2022

Projet PIC

Abderrachid Bellaali

Ikram Arous

Marina Quadu

Patson Tiako

Groupe A

17/05/2022

1. Introduction

Un microcontrôleur PIC est une unité de traitement de l'information à laquelle on a ajouté des périphériques internes de communication avec l'extérieur permettant de réaliser des montages sans nécessiter l'ajout de composants externes ou du moins avec un nombre restreint de composants. Il peut fonctionner de manière autonome après programmation.

2. Objectifs

L'application est un télémètre à ultrasons qui envoie au PIC la mesure d'une distance en cm.

- La programmation du PIC 18F458 doit se faire via l'interface RS232 avec un convertisseur vu qu'il n'intègre pas de contrôleur USB.
- L'affichage de la distance se fait sur 2 afficheurs 7-segments à cathodes communes.
- Si la distance est sur 2 chiffres, les 2 chiffres apparaissent. Si c'est sur 3 chiffres, la distance est convertie en m avec la partie entière séparée de la partie fractionnaire par le point de l'afficheur.
- Une LED rouge clignotante signale l'alerte en cas de dépassement de la distance définie par l'utilisateur ou dans le cas contraire, d'atteinte du seuil critique également défini par l'utilisateur.
- Une LED verte doit rester en permanence allumée quand aucune alerte n'est en cours.
- Le PIC sera programmé en C.
- Le PIC doit communiquer avec une application Python qui affichera la distance mesurée et qui permettra de définir le seuil de déclenchement d'alerte.

3. Eléments techniques

Circuit imprimé

Pour le PCB, nous utilisons le logiciel de conception de circuits imprimés Eagle. Nous avons réalisé le circuit sur base d'une simulation Proteus avec les composants suivants :

- Un microcontrôleur PIC 18F458
- Une sonde à ultrasons HC-SR04
- Un cristal
- Un bouton poussoir
- Une interface RS232 avec convertisseur USB
- Une LED rouge
- Une LED verte
- Deux décodeurs 74LS47
- Deux afficheurs 7-segments à anodes communes

Code C

Le code C va permettre de programmer le PIC à l'aide d'un compilateur. Le compilateur CCS que nous avons utilisé a donc généré un fichier .hex sur base du fichier .c dans lequel sont consignées toutes les instructions de programmation nécessaires au fonctionnement du PIC 18F458.

Code Python

Le code Python sert d'interface entre le microcontrôleur PIC programmé et l'autre côté du canal de transmission c'est-à-dire l'utilisateur et son ordinateur. L'interface nous permet de visualiser la distance entre la sonde et l'objet détecté mais également si alerte il y a en cas de dépassement du seuil de déclenchement. Elle doit également permettre de changer le seuil auquel l'alarme va se déclencher.

4. Les tests effectués, leurs résultats et les solutions apportées

Sur le PCB

- Estimation du temps de réalisation : plus ou moins 8h
- Temps réel de réalisation : 15h

Nous avons testé les composants, les pins et les connexions.

Tout d'abord, le programme ne reconnaissait pas le PIC, il a fallu vérifier toutes les soudures et reprendre celles qui ne semblaient pas très nettes. Après quelques efforts et sueurs froides, il l'a enfin reconnu !

Ensuite, les afficheurs 7-segments ne s'allumaient pas. En effet, au moment de l'assemblage, nous avons travaillé avec des décodeurs HCF4511 et des afficheurs 7-segments à cathodes communes. Mais la simulation a été réalisée avec des décodeurs 74LS47 et leurs afficheurs 7-segments à anodes communes correspondants.
Après réalisation de notre distraction, nous avons effectué les changements adéquats et les afficheurs ont, bien évidemment, fonctionné convenablement.

Le code C

- Estimation : plus ou moins 24h
- Temps réel de réalisation : 48h

Le programme a été réalisé au tout début du projet mais pas finalisé à ce moment-là. Nous avions alors une première ébauche que nous avons fait évoluer au fur et à mesure et nous l'avons finalisée quand nous avons pu assembler et tester notre PCB.

Le code Python

- Estimation du temps de réalisation : plus ou moins 24h
- Temps réel de réalisation : 48h

Le code Python a été développé en parallèle du code C. De même que le code C, il a évolué tout au long du projet et a été finalisé quand nous avons assemblé et testé le PCB.

A compléter Par rapport au cahier des charges ? …

5. Répartition du travail

Bien que nous ayons pratiquement tout le temps travaillé ensemble, certains d'entre nous, en fonction des affinités, se sont plus penchés sur certains aspects du projet en particulier.

Simulation proteus :

Ikram et Rachid ont peaufiné la simulation Proteus en coopération avec Marina et Patson

PCB

Marina s'est chargée du PCB et de la finalisation au niveau des soudures en collaboration avec Ikram, Rachid et Patson.

Code C

Rachid et Patson ont principalement travaillé sur le code C en collaboration avec Ikram et Marina.

Code Python

Ikram, Patson et Rachid ont travaillé sur le code Python en collaboration avec Marina.

6. Les limites du système et les améliorations

On pourrait optimiser le PCB en travaillant avec un seul décodeur et un

7. Conclusions personnelles :

Ikram :

Rachid :

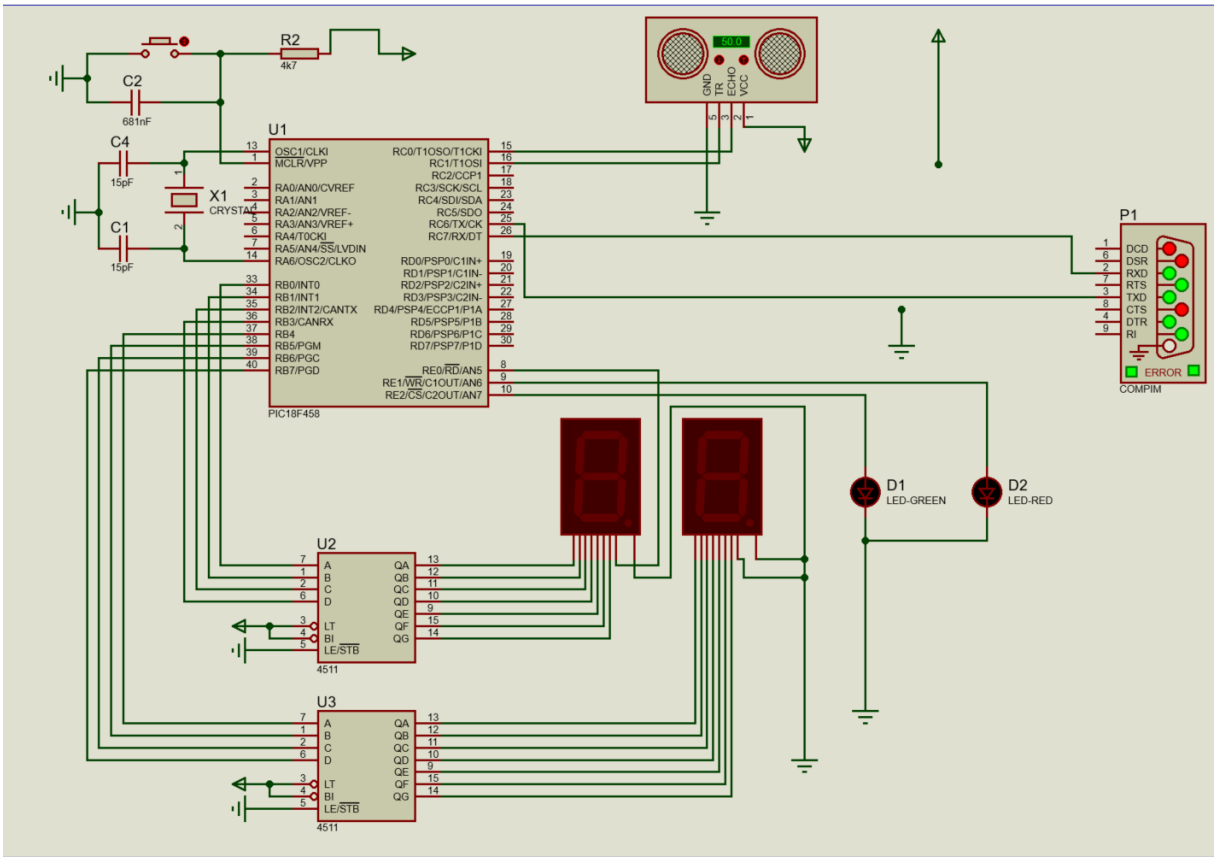
Marina :

En participant à ce projet, j'ai réalisé que la communication, bien que nous ayons toujours travaillé ensemble à peu de choses près, nous a manqué. Il n'est pas évident de partager ce que l'on a fait de manière claire et exhaustive malgré les moyens que nous avons à disposition (Github, Trello…) et le fait que l'on se voit quasi tous les jours.
De plus, entre la remise du rapport intermédiaire et la réception du PCB, le projet a été un peu laissé de côté pour se consacrer aux autres cours et la reprise n'a pas été évidente.
Le projet en lui-même était très intéressant et j'en retire que je dois mieux documenter mes actions afin de me faciliter la vie et le travail de mes collègues.

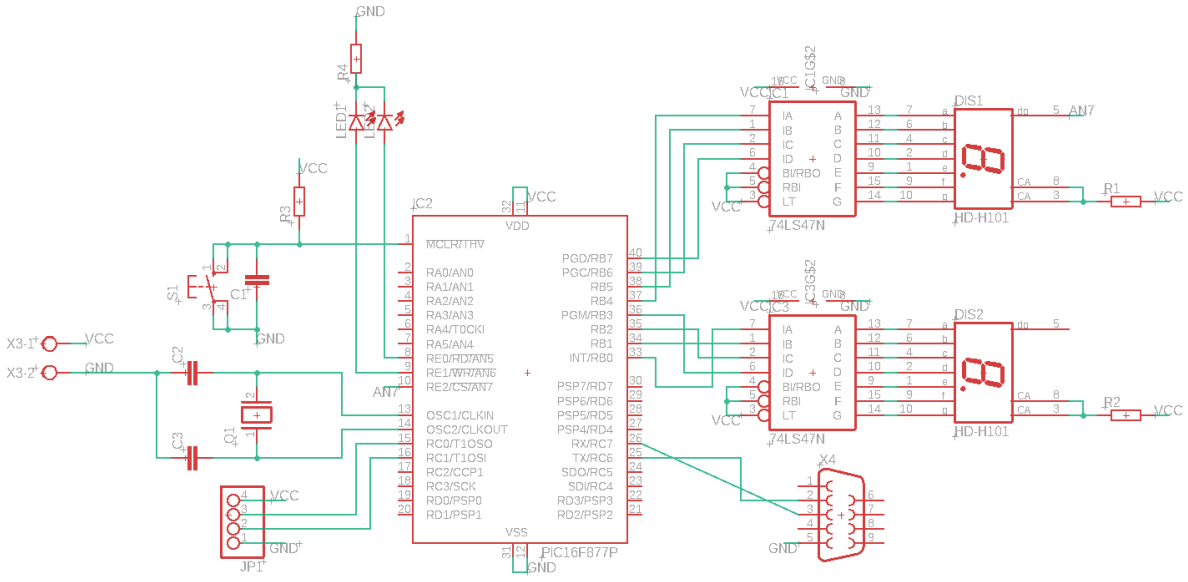
Patson :

ANNEXES

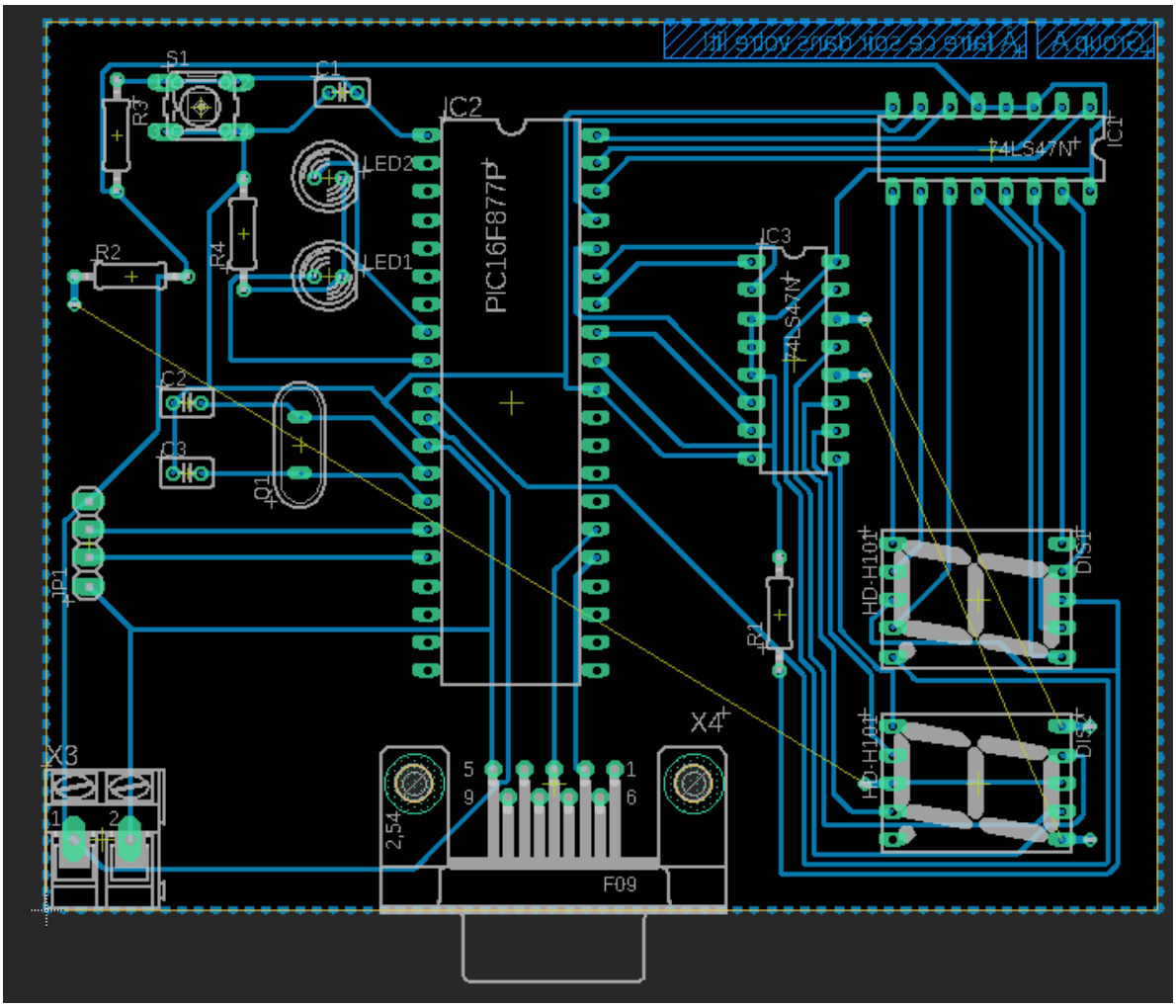
I. Schéma Proteus



II. Schéma Eagle



III. Schéma PCB



IV. Code c

```
#include
"C:\Users\Patson\Documents\Projet_Electronique\Version_LCD\Code_3\code_with_in
terrupt.h"
#include <lcd.c>
char buffer[4];
int i = 0;
int16 diz, unit;
boolean flag = 0;
int16 limitDist = 100;
int16 dist ;

int d, c,u;
#INT_TIMER1
//int stop_timer = 0;
//int timer_overflow = 0;
void  TIMER1_isr(void)
{
   set_timer1(0);
   //if (stop_timer){
     // stop_timer = 0;

   //}else{
     // set_timer1(0);
    //  timer_overflow ++;
   //}

}
#int_RDA
void RDA_isr(void)
{
   buffer[i] = getc();
    if(buffer[0] == ':' && flag == 0 ){
        i++;
        if(i>=4){
            i = 0;
            flag = 1;
        }
    }

}


void main()
{

   setup_adc_ports(NO_ANALOGS);
```

```
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_wdt(WDT_OFF);
  setup_timer_0(RTCC_INTERNAL);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  setup_timer_2(T2_DISABLED,0,1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);
  setup_oscillator(False);


    // TODO: USER CODE!!
    while(true){

     //read value of sensor
     output_high(pin_c1);
     delay_us(20);
     output_low(pin_c1);

     while(!input(pin_c0)){} //attendre l'etat haut de la pin echo

     set_timer1(0);

     while(input(pin_c0)){} //attendre l'etat haut de la pin echo

     dist = get_timer1()*0.028;
     printf("distance : %ld \n" , dist);
     printf("\n");
     delay_ms(100);

     if(dist<limitDist){
        output_high(PIN_E0);
        output_low(PIN_E1);
        printf("A1_OFF\n");
     }else{
        output_low(PIN_E0);
        output_toggle(PIN_E1);
        printf("A1_ON\n");   //tel python application not alarm
     }

     //affichage
     if(dist<100){
        output_low(PIN_E2);
        diz = dist/10;
        unit = dist - (diz*10);
```

```
         //output_b(diz+unit);
         output_b((diz<<4)+unit);
      }else{
         output_high(PIN_E2);
         diz = dist/100;
         unit = (dist - (diz*100))/10;

         printf(" Unit : %ld\n ", unit);
         output_b((diz<<4)+unit);


      }

      //A envoyer vers le lcd
      if(flag == 1){
         flag =0;
         c = buffer[1]-48;
         d = buffer[2]-48;
         u = buffer[3]-48;
         limitDist = (int16)(c*100+d*10+u);
      }
      c =  limitDist/100;
      d = (limitDist-(c*100))/10;
      u = (limitDist-(c*100))-(d*10);
      delay_ms(300);
     }

}
```

V. Code Python