Projet Electronique

Conception d'un télémètre à ultrason

Igor Vandervelden
Robin Castermane
Mugisha Tuyishime Rodrigue
Victor Cotton

Rapport Final

Année Académique 2019-2020 Groupe n°8

Table des matières

Composition du groupe

Notre groupe n°8 est composé de Igor Vandervelden (HE201536), Robin Castermane (HE201674), Mugisha Tuyishime Rodrigue (HE201493) et Victor Cotton (HE201662).

Enseignants responsables du projet : Mr Y.Bouterfa et Mr A.Dewulf.

Objectifs du projet

Le but de ce projet est de réaliser un télémètre à ultrason mesurant une distance. Ce télémètre calculera une distance pour ainsi l'afficher à l'aide de deux afficheurs à 7 segments à cathodes communes.

Cette distance sera calculée à partir d'un point A (télémètre) jusqu'à un point B et renverra l'information au PIC. Celui-ci récupérant une entrée et définissant l'état de plusieurs sorties digitales.

Répartition du travail au sein du groupe

	ROBIN	**IGOR**	**VICTOR**	**RODRIGUE**
EAGLE		5h	1h30	
SCHÉMA PROTEUS	2h			
RAPPORT	1h	1h	5h	2h30
CODE JAVA				10h
CODE C	15h – 20h			1h30

Ces heures sont des estimations et ne représentent pas l'entièreté du travail consacré. Ce sont des heures effectives, le temps consacré à la recherche et à la compréhension de certains points n'est pas pris en compte.

Au cours de ce projet, l'ensemble du groupe a eu un rôle à assurer. Nous nous sommes chacun documentés sur les différents composants, leurs spécificités et datasheets.

Nous avons aussi effectué des phases de tests pour tester l'efficacité de nos schémas et prototypes.
Les tâches ont été correctement réparties et de l'entraide était bien présente entre nous, notamment lors de la conception de la Board Eagle et le développement du code JAVA.

Concernant les outils de communications utilisés tout au long du projet, nous avons utilisé :

- Messenger : pour une communication journalière
- Discord : pour les meetings, permettant le partage d'écran entre nous
- GitHub : pour le partage des fichiers et déploiement du projet

Concernant notre planning de travail, nous ne nous sommes pas basés sur un planning avec date et échéance. Le travail et recherches ont été faites de manières irrégulières, en fonction de notre temps libres et des autres cours.

Descriptif de la carte

Voici les principaux composants de notre carte

COMPOSANTS	SPÉCIFICITÉES
1 RÉSISTANCES	330 Ohm
3 CONDENSATEURS	0.5 pF
1 BOUTON POUSSOIR	/
2 LEDS	Vert et rouge
QUARTZ CRYSTAL	Cadencé à 20 MHz
SONDE À ULTRASONS	HCSR04
PIC	18F458
2 AFFICHEURS 7 SEGMENTS	À cathodes communes
2 DÉCODEURS	4511
1 COMPIM	Port physique
1 ALIMENTATION	À courant continu

Conformité du cahier des charges

Nous avons essayé de respecter au mieux le cahier des charges mit à notre disposition.

La carte affiche bien la distance sur 2 afficheurs 7 segments. Une alerte apparait lorsque la distance dépasse un seuil critique pouvant être définit.

Le Pic communique avec l'application JAVA qui affiche la distance mesurée et avertit en cas de dépassement du seuil d'alerte.

L'alerte est bien signalée à travers la led rouge sur la carte. Et une led verte reste allumée tant qu'il n'y a pas d'erreurs.

Notre interface JAVA affiche si une alerte est en cours ou non. Elle permet aussi d'envoyer au PIC la valeur du seuil limite.

Tests effectués et résultats

Au cours du développement du projet, plusieurs séries de tests ont été effectués.

Au niveau du code C, nous avons effectués un test pour voir si les décodeurs interagissaient bien entre eux.

Celui-ci envoie des valeurs aux décodeurs qui eux par la suite affichent cette valeur via les 2 afficheurs 7 segments.

Nous avons aussi fait un simple compteur afin de voir que les décodeurs interagissaient bien avec le PIC.

```
void main()
{
   setup_low_volt_detect(FALSE);
   d_value = 0;

   while(true)
    {
     //Envoie la valeur sur l'afficheur 7 segments
     output_d(outputValueParser(d_value));

     if(d_value > 99){
     //Point de l'afficheur 7segments
     output_e(1);
     }

     output_high(PIN_B0);
     delay_ms(150);
     output_lower(PIN_B0);
     d_value++;
    }
}
```

Problèmes rencontrés

- Librairie RxTx incompatible avec la version récente d'Eclipse.
  ➔ Utilisation d'une version inférieure (Eclipse Mars).

- La communication série ne s'opérait pas et donc le programme ne recevait aucunes données.
  ➔ Problème venant du code C, modifications apportées à cet effet.

- Récupérer le seuil de distance par défaut venant du code C.
  ➔ Mettre le seuil d'alerte dans le code JAVA 70cm par défaut (parce que par défaut il est à 70cm dans le code C).

- Lors de la lecture de données, une exception interrompait le programme lorsque le seuil qui est un string était vide.
  ➔ Mettre en commentaire l'exception pour éviter l'interruption du programme intempestif.

- Lors de la finalisation de la Board Eagle, il nous a été difficile de trouver des chemins de connexions parfait à 100%.
  ➔ Il nous a fallu du temps et de la patience, du Try&Catch. Par la suite, nous avons dû placer des trous physiques sur la carte pour parfaire les connexions.

- Manque d'informations concernant les librairies à utilisées sur notre schéma Proteus. Tous les composants nécessaires au projet ne sont pas repris dans les libraires par défaut.
  ➔ Recherche et entraide entre étudiants nous ont permis l'utilisation des libraires adaptées.

- Lors de l'affichage de la distance mesurée via le code C, nous rencontrions des problèmes de conversion décimal/hexadécimal. Ex : Si la distance mesurée était de 53cm et que par la suite nous l'envoyions sur les afficheurs 7 segments, la valeur affichée sur le schéma était de 35 (53 en base 10 étant égale à 35 en base 16)
  ➔ Pour cela nous avions premièrement tenté de convertir cette valeur via une fonction (→ Echec). En deuxième lieu nous avions créé un tableau hard codé contenant les valeurs hexadécimales dans l'ordre (Ex si la distance était égale à 20 : tab[distance] = 0x20), cela fonctionnait bien mais cependant la compilation devenait très lourde. Nous avons donc continué nos recherches afin de trouver la solution adéquate. Finalement lors d'une séance question réponse avec Monsieur Dewulf ainsi qu'en demandant à d'autre étudiants, mon groupe et moi trouvions enfin la bonne solution.

- Lors de l'affichage de la distance sur le schéma Proteus, une coupure de signal entre le décodeur et les 2 afficheurs 7 segments était présente. Cela devait être dû à la librairie téléchargée des afficheurs.
  ➔ La solution a été résolue également grâce au point mentionné précédemment.

- Vérifier si notre calcul permettant de mesurer la distance était exact.
  ➔ L'utilisation de la librairie « HCSR04 » de Proteus. Celle-ci contenait une petite zone affichant la valeur de la distance en cm (pas grand intérêt de donc afficher cette distance sur des afficheurs plus bas). Après vérification nous avons donc remis notre sonde munie d'un potentiomètre.

Pistes d'améliorations

- Concernant la partie développement, une architecture en MVC (Modèle-Vue-Contrôleur) reste préférable pour une meilleure qualité de code. Par faute de moyens et de temps, nous avons opté pour un code simple de compréhension, en une seule classe JAVA.
- Au niveau du potentiomètre présent sur le schéma, l'objectif serait d'avoir une seule échelle, et pas seulement de 0 à 20%, puis de 20 à 40%, etc. Nous avons essayé la modification au niveau des compteurs (« Timer ») sans grand succès.

Conclusion

Etant donné que notre groupe était réparti dans 2 classes différentes, nous avons appris à répartir le travail de groupe et à s'organiser entre nous à travers différents outils comme GitHub et Discord. Nos connaissances théoriques de 1er et de 2e année ont pu être mises en pratique.

|**Robin**|

Au niveau de la pratique de ce projet, le soudage, lier les composants, avoir notre projet physique fonctionnel, etc. aurait été pour moi la partie la plus fun à concevoir comme la plupart des élèves je pense. Mise à part cela, les événements inattendus n'ont pas fort impacté notre projet de groupe.

Les moyens de communication ainsi que le fait de travailler ensemble aux mêmes moments se sont plutôt bien déroulé. Rodrigue et moi-même étions chargé de développer la partie programmation de ce projet. Cela se passait plutôt bien au début, mais arrivé dans le vif du sujet (communication entre les 2 applications, affichage, etc.) le temps de recherches et de tutoriels est devenu très long. Nous avions rencontré pas mal de petites erreurs (comme une mauvaise version de java) cela nous ralentissait donc encore plus dans notre travail.

De plus, l'aide de certains étudiants ayant rencontré les mêmes problèmes que nous, était fort précieux.

Finalement, mon groupe et moi sommes donc arrivé jusqu'à la fin de ce projet dans une bonne entente.

|**Rodrigue**|

Pour ma part, n'ayant pas autant contribué à la conception du PIC que les autres, je me suis principalement occupé de la partie Programmation (précisément Java et un peu du C). Ayant eu des problèmes avec mes groupes dans les projets d'autres cours (me retrouvant seul dans certains cas), cela a été difficile pour moi de pouvoir investir du temps pour ce projet, en tout cas au début.

Par la suite j'ai pu donc m'investir comme il se doit. Bien que des soucis de librairies, communication série ou même logique de programmation ont été rencontrés pour l'application Java et le C, nous avons cependant pu mettre en place les fonctionnalités demandées.

Pour conclure, j'ai trouvé ce projet fort intéressant car il mélange bien à la fois nos connaissances d'électronique et de programmation mais également la cohésion de groupe. La contribution de chacun a permis de réaliser l'entièreté du projet selon le cahier des charges.

|**Igor**|

De mon côté, j'ai trouvé le principe du projet très intéressant dans l'ensemble. Je me suis axé sur la conception du PCB en lui-même ce qui a été fortement intéressant bien que parfois un peu frustrant. Le résultat était très satisfaisant, malheureusement à cause des conditions actuel nous n'avons pas pu en profiter en vrai.

La répartition du travail s'étant faite ainsi je n'ai pas participé à la conception du programme, mais si le projet était à refaire en voyant que l'équipe responsable du programme à passer un peu plus de temps que moi à la réalisation, je m'y serais plus impliqué.

Dans l'ensemble, tout s'est plutôt bien passé et je suis personnellement content du résultat.

|**Victor**|

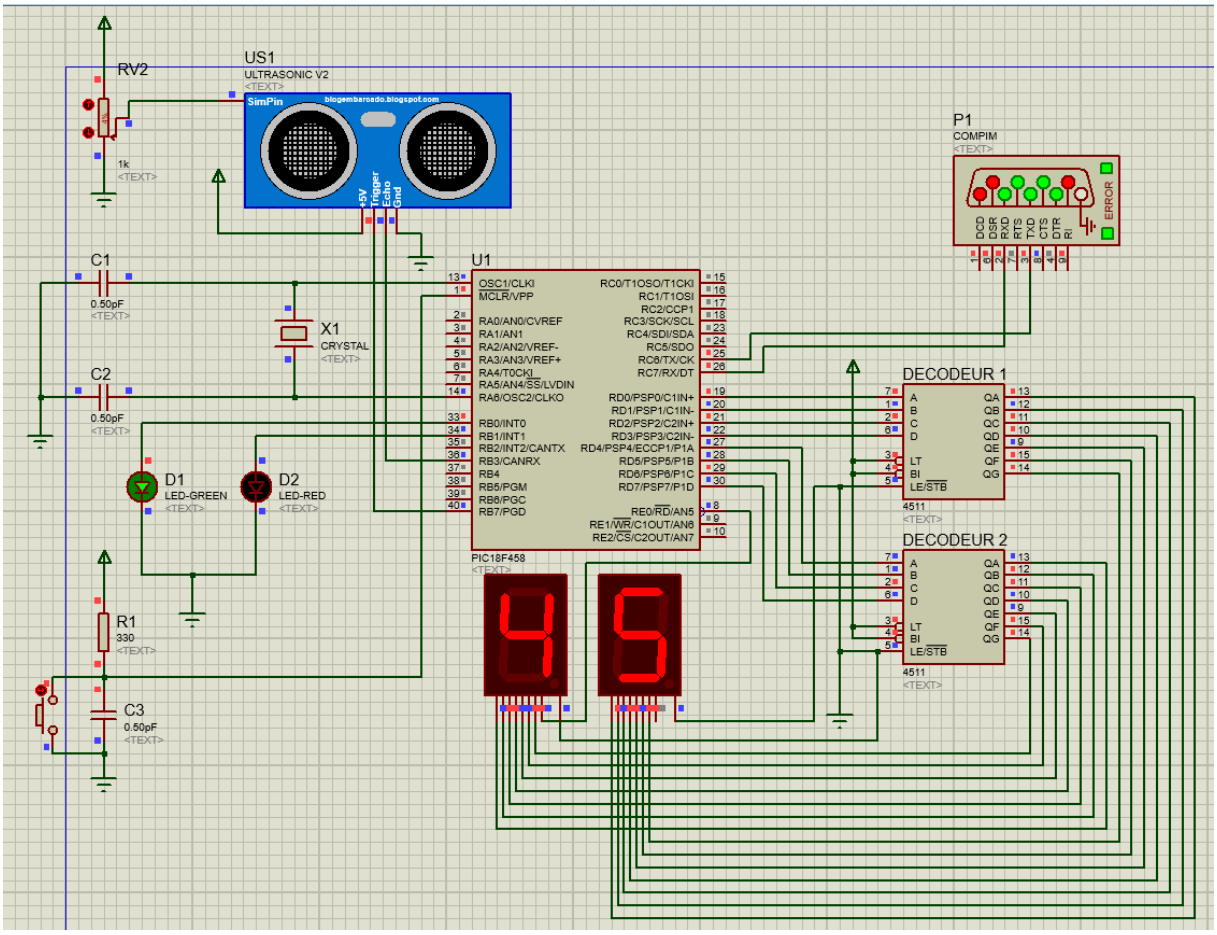
Ce projet au départ me semblait irréalisable, car nous avons dû absorber beaucoup de nouvelles informations et dû s'approprier de nouveaux logiciels, qui étaient jusque-là, encore inconnu pour la plupart d'entre nous.

Ce projet m'a fait rendre compte du potentiel et des nombreuses possibilités existantes concernant la conception d'un objet électronique. Ce projet nous ouvre les portes à de futures perspective de création de matériels physiques.

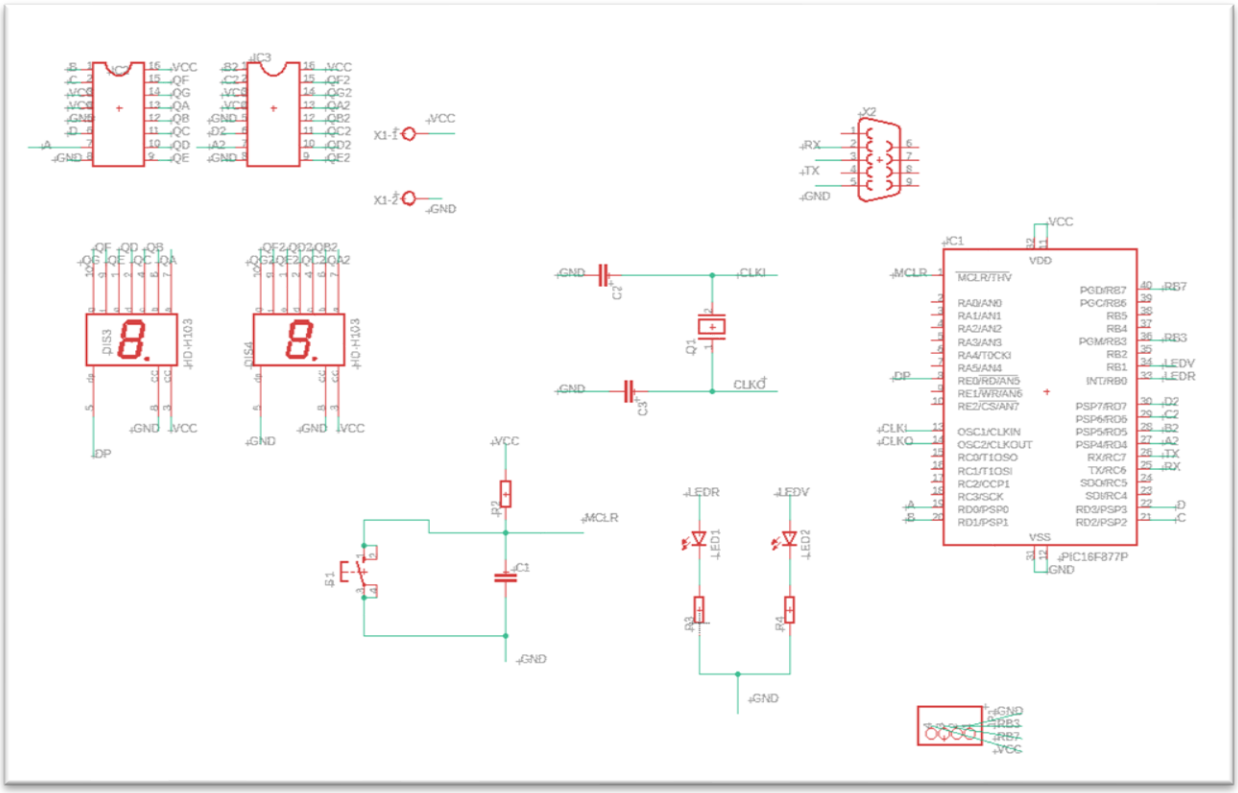
Seul petit regret, ne pas avoir eu la possibilité d'avoir eu entre les mains notre plaque avec tous les composants. Ce projet est donc resté fictif malheureusement.

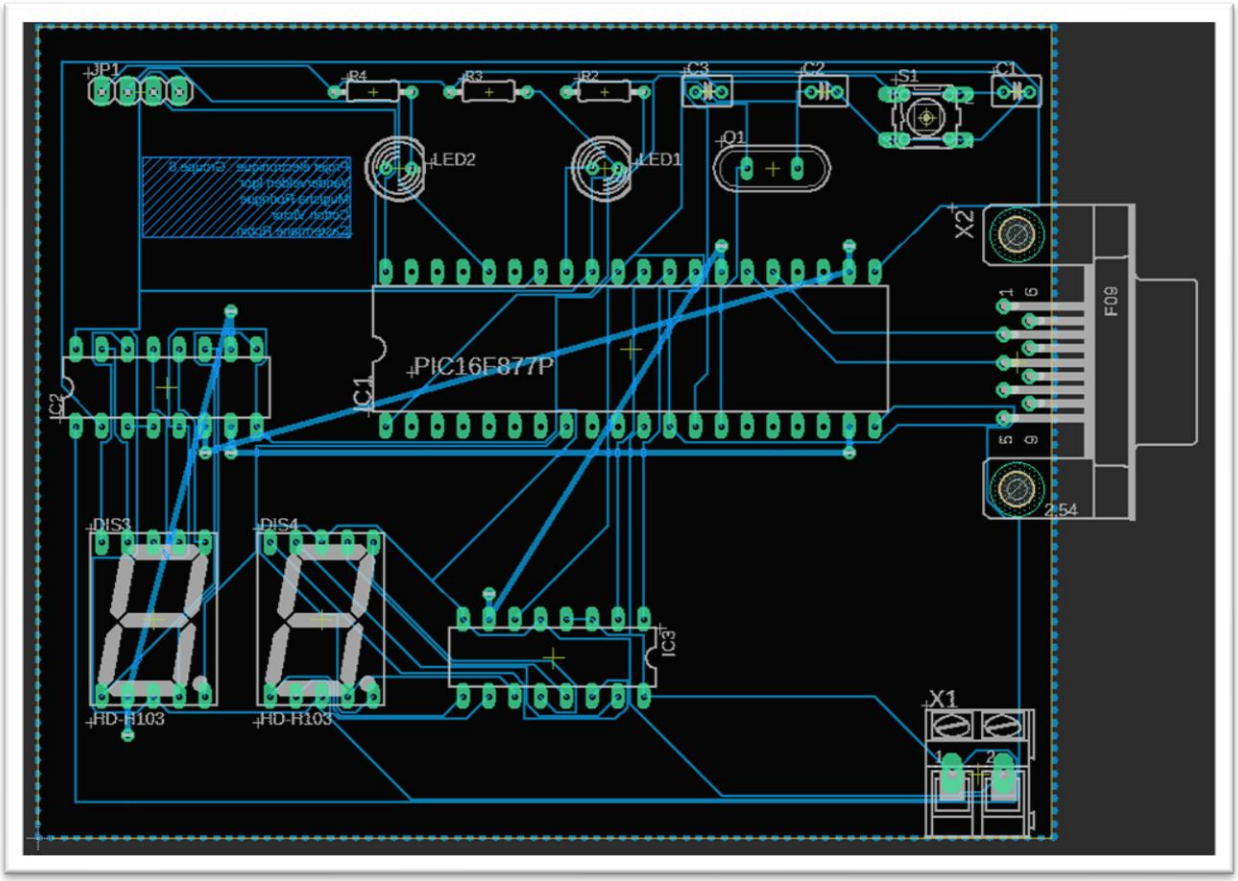
Annexe

Proteus



Eagle


IC3
VCC
X1-1
X1-2
GND
X2
RX
TX
GND
DIS3
DIS4
HD-H103
DP
CLKI
CLKO
C2
C3
Q1
IC1
MCLR
VDD
VSS
PIC16F877P
RA0/AN0
RA1/AN1
RA2/AN2
RA3/AN3
RA4/T0CKI
RA5/AN4
RB0/INT
RB1
RB2
RB3
RB4
RB5
RB6/PGC
RB7/PGD
RC0/T1OSO
RC1/T1OSI
RC2/CCP1
RC3/SCK
RC4/SDI
RC5/SDO
RC6/TX
RC7/RX
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5/PSP5
RD6/PSP6
RD7/PSP7
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
LEDR
LEDV
P2
C1
S1
LED1
LED2
R3
R4


JP1
R4
R3
R2
C3
C2
S1
C1
LED2
LED1
Q1
X2
F09
2.54
PIC16F877P
IC1
IC2
IC3
DIS3
DIS4
HD-H103
HD-H103
X1

Code C

```c
#include <main.h>

int32 seuilLimite = 70;
/*
utilisation de ce tableau pour 2 raisons :
   1) Proteus n'allumant parfois pas l'afficher à 7seg, cela permet donc de lui
   forcer à afficher la valeur voulu (problème dans les fichiers dw de proteus).
   2) La conversion de décimal à hexadécimal et permet donc un affichage exact
   de la valeur calculé.
int seg [] = {0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,
              0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,
              0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,
              0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,
              0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,
              0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,
              0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,
              0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,
              0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,
              0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99};
*/
int32 distance = 0;
/*
Fonction permettant de calculer la distance, Nous envoyons une impulsion au Trigger
de la sonde à ultrason avec un délais de 20us, lorsque la pin_b3(Echo) est à high,
nous réceptionnons à ce moment la le temps parcouru, après ça le calculer et enfin
l'envoyer pour permettre aux 7segments de l'afficher.
*/
int32 valeurDist(){
   float temps=0;
   int32 dist=0;

   output_high(pin_b7); //Impulsion envoyé sur le trigger de la sonde à ultrasons HC-sr04
   delay_us(20);
   output_low(pin_b7); //Fin de l'impulsion

   while(!input(PIN_b3)); //tant que la pin b3 est à low (pour attendre que l'impulsion revienne)
      set_timer1(0); //mise à zéro du timer 1, Initialise l'horloge/compteur timer0. La valeur peut être de 8 bits ou de 16

   while(input(PIN_b3)); //tant que la pin b3 est à high (donc que l'impulsion est revenu)
      temps=get_timer1(); //récupération donc du timer (du temps)


   dist = temps*0.00344;// Calcule de la distance en sachant que la vitesse du son est de 340m/s (*0,00344)
   return dist; //dist/2
}


void main(){
   //test des leds en les allumant au démarage
   output_high(PIN_B0);
   output_high(PIN_B1);


   //augmentera tous les 0,4us
   setup_timer_0(RTCC_INTERNAL); //Définit la source pour le timer_0, fixe l'horloge interne comme source
   //augmentera tous les 1.6us
   setup_timer_1(T1_INTERNAL); //Idem pour le timer_1, régler l'horloge interne comme source
   enable_interrupts(GLOBAL);//Permet l'interuption spécifié,
   setup_low_volt_detect(FALSE);//Règle les niveaux de déclenchement de la tension et aussi le mode
   enable_interrupts(INT_RDA);//RDA = RS232 receive data available (activer le rs232)
   enable_interrupts(GLOBAL);

   //Cette fonction nous permet de convertir la limite reçue par le code java (String en int)
   #define toint(c)  ((c) > '9' ? c - '7' : c - '0') //hexadécimale

   //Les éteinte afin de permettre un délai lors du démarage
   delay_ms(200);
   output_low(PIN_B0);
   output_low(PIN_B1);


   while(true){ //Boucle infinie afin de constament cacluler la valeur de la distance

      distance = valeurDist(); //Appel à la fonction calculant donc la distance

      if(distance<seuilLimite){ //si la distance ne dépasse pas la limite alors :
         output_low(PIN_B1); //la led rouge est éteinte
         output_high(PIN_B0); //la led verte est allumé
      }else{
         output_low(PIN_B0); //sinon la led verte est éteinte
         output_toggle(PIN_B1); //et la led rouge clignote
      }



       //si la distance est <= 100 alors la PIN_E0 s'allume
       // --> elle représente le point sur le 2ème afficheur 7seg
      if(distance <= 100){
         output_low(PIN_E0);

         //output_d(seg[distance]);
         int ten = distance/10;
         int unit = (distance - (ten*10));
         output_d(((distance/10)<<4)+unit);

         printf("%lu\r\n",distance);//Envoie de la donnée au code java
         delay_ms(300);
```

```
99
100         }else{
101             output_high(PIN_E0);
102
103             int ten = distance/100;
104             int unit = (distance - (ten*100))/10;
105             output_d(((distance/100)<<4)+unit); //permet de mettre la dizaine à gauche et l'unité à coté
106
107             printf("%lu\r\n",distance);//Envoie de la donnée au code java
108             delay_ms(300);
109         }
110
111
112     }
113 }
114
115
116 /*
117 Fonction permettant de prendre la valeur limite envoyé par le java
118 */
119 #INT_RDA //RS232 receive data available
120 void RDA_isr(void){
121     char reception[4]; //buffer de cinq caractères
122     int32 valeurLimite=0;
123
124
125     gets(reception); //C'est ici que nous attendons les char de la valeur attendu
126
127     int x = 0;
128     while(x < 4){ // Lecture du buffer
129         if(reception[x] == '\r'){ //passage à la ligne
130             break; //Si la fin su char est égale à \r cela veut dire que c'est la fin du int(géré en java)
131         }
132         x++;
133     }
134
135     int i=x-1;
136     int j=1; //compteur initialiser à 0
137     while(i > 0){ //i >= 0 Ne marche pas
138         valeurLimite += toint(reception[i])*j;
139         j*=10;
140         i--;
141     }
142
143     //ajout de la dernière valeur manuellement
144     valeurLimite += toint(reception[0])*j;
145     seuilLimite = valeurLimite;
146 }
```

Code JAVA

```
1  package java_app;
2
3  import gnu.io.*;
4
5  import java.io.BufferedReader;
6  import java.io.BufferedWriter;
7  import java.io.InputStreamReader;
8  import java.io.OutputStreamWriter;
9  import java.io.PrintWriter;
10 import java.util.ArrayList;
11
12 import javax.swing.JFrame;
13 import javax.swing.JButton;
14 import javax.swing.JLabel;
15 import javax.swing.JOptionPane;
16 import javax.swing.JSpinner;
17
18 import java.awt.Color;
19 import javax.swing.JTextField;
20 import javax.swing.SpinnerModel;
21 import javax.swing.SpinnerNumberModel;
22
23 import java.awt.event.ActionEvent;
24 import java.awt.event.ActionListener;
25
26 /**
27  *
28  * @author Robin Castermane, Igor Vandervelden, Victor Cotton & Rodrigue Mugisha Tuyishime
29  * Groupe Elec 8
30  *
31  */
32
33 public class GUI implements ActionListener, SerialPortEventListener{
34
```

```
34
35          private JFrame window;
36          private JButton sendButton,com1,com2,com3,com4;
37          private SerialPort portSerie;
38          private SpinnerModel seuil;
39          private JSpinner spinner;
40          private JLabel distance,result,alerte;
41          private PrintWriter pw;
42          private BufferedReader br;
43          private CommPortIdentifier numPort;
44
45          /**
46           *
47           * Construction interface graphique
48           *
49           */
50          public GUI() {
51                  window = new JFrame("App Java");
52                  window.setSize(600, 600);
53                  window.setResizable(false);
54                  window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
55                  window.getContentPane().setBackground(Color.lightGray);
56                  window.setLayout(null);
57                  window.getContentPane();
58
59                  com1 = new JButton("COM1");
60                  com1.setBounds(50, 300, 100, 30);
61                  com1.setFocusPainted(false);
62                  com1.addActionListener(this);
63
64                  com2 = new JButton("COM2");
65                  com2.setBounds(175, 300, 100, 30);
66                  com2.setFocusPainted(false);
67                  com2.addActionListener(this);
68
69                  com3 = new JButton("COM3");
70                  com3.setBounds(300, 300, 100, 30);
71                  com3.setFocusPainted(false);
72                  com3.addActionListener(this);
73
74                  com4 = new JButton("COM4");
75                  com4.setBounds(430, 300, 100, 30);
76                  com4.setFocusPainted(false);
77                  com4.addActionListener(this);
```

```
78
79                  sendButton = new JButton("Envoyer");
80                  sendButton.setBounds(225, 350, 150, 30);
81                  sendButton.setFocusPainted(false);
82                  sendButton.addActionListener(this);
83
84                  seuil =  new SpinnerNumberModel(70,0,1000,1);
85                  spinner = new JSpinner(seuil);
86          spinner.setBounds(225,150,150,30);
87
88                  distance = new JLabel("La distance est:");
89                  distance.setBounds(250, 225, 300, 50);
90                  distance.setForeground(Color.black);
91
92                  result = new JLabel("");
93                  result.setBounds(345, 225, 50, 50);
94                  result.setForeground(Color.black);
95
96                  alerte = new JLabel("");
97                  alerte.setBounds(245, 185, 150, 50);
98                  alerte.setForeground(Color.black);
99
100                 window.getContentPane().add(spinner);
101                 window.getContentPane().add(com1);
102                 window.getContentPane().add(com2);
103                 window.getContentPane().add(com3);
104                 window.getContentPane().add(com4);
105                 window.getContentPane().add(sendButton);
106                 window.getContentPane().add(distance);
107                 window.getContentPane().add(result);
108                 window.getContentPane().add(alerte);
109                 window.setVisible(true);
110         }
111
112         /**
113          *
114          * Méthode permettant l'ouverture d'une connexion d'un port COM donné
115          *
116          * @param port port à écouter
117          *
118          */
```

```
119         public void connexionPort(String port){
120                 try {
121                         numPort=CommPortIdentifier.getPortIdentifier(port);
122                         portSerie=(SerialPort)numPort.open("",100);
123                         portSerie.addEventListener(this);
124                         portSerie.notifyOnDataAvailable(true);
125                         portSerie.setSerialPortParams(9600, SerialPort.DATABITS_8, SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
126
127                 }catch(Exception e) {
128                         JOptionPane.showMessageDialog(null, "Connexion au port "+ port +" impossible");
129                         }
130         }
131
132         /**
133          *
134          * Méthode permettant la lecture des données arrivant sur le port COM et affichage de ces données dans l'UI.
135          * Une alerte est affiché si le seuil est dépassé.
136          *
137          */
138         @Override
139         public void serialEvent(SerialPortEvent event) {
140                 try {
141                         br = new BufferedReader(new InputStreamReader(portSerie.getInputStream()));
142                         String str = br.readLine();
143                         result.setText(str + " cm");
144
145                         if((Integer)seuil.getValue() > Integer.parseInt(str)){
146                                 alerte.setText("Aucune alerte");
147                                 alerte.setForeground(Color.green);
148                         }
149                         else if((Integer)seuil.getValue() < Integer.parseInt(str)){
150                                 alerte.setText("Alerte, seuil dépassé !!");
151                                 alerte.setForeground(Color.red);
152                         }
153                         br.close();
154
155                 }catch(Exception e) {
156                         /*JOptionPane.showMessageDialog(null, e.toString());
157                          * Suite à des difficultés rencontrées, j'ai enlevé l'exception qui ouvrait une fenêtre message toutes les 5s due à
158                          * dont j'ignore l'origine, ainsi le programme peut fonctionner sans être interrompu constamment.
159                          * */
160                 }
161         }
162
```

```
162
163         /**
164          *
165          * Lors d'un clic sur le bouton "Envoyer", la valeur introduite dans le champ texte est envoyé au PIC
166          * afin de définir le seuil au-delà (ou en dessous) duquel l'alerte est déclenchée.
167          *
168          * Pour la connexion au port COM, quatres boutons sont mis à disposition afin de permettre à l'utilisateur
169          * de choisir entre les ports COM1 à COM4.
170          *
171          */
172         @Override
173         public void actionPerformed(ActionEvent event) {
174                 if(event.getSource() == sendButton) {
175                         String str = seuil.getValue() + "\r";
176                         try {
177                                 pw = new PrintWriter(new BufferedWriter(new OutputStreamWriter(portSerie.getOutputStream())), true);
178                                 pw.print(str);
179                                 pw.close();
180                         }catch(Exception e) {JOptionPane.showMessageDialog(null, e.toString());}
181                 }
182                 else if(event.getSource() == com1){
183                         connexionPort("COM1");
184                 }
185                 else if(event.getSource() == com2){
186                         connexionPort("COM2");
187                 }
188                 else if(event.getSource() == com3){
189                         connexionPort("COM3");
190                 }
191                 else if(event.getSource() == com4){
192                         connexionPort("COM4");
193                 }
194         }
195
196         public static void main(String[] args) {
197
198                 GUI app = new GUI();
199                 System.out.println(app);
200         }
201
202
203
204 }
205
```

Bibliographie

Lié au code C :

|1| https://www.carnetdumaker.net/articles/mesurer-une-distance-avec-un-capteur-ultrason-hc-sr04-et-une-carte-arduino-genuino/

|2| https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf

|3| https://www.youtube.com/watch?v=MgfYRKxZPlg

|4| https://www.youtube.com/watch?v=2VyxHyt9H8g

|5| https://www.youtube.com/watch?v=dA6g17E3NmM

|6| https://www.youtube.com/watch?v=gNgSoQ_W3Yc

|7| https://simple-circuit.com/pic16f84a-hc-sr04-ultrasonic-sensor-ccs/?fbclid=IwAR0oQ3hYch1c-5sCYBDsLdF5n3gpz7qYtK7uu0e9R9HLNIFn2358uZER_PQ

|8| http://www.robot-maker.com/shop/img/cms/datasheet-capteur-ultrasons-hc-sr04.pdf