
# TP DOMOTIQUE

## Durée : 8H

### Salles G45-G46 Bât 3A (voir plan fac page 2)

**Responsables TPs :**

**Hélène LEYMARIE** **helene.leymarie@univ-tlse3.fr**

**Thierry PERISSE** **thierry.perisse@univ-tlse3.fr**

**Technicien :** **Franck Lacourrège**

## TP1 a : Mise en œuvre d'un capteur de température I2C.(4h)

## TP1 b : Envoie d'informations à distance à l'aide d'émetteur /récepteur XBEE (4h)

***Documentations dans le répertoire TP-domotique-docs:***

***Norme I2C / Capteur de Température I2C DS1621***

***Norme Zigbee / Modules Xbee***

***Carte développement PIC 16F877***

**Salles de TP**

**G45/G46**


***Plan : Salles G45/G46***

***Pour ne pas vous tromper de salle***

***et donc arriver à l'heure !!!***

# *TP1 a Mise en œuvre d'un capteur de température I2C*

1. **: Projet n°1 : ledmaquette**

2. ***: Projet n°2 : lcdmaquette***

3. ***: Projet n°3 : lcd_ds1631maquette***

4. ***: Projet n°4 : lcd_ds1631maquette en autonomie***

# *TP1 b Envoie d'informations à distance à l'aide d'émetteur /récepteur XBEE (3h)*

5. ***: Projet n°5 : Faire converser 2 modules Xbee.***

6. ***: Projet n°6 : bonjourxbee1***

7. ***: Projet n°7 : Projet complet cad : envoi à distance de l'info température***

   ***(partie émission : capteur I2C DS1621, PIC16F877, Xbee_émetteur, autonomie (accu+régulateur)***

   ***(partie réception : Xbee_récepteur, interface Xbee/miniUSB, Terminal de X-CTU)***

8. ***: Projet n°8 : Consommation // Portée // Interférences***

9. ***: Projet n°9: mise en veille de l'émetteur Xbee***

# ***PREPARATION et MANIPULATION***

## A– PREPARATION (la préparation doit être jointe au compte rendu en fin de séance)

### Préparation projet n°1 :

Récupérer et analyser le programme permettant de faire clignoter la LED.

Repérer sur la maquette les différents ports d'E/S (à l'aide du schéma électrique).

### Préparation projet n°2 :

Récupérer et analyser le programme permettant d'afficher « test» sur le LCD.

### Préparation projet n°3 :

Etude de la norme I2C.

Repérer sur le microcontrôleur les broches SDA et SCL.

A l'aide du document constructeur donner le schéma de câblage du DS1621

### Préparation projet n°4 :

Donner le schéma de câblage si on alimentela maquette par une pile de 9V et un régulateur 7805.

### Préparation Projet n°6 :

Donner les caractères présents sur les différentes trames

### Préparation Projet n°7 : Projet complet d'envoi à distance de la température.

A l'aide des différents programmes validés : Réaliser la programmation complète du projet 7.

### Préparation Projet n°8 : Consommation // Portée // Interférences

Réfléchir aux différentes mesures demandées (on précisera le câblage) et à une diminution possible de la consommation

### *Projet n°9: mise en veille de l'émetteur Xbee*

Repérer l'entrée permettant la mise en veille du Xbee et modifier le programme précédent.

## B– MANIPULATION

Une validation de chaque partie expérimentale doit être faite avec un responsable. Les programmes doivent être commentés.

Les câblages doivent être soignés.

Attention à respecter les couleurs

**+Vcc —> Rouge**

**-Vcc —> Bleu**

**Masse —> Noir**

**Un compte rendu (.pdf) doit être rendu en fin de séance.**

## TP 1-a

# Afficher sur un LCD la valeur de la température donnée par un capteur I2C (DS1621)

Objectifs : Acquérir les bases pour utiliser un logiciel (mikroC PRO for PIC) permettant de programmer (programmateur mikroProg) un microcontrôleur (PIC16F877ou pic16F877A) avec un langage de haut niveau (langage C).

Prise en main du système (logiciel, matériel, …).

Lancer le Logiciel mikroC PRO for PIC présent sur le bureau du PC :

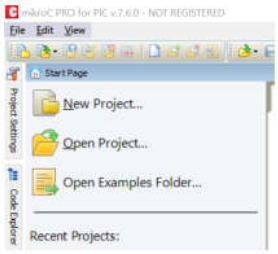
## Projet n°1 : led

Création du premier projet : led.mcppi

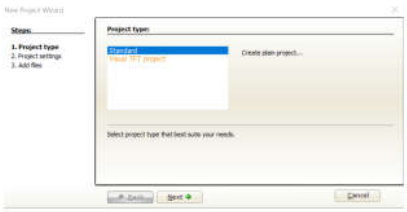
Pour chacun des programmes créer un répertoire du même nom **led** que l'on rangera dans le répertoire

**Mes documents/TPDomotique202X/Nom1-Nom2/led/*****

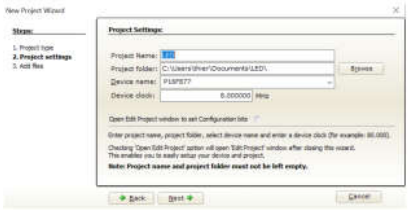
But: Faire clignoter une LED toutes les secondes (sortie7 du PORTD). (**Lire le nom du pic et la fréquence du quartz sur votre maquette)**



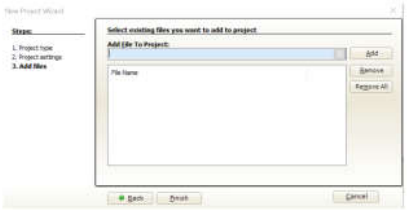

Créer un nouveau projet



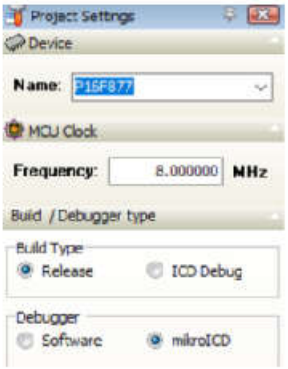
Choisir Projet standard



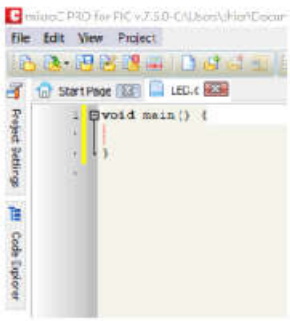
Choisir le nom du projet LED, le répertoire de sauvegarde .../LED/, le microcontrôleur 16F877, l'horloge 8Mhz



Pas de fichier à ajouter au projet

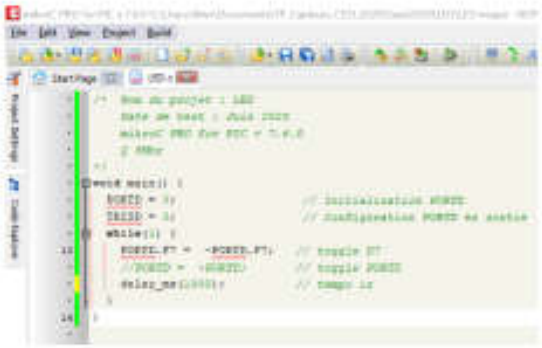



Choisir les paramètres de Project Settings



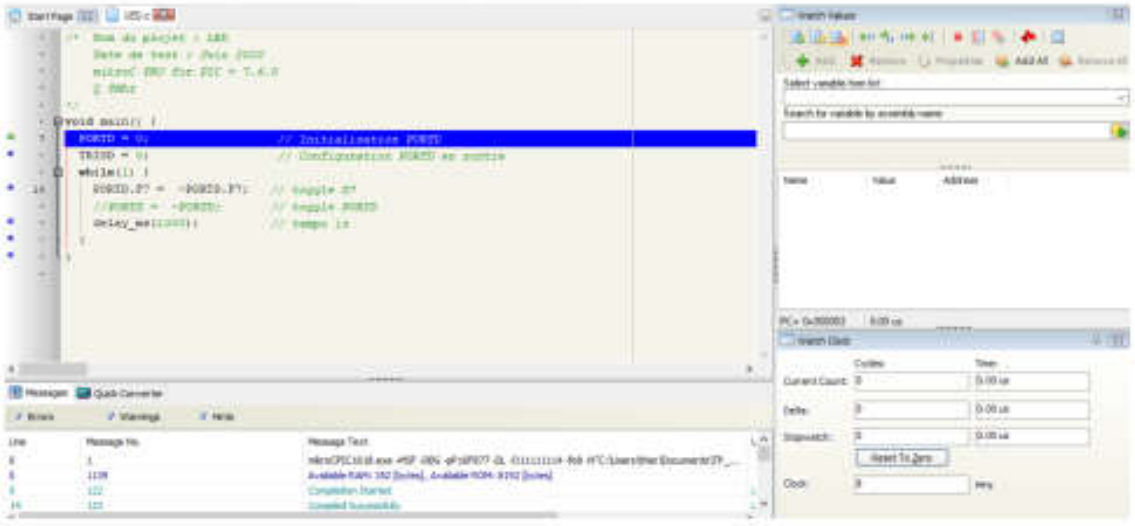

Projet vide

```
*/
void main() {
  PORTD = 0;                  // Initialisation PORTD
  TRISD = 0;                  // Configuration PORTD en sortie
  while(1) {
    PORTD.F7 =  ~PORTD.F7;   // toggle D7
    //PORTD =  ~PORTD;        // toggle PORTD
    delay_ms(1000);           // tempo 1s
  }
}
```

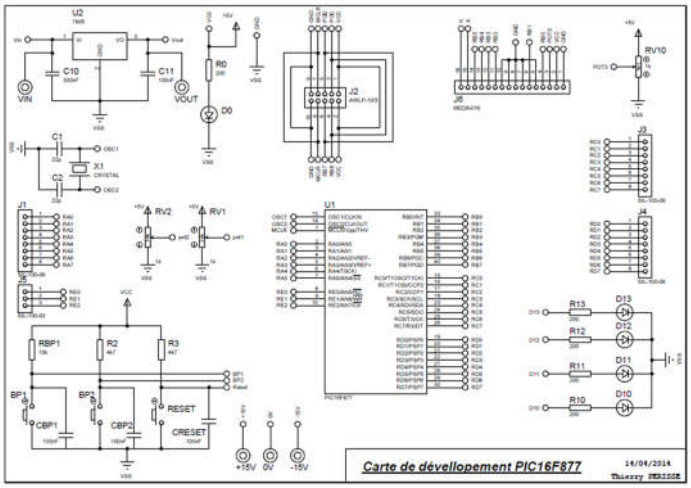


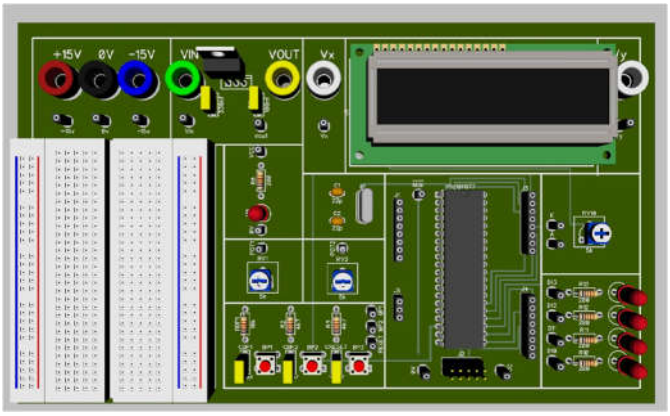
Copier / Coller du programme ci-dessus LED  Compiler le projet "Build", vérifier dans la fenêtre "Messages" il y a des erreurs.

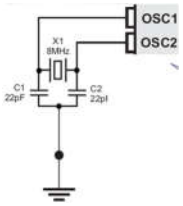
Regarder maintenant dans le répertoire LED les fichiers qui ont étés crées (en particulier "LED.hex" qui va nous servir pour programmer le microcontrôleur dans PROTEUS)

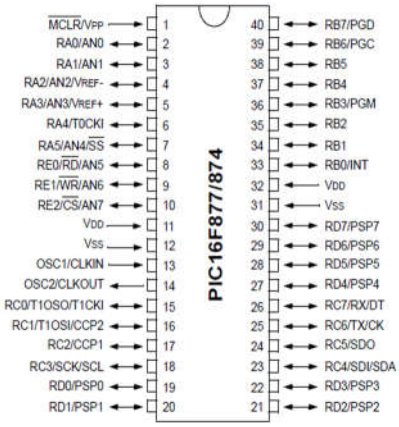


Réaliser le câblage sur la carte de développement PIC16F877 en se servant du schéma électrique en annexe (fin du TP).



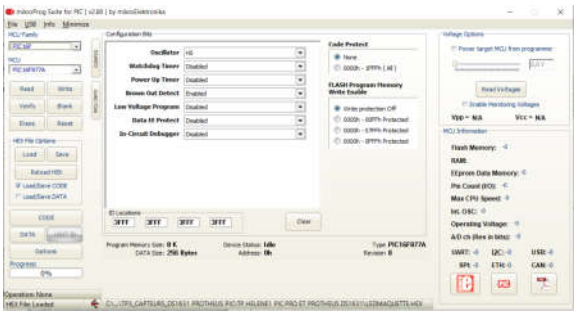



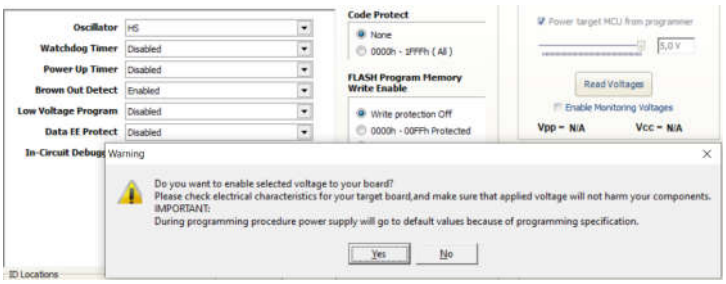




Branchons maintenant le programmateur mikroProg sur le connecteur (fil bleu à droite)

afin de transférer le programme dans le microcontrôleur PIC16F877.

Transférer le programme en cliquant sur Program. Le logiciel mikroProg suite for Pic est lancé et télécharge le fichier led.hex à transférer.






Write

Mettre le curseur à 5V

Alimenter en cochant le power target Mcu programmer et répondre yes

La maquette est alimentée (voyant allumé).

le programme fonctionne et on le visualise avec la led qui clignote sur le port D.F7 que l'on aura câblé.

Si le programme n'est pas lancé, il faut une impulsion sur la broche MCU_MCLR à 5V brièvement.

(Sur certains programmateurs, il faut laisser la broche MCU_MCLR à 5V de façon permanente. On retira le 5V du MCU_MCLR quand il faudra de nouveau programmer le pic)

## Projet n°2 : lcdmaquette

Création d'un projet : lcdmaquette que l'on enregistrera dans le répertoire **Mes documents/TPDomotique202X/Nom1-Nom2/lcdmaquette/*****

Aller chercher le programme permettant d'afficher « test: pause café» et de faire clignoter une LED sur le portD.

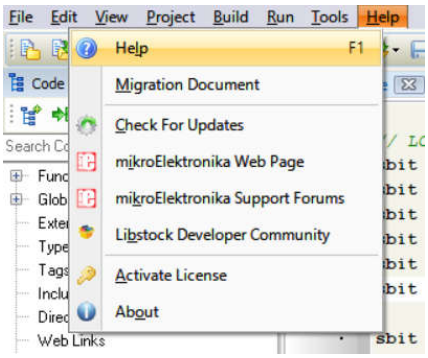
Pour utiliser l'afficheur, on souhaite utiliser le PortB avec les affectations suivantes : 0,1,5,4,3,2 pour : RS, EN, D7, D6, D5, D4.

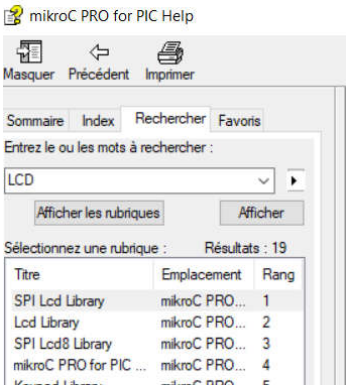
En utilisant les routines du LCD ( voir ci-dessous) :
Justifier les lignes de codes du programme ci-contre.

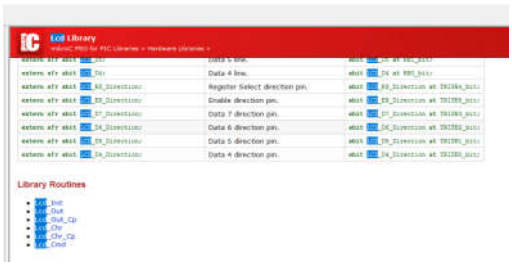
```
// LCD module connections
sbit LCD_RS at RB0_bit;
sbit LCD_EN at RB1_bit;
sbit LCD_D4 at RB2_bit;
sbit LCD_D5 at RB3_bit;
sbit LCD_D6 at RB4_bit;
sbit LCD_D7 at RB5_bit;

sbit LCD_RS_Direction at TRISB0_bit;
sbit LCD_EN_Direction at TRISB1_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;
// End LCD module connections

char *text = "TEST";
char *text2 = "PAUSE CAFE";

void main() {

  Lcd_Init();
  Lcd_Cmd(_Lcd_CLEAR);               // Clear display
  Lcd_Cmd(_Lcd_CURSOR_OFF);          // Turn cursor off

  PORTD = 0;                        // Initialize PORTC
  TRISD = 0b01000000;               // Configure D6 en entrée et les autres en sorties
```

Pour avoir les détails sur les routines pilotant le LCD cliquer sur le Help

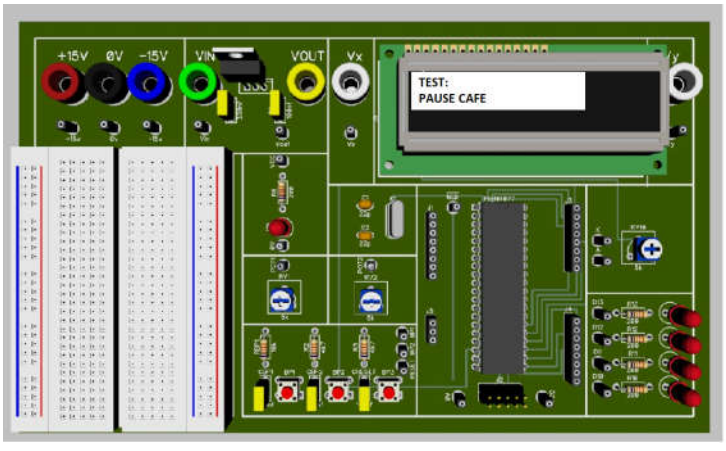
Rechercher / Taper LCD / Afficher les rubriques / LCD Library



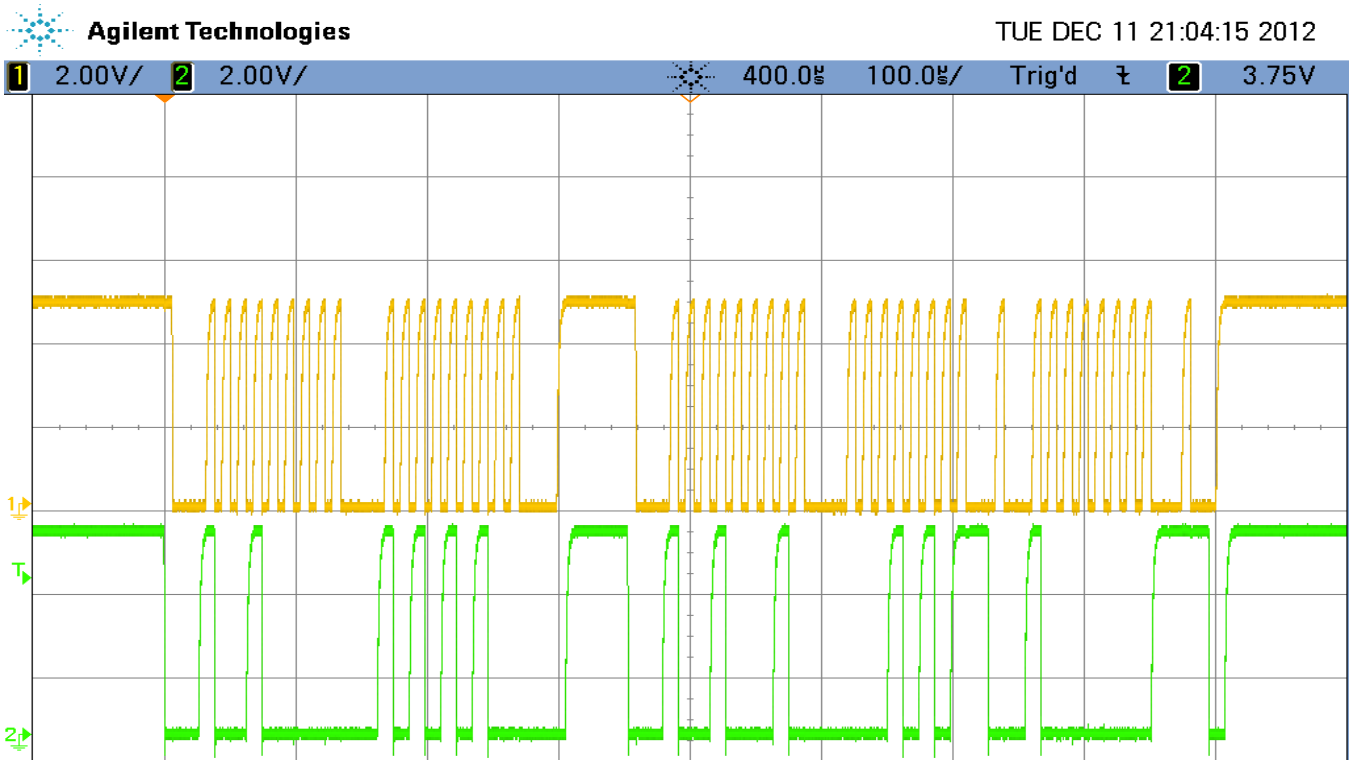

Aucun câblage supplémentaire n'est nécessaire pour tester le fonctionnement du LCD.




Compiler le programme, transférer le programme dans le microcontrôleur et vérifier le bon fonctionnement du projet **lcdmaquette**. Faites valider

## Préparation projet n°3 : lcd_ds1621maquette

Ci-dessous un enregistrement des lignes SDA et SCL dans un exemple de communication entre le PIC16F877 et le DS1621 :




```
        PORTD = 0;                          // Initialize PORTC
        TRISD = 0x01000000;                 // Configure D6 en entrée et les autres en sorties*/

80  while(1) {
        I2C1_Start();         //Détermine si l'I2C est libre et lance le signal
        I2C1_Wr(0x90);        //Mode de contrôle en mode écriture
        I2C1_Wr(0xAA);        //Lecture de la température
        I2C1_Stop();          //Arrêt du sigal
        Delay_ms(10);
        I2C1_Start();         //Détermine si l'I2C est libre et lance le signal
        I2C1_Wr(0x91);        //Mode de contrôle en mode lecture
        MSB = I2C1_Rd(1);     //Nombre signé [température entre +125° et -55°C]
        LSB = I2C1_Rd(0);     //Si bit 7 = 1 température MSB +0,5°C
90      //MSB = 0x25;
        //LSB = 0x80;
        I2C1_Stop();          //Arrêt du sigal
        Delay_ms(10);
        // Écrire sur l'écran LCD.
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1, 5, "DS1631");
        Lcd_Out(2, 1, "Temp: ");

        IntToStr(MSB, msg);                 // Convertir la partie entière de la température en chaîne.
100     Lcd_Out_CP(msg + 3);                // ignorer les 3 blancs au début de la chaîne
        Lcd_Chr_CP('.');                    // Point décimal.
```

***A l'aide de la boucle infinie programmée ci-dessus et des signaux SDA et SCL ci-dessus :***

***Repérer les différents octets présent sur la ligne SDA et en donner leurs significations ?***

***- Sur cet exemple quelle est la température présente sur le capteur DS1621 ?***

## Projet n°3 : lcd_ds1621maquette

Création d'un projet : lcd_ds1621maquette ranger dans le répertoire **Mes documents/TPDomotique202X/Nom1-Nom2/lcd_ds1621maquette/*****

Câblage supplémentaire : DS1621

Alimentation : 5V pour VDD et 0v pour GND

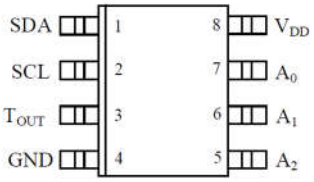
Adresse physique à câbler d'après la programmation: 3 broches $A_0A_1A_2$

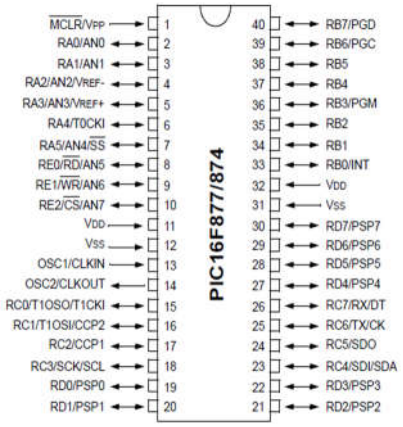
SDA et SCL à relier au PIC 16F877

**DS1621**

**Digital Thermometer and Thermostat**

**PIN ASSIGNMENT**



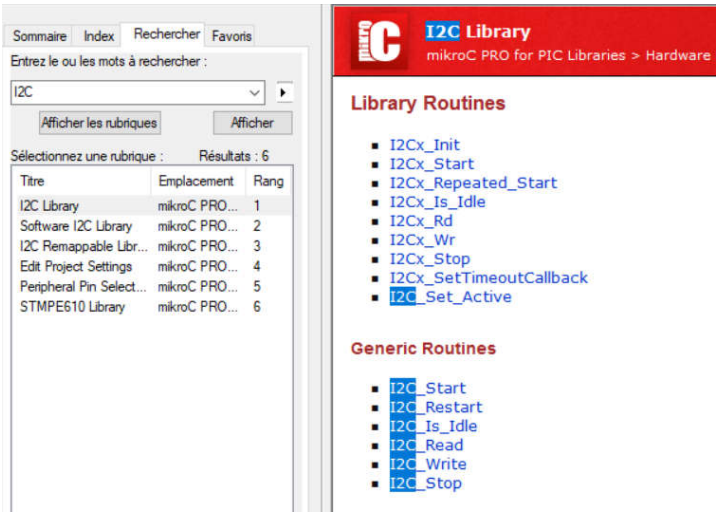




Recherche d'aide sur la programmation à partir :

-de la bibliothèque mikroC Pro

- de la notice technique DS1621:

- du programme donné ci dessus :




Repérer et commenter les lignes de commande concernant l'I2C.

Vérifier le bon fonctionnement du projet et faites valider.

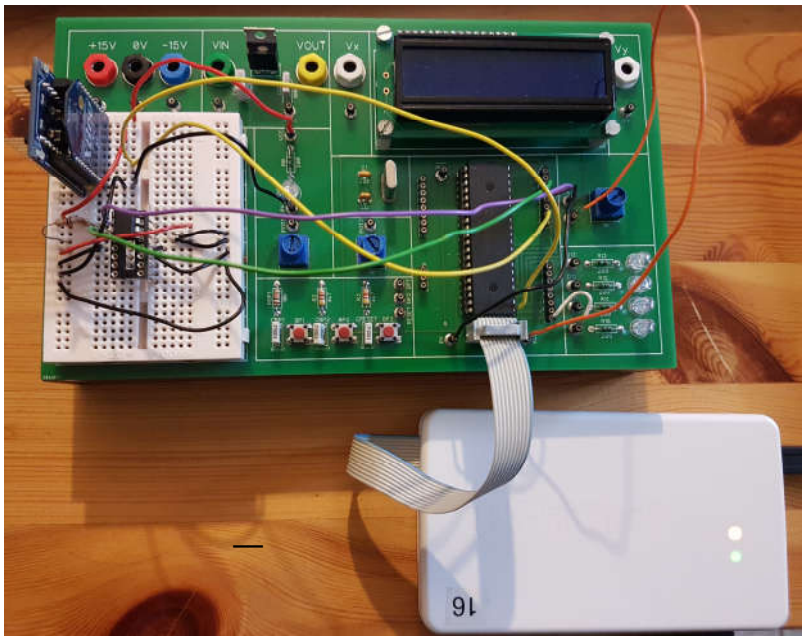
## Projet n°4 : lcd_ds1621maquette en autonomie

On travaille exactement sur le même projet que précédemment.

But : Rendre le système autonome

Jusqu'à présent la maquette est alimentée via le programmateur. Le but est d'enlever cette connexion .

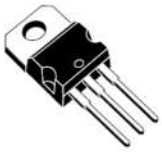
on alimente le montage via une pile de 9V et un régulateur de tension 7805 de telle sorte qu'on possède en sortie du régulateur une tension de 5V.

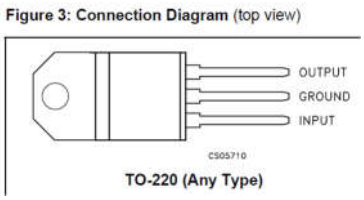


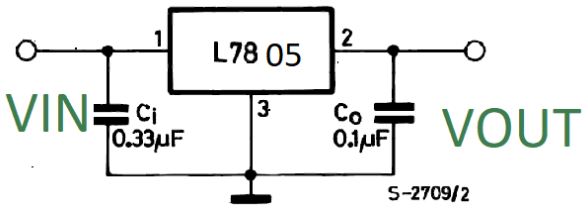

enlever le connecteur .

relier MCUMCLR à +VCC ( pin MCR barré sur la maquette)

repérer sur la maquette le régulateur 7805 et étudier sa documentation constructeur. Câbler.




Connecter la pile 9V .

Vérifier le bon fonctionnement et faites valider.

Sortir dehors avec la maquette et relever une dizaine de mesures de température jusqu'au régime permanent. En déduire le temps de réponse du capteur .

## Projet n°5 : Xbee1 : Faire converser 2 modules Xbee

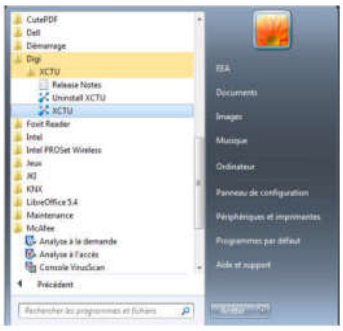


Matériel : deux émetteurs /récepteurs XBEE XBP24-API-001 , deux interfaces USB/XBEE et deux cordons USB/miniUSB . Attention au sens de connexion entre le XBEE et l'interface!

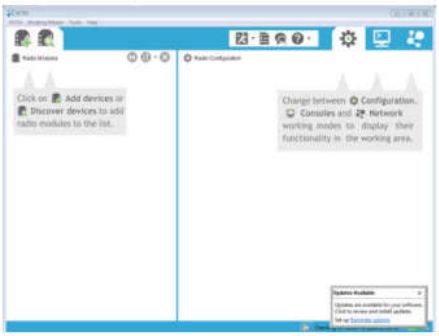
Le logiciel XCTU permet de configurer le réseau entre les différents composants XBEE.


Voici le tutoriel pour apparier deux composants point à point.

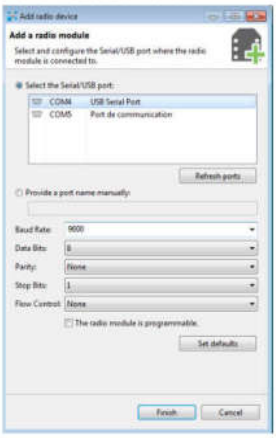
1) Ouvrir le logiciel XCTU



2) Attendre la fin de la mise à jour du logiciel



3) Connecter le premier module XBEE sur le port USB du PC et attendre si besoin que le pilote du périphérique soit installé. Cette fenêtre s'affiche.

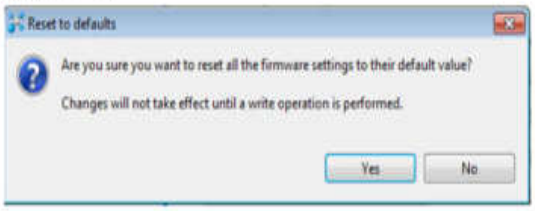


4) Refresh port puis choisir le « port USB Serial Port » affiché. Laisser les paramètres par défaut. Puis Finish

5) Le module Xbee est détecté et s'affiche dans la colonne « Radio Modules ». Cliquer  .

6) Ouvrir le panneau « configuration » en cliquant sur  . Remettre les paramètres d'usine par défaut en cliquant sur 

.



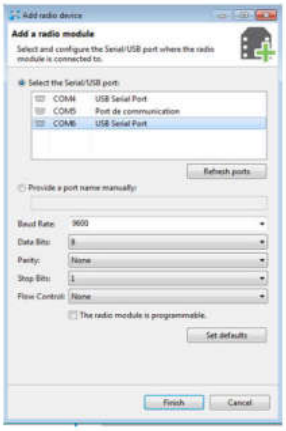

Yes

7) Transférer les paramètres sur le module Xbee en cliquant sur Write.

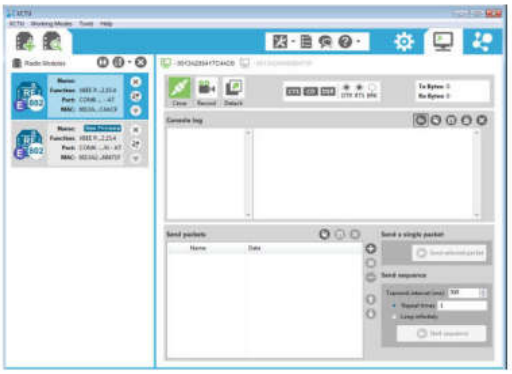
8) Choisir un canal de communication par exemple 3336 sur la ligne « ID Pan ID » puis Write de nouveau

9) Ouvrir le panneau « fenêtre d'affichage » représenté par  et cliquer sur Open.
Le premier Xbee est prêt à être connecté.

10) Reprendre les étapes 3 à 9 pour le **deuxième** Xbee. Il faut choisir le **nouveau** port USB affiché.

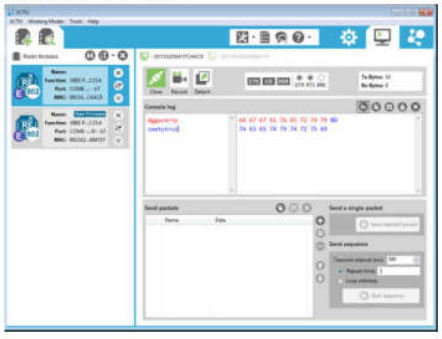



Finalement les deux Xbee sont connectés sur le même canal C3336 et voici la fenêtre finale.



11) On teste la communication :
Sélectionner le Xbee sur le port Com 4, écrire un message dans le tableau « Console log » .
Sélectionner le Xbee du port Com 6 et vérifier que le message précedent est bien arrivé dans le tableau « Console log ».

12) Tester la communication dans l'autre sens.

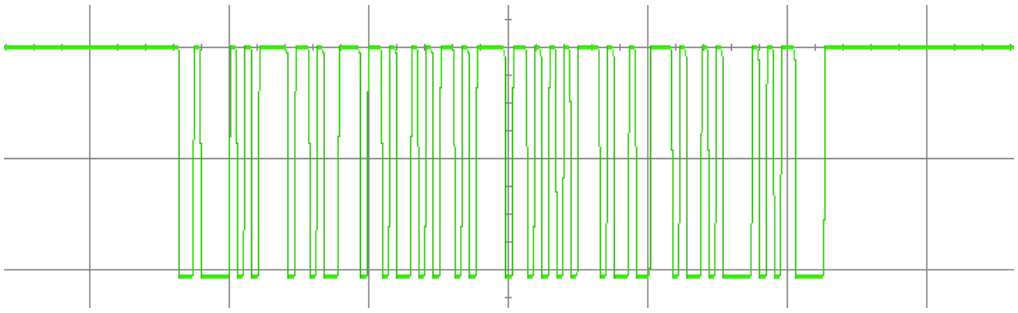


Ça marche ! Déconnecter **un des deux câbles USB** sans fermer le logiciel puis continuer le TP.

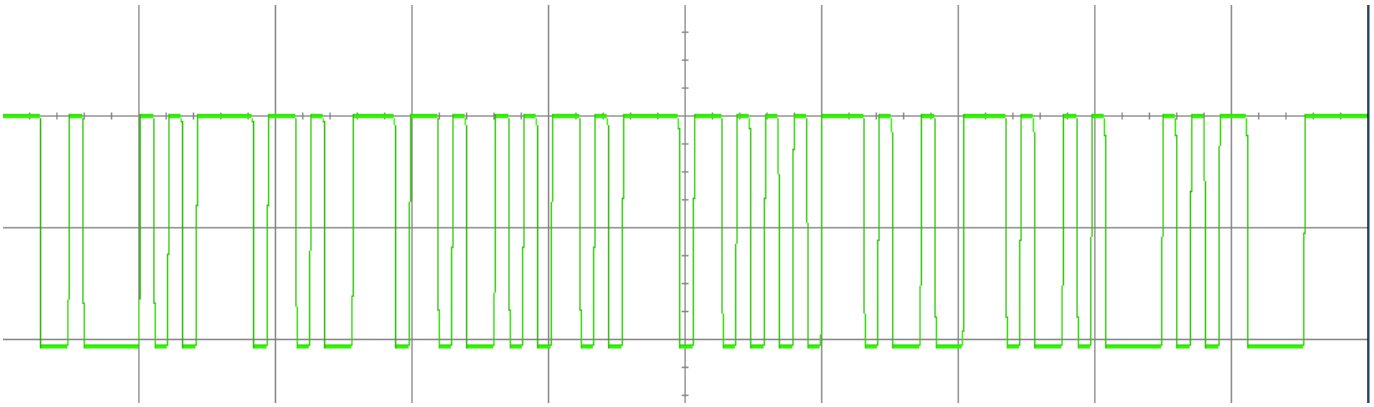
## Préparation Projet n°6 : bonjourxbee1

Envoi de « bonjour » sur la broche Tx du microcontrôleur 16F877 :

**2ms/div**



**1ms/div**

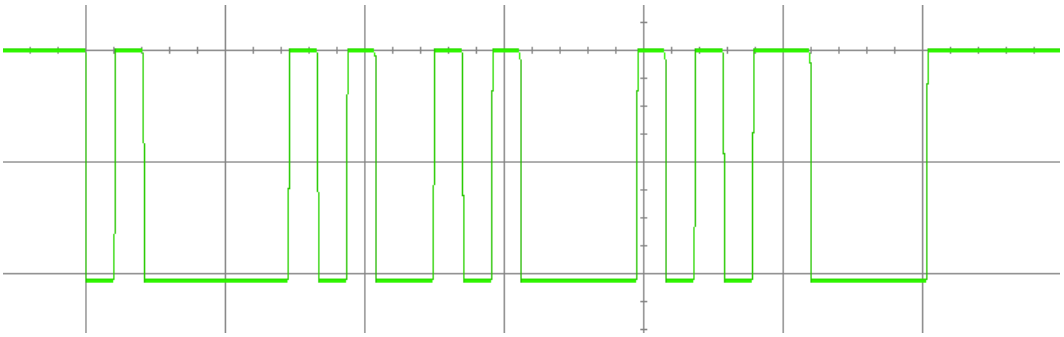


A l'aide de la norme ASCI montrer les différents mots présents sur cette trame ?

Faire la même chose sur les 2 trames suivantes ? Quels sont les caractères transférés sur ces trames ?

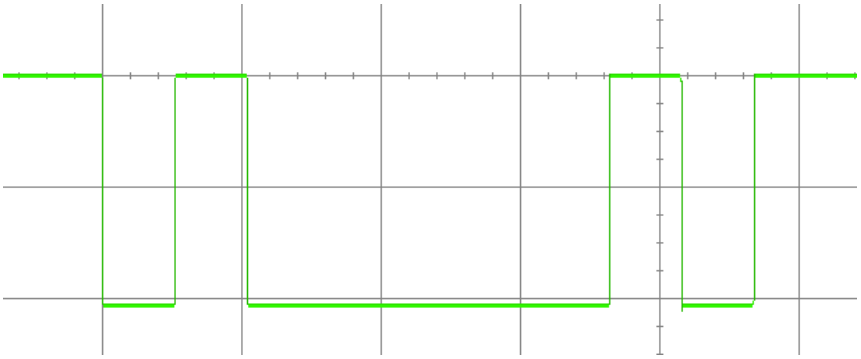
**500us/div.**

**Trame 1**



**200us/div.**

**Trame2**



# Projet n°6 : bonjourxbee1

Création d'un projet : bonjourxbee1 ranger dans le répertoire Mes documents/TPDomotique202x/bonjourxbee1/

Envoyer « Bonjour » avec un émetteur Xbee1 relié au PIC16F877 (Din du Xbee1 // TX du PIC16F877) et récupérer l'info avec un Xbee2 monté sur un adaptateur Xbee/miniUSB via Terminal de X-CTU.

***Hard :***

Câbler l'alimentation du Xbee1 Vin=5V. et la masse avec celle du reste du montage.

Tx du PIC16F877 relié au Din du Xbee1.

***Soft :***

Faire un copier / coller du programme bonjourxbee1.c

Start Page | bonjourxbee1.c

```
// testXbeebonjour ok le 09122012
char i;
int cpt=0;
char tableau[] = "Bonjour";

void saut_ligne()          {
        i = '\n'  ;
        Uart1_Write(i);
        i = '\r'  ;
        Uart1_Write(i);    }
void main()                {
  PORTD = 0;                 // Initialize PORTD
  TRISD = 0;                 // Configure PORTD as output
//----------Initialisation USART--------------
  Uart1_Init(9600);   //(8 bit, 2400 baud rate, no parity bit..)
//----------Programme principal-------------
  do {
      if (Uart1_Data_Ready()) {   // If data is received
        for (cpt=0; cpt<7; cpt++){
            i =tableau[cpt];
            Uart1_Write(i);
            }
        saut_ligne() ;
      }
      delay_ms(1000) ;
      PORTD =  ~PORTD ;
}while(1);                 }
```

Analyser le programme

Compiler

Transférer

Lancer le programme.

Bonjour s'affiche sur le terminal de l'XBEE2

Vérifier le bon fonctionnement du projet et faites valider.

## *Projet n°7 : Projet complet d'envoi à distance de l'info température.*

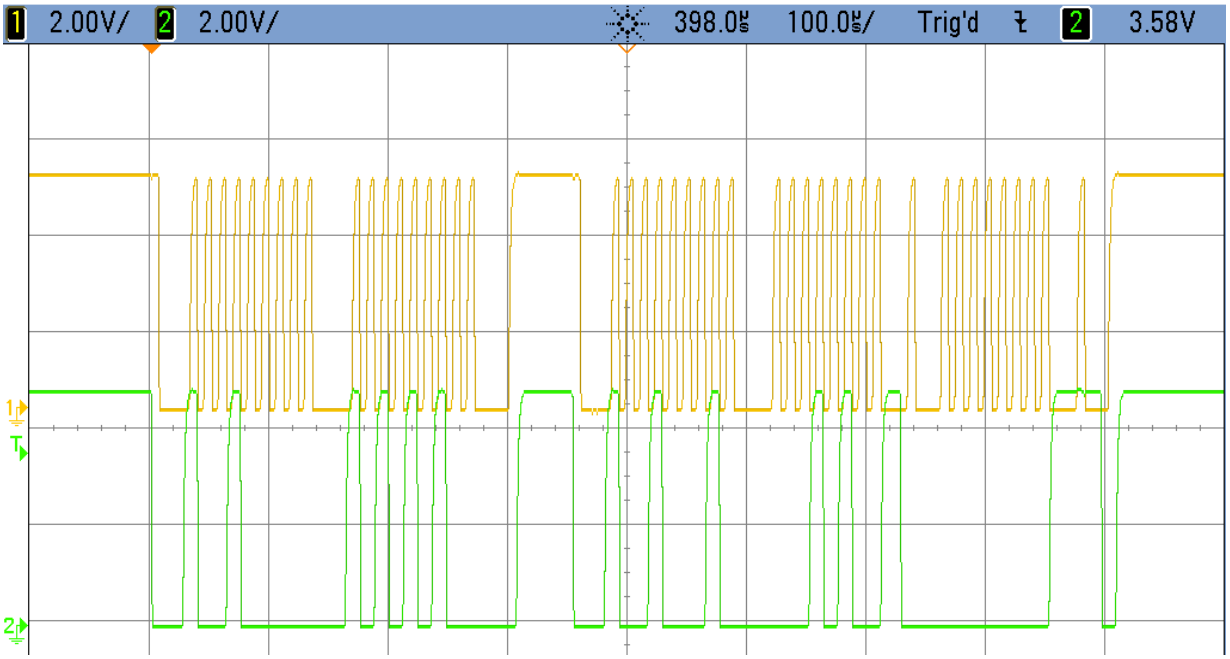
***(partie émission : capteur I2C DS1621, PIC16F877, Xbee_émetteur, (partie réception : Xbee_récepteur, interface Xbee/miniUSB, Terminal de X-CTU)***

## ***A l'aide des différents programmes validés; Réaliser le programme complet du projet 7 ?***

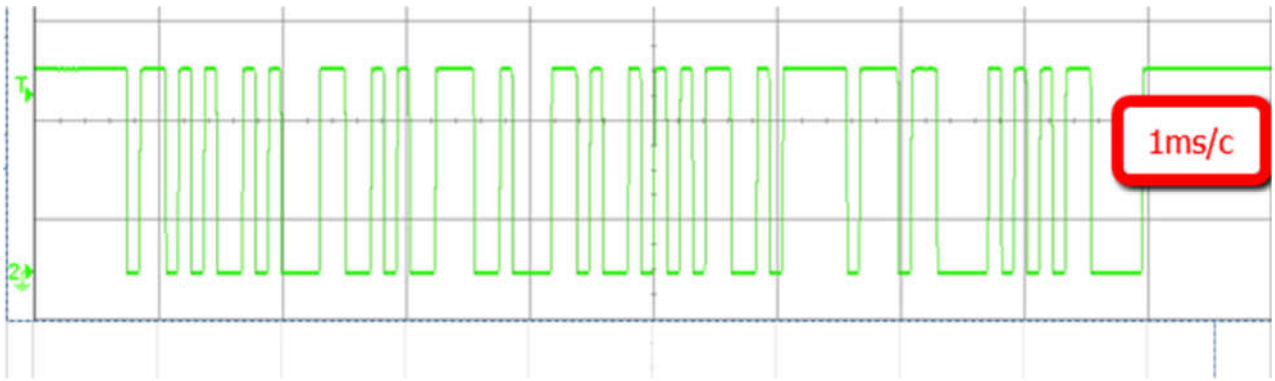
Récupération des voies SDA et SCL à l'oscilloscope.

Récupération du signal présent sur la broche Din du Xbee_émetteur (ou du Tx du 16F877).

Ci-dessous un exemple de signaux SDA et SCL.




et de un exemple signal Tx.




Vérifier le bon fonctionnement du projet complet.

Faites valider

***Projet n°8 : Consommation // Portée // Interférences***

# *A-Faire tourner le projet 7 en autonomie avec la pile 9V*

***Récupérer et faire tourner le programme : LED_LCD_DS1621_UART (site web ou répertoire)***

## *A-1 Etude de portée :*

***Mesure de la portée Emission // Réception*** *(Sans obstacle, avec obstacle)*

*Vérifier par rapport à la documentation technique du XBEE.*

## *A-2 Bilan énergétique de la partie émission :*

***Mesurer la consommation, calculer la durée de fonctionnement***

- *donner le schéma de câblage permettant de visualiser l'image du courant (Rshunt proche de 1Ω))*
- *A l'aide de $V_{Rshunt}$ mesurer à l'oscillo donner le courant consommé par le montage.*
- *A l'aide de l'ACCU 300mA/9V donner la durée de fonctionnement du projet.*

## *A-3 Interférences (portable, wifi, …)*

# *B-Faire tourner le projet 7 en autonomie et en veille*

## *B-1 Programmation du Xbee en veille :*

- *Pour que le Xbee1 puisse se mettre en veille il faut le programmer à partir du logiciel X-CTU avec Sleep mode (1) cad Hibernate. Il faut modifier le programme (veille /réveil) et câbler la pin 9 « pin sleep control line ».*

## *B-2 Bilan énergétique de la partie émission :*

***Mesurer la consommation, calculer la durée de fonctionnement***

- *Garder le schéma de câblage permettant de visualiser l'image du courant (Rshunt proche de 1Ω))*
- *A l'aide de $V_{Rshunt}$ mesurer à l'oscillo donner le courant consommé par le montage.*
- *A l'aide de l'ACCU 300mA/9V donner la durée de fonctionnement du projet.*

***Carte de développement PIC 16F877 :***

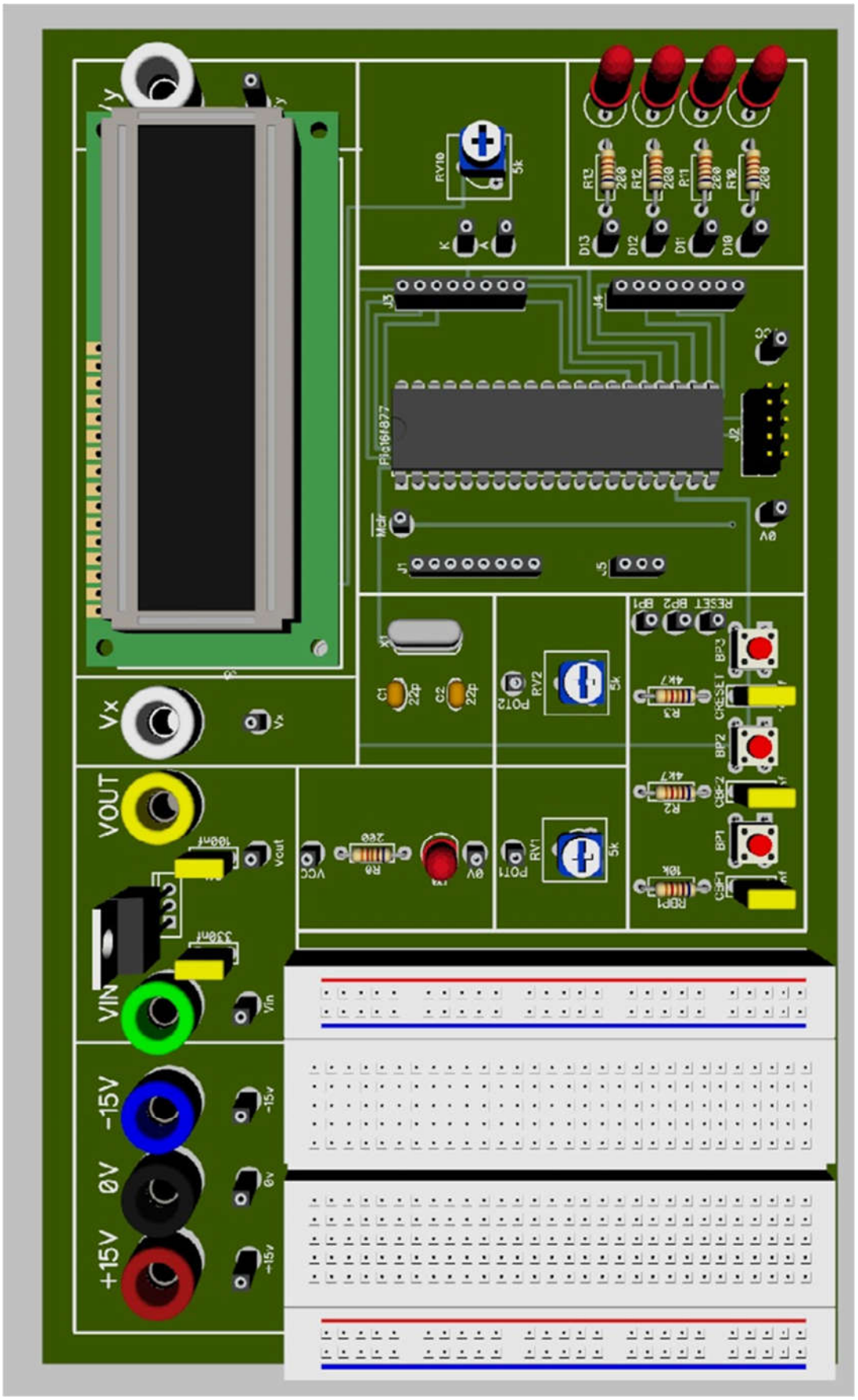


***Schéma de la carte de développement PIC 16F877 :***