14/05/2018

Rapport Final – Projet Electronique

Masson J.

Raemdonck S.

Hancquet B.

**Groupe**

Groupe 4

Masson Julien 2TL2

Raemdonck Sébastien 2TL2

Hancquet Brian 2TL1

**Cahier des Charges**

L’objectif du projet est de fabriquer un télémètre à ultrason. Celui-ci doit pouvoir calculer une distance entre lui-même et la direction vers laquelle il pointe grâce à sa sonde.

Pour ceci nous utilisons un PIC18F458 qui sera codé en C et relié à une application Java afin de faire sa représentation visuelle.

Premièrement il nous était demandé de crée une simulation du projet avec le logicielle Proteus. Après les tests de celle-ci, nous devions créer une board avec le logicielle Eagle afin de pouvoir en récupérer une plaquette physique pour la représentation réelle du projet.

Les composants doivent être proprement soudé sur la plaquette. Le télémètre doit pouvoir signaler une alerte par le clignotement d’une led rouge lorsque la distance dépasse un seuil défini par l’utilisateur. Si tout se passe bien une led verte allumé en continu indique le bon fonctionnement de la sonde.

L’interface Java doit également afficher un message d’alerte en cas de problème de distance et également afficher la distance calculée en cas de bon fonctionnement. Elle communiquera avec le PIC soit par le port RS232 du pc, soit par le port USB, cela dépendra de ce que le groupe a choisi.

**Répartition du travail**

Masson Julien

Tout

Raemdonck Sébastien

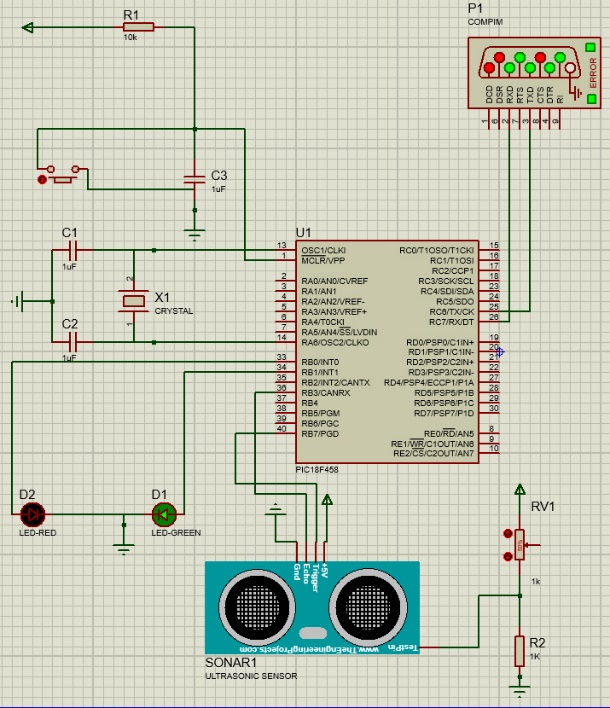
A essayé de souder et a raté

Hancquet Brian

Qui est-ce ?

* **DESCRITPIF CARTE**

**2. Schémas Electronique**

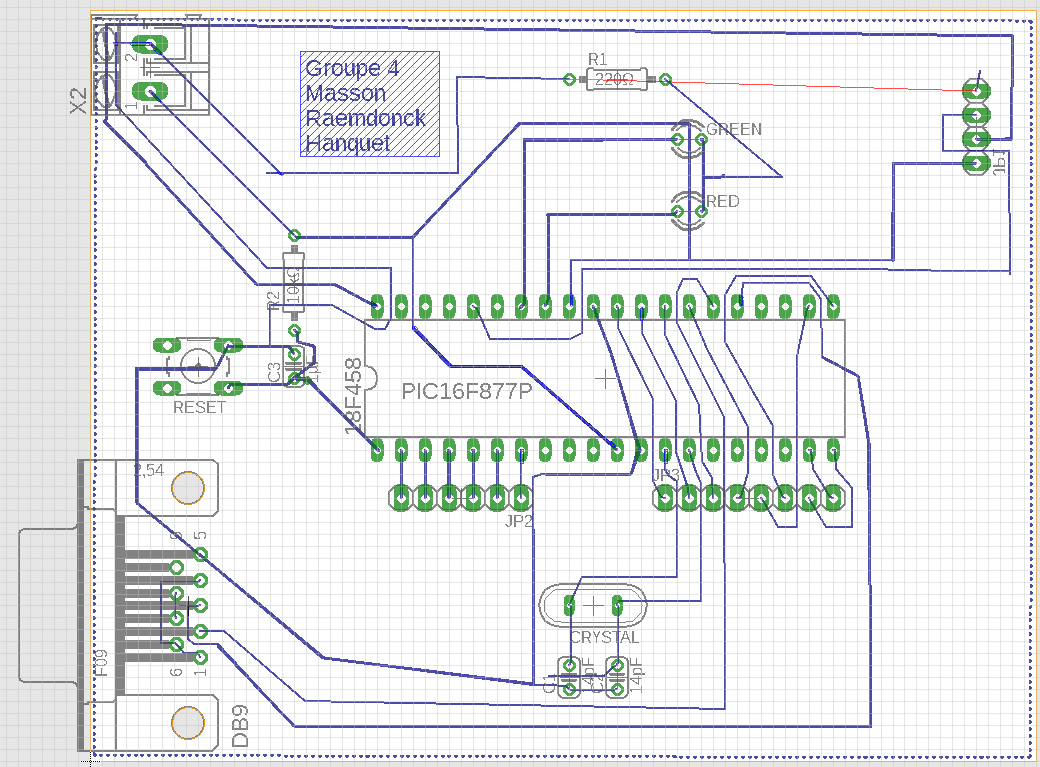
** 2.1 Schéma Protéus**

Ci-dessus se trouve l’image du schéma de simulation de Proteus. Il est composé de tout les éléments importants afin de faire fonctionner notre télémètre à ultrason.

Celui-ci à été réalisé grâce aux fiches de cours « Electronique digitale et analyse des signaux partie pratique ».

Vous pourrez trouver plus de détail à propos de ses composants, de leur utilité et de leur configuration dans le point « 6. Caractéristiques techniques des éléments. »

**2.2 Schéma Eagle**



Ci-dessus se trouve l’image de la board Eagle. Cette image a été réalisé grâce au schéma de Proteus.

Nous avions le choix de travailler avec un connecteur USB et un DB9 et nous avons décidé de travailler avec un DB9 en vue de pouvoir se laisser du RS232 dans notre schéma et sur notre PCB. Les informations à propos du pourquoi le DB9 n’a pas besoin de RS232 se trouvent dans le point « 6. Caractéristiques techniques des éléments. »

Ce schéma a été réalisé en vue de tiré une PCB physique.

**3. Les codes**

**3.1 Le code C (PIC)**

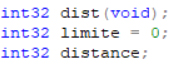
Nous allons scinder le code en ses différents points pour une meilleur explication. Le code sera également en fin de partie 3.1 en entier pour une vision plus globale de celui-ci.

*3.1.1 Include*



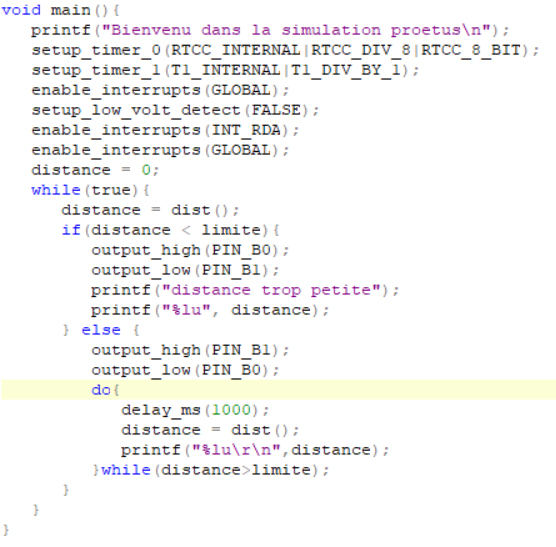
Nous incluons un seul fichier qui sera nécessaire pour l’application du code pour le PIC. Cette include inclura toutes les données nécessaire pour que le code soit exécuté pour un PIC18F458.

*3.1.2 Déclarations*



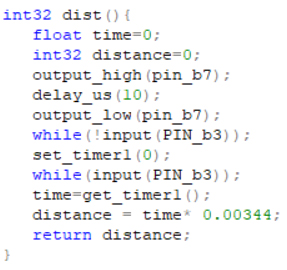
Nous avons que deux déclarations de variables et une déclaration de fonction.

*3.1.3 Fonction main*



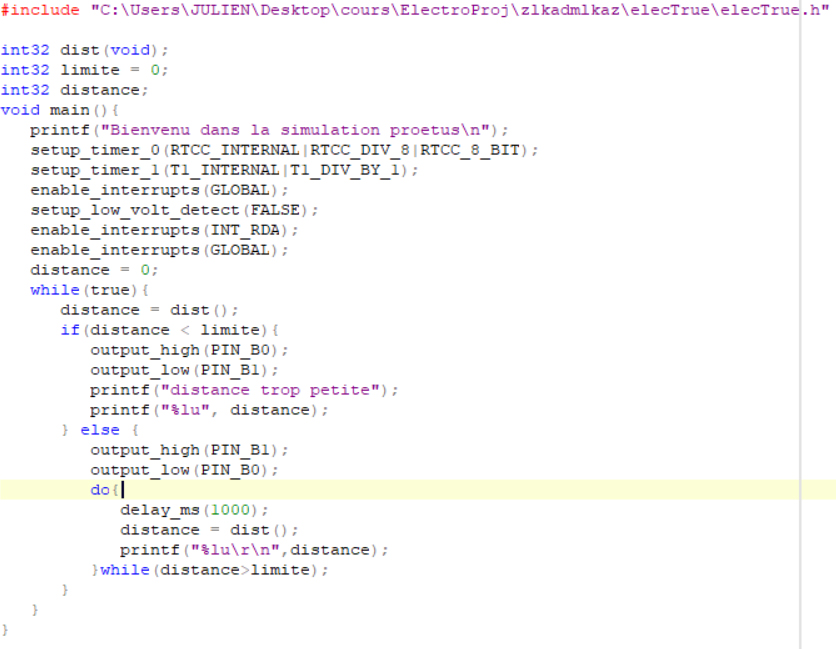
Cette partie est la partie principale du code, celle qui sera exécuté et qui fera appel aux autres fonctions. Après la mise en place de tout les paramètres nécessaires au bon fonctionnement du PIC, nous initialisons la distance à zéro et nous entrons dans une boucle While(true) calculer la distance en continu. Dans cette boucle nous appellerons la fonction dist() l’affecter à notre variable distance. La boucle nous emmène ensuite dans une condition où si la distance est inférieur à la limite que l’utilisateur à défini (ici 0) le led verte s’éteint et la led rouge s’allume. Si la condition est respectée la led verte s’allume et la led rouge reste éteinte, et la distance calculée s’affiche.

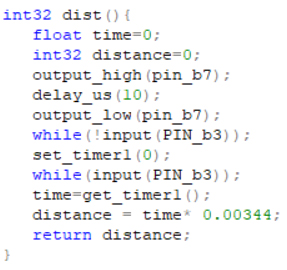
*3.1.4 Calcule de la distance*



Dans cette fonction, nous allons utiliser deux variables ; time et distance, toutes deux variable local et initialisée à zéro. Nous allons envoyer une impulsion sur le trigger de la sonde ultrason (PIN\_B7) et attendre sa fin, ensuite tant que la PIN\_B3 n’a pas reçu de retour un timer calcule le temps qu’il lui faut pour que l’ultrason revienne, quand celui-ci est à l’état haut cela veut dire que l’ultrason est revenu. Il nous suffit alors de récupérer le temps du timer et le calculer pour obtenir une distance en centimètre. Cette fonction return la distance calculée.

*3.1.5 Code complet*



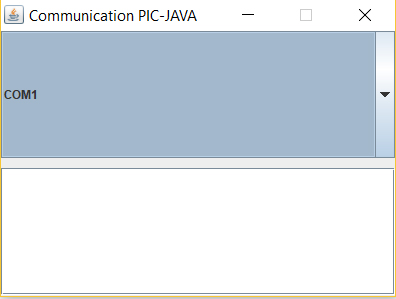


**3.2 Le code Java**

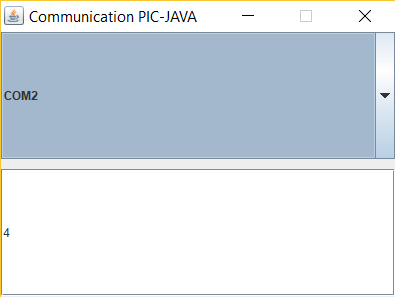
Nous allons scinder le code en ses différents points pour une meilleur explication ainsi qu’un aperçu visuel de l’application en cours de fonctionnement. Le code en sera également mis en fin de partie 3.2 en entier pour une vision plus globale de celui-ci.

*3.2.1 Aperçu de l’application*

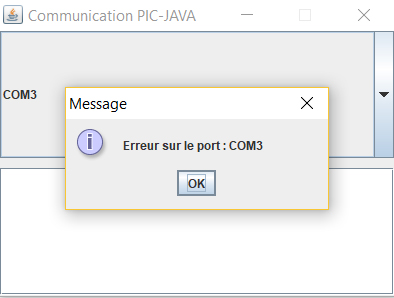
*3.2.1.1 à l’arrêt.*

**

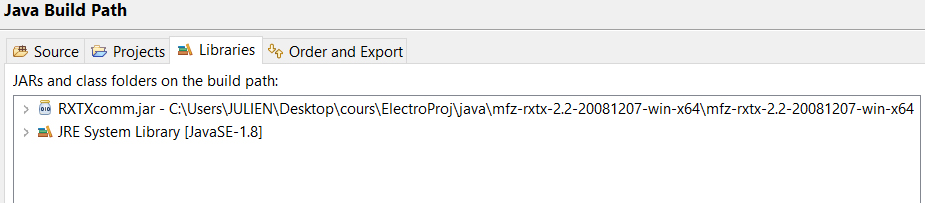
*3.2.1.2 en fonctionnement*

**

*3.2.1.3 en cas d’erreur*

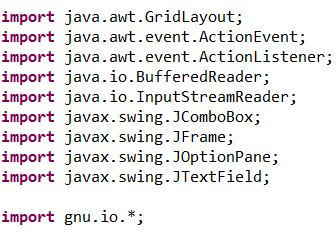
**

*3.2.2 Les librairies*



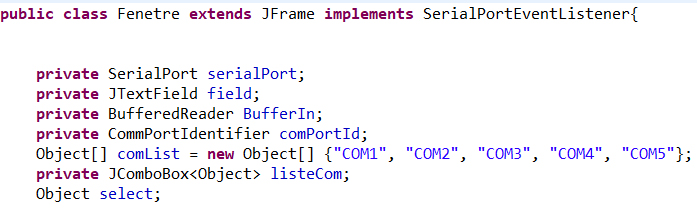
Pour les librairies nous avons utilisé qu’une librairie nommée RXTXcomm.jar qui a servi pour la liaison entre l’application Java et les ports serials de VSPE. Cette librairie sert également à la connexion sur le PCB physique.

*3.2.3 Les Imports*



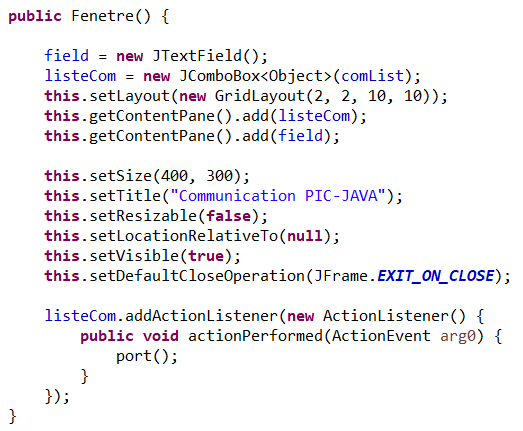
Nous avons effectué plusieurs imports dans le code nécessaire au bon fonctionnement de celui-ci envers Proteus et envers notre PCB. Certains de ces imports sont directement importé de librairies internes à Java et d’autres à notre librairie RXTXcomm.jar

*3.2.4 Déclaration de la classe principale & déclarations*

**

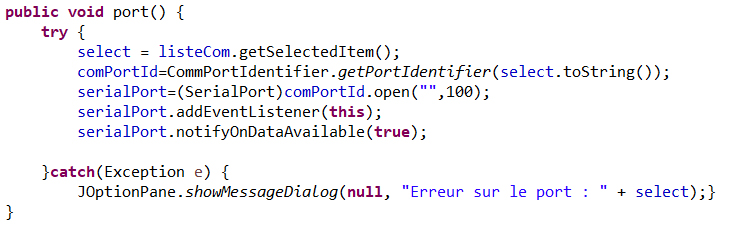
Dans la déclaration de la classe nous devons informer celle-ci qu’il y aura une JFrame avec le « exetends JFrame et également des événements impliquant des ports avec le « Implements SerialPortEventListener ». Lors de la déclaration nous déclarons un champ où sera afficher la mesure, un BufferedReader qui lira les données envoyé par Proteus, un CommPortIdentifier qui servira à identifier le port ouvert, une liste avec différents ports dedans, une JComboBox qui servira à l’affichage de la liste dans la JFrame et enfin un Objet.

*3.2.5 La classe*



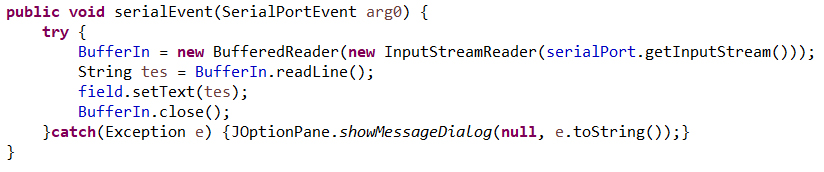
La classe servira à construire la JFrame. Cette JFrame constituera la représentation graphique de toutes les fonctions de la Classe. Dans cette partie du code on retrouvera donc tous les éléments qui construiront la fenêtre.

*3.2.6 Connexion avec le port Comm*



Cette fonction permettra l’ouverture de la connexion avec le port Comm. Cette fonction contient un TRY/CATCH qui sera utile au cas où il y aura un problème lors de la connexion au port Comm. Si cela arrive un message d’erreur s’affichera indiquant quel le port n’est pas ouvert.

3.2.7 Reçu des informations



Cette fonction servira à recevoir les mesures qui sont envoyées par le PIC ou par Proteus et les affichera dans le zone de texte de l’application.

3.2.8 Code complet

