RAPPORT

PROJET ELECTRONIQUE

DIGITALE

Pourbaix Michaël

Nechifor Bogdan

Borca Mark

Geonet Antoine

***Groupe D***

Mai 2022

Table des matières

**Objectifs du projet** **3**

**Outils utilisés4**

**Description de la carte5**

**Tests effectués et résultats5**

**Conformité par rapport au cahier des charges6**

**Répartition du travail7**

**Limites et améliorations à apporter7**

**Conclusions personnelles8**

**Annexe9**

1.Objectifs du projet

L’objectif de ce projet d’électronique digitale est de construire un capteur de distance avec alarme sur base du microcontrôleur PIC18F458.

Ce microcontrôleur est programmé à partir d’un code C et interagit avec un code Python.

Le code C gère tous ce qui est interne au PCB c’est-à-dire toutes les actions que devra effectuer le microcontrôleur en fonction des évènements.

Le code python sert d’input pour l’utilisateur, il lui permet de rentrer la distance qu’il souhaite pour faire office de seuil. Ce valeur de seuil est transmise au code C pour qu’il l’utilise.

Il y a également d’autres éléments mais cette fois d’ordre purement électronique. Voici la liste de ces derniers ainsi que leurs rôles :

* ***Afficheur et décodeurs 7 segments*** ; Leurs rôles est d’afficher la distance en cm fournie par le microcontrôleur. Cette distance ne pourra pas être affichée si elle excède 100cm (soit 1m).
* ***Une sonde à ultra-son*** ; qui renvoi le temps mis par le signale ultra-son pour faire un aller-retour . Cette information est envoyée au microcontrôleur qui va convertir ce temps en distance.
* ***Une LED rouge et une LED verte***; Ces deux éléments ont un but de retour visuel pour l’utilisateur . En effet, quand la LED verte, s’allume cela signifie que la distance actuellement mesurée par la sonde est en dessous de celle fournie par le code Python. Si, en revanche, la LED rouge s’allume, c’est le cas contraire. Cela veut alors dire que la distance est plus grande que le seuil fournis par le code Python.
* ***Une interface RS232/USB ;*** elle nous permet d’envoyer et des informations au code python ainsi que d’en recevoir. C’est également grâce à elle que nous pouvons injecter du code C dans le microcontrôleur.

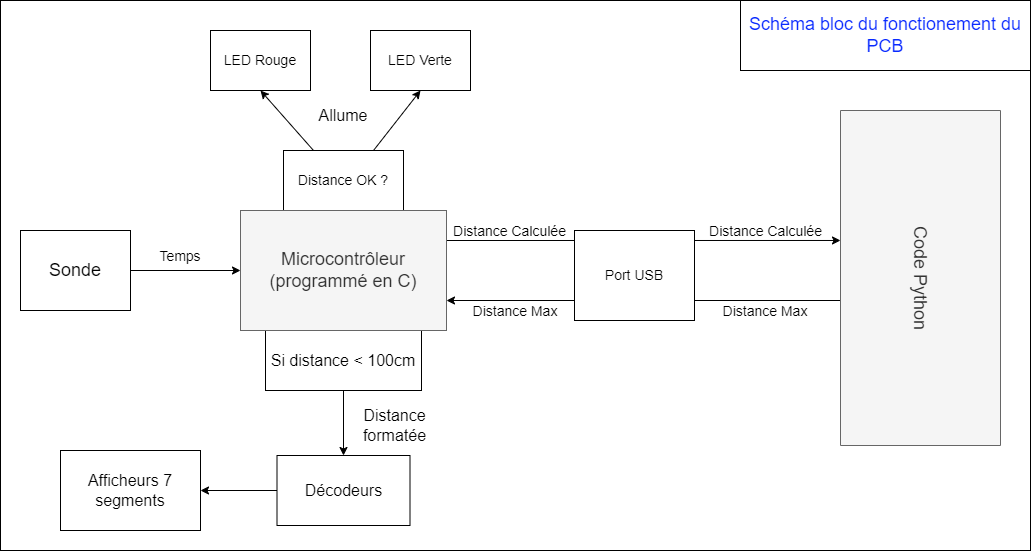
2.Outils utilisés

Pour effectuer ce projet nous avons dû utiliser une ribambelle d’outils et d’objets qui nous on permit d’arriver à nos fin :

* Le PCB qui est l’élément central du projet et sans quoi rien ne fonctionne. Il a été désigné en début de projet à l’aide de deux logiciels Proteus et Eagle.
* ***Proteus*** est un logiciel de simulation électronique qui permet de simuler le projet dans une certaine échelle. Ce logiciel nous à permit de simuler notre code C pour en valider le bon fonctionnement et nous a également servit de base pour la conception du schéma Eagle du PCB.
* ***Eagle*** est un logiciel de réalisation de schéma électronique qui nous a permis de créer le schéma PCB de notre puce et l’envoyer en production.
* ***PIC-C*** est un logiciel permettant d’écrire du code C et de le compiler pour qu’il soit lisible par le microcontrôleur PIC.
* ***Pycharm*** qui est un environnement de développement et qui nous a servi pour développer la partie Python du projet.
* Finalement, ***TinyBootLoader*** nous a permis de mettre le code dans la mémoire du microcontrôleur Pic. Il est donc l’un des outils les plus utiles pour tester du code sur le PIC.

Ce sont tous ces outils regroupé qui nous ont permis de mener le projet à bien.

3.Descriptif de la carte



Sur cette illustration nous pouvons voir tous (ou quasiment tous) les composants de la carte PBC et comment ils interagissent entre eux.

4.Tests effectués et résultats

***Test sur le code Python :***

**Antoine**

***Test sur le code C :***

Nous avons commencé par mettre en place la fonction permettant l’affichage des temps sur les deux afficheurs. La difficulté résidait dans le fait qu’il fallait utiliser un modulo pour avoir le bon format.

**Bogdan et Mark**

***Test sur la board :***

Au départ nous avons eu une frayeur avec la plaque car le TinyBootLoader ne la reconnaissait pas. Nous avons alors réalisé que le PIC était mal enfoncé et que c’était cela qui causait le problème.

Ensuite nous avons cru être sorti d’affaire mais voilà, la plaque ne fonctionne pas. **Suite Pour Bogdan et Mark**

5.Conformité par rapport aux cahier des charges

Est-ce que notre projet remplit les points demandé dans le cahier des charges ?

Voici les points respectés par notre projet :

Voici les points non respectés par notre projet :

6.Répartition du travail :

|  |  |
| --- | --- |
| Membres | Travail effectué |
| Pourbaix Michaël | Schéma de câblage + Simulation Proteus(Eagle) + Rapport |
| Nechifor Bogdan | Rapport + Code C |
| Geonet Antoine | Code C + Python + Rapport |
| Borca Mark | Code C + Python + Rapport |
| DE SOUSA FEREIRA Nuno | **Aucune participation** + pas Rapport |

7.Limites et améliorations à apporter

On pourrait refaire un PCB sans faux-contacts cette fois-ci ☹.

**A étoffer**

8.Conclusions personnelles

Michaël :

Personnellement, malgré que notre PCB n’est pas fonctionné, je trouve que ce projet à encore plus apporté à l’expérience des cours ‘pratique/manuel’ de l’EPHEC. Je trouve que se projet m’a permis d’apprendre de nouvelles choses notamment au niveau de la conception de PCB. Un très bon projet en globalité.

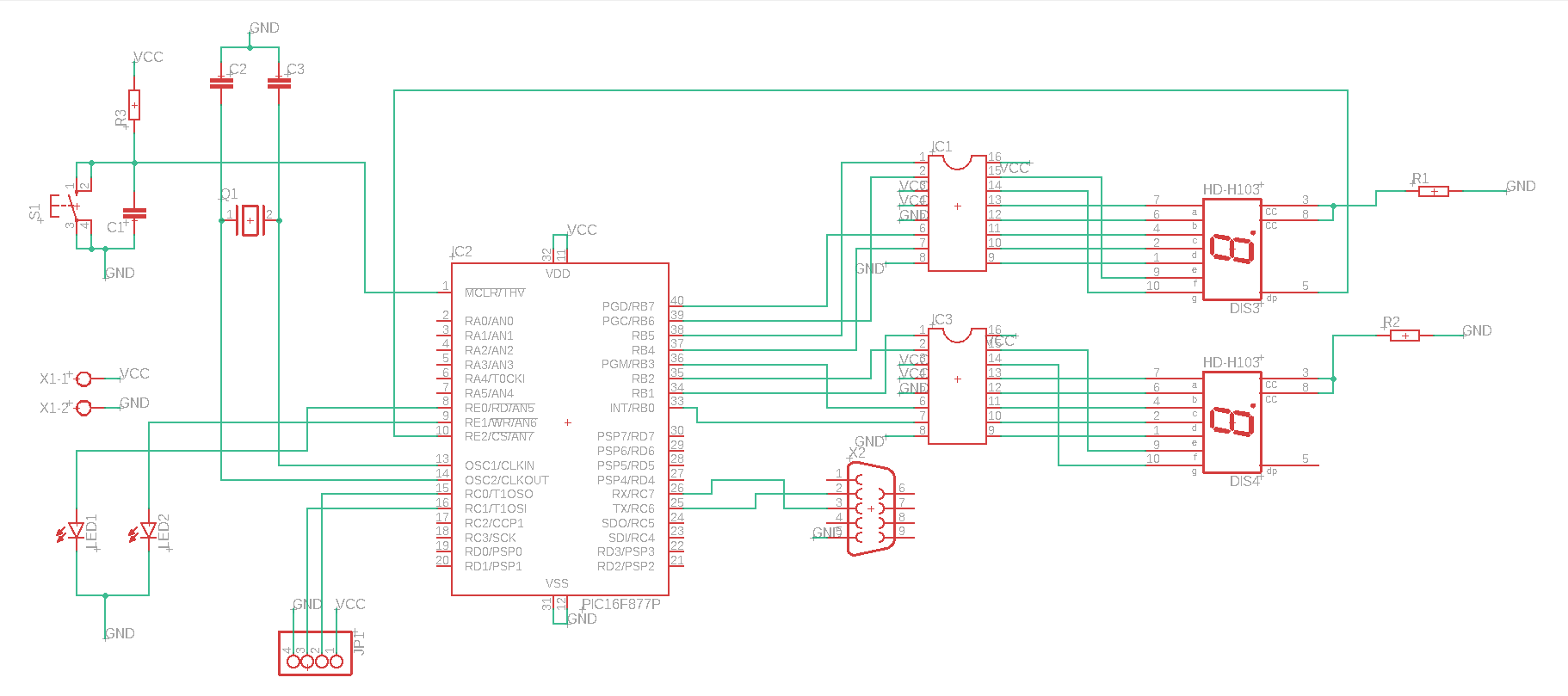
Mark :

Antoine :

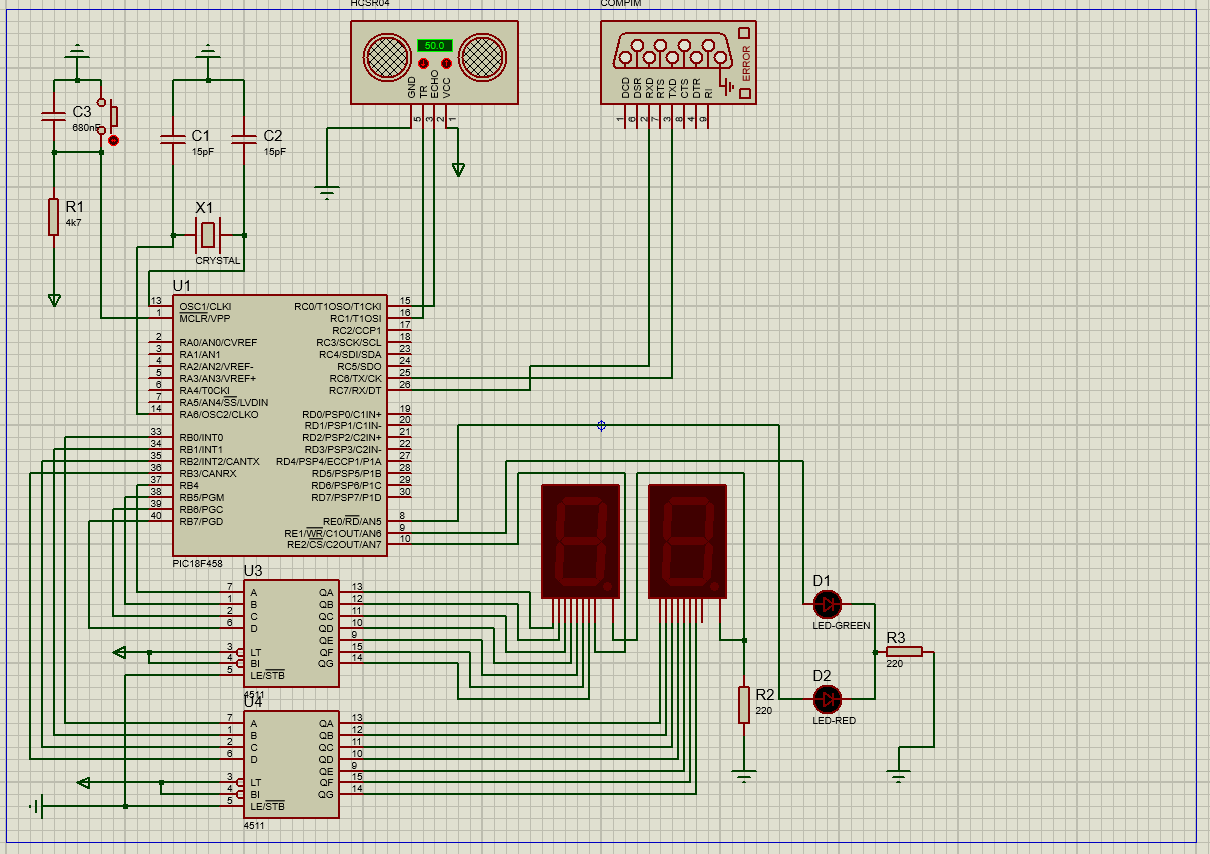
Bogdan :

9.Annexe

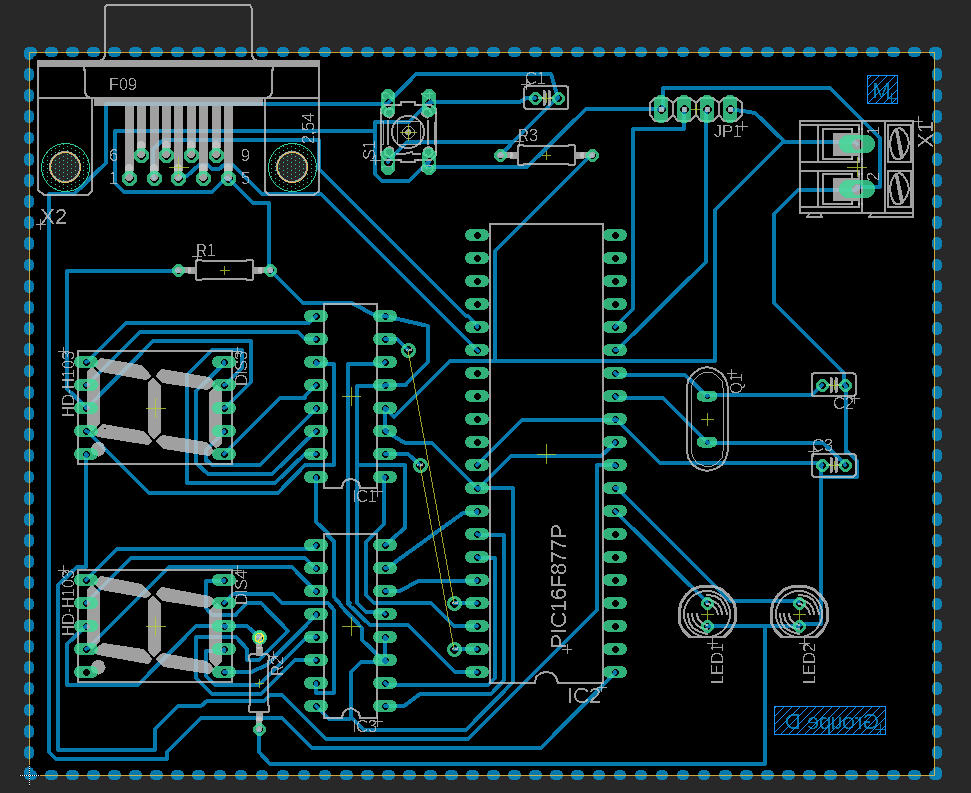
A.Schéma Eagle :



B.Schéma Proteus :



C.Schéma PCB :



AJOUTER LE CODE PYTHON ET C