Groupe 2 (Etudiants) :
. 2TL1 : Adrien Nini Pereira (HE201440)
Shantou François (HE201432)
. 2TL2 : Benoît de Mahieu (HE201458)
Sébastien Blacks (HE201446)

Enseignants responsables du projet :
. Mr Bouterfa Youcef
. Mr Dewulf Arnaud

Rapport final – Projet d'électronique (2017-2018) : Télémètre à ultrason

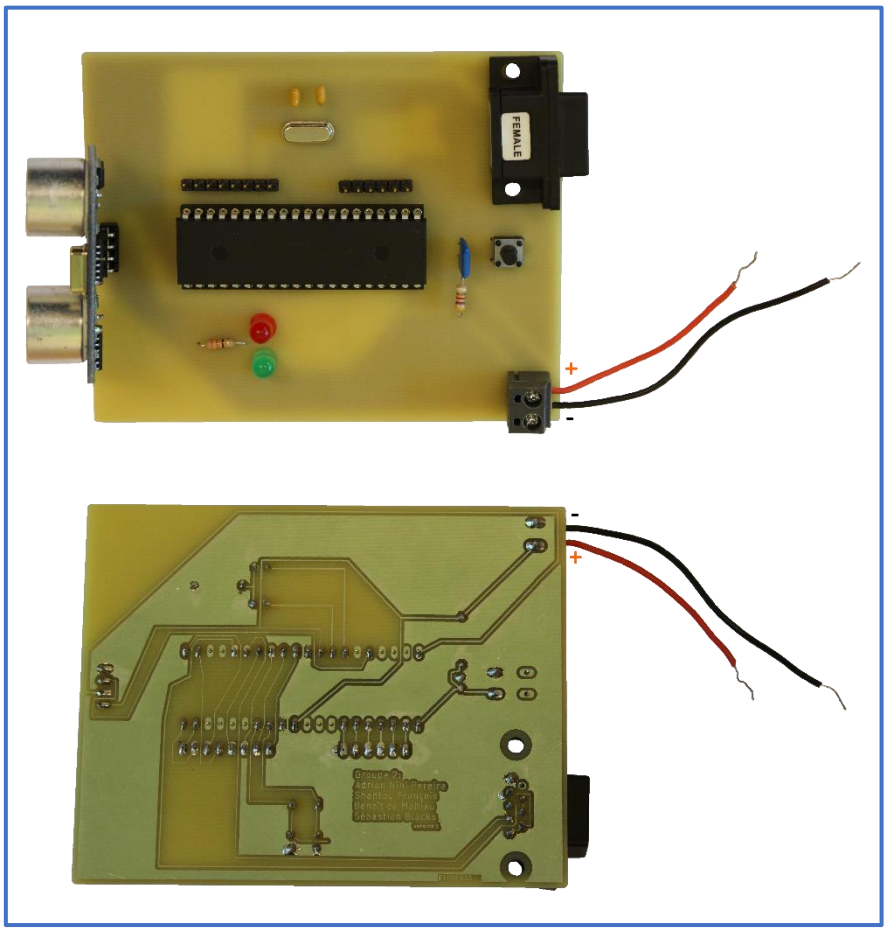


Table des matières :

1) Introduction

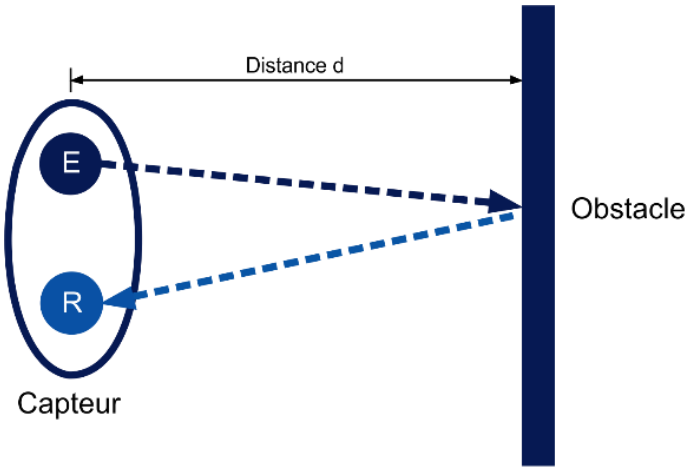
Qu'est-ce qu'un ultrason ?
"Un ultrason est une vibration sonore de fréquence très élevée, non perceptible par l'oreille humaine." (1)

Sonde à ultrason HC-SR04 (2)

Débouchés :
Les ultrasons sont utilisés dans de nombreuses applications :
En médecine (échographie, thermothérapie), en milieu industriel (soudures, contrôles d'équipements), en milieu aquatique (le sonar), en milieu aérien (écholocalisation, **télémétrie**).

Fonctionnement du télémètre à ultrason :
L'émetteur "E" du télémètre envoie une onde sonore qui rebondit sur un obstacle et qui est renvoyée vers le récepteur "R". Ce récepteur délivre un signal électrique qui est transformé en une série de "0" et de "1" et est analysé par un ordinateur.

La vitesse de l'onde sonore dans l'air est d'environ 340 mètres par secondes à 20°C.



t = 2d / V

t : Temps entre l'émission et la réception
d: Distance entre la source et l'obstacle
V: Vitesse des ultrasons dans l'air

Schéma du fonctionnement d'une sonde à ultrason (3)

La distance estimée = temps où le signal de sortie est au niveau haut x vitesse du son(340m/s) /2.

(1) *Source : Le Robert illustré 2018 | Version livre | Edition millésime 2018 | Paris*
(2) *Photo prise par S. Francois au laboratoire de l'Ephec en février 2018.*
(3) *Source : Schéma du fonctionnement d'une sonde à ultrason | Consulté le 26/02/18*
https://gastondanslespace.wordpress.com/2015/10/14/capteurs-de-distance-lequel-choisir

2) Objectifs

Conception d'un télémètre à ultrason qui mesure la distance par rapport à un obstacle et qui renvoie l'information numériquement sur un ordinateur connecté en USB via un port FTDI choisi à la place d'un port série DB9 (décision de groupe).

De plus, au travers de cette mission pratique, nous avons pu combiner toutes les notions vues dans les différents cours.

3) Planning

- Semaine 1 (début février) :

. Réflexion sur le projet
=> (durant les 4 mois | toute l'équipe)
. Outils de communication à utiliser (cfr * ci-dessous)
=> (durant les 4 mois | tous)
. Découpage en étapes de la mission
=> (tous)
. Test de la sonde en labo
=> (30 min | 2 personnes)

- Semaine 2 et 3 (mi-février) :

. Simulation électrique du projet sous "Proteus ISIS"
=> (2 semaines | 1 personne)
. Langage C pour la communication avec le PIC
=> (2 semaines | 1 personne)

- Semaine 4 et 5 (fin-février) :

. Conception du circuit imprimé sous "EAGLE"
=> (2 semaines | 1 personne)
. Test "Naked proto" chez Eurocircuit
=> (30 min | 1 personne)
. Récolte d'éléments et rédaction du rapport intermédiaire
=> (4 semaines | 1 personne)

- Semaine 6 (début mars) :

. Dépôt du rapport intermédiaire
. Dépôt du fichier Eagle "*.brd" (pour impression chez Eurocircuits)
. Fin de la programmation du PIC en langage C

- Semaine 7 et 8 (mi-mars) :

. Conception du code Java pour l'interaction entre l'utilisateur et le télémètre à ultrason
=> (2 semaines | 1 personne)

* Suite à notre composition de "groupe 2" (classes : 2TL1 et 2TL2), nous avons communiqué via GitHub, Facebook, Trello et via des réunions en réel.

- Semaine 9 et 10 (fin-mars) :

  . Mise à jour du code de la programmation du PIC en langage C

  => (2h | 2 personnes)

  . Mise à jour du code Java

  => (2h | 2 personnes)

- Semaine 11 (début avril) :

  . Réception des deux PCB imprimés et soudure des composants sur le 1er PCB – partie 1

  => (1h | 2 personnes)

- Semaine 12 (mi-avril) :

  . Soudure des composants sur le premier PCB – partie 2

  => (30 min | 1 personne)

- Semaine 13 (fin-avril) :

  . Soudure des composants sur le deuxième PCB – partie 1 et 2

  => (1h min | 1 personne)

  . Installation des drivers FTDI et Tests du PCB

  => (30 min | 2 personnes)

- Semaine 14 (début mai) :

  . Debug / mise à jour du code de la programmation du PIC en langage C

  => (2h | 2 personnes)

  . Récolte d'éléments et rédaction du rapport final

  => (8 semaines | 1 personne)

- Semaine 15 (mi-mai) :

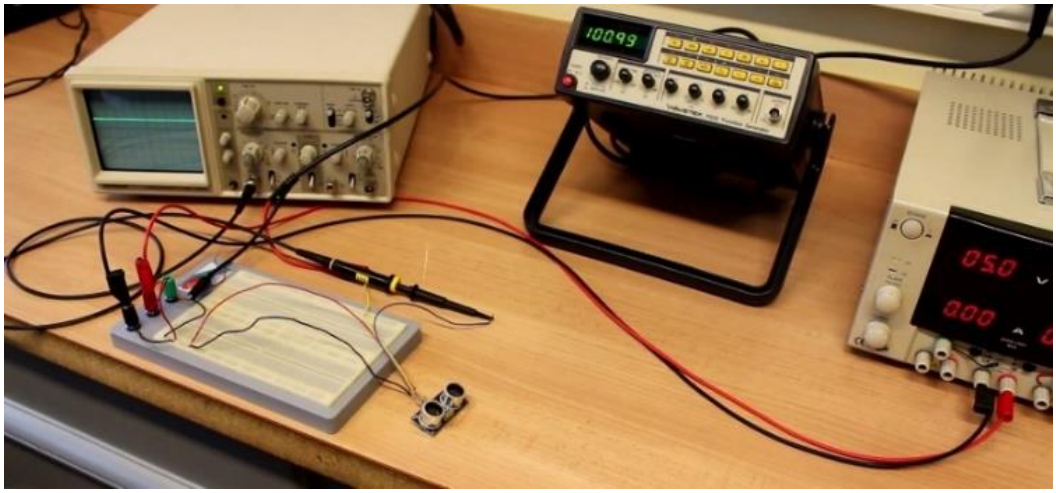
  . Dépôt du rapport final

  . Défense et démonstration du projet final

Remarque 1 : Feedbacks d'équipe réalisés à la fin d'une à deux semaines.

4) Mission

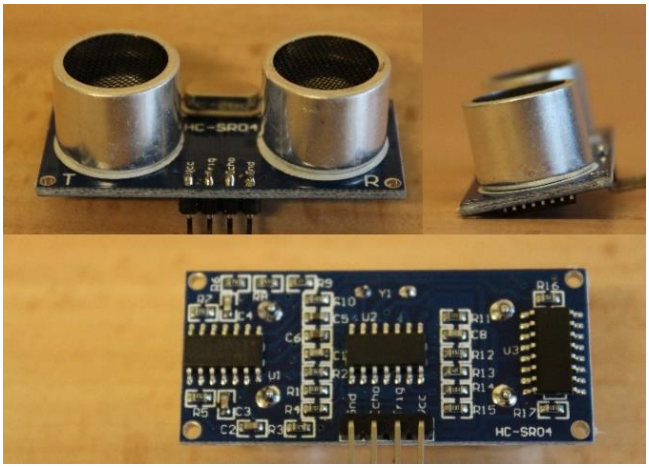
4.1 Sonde à ultrason "HC-SR04" : Tests au laboratoire de l'Ephec



Test de la sonde à ultrason "HC-SR04" (4)

Observation :

Le fil rouge est branché sur le +5v et le fil noir sur la masse de l'alimentation (située à droite sur la photo). Le fil jaune est branché sur le générateur de fonction (le boiter du milieu) et l'autre côté est branché sur le port "trigger " de la sonde. Ce générateur (réglé sur 100kHz) envoie une impulsion TTL toutes les 10 microsecondes à la sonde.
Le fil mauve est branché sur le port "echo" de la sonde et est connecté, à l'autre bout, sur le Channel 1 de l'oscilloscope. La sonde émet une onde qui réfléchit sur un obstacle et revient dans le récepteur pour afficher l'allure du signal à l'écran. Lorsque nous approchons notre main du capteur, nous apercevons que la modulation du signal change. En dessous d'un seuil minimum ("treshold"), la sonde ne retransmet plus rien. Ce test nous montre l'importance d'utiliser un microcontrôleur PIC avec cette sonde.



Problème rencontré :

- Aucuns soucis rencontrés durant cette étape.

<= La sonde à ultrason "HC-SR04" est une solution peu couteuse pour mesurer des distances (environ 2$/pièce). Elle est capable, en théorie, de mesurer de 2cm à 400cm avec une précision de 3mm. Ce module inclus un émetteur et un récepteur à ultrasons. Des puces de contrôle sont situées à l'arrière de la sonde et elle est équipée de 4 pins : 1x VCC +5v, 1x TRIG (Trigger pin), 1x ECHO (Echo pin), 1x GND. T = Transmitter = émetteur. R = Receiver = récepteur. (5)

(4) Test de la sonde à ultrason "HC-SR04" | Photo prise par S. Francois au laboratoire de l'Ephec en février 2018.
(5) *Deuxième série de photos de la sonde prise par S. Francois au laboratoire de l'Ephec en mai 2018.*

4.2 "Proteus ISIS" : La simulation électrique

Pour cette simulation sous Proteus, nous avons pris, comme point de repère, le schéma sur le "PIC 18F458" (*dans le syllabus d' "Electronique digitale et analyse des signaux" de Mr Dewulf à la page 89).*

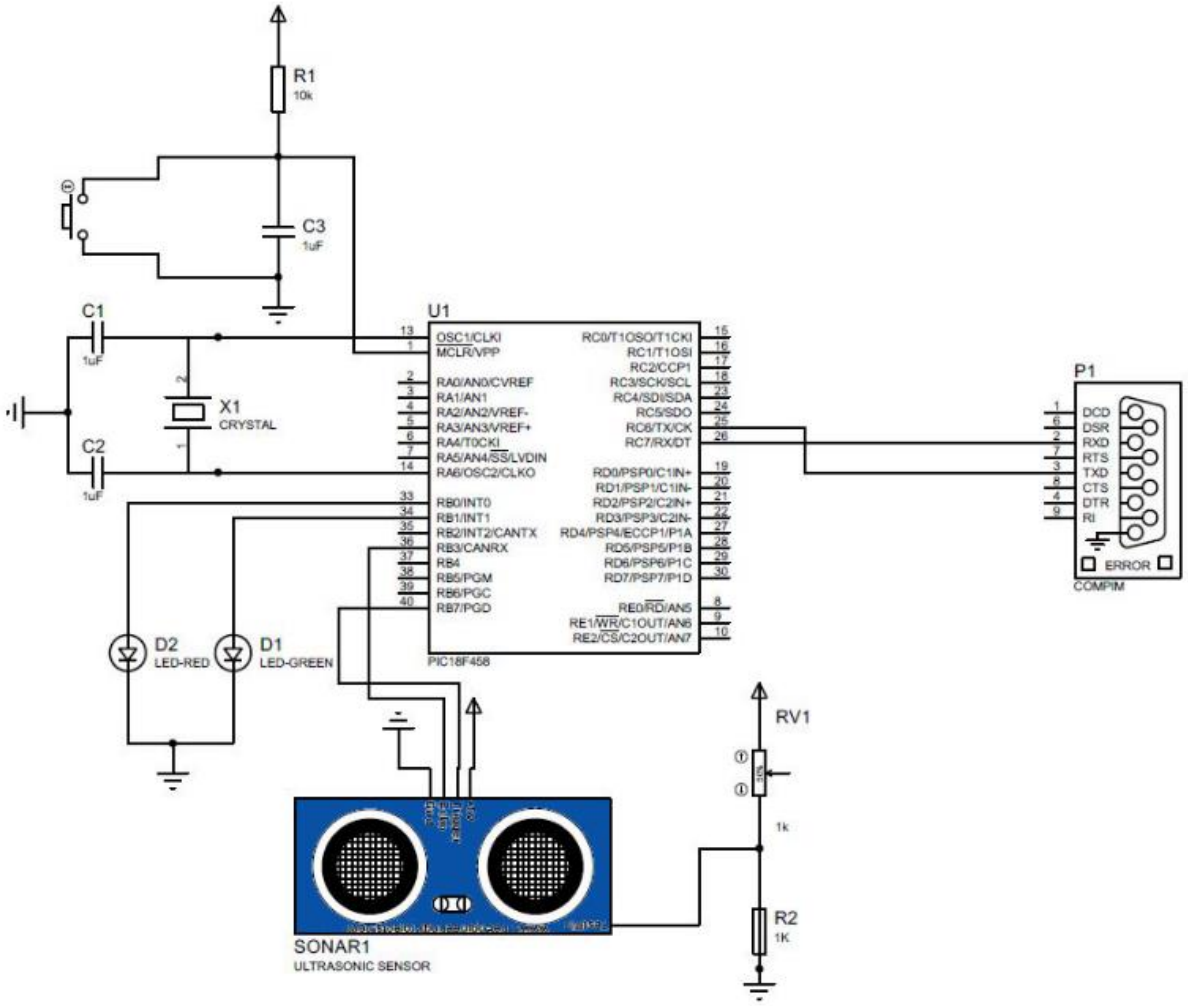


Schéma logique de la simulation électrique du projet "Télémètre à ultrason" (6)

Observation :

Comme nous l'avons mentionné durant notre test précédent, le microcontrôleur PIC est l'élément clé de notre projet. Il est composé d'un processeur, de mémoires, d'interfaces d'entrée et sorties. Comparé à un ordinateur, il a l'avantage d'avoir une faible consommation (de l'ordre de quelques milliwatts). Il est couramment utilisé pour des systèmes embarqués (électroménager …). Le PIC est programmable. Nous parlerons de cet aspect plus loin dans ce rapport.

Problème rencontré :

- Le seul souci rencontré était la fréquence du PIC => résolu en mettant la bonne fréquence à 20MHz. Pour le reste, tout s'est bien passé.

(6) *Le schéma logique de la simulation électrique du projet a été accompli sur "Proteus ISIS 8.6" (avec les librairies "Arduino.lib", "Arduino.idx" et "UltraSonicSensor.hex" (pour émuler la sonde à ultrason)).*

"Proteus ISIS" : suite

Photo d'un PIC18F458 | Source : site www.microchip.com | consulté en mail 2018.

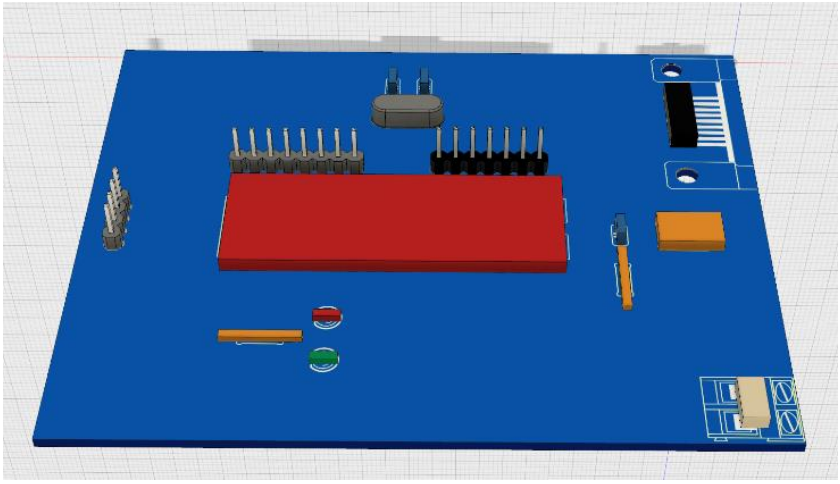
PIC – complément d'info :
La dénomination "PIC18" représente un PIC de la famille 18 (càd avec d'avantages d'instructions que d'autres PIC), le "F" signifie une mémoire Flash, le nombre "458" est le modèle.
Ce PIC n'intègre pas de contrôleur USB en natif, sa programmation se fait par l'interface RS232 (port COM).
Dans notre situation, nous avons choisi d'opter pour un port USB femelle FTDI qui joue le même rôle qu'un RS232 DB9. Nous reparlerons de ce FTDI lorsque nous aborderons la partie soudure des composants.

D'autres composants viennent se connecter au PIC :

- A la pin 1 (MCLR) => Le bouton reset sert à l'initialisation du PIC (en cas d'erreurs de mémoire, de pannes de tension …).
- Aux pins 13 (CLK1) et 14 (CLK0) => Une horloge à quartz de 20MHz qui sert à cadencer le PIC.
- A la pin 33 (INT0) => led rouge clignotante signalant une erreur.
- A la pin 34 (INT1) => led verte signalant que tout est OK.
- A la pin 36 (CANRX) => la pin "echo" de la sonde à ultrason "HC-SR04" vient se brancher sur la pin de réception du PIC et convertit ce signal analogique vers un signal numérique.
- A la pin 40 (PGD) => la pin "trigger" de la sonde à ultrason "HC-SR04" vient se brancher sur la pin PGD du PIC
- A la pin 25 (RC6/Tx/CK) => vers la pin TXD du port DB9 (FTDI dans notre projet)
- A la pin 26 (RC7/Rx/DT) => vers la pin RXD du port DB9 (FTDI dans notre projet)

4.3 "EAGLE" - CAO : circuit imprimé

Eagle (Easily Applicable Graphical Layout Editor) est un outil de dessin de circuits imprimés.
Dans le cadre de ce projet, nous travaillons avec la version de démonstration gratuite, limitée à deux couches au format européen 10 x 8 cm. Ce logiciel permet de faire du routage manuel et automatique de circuits imprimés.



Rendu 3D du dessus du PCB sous Eagle 8.6.3.

Observation :
La première chose que nous avions faite, a été de vérifier si les librairies dont nous avions besoin pour le projet étaient bien activées (càd en "In use "). Pour la conception du circuit imprimé, nous avons sélectionné et disposé les composants de manière à ce que leurs soudures viennent se faire sur la face de dessous du PCB. Dans Eagle, ce choix se traduit par la sélection de composants avec étiquettes vertes uniquement (= soudure en dessous).
La suite s'est passé sans soucis.
Le circuit imprimé a passé avec succès les tests de contrôles sur le site en ligne d'Eurocircuits.

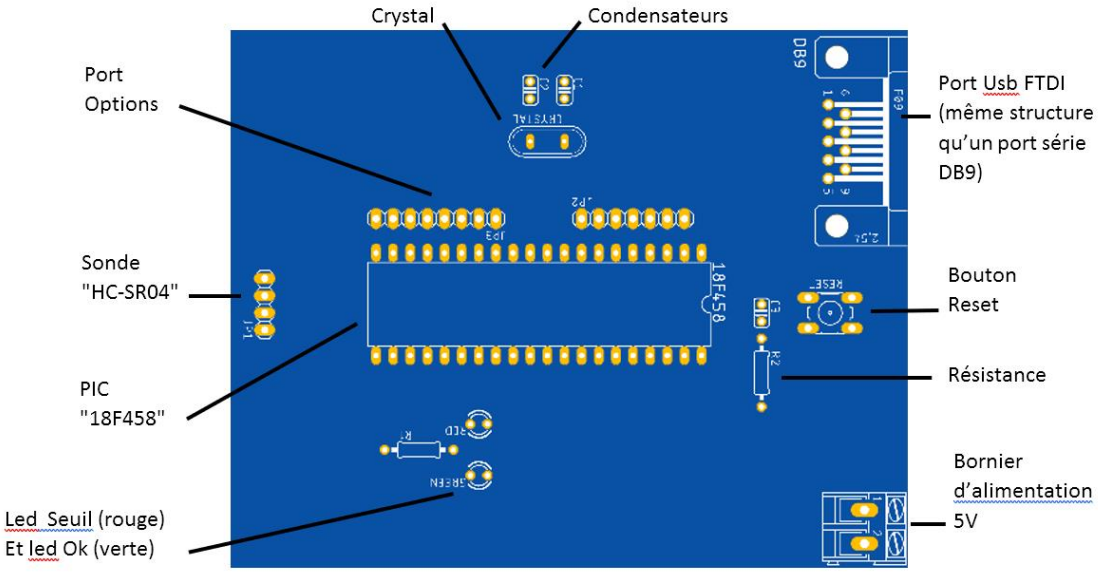


Schéma physique du PCB (en vue de dessus)

"EAGLE" - CAO : suite

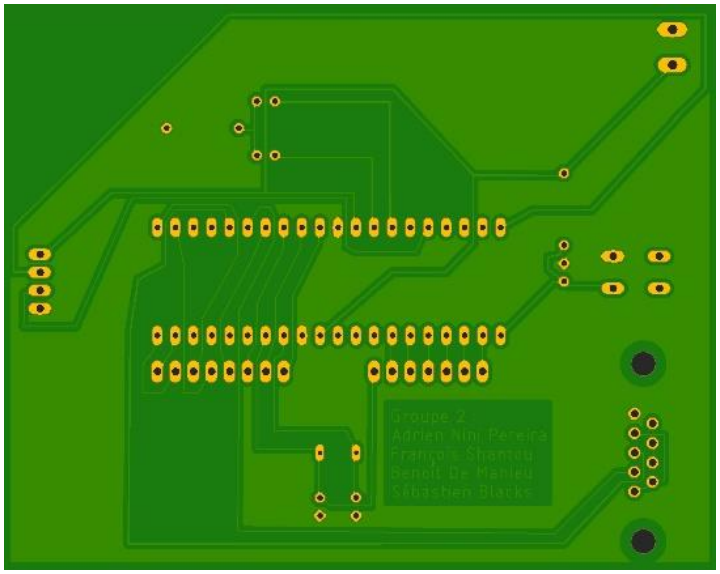


Schéma physique du PCB (en vue du dessous) nous montrant les pistes et les zones conductrices en cuivres sur lesquelles nous viendrons déposer les soudures.

Problème rencontré :

- Un seul pépin rencontré était la position des pins de la sonde qui n'était pas dans le bon sens => résolu en coupant les pistes conductrices et en remettant la sonde à sa bonne place et ensuite, en reconnectant les chemins coupés, sauvant le boulot de devoir tout recâbler le circuit.

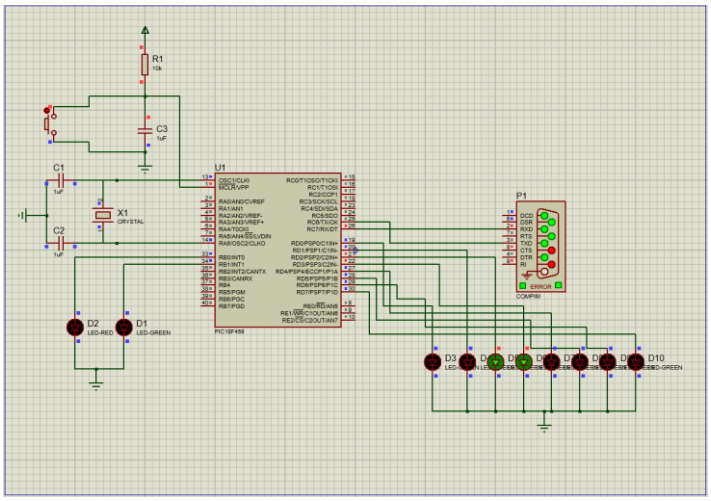
4.4 "Langage C" : communication avec le PIC

L'écriture du langage C a été utilisée pour communiquer avec le "PIC 18F458".
Le code C est compilé en héxadécimal et est envoyé dans le PIC via le logiciel libre "Tiny Bootloader" par le port série RS232 ou une conversion vers le port USB (recours au FTDI USB vers DB9 dans notre projet).

Observation :
Après les déclarations d'une méthode de conversion char vers int, d'un timer, d'une interruption pour lire les données arrivant de l'application Java, la boucle infinie du programme commence.

La fonction *dist()* est appelée en premier lieu afin de gérer l'utilisation de la sonde à ultrasons. Pour ce faire, le PIC envoie une impulsion sur la pin b7, qui est reliée au trigger de la sonde. Lorsque la sonde renvoie une impulsion sur la pin b3 du PIC, celui-ci démarre un timer le temps que l'impulsion soit haute et arrête ce même timer une fois que l'impulsion envoyée par la sonde à ultrasons soit finie. La valeur de ce timer multipliée par 0.00344 correspond à la distance entre la sonde à ultrasons et l'obstacle en centimètre. Si la distance calculée par la sonde est inférieure à la limite fixée par l'application Java, nous allumons la led verte. Si la distance est supérieure ou égale à la limite de l'application Java, nous changeons d'état la led rouge et nous vérifions que la led verte soit bien éteinte. La distance est enfin envoyée par le port Com à l'application Java afin de l'afficher.

L'interruption définie lorsqu'une donnée arrive sur le port Com fait appel à une fonction lecture. Dans cette fonction, un buffer de cinq charactères est défini afin de récupérer, grâce à la fonction *gets,* les infos envoyées par l'application Java. Une boucle parcourt ensuite ce buffer de cinq charactères afin de savoir combien de charactères sont utilisés. Une autre boucle ensuite transforme chaque caractère en un chiffre. Après, le tout mis ensemble forme la limite.



Opération de test du code C sur la sortie D via huit Leds.

Problèmes rencontrés :
- Lors de cette partie du projet plusieurs problèmes se sont posés. La bonne maitrise de CCS qui nous a ralenti un certain temps avant de bien comprendre les possibilités de cet IDE. Le raccordement entre l'application Java et le code C a aussi fait surgir quelques problèmes de compatibilité entre les deux codes. Mais après concertation, nous avons pu trouver certaines fonctions permettant de faire fonctionner ces deux programmes correctement. Nous avons aussi eu une incohérence lors des essais du code C sur simulation Proteus ISIS, où la distance donnée par la sonde différait de manière anormale due à notre mauvais calcul de distance.

"Langage C" : suite C.1

Code source langage C (fichier "main.c" | partie 1 - Début) :

```
#include <main.h>

int32 lecture(void);
int32 dist(void);
int32 limite = 10;

#INT_RDA
void RDA_isr(void){
  limite = lecture();
}

void main(){
  output_high(PIN_B0);
  output_high(PIN_B1);
  #define toint(c)  ((c & 0x5F) > '9' ? c - '7' : c - '0') //conversion d'un char en int (0,9)
  setup_timer_0(RTCC_INTERNAL);  // 409us (us = microseconde)  overflow
  setup_timer_1(T1_INTERNAL);  //13,1 ms overflow
  enable_interrupts(GLOBAL);
  setup_low_volt_detect(FALSE);
  enable_interrupts(INT_RDA);
  enable_interrupts(GLOBAL);
  delay_ms(100); //délai initialisation
  output_low(PIN_B0);
  output_low(PIN_B1);
  int32 distance=0;

while(true){
    distance = dist(); //calcul de la distance
    if(distance<limite){ //si la distance est assez grande
      output_low(PIN_B0); //si la sortie est basse led rouge
      output_high(PIN_B1); //si la sortie est haute led verte
    }else{
      output_low(PIN_B1); //vérifie que la led verte est bien éteinte
      output_toggle(PIN_B0); //la led rouge clignote
    }
    printf("%lu\r\n",distance);  //envoi du chiffre par le port Comm
    delay_ms(300);
  }
}

int32 lecture(){
    char entre[5]; //buffer de cinq caractères
    int32 sorti=0;
    int i=0,x=0,y=1; //sorti=int voulu
    gets(entre); //attend une chaîne de caractères finie par CR carriage return
      for(x=0;x<5;x++){ //boucle lisant le buffer
        if(entre[x]=='\r'){break;} //quand le char == \r => fin du int
      }
      for(i=x-1;i>0;i--){ //boucle traduisant les char en un int
        sorti+=toint(entre[i])*y; //la variable sorti est incrémentée du int* dizaine /centaine
        y*=10;
      }
      sorti+=toint(entre[0])*y; //la boucle ne veut pas se finir à zéro alors voilà :'(
      return sorti;
}
```

"Langage C" : suite C.2

Code source langage C (fichier "main.c" | partie 2 - Fin) :

```
int32 dist(){
  float time=0;
  int32 distance=0;
  output_high(pin_b7); //début de l'impulsion sur le trigger de la sonde à ultrasons
  delay_us(20);
  output_low(pin_b7); //fin de l'impulsion
  while(!input(PIN_b3)); //tant que la pin b3/canrx du PIC est à low  => reliée au echo de la  sonde
  set_timer1(0); //mise à zéro du timer 1
  while(input(PIN_b3)); //tant que la pin b3/canrx  du PIC est à high
  time=get_timer1(); //récupération du timer
  distance = time*0.00344; //calcul de la distance
  return distance;
}
```

"Langage C" : suite C.3

Code source langage C (fichier "main.h") :

```
#include <18F458.h>
#device adc=16

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES WDT128                //Watch Dog Timer uses 1:128 Postscale
#FUSES RC_IO                 //Resistor/Capacitor Osc
#FUSES NOOSCSEN              //Oscillator switching is disabled, main oscillator is source
#FUSES NOPUT                 //No Power Up Timer
#FUSES NOBROWNOUT            //No brownout reset
//#FUSES BORV20              //Brownout reset at 2.0V
#FUSES STVREN                //Stack full/underflow will cause reset
#FUSES NOLVP                 //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NODEBUG               //No Debug mode for ICD
#FUSES NOPROTECT             //Code not protected from reading
#FUSES NOCPB                 //No Boot Block code protection
#FUSES NOCPD                 //No EE protection
#FUSES NOWRT                 //Program memory not write protected
#FUSES NOWRTC                //configuration not registers write protected
#FUSES NOWRTB                //Boot block not write protected
#FUSES NOWRTD                //Data EEPROM not write protected
#FUSES NOEBTR                //Memory not protected from table reads
#FUSES NOEBTRB               //Boot block not protected from table reads

#use delay(clock=20000000) //horloge à 20MHz
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,restart_wdt)
```

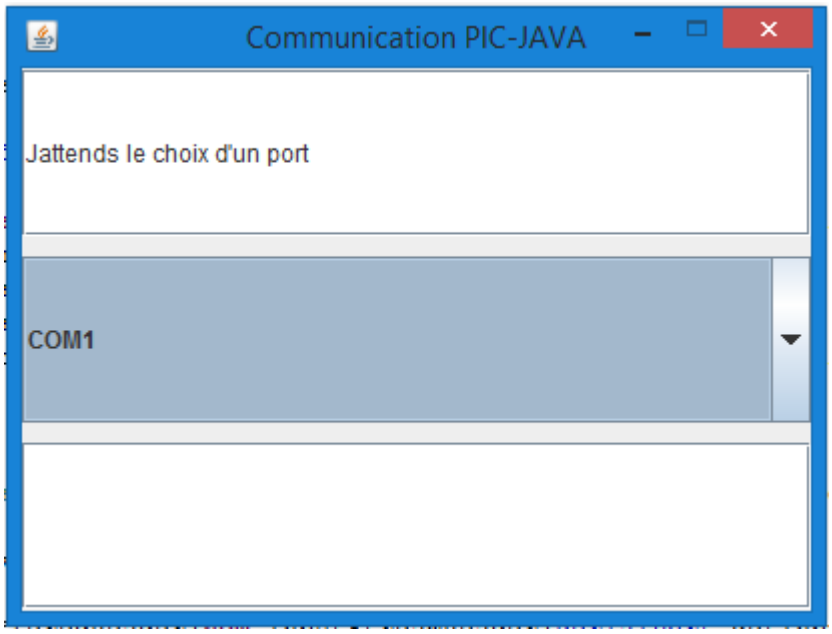
4.5 "JAVA" : interface utilisateur

Du côté utilisateur, nous employons une application JAVA qui communique avec le PIC, qui affiche la distance mesurée et qui permet de définir un signal d'alerte lorsque cette distance tombe en dessous d'un seuil critique ("treshold"). Si aucune alerte n'est en cours, une LED verte reste allumée. S'il y a une alerte, une LED rouge clignote.

Nous utilisons l'API "RxTx" pour piloter le port RS232 (remplacé par le FTDI USB to DB9 dans notre projet – chip 751-1197) et qui émule le port Com.

Observation :

Concrètement, nous avons utilisé la programmation Java apprise lors du premier quadrimestre. Avant d'attaquer le code à proprement dit, nous avons regardé les librairies requises et vérifier si elles sont compatibles avec notre hardware. Ensuite, après de longues recherches sur le sujet et avoir revu l'exemple du chat, du premier quadrimestre, nous avons pu faire un lien entre les deux. Ce travail nous a permis d'appliquer encore ce que nous avions fait lors du cours de Java. De plus, nous avons pu coordonner la couche "Java" pour qu'elle fonctionne avec la couche "langage C" (communications des deux codes l'un vers l'autre et vice versa).



Prototype alpha de notre logiciel Java de selection de port.

Problèmes rencontrés :

- Trouver le moyen pour ouvrir/envoyer les ports => résolu en trouvant son instruction.
- Incompatibilité de librairie (32 vs 64 bits) => résolu en chargeant la bonne version de libraire x64 bits.
- Envoie limité à un caractère => résolu grâce à des buffers et l'exemple du chat du premier quadri.
- Problème dans la fin de chaîne => envoie du CR au bon endroit.
- Autres problèmes => envoie dans un buffer.

"JAVA" : suite J.1

Code source Java (fichier "JavaApplication.java" | partie 1 - Début) :

```
package projetElec;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import gnu.io.*;

@SuppressWarnings("serial")
public class JavaApplication extends JFrame implements ActionListener, SerialPortEventListener{


	//Eléments pour construire la fenêtre
	private SerialPort serialPort;
	private CommPortIdentifier portId;
	private JTextField field1;
	private JTextField field2;
	private PrintWriter out;
	private BufferedReader in;
	private JComboBox<Object> listeCom;
	Object[] com = new Object[] {"COM1", "COM2", "COM4", "COM5"};
	Object selected;


	//Début de la classe
	public JavaApplication() {

		field1 = new JTextField();
		field2 = new JTextField();
		listeCom = new JComboBox<Object>(com);
		this.setLayout(new GridLayout(3, 2, 10, 10));
		this.getContentPane().add(field1);
		this.getContentPane().add(listeCom);
		this.getContentPane().add(field2);

		field1.addActionListener(this);
		this.setTitle("Communication PIC-JAVA");
		this.setSize(400, 300);
		this.setResizable(false);
		this.setLocationRelativeTo(null);
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		//this.setAlwaysOnTop(true);
		this.setVisible(true);
		field1.setText("Jattends le choix d'un port");
```

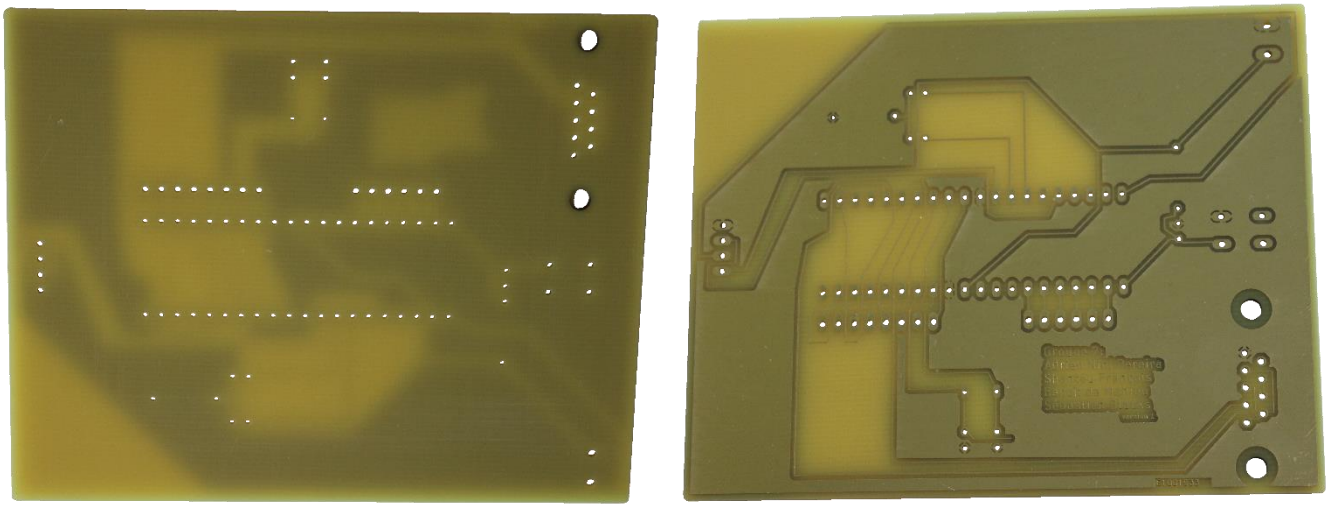
"JAVA" : suite J.2

Code source Java (fichier "JavaApplication.java" | partie 2 - Fin) :

```
    //action du choix du port sur lequel l'application java fonctionne
                listeCom.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent arg0) {
                                port();
                        }
                });
        }

        //Méthode pour ouvrir la connexion avec le port Comm
        public void port() {
                try {
                        selected = listeCom.getSelectedItem();
                        portId=CommPortIdentifier.getPortIdentifier(selected.toString());
                        serialPort=(SerialPort)portId.open("",100);
                        serialPort.addEventListener(this);
                        serialPort.notifyOnDataAvailable(true);
                        field1.setText("");

                }catch(Exception e) {
                        JOptionPane.showMessageDialog(null, "Le port " + selected + " n'est pas ouvert");}
        }

    //Action pour envoyer le message vers l'application Proteus
        @Override
        public void actionPerformed(ActionEvent m) {
                if(m.getSource() == field1) {
                        String x = field1.getText() + "\r";
                        try {
                                out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(serialPort.getOutputStream())), true);
                                out.print(x);
                                out.close();
                                field1.setText("");
                        }catch(Exception e) {JOptionPane.showMessageDialog(null, e.toString());}
                }
        }

    //Méthode qui écoute sur le port série pour recevoir les messages venant de Proteus
        @Override
        public void serialEvent(SerialPortEvent arg0) {
                try {
                        in = new BufferedReader(new InputStreamReader(serialPort.getInputStream()));
                        String tes = in.readLine();
                        field2.setText(tes);
                        in.close();
                }catch(Exception e) {JOptionPane.showMessageDialog(null, e.toString());}
        }
}
```

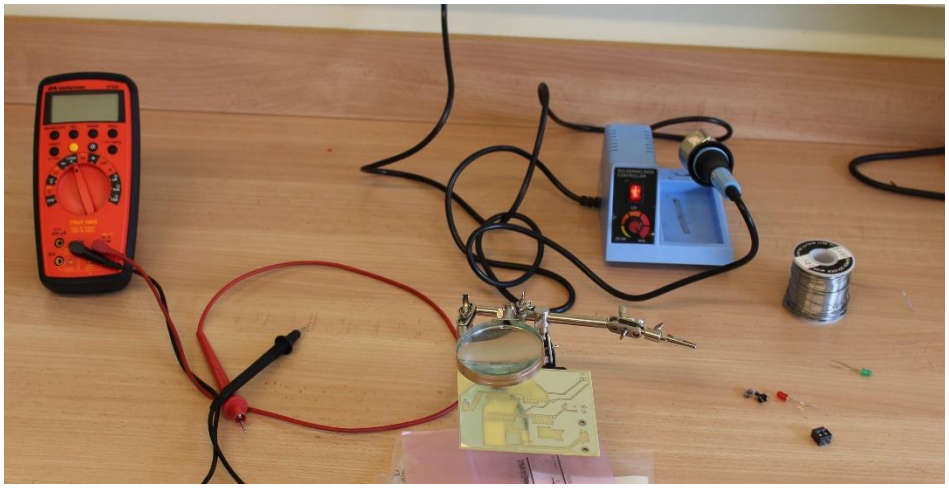
4.6 "Soudure des composants" : PCB

Après la fin des vacances de Pâques, nous avons reçu, avec joie, nos circuits imprimés (Eurocircuits).



PCB nu en vue de dessus (à gauche), PCB nu en vue de dessous (à droite).

La première étape, a été de réunir les composants dont nous avions besoin pour le projet et de préparer le matériel de soudure.



Soudure - Plan de travail. (7)

Fil à souder :
=> Features :
Size : 60% | Dia : 0.8 mm

Observation :
Quelques semaines auparavant, nous nous sommes entraînés à souder et dessouder des composants de toutes sortes (fils, résistances …) pour prendre un peu la main.

Remarque 2 : Pour ce projet, toutes les soudures se font du côté du dessous du PCB.
Remarque 3 : Le multimètre est à proximité pour contrôler les soudures du circuit.

(7) Photo de l'installation de soudure prise par S. Francois au laboratoire de l'Ephec en avril 2018.

"Soudure des composants" : suite

Liste des composants :

- 1 x Bornier d'alimentation (avec 2 pôles (1x VCC, 1x GND) et un pas de 5mm) sur lequel nous viendrons y connecter une alimentation 5V.
- 1 x Led verte (de 3mm et un pas de 2,54mm).
- 1 x Led rouge (de 3mm et un pas de 2,54mm).
- 1 x Résistance de 330 Ohms (du côté des Leds).
- 1 x Résistance de 10k Ohms (du côté du bouton Reset).
- 1 x Crystal/Quartz (de 20 MHz et un pas de 5 mm).
- 1 x DIP IC socket - Turned Pin Contact 40 (DIP = Dual Inline Package et un pas de 2,54 mm).
- 1 x Bornier 6 broches mâles (pour des accessoires optionnels au PIC | avec un pas de 2,54 mm).
- 1 x Bornier 8 broches mâles (pour des accessoires optionnels au PIC | avec un pas de 2,54 mm).
- 1 x Bornier 4 broches femelles (pour y connecter la sonde à ultrasons HC-SR04).
- 1 x Bouton Reset (avec un pas de 4,5 mm sur 6,5 mm)
- 2 x Condensateurs de 15 pF (du côté du Crystal et avec un pas de 2,54 mm)
- 1 x Condensateur de 684 uF (du côté du bouton reset et avec un pas de 2,54 mm)
- 1 x FTDI (pour l'interface "Java" de communication utilisateur, sélection du niveau de "seuil".

Petit mot sur le FTDI :

Le FTDI est utilisé ici pour remplacer un connecteur DB9 (RS-232/UART TTL) par une connexion USB. Il possède le même espacement entre les pins qu'un DB9. Ce module FTDI inclus un circuit imprimé qui sert de passerelle série-USB et tous les composants électroniques nécessaires à la conversion entre les signaux USB et RS-232/UART.

Photo provenant du site : http://media.futureelectronics.com/SEMICONDUCTORS/SIGNAL-INTERFACE | Consulté en mai 2018.

Problèmes rencontrés :

- En soudant des pièces, les mains tremblaient. => résolu en ajustant son assise, sa respiration.
- En voulant dessouder un composant sur notre première version de PCB, nous avons glissé légèrement avec le fer à souder et cassé une piste du circuit. => résolu en rajoutant un fil conducteur.
- Durant une autre intervention, nous avons cassé, cette fois-ci, un plus grand tronçon de circuit.

=> Malgré la qualité moyenne des PCB, cette expérience a été enrichissante. Ce problème a été résolu en prenant notre 2ème circuit imprimé et en refaisant les étapes de soudures des composants.

- Extraction du PIC pour le reprogrammer => Aller délicatement pour sortir le PIC de son socket (l'idéal étant de le faire avec un outil spécialisé d'extraction de chipset) si nous ne voulons pas abîmer des pattes de connexion.

5) Conclusion :

Au terme de cette mission de quatre mois, nous indiquons ici une liste des réalisations qui fonctionnent à ce jour :

- La simulation électrique.
- La réalisation du schéma électrique du circuit imprimé.
- La conception du code C en simulation.
- Le code java.
- La conception du circuit imprimé et la soudure des composants.

Par ailleurs, nous avons rencontré certaines difficultés.
D'un point de vue tests réels, après injection du code C, dans la rom du PIC, sur notre circuit imprimé, rien ne se passe. Mais l'alimentation est bonne, le java et les drivers aussi.
Peut-être, s'agit-il d'une incompatibilité avec le port FTDI DB9 to USB ?

D'un point de vue debug et temps, nous avons bien avancé. Cependant, nous avons été bloqués par l'absence de réponse du FTDI dans les deux sens et nous n'avons pas pu passer aux séries de tests (benchmark de mesures de la précision de la sonde à ultrasons). Nous aurions eu besoin de temps pour dessouder et tester les composants un par un, si l'un d'eux était défectueux.
Nous pourrions aussi remplacer le FTDI par une carte additionnelle, contenant à la place, un DB9 avec ses composants nécessaires à son fonctionnement.
En cas de réussite, cela aurait pu nous confirmer s'il s'agissait d'une incompatibilité avec le FTDI.

Dernière difficulté rencontrée, lors de la conception du code C, celui-ci ne fonctionne pas en réel.

Comme depuis le rapport intermédiaire, nous avons pu continuer à mettre nos connaissances de première et de deuxième année, en pratique, nous rapprochant, chaque jour, de plus en plus vers le monde professionnel.

5.1 Pistes envisagées pour la suite du développement du projet :

- Tester dans différents environnements et vérifier le taux de précision du télémètre.
- Tester la surchauffe des composants de la carte à l'aide d'une caméra thermique en Idle (faible charge), en moyenne charge (usage normal) et en full load (usage intensif).
- Réaliser un télémètre portatif sur batterie et/ou avec un panneau solaire en option.
- Concevoir un boitier de protection pour le circuit imprimé.
- Inclure un écran LCD pour avoir une lecture directe en temps réels (sans avoir besoin d'être connecté à un ordinateur).
- Ajouter une possibilité de sauver les relevés de mesures de manière Offline (via un port microSD card) et Online (via une puce Wi-Fi/3G/4G) (Platform Web / iOS / Android).
- Prévoir une version robuste du circuit imprimé pour un usage dans des conditions industriels, en extérieur, sur chantier ou encore dans des conditions extrêmes (avec une sélection de composants de classe militaire et avec une résistance à l'eau (IP68) et à la poussière).

6) Sources :

- **6.1 Biblographie :**

=> Christian Tavernier, Microcontrôleurs PIC 18 (2è Edition février 2012), DUNOD, 360 pages.
www.dunod.com/sciences-techniques/microcontroleurs-pic-18-2e-edition-description-et-mise-en-oeuvre

- **6.1 Software :**

Outils de communication
=> Facebook => https://fr-fr.facebook.com/
=> GitHub => https://github.com/
=> Trello => https://trello.com/

Electronique
=> "Proteus ISIS" : La simulation électrique => https://www.labcenter.com/downloads/
Avec librairies arduino.lib, arduino.idx (sonde à ultrasons) et UltraSonicSensor.hex pour émuler la sonde.
=> Proteus Tutoriel de 249 pages => https://labcenter.s3.amazonaws.com/downloads/Tutorials.pdf
=> Eagle => www.autodesk.com/education/free-software/eagle

Programmation
Langage C
=> Langage C avec "C Compiler for PIC 9.80" => CCS PCWH Compiler => www.ccsinfo.com/downloads.php
Manuel de 597 pages (version Fevrier 2018) => http://www.ccsinfo.com/downloads/ccs_c_manual.pdf
=> Tiny Bootloader => www.etc.ugal.ro/cchiculita/software/picbootloader.htm
=> Terminal (Emulation du port série) => https://sites.google.com/site/terminalbpp/

JAVA
=> Java Eclipse Mars
=> API "RxTx" => port RS232 (ou le FTDI USB dans notre projet) => gestion du port COM =>
http://users.frii.com/jarvi/rxtx/

- **6.1 Autres :**

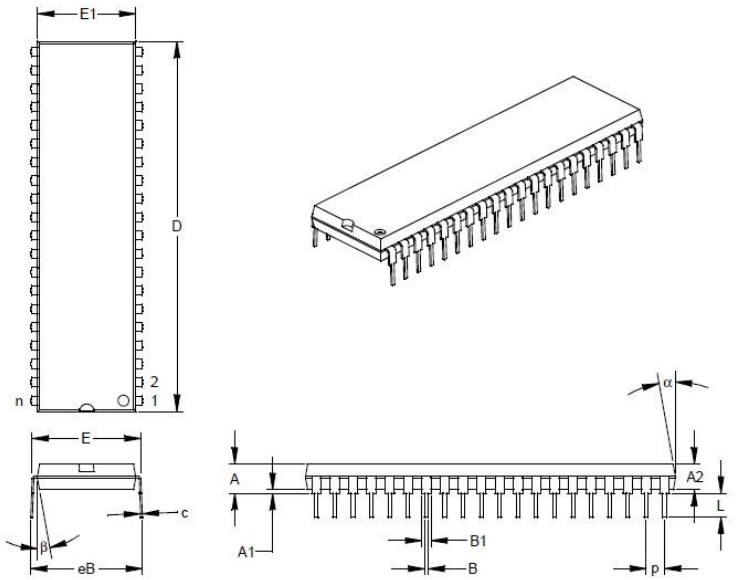
=> Eurocircuits – test online et impression des PCB => www.eurocircuits.com

- **Datasheets**

6.2 Composants :

. Microcontrôleurs "PIC 18F458" (40 pins avec un pas de 2,54 mm)
=> www.microchip.com/wwwproducts/en/PIC18F458#additional-features
=> Datasheet (402 pages) : ww1.microchip.com/downloads/en/DeviceDoc/41159e.pdf
=> Features :
Program Memory Type : Flash
Program Memory Size : 32 kb
CPU Speed (MIPS/DMIPS) : 10
SRAM : 1536 bytes
Data EEPROM/HEF : 256 bytes
Digital Communication Pripherals : 1 UART, 1 SPI, 1 I2C1-MSSP(SPI/I2C)
Capture/Compare/PWM Peripherals : 1 CCP, 1 ECCP
Timers : 1 x 8 bits, 3 x 16 bits
ADC Input : 8 ch, 10 bits
Number of Comparators : 2
Number of CAN Modules : 1 CAN
Temperature Range : -40 °C to 125 °C
Operating Voltage Range : 2 V to 5.5V
Pin Count : 40

=> Additional Features :
- 5 x 10 bits PWM
- 40 MHz Max. Speed
- Full CAN 2.0B Active 3Tx Buffers
- 2Rx Buffers
- 6 Full Acceptance Filters
- 2 Filter Masks
- PSP
- ICD
- Self-Programming

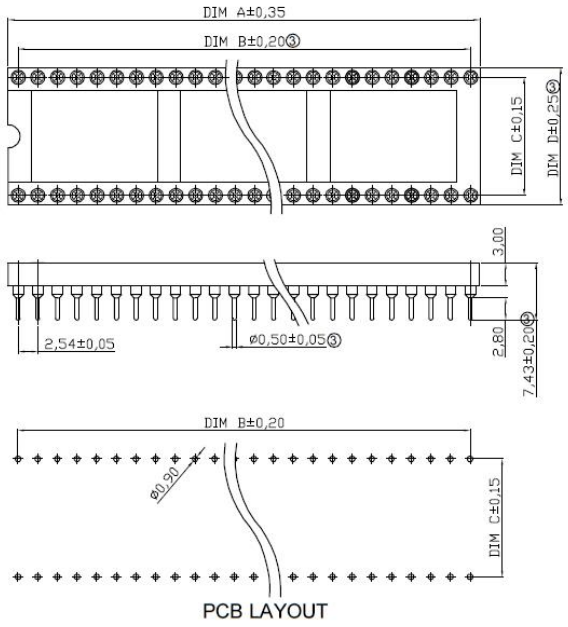


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-011
Drawing No. C04-016

Datasheets : suite D.1

. Bornier femelle 40 pins pour le PIC
DIP IC socket - Turned Pin Contact 40 (avec un pas de 2,54 mm)
Les dessus des pins sont de forme conique. DIP = Dual Inline Package.
=> Datasheet : http://www.assmann-wsw.com/fileadmin/datasheets/ASS_4852_CO.pdf
=> Features :

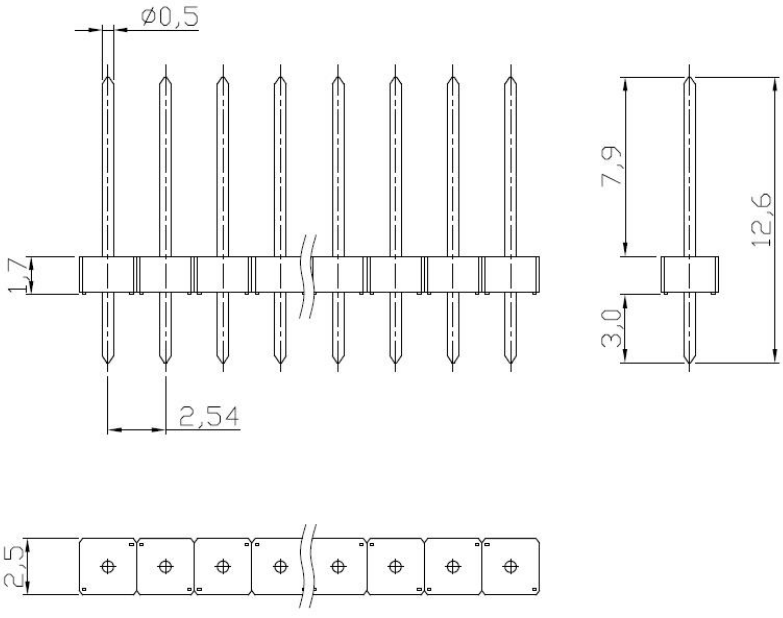


NOTES:
MATERIAL
1.Pin(outer sleeve): Brass, machined CuZn38Pb2
③2.Clip(contact 4 finger): Beryllium copper, heat treated
3.Plating(outer sleeve): HZL: 2um/80u"nickel, 5um/200u"Tin
HGL: 2um/80u"nickel, Gold flash
4.Plating(contact): 2um/80u"nickel, Gold flash or 10u", 30u" Gold
5.Insulator Material: Glass filled PBT, UL94V-0 Black
ELECTRICAL
1.Current Rating: 3Amps/contact max.
2.Contact Resistance: 4mΩ/contact max.
3.Insulation Resistance: ≥10000MΩ at 500VAC
4.Operating Voltage: 100 VRMS/150VDC
MECHANICAL
1.Average Insertion force with steel pin of ø0.43mm/0.017": <250g
2.Average Withdrawal force with steel pin of ø0.43mm/0.017": >50g min.
3.Mechanical life cycle: 200 min.
4.Operation Temperature: -40°C to +105°C
④5.Wave Soldering temperature: +245°C, 5-10s max.

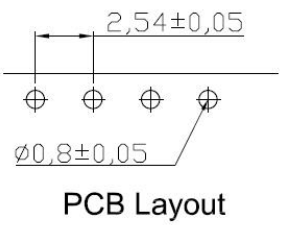
Packing: Tube

Contact	Dim A	Dim B	Dim C	Dim D
36	45.72	43.18	15.24	17.78
40	50.80	48.26	15.24	17.78
42	53.34	50.80	15.24	17.78
48	60.96	58.42	15.24	17.78
50	63.50	60.96	15.24	17.78
52	66.04	63.50	15.24	17.78
50	63.50	60.96	22.86	25.40
52	66.04	63.50	22.86	25.40
64	81.28	78.74	22.86	25.40

. 1x Bornier 6 broches mâles (pour des accessoires optionnels au PIC | avec un pas de 2,54 mm).
. 1x Bornier 8 broches mâles (pour des accessoires optionnels au PIC | avec un pas de 2,54 mm).
=> Datasheet : http://www.assmann-wsw.com/fileadmin/datasheets/ASS_1223_CO.pdf
=> Features :



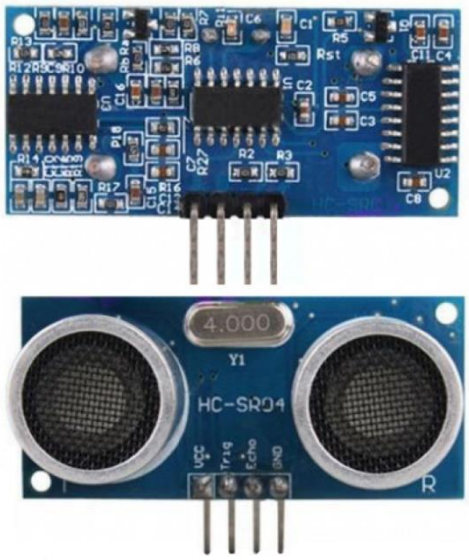
Notes:
1.Insulator: PA4.6(UL94V-0)
2.Contact: CuZn30
3.Plating: 2u Ni+0,1-0,25u Au
4.Insulation Resistance: > 10^9 Ω
5.Contact Resistance: <20m Ω
6.Current Rating :2.5AM
7.Operating Voltage : 250V AC
8.Temperature : -55°C to +125°C
9.Solder Temperature : 235°C(30-60 sec)
max. 260°C(10 sec)

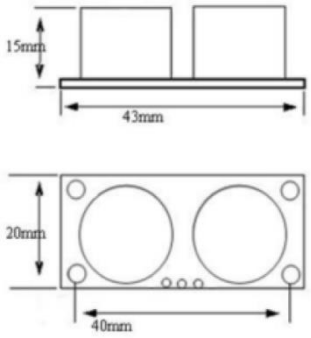


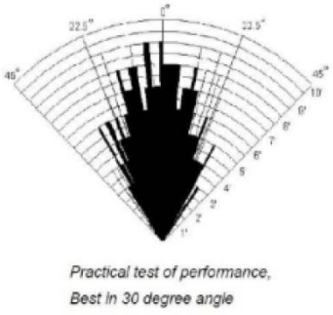
PCB Layout

Datasheets : suite D.2

. Sonde à ultrasons "HC-SR04" (avec un pas de 2,54 mm)
=> Attention : D'après plusieurs sites d'électroniques, il est important de connecter la masse (GND) avant de placer l'alimentation sur VCC car cela impacte fortement le fonctionnement du module.
=> Datasheet : http://web.eece.maine.edu/~zhu/book/lab/HC-SR04%20User%20Manual.pdf
=> Features :
Power Supply :+5V DC
Quiescent Current : <2mA
Working Currnt: 15mA
Effectual Angle: <15°
Ranging Distance : 2cm – 400 cm or 1" - 13ft
Resolution : up to 3 mm
Measuring Angle: 30 degree
Trigger Input Pulse width: 10uS
Dimension: 45mm x 20mm x 15mm
T : Transmitter
R : Receiver

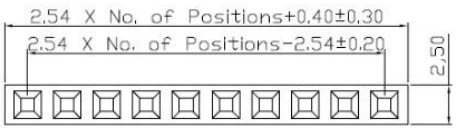


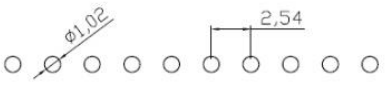




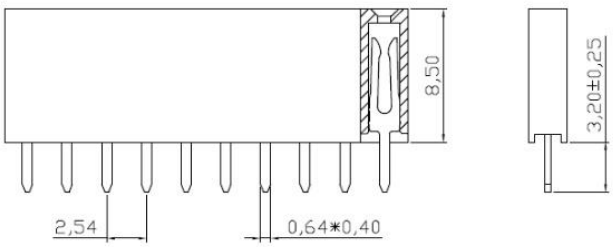
VCC = +5VDC
Trig = Trigger input of Sensor
Echo = Echo output of Sensor
GND = GND

.Bornier 4 broches femelles pour la sonde (avec un pas de 2,54 mm)
=> Datasheet : http://www.assmann-wsw.com/fileadmin/datasheets/ASS_1034A_CO.pdf
=> Features :





P.C.B. LAYOUT



NOTES:

MATERIAL
1.Insulator Material:PBT+30%G.F(UL94V-0),Black
2.Contact Material : Brass
3.Contact Plating : Gold Flash or Tin plated
ELECTRICAL
1.Current Rating : 3.0Amps/contact max
2.Withstand Voltage : 1000V AC/minute
3.Contact Resistance : 20mΩ/contact max
4.Insulation Resistance: 1000MΩ min at Voltage=100V
ENVIRONMENTAL
1.Operation Temperature: -55°C to +105°C (Gold plated)
-40°C to +105°C (Tin plated)
2.Mechanical life cycle: min.200
Packing: Tray

Order code: A-BL254-EG-xxxD
Contact plating — No. of Contacts
G: Gold flash
Z: Tin plated

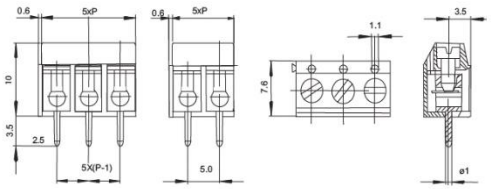
Datasheets : suite D.3

. Bornier d'alimentation (avec 2 pôles et un pas de 5 mm)
=> Datasheet : http://data.leocom.kr/datasheets/329719_611087.pdf
=> Features :

5mm Pitch Low Profile Screw Terminal

- 2 & 3 pole interlocking
- Test probe facility
- Moulded in black UL94V-0 PA (GF)
- Wire protector terminals
- UL approved
- Guided pin alignment

CTB5202 series

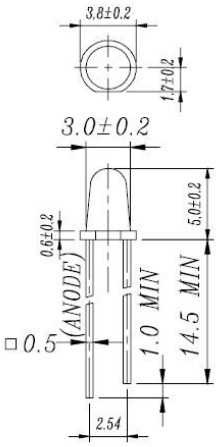


Type No.	Poles	Length
CTB5202/2	2	10mm
CTB5202/3	3	15mm
CTB5202/4	4	20mm
CTB5202/5	5	25mm
CTB5202/6	6	30mm
CTB5202/7	7	35mm
CTB5202/8	8	40mm
CTB5202/9	9	45mm
CTB5202/10	10	50mm
CTB5202/11	11	55mm
CTB5202/12	12	60mm

Rating	300V/15A
Insulation resistance	5000M Ω at 1000V
Operating temperature	-33°C to 120°C
Cable entry	1.5mm²
Withstand Voltage	AC2000/1Min
Recommended PCB hole size	ø1.2mm

.Diode rouge (3 mm et un pas de 2,54 mm)
.Diode verte (3 mm et un pas de 2,54 mm)
=> Datasheet : www.distrelec.be/Web/Downloads/_t/ds/1254-10sygd-s530-e2_eng_tds.pdf
=> Features :

Package Dimensions



Notes: 1. All dimensions are in millimetres
2. The height of flange must be less than 1.5mm(0.059").
3. Without special declared,the tolerance is ±0.25mm.

■ Absolute Maximum Ratings at Ta = 25℃

Parameter	Symbol	Rating	Unit
Forward Current	IF	25	mA
Operating Temperature	Topr	-40 to +85	℃
Storage Temperature	Tstg	-40 to +100	℃
Soldering Temperature	Tsol	260 ± 5	℃
Electrostatic Discharge	ESD	2000	V
Power Dissipation	Pd	60	mW
Reverse Voltage	VR	5	V

Datasheets : suite D.4

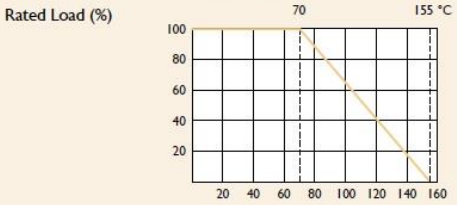
.1 x résistance de 10k Ohms (du côté du bouton Reset).
. 1 x résistance de 330 Ohms (du côté des Leds).
=> Datasheet : http://www.yageo.com/documents/recent/Yageo%20LR_CFR_2013.pdf
=> Features :

FEATURES

Power Rating	1/6W, 1/4W, 1/2W, 1W, 2W, 3W
Resistance Tolerance	±2%, ±5%
T.C.R.	see Table 1

DERATING CURVE

For resistors operated in ambient temperatures above 70°C, power rating must be derated in accordance with the curve below.



INTRODUCTION

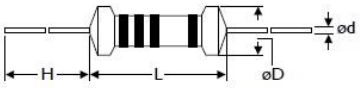
The CFR Series Carbon Film Resistors are manufactured by coating a homogeneous film of pure carbon on high grade ceramic rods. After a helical groove has been cut in the resistive layer, tinned connecting leads of electrolytic copper are welded to the end-caps. The resistors are coated with layers of tan color lacquer.

TABLE 1 TEMPERATURE COEFFICIENT

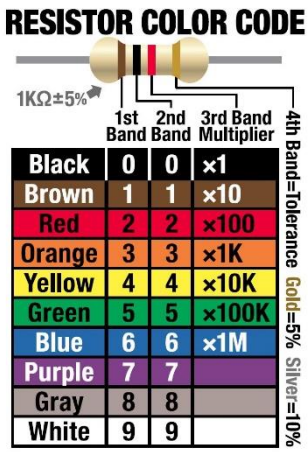
STYLE	TEMP. COEFFICIENT (ppm/°C)		
	under 100KΩ	100K Ω - 1MΩ	1MΩ - 10MΩ
CFR100, CFR200, CFR2WS, CFR3WS	-350~350	-500~0	-1,500~0
CFR-12, CFR-25, CFR-50, CFR25S, CFR50S, CFR1WS	-350~500	-700~0	-1,500~0

DIMENSIONS

Unit: mm



STYLE		DIMENSION			
Normal	Miniature	L	øD	H	ød
CFR-12	CFR25S	3.4±0.3	1.9±0.2	28±2.0	0.45±0.05
CFR-25	CFR50S	6.3±0.5	2.4±0.2	28±2.0	0.55±0.05
CFR-50	CFR1WS	9.0±0.5	3.3±0.3	26±2.0	0.55±0.05
CFR100	CFR2WS	11.5±1.0	4.5±0.5	35±2.0	0.8±0.05
CFR200	CFR3WS	15.5±1.0	5.0±0.5	33±2.0	0.8±0.05

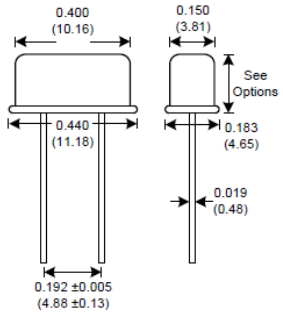


=> https://github.com/adafruit/Reference-Cards/blob/master/Card%202%20Front.pdf

Datasheets : **suite D.5**

.Crystal Quartz (20 MHz et un pas de 5 mm)
=> Datasheet : http://www.crystek.com/crystal/spec-sheets/crystal/CYSxx.pdf
=> Features :

CYSxx Model
Low Profile HC49S Leaded Crystal



Dimensions inches (mm)
All dimensions are maximum unless otherwise specified

Resistance at series resonance	
Freq. (MHz)	Max ESR
3.579545 - 4.0	150
4.1 - 4.9	120
5.0 - 5.9	100
6.0 - 11.9	90
12.0 - 14.9	70
15.0 - 30.0	50
30.1 - 86.0	100

Table 1

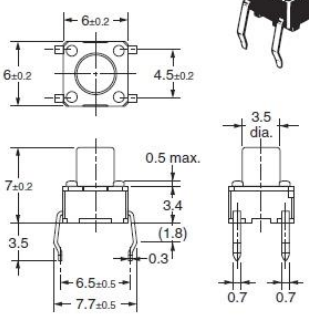
Part number	Freq. (MHz)	CL	Max ESR
CY3DMS	3.579545	18pF	150
CY3JS	3.686400	series	150
CY3JMS	3.686400	18pF	150
CY3JNS	3.686400	20pF	150
CY3AS	4.000	series	150
CY3APS	4.000	20pF	150
CY4ES	4.194304	12pF	120
CY4DS	4.915200	series	120
CY7AS	5.000	series	100
CY7APS	5.000	20pF	100
CY5BS	5.0688	series	100
CY6BS	6.000	series	90
CY6BPS	6.000	20pF	90
CY8GS	8.000	series	90
CY8GPS	8.000	20pF	90
CY9BS	9.830400	series	90
CY12AS	10.000	series	90
CY11BS	11.059200	series	90
CY11BPS	11.059200	20pF	90
CY12BS	12.000	series	70
CY12BPS	12.000	20pF	70
CY14AS	14.318180	series	70
CY14ACS	14.318180	18pF	70
CY14APS	14.318180	20pF	70
CY14BS	14.745600	series	70
CY14BPS	14.745600	20pF	70
CY15AS	15.000	series	50
CY16BS	16.000	series	50
CY16BPS	16.000	20pF	50
CY19AS	18.000	series	50
CY19BS	18.432	series	50
CY19BPS	18.432	20pF	50
CY20AS	19.660800	series	50
CY20APS	19.660800	20pF	50
CY22AS	20.000	series	50
CY22APS	20.000	20pF	50
CY24AS	24.000	series	50
CY24APS	24.000	20pF	50
CY25AS	25.000	series	50

Frequency Range:	3.579545 MHz to 40 MHz (fund) 27 MHz to 86 MHz (3rd O/T)
Calibration Tolerance:	±50ppm (Standard p/n)
(Option)	±10ppm to ±100ppm
Frequency Stability:	±100ppm (Standard p/n)
(Option)	±15ppm to ±100ppm
Operating Temp. range:	0 to 70°C (Standard p/n)
(Option)	-20 to 70°C
(Option)	-40°C to 85°C

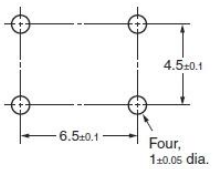
Storage Temp. range:	-45°C to 90°C
Shunt Capacitance:	7.0pF Max
Drive level:	100uW Typical
ESR:	See table 1
Aging:	<3ppm 1st year Max
Insulation Resistance:	500 Megaohms Min at 100Vdc

. Bouton reset (avec un pas de 4,5 mm sur 6,5 mm)
=> Datasheet : http://omronfs.omron.com/en_US/ecb/products/pdf/en-b3f.pdf
=> Features :

Standard, Flat Plunger Type (without Ground Terminal)
B3F-1060, B3F-1062, B3F-1062-G



PCB Processing Dimensions (Reference Only) (Top View) (PCB thickness, t=1.6)



Terminal Arrangement/ Internal Connections (Top View)

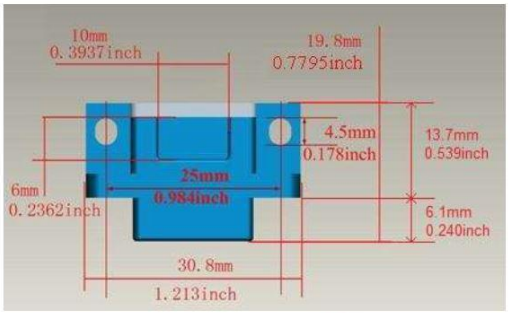
Datasheets : suite D.6

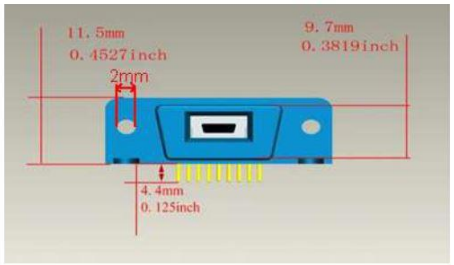
.2 x Condensateurs de 15 pF (du côté du Crystal et avec un pas de 2,54 mm)
=> Datasheet : http://www.vishay.com/docs/45171/kseries.pdf

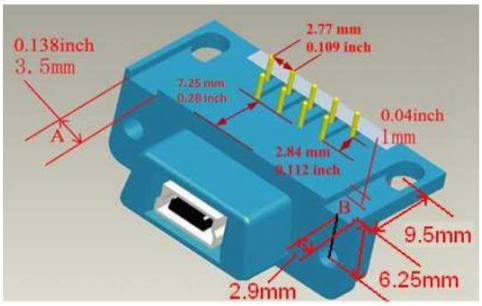
.1 x Condensateur de 684 uF (du côté du bouton reset et avec un pas de 2,54 mm)
=> Datasheet : https://www.vishay.com/docs/28105/mkt467-mkt468.pdf

.1 x Connecteur FTDI (Female)
=> Datasheet : http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_DB9-USB.pdf
=> Features :
Adds one USB serial port by connecting to the RS232 DB9 footprint of a device
Easy placement for an standard Male and Female RS232 DB9 footprint of a device
A selection of RS232, 3.3V or 5V UART signal levels.
Works with USB 1.1 & 2.0 Host and Hub ports
Industry Standard FTDI chip set & device drivers for maximum compatibility
Microsoft Windows® WHQL-certified, Mac OS X, Linux and Windows CE device drivers
Installs as a standard Windows COM port
128 byte transmit buffer, 256 byte receive buffer
UART data signals: TxD, RxD, RTS, CTS, DSR, DTR, DCD, RI, GND
Powered by USB port. No external power adapter required.
Serial port speed up to 1Mbps (RS232 levels) or 3Mbps (3.3V/5V levels)
Serial Communication Parameters
- Parity: None, Even, Odd
- Data bits: 7, 8
- Flow control: RTS/CTS , DSR/DTR, X-ON/X-OFF, None

Operating temperature of -40°C to +85°C







Equipement de laboratoire

Fer à souder :

=> Manuel : www.stahltools.com/docs/manuals/374-100-stahl-tools-stssvt-manual-8654.pdf

=> Features :

Working temperatures :

Melting point 419°F (215°C) | Normal operation 419°F-572°F (215°C-300°C) |

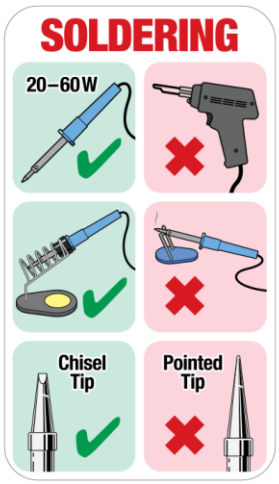
Production line operation 608°F-716°F (320°C-380°C) | Desoldering for small joint 599°F (315°C) |

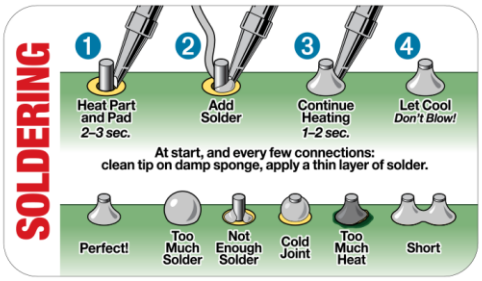
Desoldering for large joint 752°F (400°C)

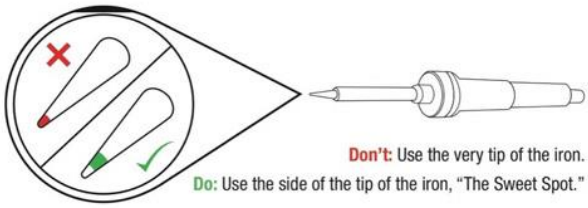
Deux astuces pour bien souder :

=> Tips 1 : https://cdn-blog.adafruit.com/uploads/2016/12/solder-card.png

=> Tips 2 : https://learn.sparkfun.com/tutorials/how-to-solder-through-hole-soldering







Do: Touch the iron to the component leg and metal ring at the same time.

Do: While continuing to hold the iron in contact with the leg and metal ring, feed solder into the joint.

Don't: Glob the solder straight onto the iron and try to apply the solder with the iron.

Do: Use a sponge to clean your iron whenever black oxidization builds up on the tip.