de données venant du coordonnateur
  Wire.onReceive(i2c_receiveEvent);
  // Fonction pour traiter une requête de données venant du coordonnateur
  Wire.onRequest(i2c_requestEvent);

  // Indiquer que le noeud est prêt
```

```
116   Serial.print(F("Noeud à l'adresse 0x")); Serial.print(ADR_NOEUD, HEX);
117   Serial.println(F(" prêt à recevoir des commandes"));
118 }
119
120 /* ------------------------------------------------------------------
121    Boucle principale
122    ------------------------------------------------------------------ */
123 void loop()
124 {
125   // Échantillonner la température interne si la commande est Go
126   if (cmd == CMD::Go) {
127     // L'objet leq "sait" à quel moment il doit accumuler les valeurs
128   // du signal sonore.
129   leq.Accumulate();
130   //Serial.println(leq.GetTotalSamples());
131   // L'objet leq sait à quels moments il faut calculer Vrms, Li et Leq
132   if (leq.Compute() ) {
133     intensiteSon = leq.GetLeq();
134
135     // Section critique: empêcher les interruptions lors de l'assignation
136     // de la valeur de la température à la variable dans la carte des registres.
137     // Recommandation: réaliser la tâche la plus rapidement que possible dans
138     //                 la section critique.
139       ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
140       // Assigner la température lue dans cr.champs.temperature
141         cr.champs.son = intensiteSon;
142       // Augmenter le compte du nombre d'échantillons
143         cr.champs.nb_echantillons++;
144       }
145       Serial.print(F("#")); Serial.print(cr.champs.nb_echantillons);
Serial.print(F("  "));
146       Serial.print("Leq : "); Serial.print(cr.champs.son);
147       Serial.print(" Vrms: "); Serial.print((((float)TS)/1000) *
leq.GetVrmSamples()); Serial.println(F(" secondes"));
148       Serial.print(" Li: "); Serial.print((((float)TS)/1000) * leq.GetVrmSamples() *
leq.GetLiSamples()); Serial.println(F(" secondes"));
149
150       // Attendre la prochaine période d'échantillonnage
151       //waitUntil(cr.champs.Ts * 1000);
152     }
153   }
154 }
155
156 /* ------------------------------------------------------------------
157    i2c_receiveFunc(int n)
158    Cette fonction est exécutée à la réception des données venant
159    du coordonnateur. Le paramètre n indique le nombre d'octets reçus.
160    ------------------------------------------------------------------
161    Note: Normalement on ne doit pas afficher des messages utilisant
162          le port série - il y a risque de conflit entre les inter-
163          ruptions. Donc, après débogage, n'oubliez pas de mettre les
164          Serial print en commentaires ;-).
165    ------------------------------------------------------------------ */
166 void i2c_receiveEvent(int n) {
167   // Traiter les commandes ou les adresses de registre (1 octet)
168   if (n == 1) {
169     // Un seul octet reçu. C'est probablement une commande.
170     uint8_t data = Wire.read();
171     switch (data) {
172       case 0xA1:
```

```
173         cmd = CMD::Stop;
174         cr.champs.nb_echantillons = 0;
175         Serial.println(F("commande 'Arrêter' reçue"));
176         break;
177     case 0xA2:
178         cmd = CMD::Go;
179         Serial.println(F("Commande 'Démarrer' reçue"));
180         break;
181     case 0xA5:
182         cmd = CMD::pause;
183         Serial.println(F("Commande 'Pause' reçue"));
184         break;
185     case 0xA6:
186         cmd = CMD::Go;
187         Serial.println(F("Commande 'Redémarrer' reçue"));
188         break;
189     case 0xA7:
190         Serial.println(F("Un courriel a été envoyé suite au crash du programme"));
191         break;
192     default:
193         // Sinon, c'est probablement une adresse de registre
194         if ((data >= 0) && (data < NB_REGISTRES)) {
195           adrReg = data;
196         }
197         else
198           adrReg = -1; // Il y sans doute une erreur!
199     }
200   }
201   else if (n == 2) {
202     // Deux octets reçus. C'est probablement pour changer un parametre.
203     uint8_t data1 = Wire.read();
204     uint8_t data2 = Wire.read();
205     switch (data1) {
206       case 0xA0:
207         Serial.println(F("Commande 'Changer Ts' reçue"));
208         if ((data2 >= MIN_Ts) && (data2 <= MAX_Ts)) {
209           cr.champs.Ts = data2;
210           Serial.print(F("La nouvelle valeur est: "));
211           Serial.print(cr.champs.Ts); Serial.println(F(" secondes"));
212         }
213         break;
214       case 0xA3:
215         Serial.println(F("Commande 'Changer nVrm' reçue"));
216         if (data2 >= 1) {
217           leq.SetNbVrmsSamples(data2);
218           Serial.print(F("La nouvelle valeur est: "));
219           Serial.print(leq.GetVrmSamples()); Serial.println(F(" nombre de Vrms
    échantillons"));
220         }
221         break;
222       case 0xA4:
223         Serial.println(F("Commande 'Changer nLi' reçue"));
224         if (data2 >= 1) {
225           leq.SetNbLiSamples(data2);
226           Serial.print(F("La nouvelle valeur est: "));
227           Serial.print(leq.GetLiSamples()); Serial.println(F(" nombre de Li
    échantillons"));
228         }
229         break;
230
```

```
    }
  }
  else {
    // Ignorer la réception n > 2 octets.
    Serial.println(F("Erreur: ce noeud n'accepte\
    pas de communication/commande à trois octets"));
  }
}

/* ---------------------------------------------------------------
   i2c_requestEvent()
   Cette fonction est exécutée à la réception d'une requête de
   données venant du coordonnateur.
   ---------------------------------------------------------------
   Note: Normalement on ne doit pas afficher des messages utilisant
         le port série - il y a risque de conflit entre les inter-
         ruptions. Donc, après débogage, n'oubliez pas de mettre les
         Serial print en commentaires ;-).
   --------------------------------------------------------------- */
void i2c_requestEvent(){
  // Le coordonnateur veut la valeur d'un registre. L'adresse du
  // registre a été reçue précédemment.
  if ((adrReg >= 0) && (adrReg < NB_REGISTRES)){
    Serial.print("");
    // Envoyer le contenu du registre au coordonnateur
    Wire.write(cr.regs[adrReg]);
  }
}

/* ---------------------------------------------------------------
   Fonction pour créer un delai de w millisecondes

   La fonction delay() est utilisée dans bien des tutoriels pour
   créer un delai temporel. On peut aussi créer notre propre délai
   et utiliser une unité de temps à notre guise.
   --------------------------------------------------------------- */
void waitUntil(uint32_t w) {
  uint32_t t{millis()};
  // Attendre w millisecondes
  while (millis() < t + w) {}
}

```