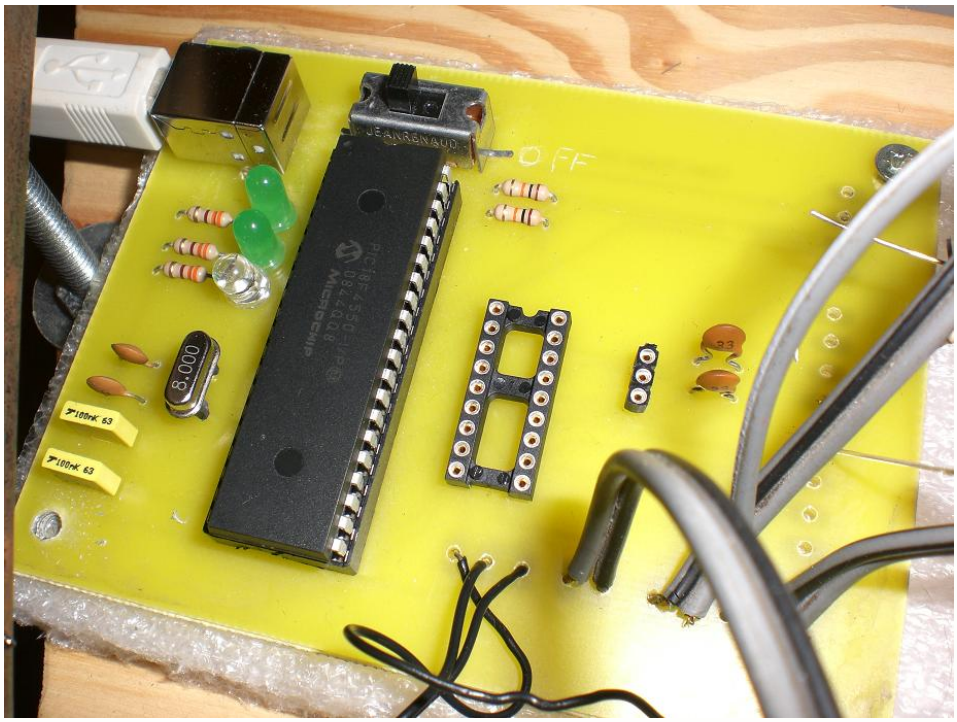
Projet de Physique P6-3
STPI/P6-3/2009 – 44

PROGRAMMATION D'UN MICROCONTRÔLEUR



Etudiants :
Pierre HOUSSIN
Peng GE
Aël GAIN

Enseignant-responsable du projet :
Faouzi DHAOUADI

Date de remise du rapport : 17***/06/09***

Référence du projet : ***STPI/P6-3/2009 – 44***

Intitulé du projet :

Mise en œuvre d'une application à base de microcontrôleur (programmation de circuit intégré); réalisation d'une carte électronique.

Type de projet : Expérimental

Objectifs du projet :

Réaliser une carte électronique à base de micro-contrôleur permettant d'interfacer des périphériques tels que des capteurs de distance, des diodes électroluminescentes ou une carte de contrôle de moteur. La carte devra aussi permettre une communication avec un ordinateur au moyen d'une liaison USB pour récupérer les mesures des capteurs de distance et le contrôle des diodes et de la carte moteur.

TABLE DES MATIERES

NOTATIONS, ACRONYMES

PIC : Programmable Integrated Circuit

USB : Universal Serial Bus

I2C : Inter Integrated Circuit Bus

ROM : Read Only Memory

RAM : Random Acces Memory

CPU : Central Processing Unit

UART : Universal Asynchronous Receiver Transmitter (équivalent du bus série)

HID : Human Interface Device

MIPS : million Instruction Per Second

LED : Light Emiting Diode

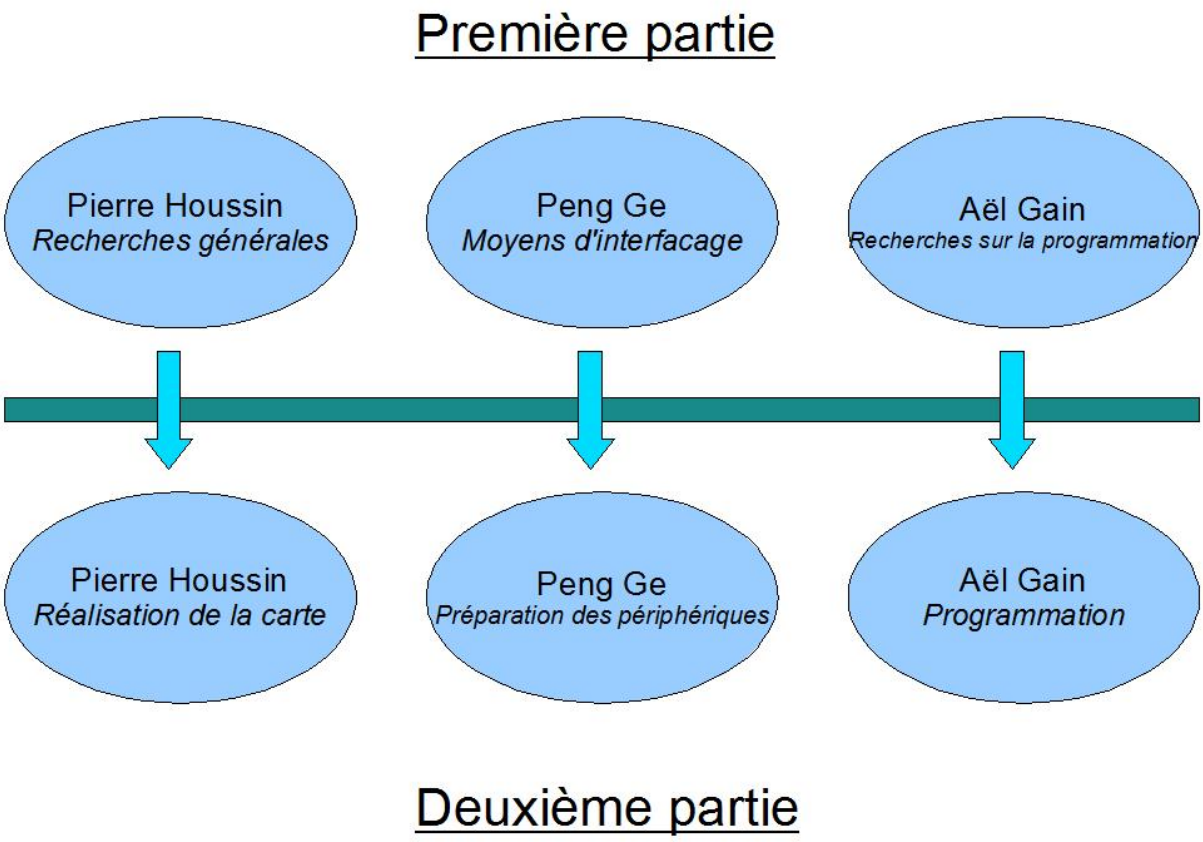
1) INTRODUCTION

Dans tout type d'application électronique, il est courant d'utiliser des capteurs ou des actionneurs pour interagir avec l'environnement. Seuls, ces matériels peuvent être difficiles à utiliser et nécessiter, pour chacun, d'autres composants dédiés à la présentation des mesures ou à la communication. L'objectif de ce projet est donc d'utiliser un micro-contrôleur pour centraliser les communications et le contrôle de différents matériels électroniques. Le micro-contrôleur est doté d'une faculté de traitement des informations qui lui parviennent et de nombreux moyens de communication ; nous souhaitons donc créer une carte électronique qui servira d'interface de contrôle et de communication entre plusieurs périphériques. Nous choisissons pour notre projet le micro-contrôleur PIC 18F4550 pour ses capacités de traitement et ses nombreuses liaisons, dont une liaison USB qui permettra la communication avec un ordinateur.

2) MÉTHODOLOGIE / ORGANISATION DU TRAVAIL

Le sujet nous demande d'abord un temps de recherche important afin de comprendre le fonctionnement des micro-contrôleurs et des périphériques qui lui seront interfacés. Le travail des membres sur le projet comprend deux parties. Une première dans laquelle Pierre se chargera des recherches générales sur les micro-contrôleurs, Peng des moyens d'interfacage des périphériques et Aël de la programmation des micro-contrôleurs. Dans la seconde partie, nous commencerons la réalisation de la carte électronique et la programmation, qui sont deux tâches très liées, nous partagerons donc chacun notre temps entre ces deux tâches même si majoritairement, Pierre réalisera la carte électronique, Peng la préparation des périphériques et Aël la programmation du micro-contrôleur.

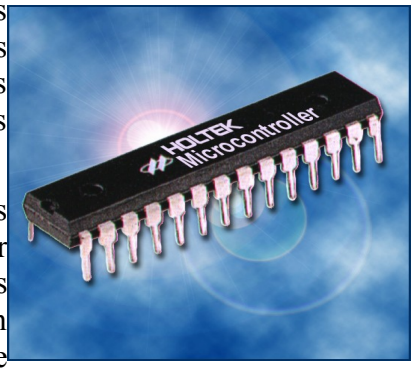
Ci-dessous l'organigramme des tâches :



3) TRAVAIL RÉALISÉ ET RÉSULTATS

A) Recherches

a) Préliminaires

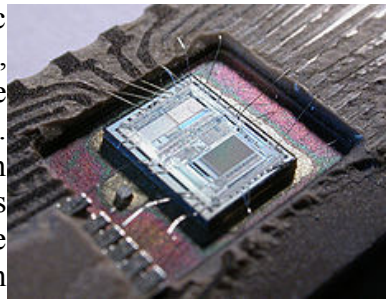


Nous avons travailler sur un micro-contrôleur 18F. Nous allons dans cette partie vous présenter le fonctionnement général des PICs. Pour être le plus clair et pragmatique possible, nous n'insisterons pas sur la partie interne et électronique des PICs mais plutôt sur ses concepts.

Les micro-contrôleurs sont utilisés dans tous les appareils électronique dès qu'ils nécessitent un peu de précision. Par exemple... systèmes embarqués, clavier, souris, dans tous les automates, dans le domaine de la numérisation... Leurs utilisation au cours des 30 dernières années est exponentielle. Il en existe de toute sorte, et même programmable en C. Par exemple , un foyer moyen est susceptible d'être équipé de deux douzaines de microcontrôleurs (appareils électroménagers) et une automobile de milieu de gamme est équipée d'au moins 70 microcontrôleurs.

L'avantage net qu'a ce composant électronique est la numérisation de tout le domaine de l'électricité, de plus il permet d'obtenir une précision, une rapidité, et une utilisation améliorée et facilitée par rapport à l'analogique.

Le micro-contrôleur, objet technique, intégrant de l'électronique, fait souvent apparaître des fonctions ayant pour rôle le traitement d'informations : grâce à des opérations arithmétiques (addition, multiplication...) ou logiques (ET, OU...) entre plusieurs signaux d'entrée il permet de générer des signaux de sortie. Les applications sont inimaginables!! C'est en fait un ordinateur miniaturisé!



La fonction qui nous a séduit dans ce projet est la facilité avec laquelle il nous a permit de connecter des « modules » (télémètre, led, moteur, servo-moteur, ecran LCD....) à un outil de commande, en l'occurrence un ordinateur par liaison USB. L'équivalent de notre projet en analogique était considérable en terme de coût de composant, espace disponible, temps d'implémentation et la complexité aurait été je pense conséquente. Le fait de pouvoir programmer en C fut aussi un aspect très attractif.

b) Anatomie d'un microcontrôleur

Un microcontrôleur se présente sous la forme d'un circuit intégré réunissant tous les éléments d'une structure à base de microprocesseur. Voici généralement ce que l'on trouve à l'intérieur d'un tel composant :

- un processeur CPU avec une largeur du chemin de données (bus) allant de 4 bits pour les modèles les plus basiques à 32 ou 64 bits pour les modèles les plus évolués ;
- de la RAM pour stocker les données et variables ;
- de la ROM (mémoire morte) pour stocker le programme. Différentes technologies peuvent être employées : EPROM, EEPROM, FLASH ;
- souvent un oscillateur interne pour le cadencement. Il peut être aussi réalisé en externe avec un quartz.
- des périphériques, capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :
- les convertisseurs analogiques-numériques; CAN (donnent un nombre binaire à partir d'une tension électrique) ;
- les convertisseurs numériques-analogiques (CNA) (effectuent l'opération inverse) ;

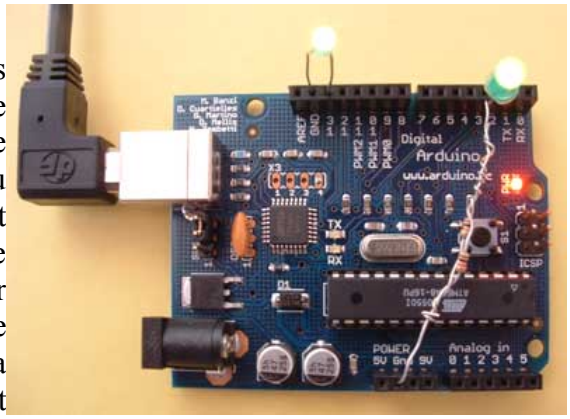
- les générateurs de signaux à modulation de largeur d'impulsion; PWM pour *Pulse Width Modulation*) utilisé pour les servo-moteur par exemple ;

- les timers, compteurs (compteurs d'impulsions d'horloge interne ou d'événements externes)
  ils permettent de réaliser les fonctions suivantes :
  -Génération d'un signal périodique modulé ou non en largeur d'impulsion,
  -Génération d'une impulsion calibrée,
  - Temporisation,
  - Comptage d'événements.

- les chiens de garde ou watchdog. Il s'agit d'une horloge interne. Cette horloge compte et est
  régulièrement remise à zéro. Dans le cas où le compteur de l'horloge ferait un "overflow", c'est-à-dire qu'il dépasserait sa valeur maximale, alors cela voudrait dire qu'il y a eu un problème dans l'exécution du programme. Le WATCHDOG effectue alors un reset logiciel ;

- les comparateurs (comparent deux tensions électriques) ;

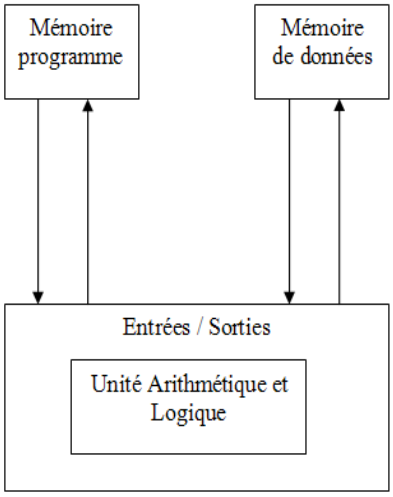
- les contrôleurs de bus de communication il existe plusieurs noms; UART,I²C,SSP,CAN,FlexRay,USB,Ethernet...

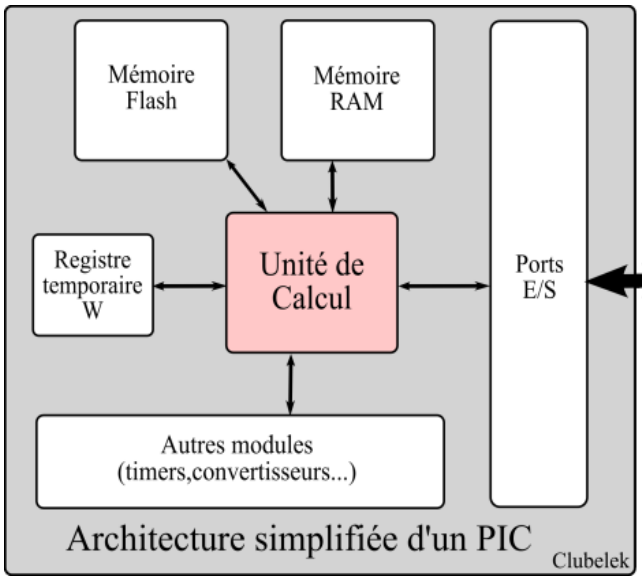


Ces microcontrôleurs PIC intègrent des quantités de composants très différentes. Le fonctionnement des périphériques peut être paramétré et commandé par le programme et/ou les entrées-sorties. Les périphériques peuvent générer une interruption qui, contraint le processeur à quitter le programme en cours pour effectuer une routine de traitement de l'interruption, lorsque l'événement qui la déclenche survient. Les microcontrôleurs peuvent généralement se placer dans un état de sommeil, dans lequel ils présentent une très faible consommation électrique. Un signal envoyé par l'un de leurs périphériques (timer, broche d'entrée-sortie, watchdog, ...) permet de les faire sortir de cet état de sommeil. Certains microcontrôleurs ont un nombre très restreint de broches, si bien qu'une broche donnée peut correspondre à plusieurs périphériques internes. La fonction choisie doit alors être sélectionnée par logiciel.

c) Architecture générale

Les pics utilisent l'architecture d'Harvard. Les flèches entre les unités de mémoires et de calcul représente les BUS notre PIC possède des BUS de 8 bits, cela veut dire qu'il a accès à 256 octect de mémoire vive. Par comparaison un ordinateur de bureau peut possèder des BUS allant jusqu'à 32 bit donc ayant accès à 4 Go de mémoire vive, ou même 64 bit l'équivalent de 18 millions de To! Mais bien sûr, la mémoire vive est limité physiquement, rendant impossible une utilisation complète.





Les PIC sont des processeurs dits RISC, c'est-à-dire processeur à jeu d'instruction réduit. Plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne.

Un cycle d'instruction d'un PIC dure 4 temps d'horloge. La plupart des instructions durent un cycle, sauf les sauts qui durent deux cycles. On atteint donc des vitesses élevées.

Avec un quartz de 4 MHz (ou l'horloge interne), on obtient donc 1 000 000 de cycles/seconde, or, comme le PIC exécute pratiquement 1 instruction par cycle, hormis les sauts, cela donne une puissance de l'ordre de 1 MIPS (1million d'instruction par seconde).

Pour notre PIC si l'on le faisait tourner à son maximum i-e 96MHz il pourrait atteindre 24 millions d'opération par seconde.

Pour un ordre d'idée, les supercalculateurs tournent en ce moment à 10^{15} opérations à virgule flottante par seconde, et le cerveau humain à 10^{17} opérations par seconde.

d) PIC 18F4550 et fonctionnalités

Alimentable de 2 à 6 V continu, notre PIC est compatible avec le mode de programmation ICSP (In Chip Serial Programming), et dispose d'un oscillateur interne pouvant monter jusqu'à 8 Mhz. Sur oscillateur externe, le 18F peut monter jusqu'à 96 MHz.

Le PIC communique grâce à des « bits » (signal 0 ou 1 courant - pas de courant) qu'il reçoit ou envoi sur l'un de ses « pin ». Un pin est une patte du micro-contrôleur. Notre micro-contrôleur comporte 40 pattes, dont 2 servent à l'alimentation. Il possède 5 port E/S. (Un port est en général constitué de 8 pin.) Il représente un fait un octet. Ce PIC dispose aussi de 13 CAN, de 2 comparateurs, de 2 PWM 10 bits, d'1 module de communication série synchrone et asynchrone, d'un module de communication USB, de 4 timers ...
Il faut savoir que les PICs ne font pas la différence entre les niveaux logiques TTL et CMOS. Il est donc alors préférable d'utiliser les deux extrêmes de signaux logiques TTL (0 et 5 volts), pour éviter tous problèmes possibles.

On ne peut affecter que deux valeurs différentes de configuration à chaque patte : un '1' pour la mettre en entrée, ou un '0' pour une sortie. Enfin, autre point fort du PIC : la consommation, car en tant que système embarqué, il faut que ce microcontrôleur consomme peu.

Ainsi, des quelques mA de consommation en fonctionnement normal, il peut passer sous les 10 µA en fonction veille ou sommeil, comme par exemple sur un téléphone portable, quand on ne s'en sert pas, la lumière reste éteinte : il est en veille. En revanche, au moindre appui sur un bouton (ex: interruption externe), il se remet en marche. Le Pic fonctionne de la même manière.

Il faut savoir que le PIC peut fournir jusqu'à 20 mA par sortie.

e) Les interruptions

Une interruption est une action, déclenchée par un événement précis, qui va arrêter le programme principal, s'exécuter, puis reprendre le programme principal à l'endroit où il s'était arrêté. Les interruptions peuvent être déclenchées par:

- Les timers
- Les pins interruptibles
- Les interfaces de communication
- Le convertisseur analogique-numérique

Les interruptions sont fondamentales dans le fonctionnement du PIC car elle permettent d'effectuer plusieurs choses à la fois, et de définir des priorités dans les différentes tâches à accomplir.

Pour utiliser les interruptions, il faut tout d'abord les activer. Il faut pour cela fixer les valeurs des registres suivant :

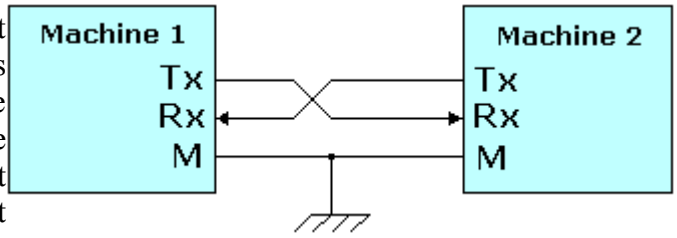
La routine d'interruption est structurée de la façon suivante:

- Sauvegarder les registres
- Identifier la source de l'interruption
- Effectuer l'action associée à cette interruption
- Restaurer les registres
- Revenir au programme principal

Par exemple, nous avons utilisé les interruptions pour créer une PWM afin de faire fonctionner le servo-moteur, de retourner un « keep alive » (vérifie le bon état du système....) Elles constitue le point le plus important dans la programmation.

f) Liaisons utilisées

Liaison série



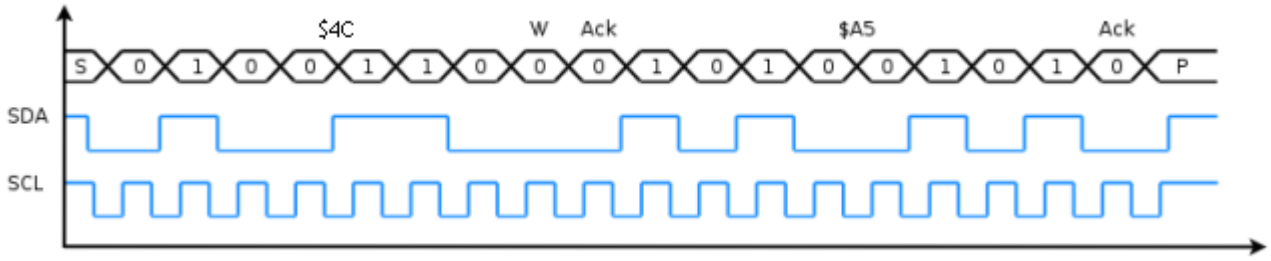
La liaison série est, en télécommunications et en informatique, l'action d'envoyer des données bit par bit sur un canal de communication ou un bus informatique. Le schéma traditionnel d'une liaison série est présenté sur la droite. L'octet à transmettre est envoyé bit par bit par l'émetteur sur la ligne Tx, vers le récepteur sur la ligne Rx qui le reconstitue. La vitesse de transmission de l'émetteur doit être identique à la vitesse d'acquisition du récepteur. La communication dans le cadre de notre projet ne se fera que dans un sens, du micro-contrôleur vers la carte de contrôle moteur. La transmission étant du type asynchrone, c'est à dire qu'il n'y a pas d'horloge commune entre l'émetteur et le récepteur, des bits supplémentaires sont indispensables au fonctionnement: bit de début de d'octet, bit de fin d'octet.

Liaison I2C

Un bus I²C contient trois fils :

- un signal de donnée (SDA) ;
- un signal d'horloge (SCL) ;
- un signal de référence (masse).

Le périphérique qui gère la communication est le maître, ici le micro-contrôleur, c'est lui qui génère l'horloge sur la broche SCL et qui envoie les données sur la broche SDA. Au début de la communication SDA passe de 1 à 0 alors que SCL reste à 1, c'est le StartBit. Après avoir imposé la condition de départ, le maître passe SCL à 0 et emet l'octe désiré. Il valide la donnée en appliquant pendant un instant un niveau 1 sur la ligne SCL. Lorsque SCL revient à 0, il recommence l'opération avec le reste de l'octet jusqu'à ce que l'octet complet soit transmis. Il redéfinit ensuite SDA comme une entrée et scrute son état ; l'esclave doit alors imposer un niveau 0 pour signaler au maître que la transmission s'est effectuée correctement,



c'est l' acquittement, la communication peut donc continuer.
Afin de clarifier le processus de communication, en voici un schéma :

Liaison USB

L'USB est un bus informatique à transmission série servant à connecter des périphériques informatiques à un ordinateur. Sa complexité est grande et nécessiterait un dossier complet afin d'être explicitée. Ainsi, nous nous contenterons d'utiliser un profil particulier de l'USB nommé HID servant à simplifier la gestion du protocole. HID est par exemple utilisé pour piloter des matériels simples tels que souris et clavier et permettre une communication « plug and play » avec le périphérique. L'initialisation du profil HID sur le pic est simplifié grace à un firmware qu'il suffit de piloter au moyen de fonctions de bibliothèque proposées par le compilateur que nous utiliserons.

g) Conclusion sur la partie théorique

Difficile à appréhender en premier lieu, les PIC s'imposent comme éléments indispensables pour de nombreux montages, tant par leur rapport qualité/prix, que par leurs capacités ; leurs principaux handicaps étant en général le nombre limité de leurs entrées/sorties et/ou leurs capacités mémoires pour le programme. Ils sont programmables avec un minimum de difficultés, il n'y a auxune limite ormis l'imagination de son propriétaire. De même, le matériel de programmation et de test, requiert le minimum de connaissances que l'on peut demander à un amateur. De fait, l'amateur appréciera utiliser les PICs tant ils sauront lui simplifier la vie pour tous ses montages, quels qu'ils soient. Pour conclure, nous pouvons dire que dans bien des cas, les PICs nous permettront d'arriver à nos fins avec un minimum de difficultés, et un maximum de simplicité.

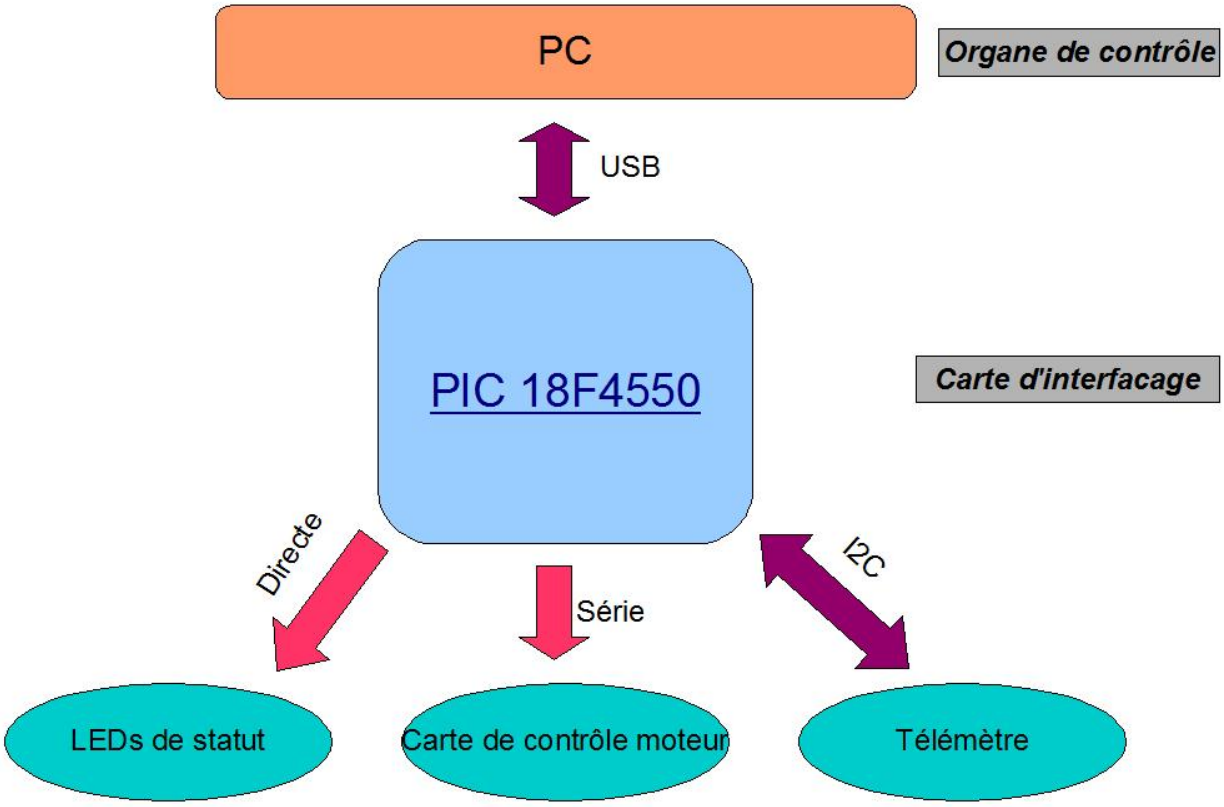
B) Application

Nous réaliserons donc une carte d'interfacage à base de micro-contrôleur entre un PC et différents capteurs et actionneurs électroniques. La présentation des mesures et le contrôle sera réalisé sur le PC au moyen d'une liaison USB. Même si les matériels pouvant être interfacés à la carte sont multiples, nous utiliserons dans le cadre de notre projet les périphériques suivants :

- 3 LEDs de statut
- Une carte de contrôle moteur sur liaison série (SyRen10)
- Un télémètre à ultrason sur liaison I2C (SRF02)
- Un ordinateur disposant d'un port USB

La carte sera réalisée sur la base d'un PIC 18F4550.

Le schéma des communications de ces différents matériels peut être modélisé comme suit :

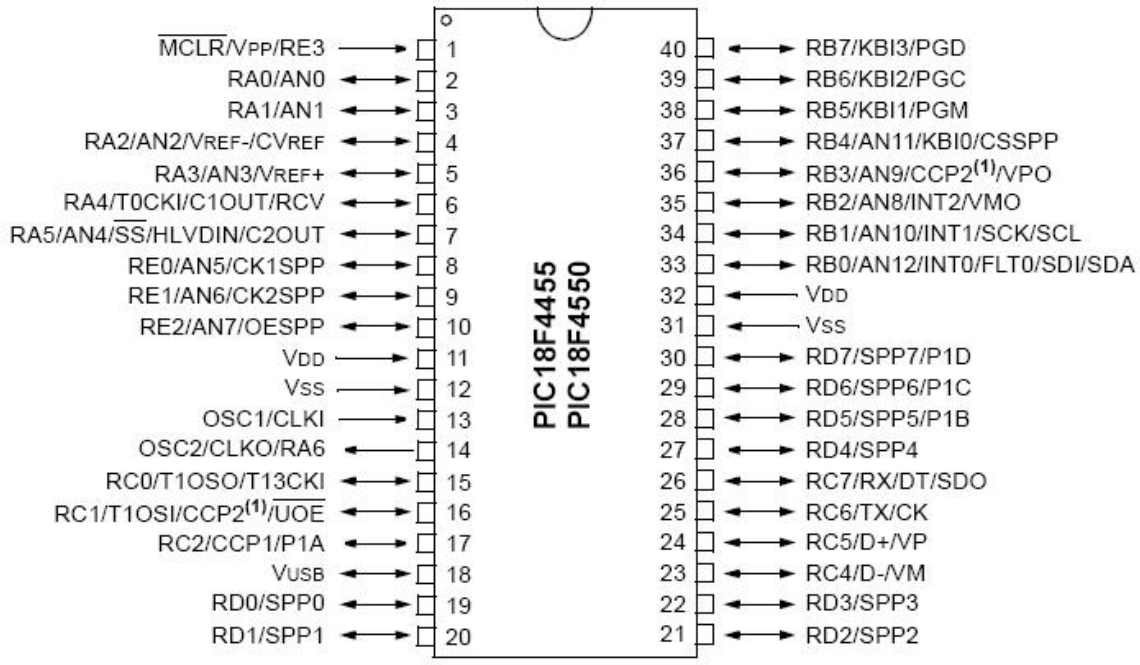


a) Réalisation d'un premier circuit de test

Nous choisissons dans un premier temps de créer un circuit simple qui servira à s'assurer du bon fonctionnement du microcontroleur et de la liaison USB. Nous commencerons par utiliser une alimentation exterieur, au lieu de la tension fournie par le port USB maitre, pour minimiser les risques sur le PC.

Définition des besoins

Avant de poursuivre, observons la structure générale du PIC :



Les caractéristiques qui nous intéressent pour le moment sont la tension d'alimentation et l'emplacement des ports de communication USB. Nous trouvons dans la documentation :

- VDD = 5v
- USB : D+ et D-

En effet, le PIC étant un composant TTL, il doit être alimenté en 5v sur sa broche VDD. La documentation stipule également que la broche MCLR doit être mise à 5v et que la broche Vusb doit être reliée à un condensateur d'environ 100nF. Le signal d'horloge, ici de 8MHz, doit être fourni aux broches OSC1 et OSC2.

Les broches RE0, RE1 et RE2 serviront à brancher les LEDs de statut.
Un régulateur 7805 est utilisé pour générer 5v à partir d'une tension de 12v.

Nous réalisons le premier schéma, présent en annexe à l'aide du logiciel libre Kicad.

Après une première programmation du PIC, qui sera explicitée en deuxième partie, et plusieurs essai de connexion USB, nous constatons le bon fonctionnement de la carte et la réponse des LEDs de statut aux ordres venant du PC. Nous commençons alors la réalisation d'une seconde carte qui doit cette fois ci pouvoir piloter le télémètre, les LEDs et la carte de contrôle moteur.

b) Réalisation du deuxième circuit

Pour alléger le circuit et le rendre plus autonome, nous alimenterons maintenant le micro-contrôleur par le bus USB. Il faut aussi rajouter à notre circuit précédent les composants nécessaires à une liaison I2C, c'est à dire deux résistance de rappel, et faciliter la connexions des périphériques à la carte en ajoutant des fils de connexion.

Le deuxième schéma est rapidement réalisé, il est présenté en annexe.

c) Partie programmation

Définition des besoins

Dans cette partie, nous étudions les mécanismes qui permettent de piloter les périphériques au niveau du micro-contrôleur et de rendre fiables les communications.

Pour programmer notre micro-contrôleur, le langage par défaut est l'assembleur PIC. Ce langage est directement compris, après transcription en fichier binaire, par le micro-contrôleur. Il produit donc un code légé et rapide. Pourtant les objectifs de notre projet et particulièrement la gestion de la liaison USB, rendent difficilement utilisable ce langage, qui nécessiterait alors un grand nombre de ligne de code. Nous optons pour un compilateur C, mikroC, qui propose à l'utilisateur de nombreuses bibliothèque de fonctions dont la gestion USB, I2C et série. L'écriture du programme dans la mémoire du micro-contrôleur se fait grace à une carte de dévloppement, easyPIC, proposée avec le compilateur mikroC.

Le programme doit pouvoir être lancé à partir d'un ordre venant du PC. Pour rendre le programme fiable, nous implementerons un mécanisme qui s'assurera du bon fonctionnement du logiciel de contrôle, situé sur le PC. Toutes les 50 milisecondes, une commande est envoyée au micro-contrôleur pour lui signifier que tout fonctionne bien. Si le PC plante, la commande n'est plus envoyée et au bout de 200 milisecondes le micro-contrôleur se réinitialise et s'arrête. Pour chaque commande reçu, le micro-contrôleur réagit en interrogeant les périphériques et en les commandant.

Programme principal

Nous ne parlerons ici que du programme situé sur le PIC, le programme situé sur le PC ne faisant pas l'objet de ce projet.

Commençons par observer les fonctions de bibliothèque qui nous permettrons de communiquer avec les périphériques:

Programmation des différentes liaisons

I2C :

`I2C_Init()`	Initialise le bus I2C à la fréquence choisie.
`I2C_Start()`	Lance une communication I2C.
`I2C_Rd()`	Lit un octet sur le bus I2C
`I2C_Wr()`	Ecrit un octet sur le bus I2C
`I2C_Stop()`	Arrêt d'une communication I2C

Série :

`Usart_Init()`	Initialise le port série au débit choisi
`Usart_Write()`	Ecrit un octet sur le port série

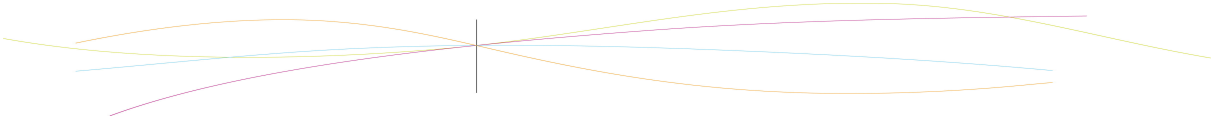
USB :

`Hid_Enable()`	Initialise le bus USB sur le profil HID
`Hid_Read()`	Lit une séquence d'octet su le bus USB
`Hid_Write()`	Ecrit une séquence d'octet sur le bus USB

Fonctions utilisées

```
void main() {
  Init_Main();
  re_init();

  while(1) {
    Hid_Read();
    if( userWR_buffer[0] == CMD_KEEPALIVE_CLIENT ) { kernel(); }
    }
}
```

Cette fonction est lancée au démarrage du PIC. Elle éxecute deux autres fonctions Init_main() et re_init() qui se charge de l'initialisation des différentes liaisons et des compteurs de temps. Elle entre ensuite dans une boucle infinie dans laquelle elle attends un message venant de la liaison USB, si celui-ci correspond à la commande de connexion, elle lance la fonction kernel, chargé du traitement de l'ensemble des données.



Explicitons maintenant les deux fonctions d'initialisation :

```
void Init_Main()
{
      INTCON = 0;
      INTCON2 = 0xF5;
      INTCON3 = 0xC0;
      ADCON1 = 0x0F;

      TRISB = 0;
      PORTB = 0;

      TRISE = 0;
      TRISD = 0;
      PORTD = 0;

      Usart_Init(9600);
      HID_Enable(&userWR_buffer, &userRD_buffer);
      I2C_init(100000);
      Delay_ms(2000);
      PORTE.F0 = 1;
}
```

Après réglage des sources d'interruptions, nous plaçons les ports B, D et E du PIC en sortie et initialisons la liaison série, USB et I2C. La LED située sur la broche 0 du port E est allumée pour signifier que les premières initialisations se sont bien déroulées.

```
void re_init()
{
   PIE1.TMR1IE = 0;
   Usart_Write(128);
   PORTE.F1 = 0;
   PORTB = 0;
   PORTD = 0;
}
```

Cette fonction est appelée à la fois au premier lancement du PIC et à la déconnexion d'un client sur le PC. Elle envoie la valeur 128 à la carte de contrôle moteur pour la mettre en position stop. Elle réinitialise également le compteur de temps.

```
int kernel()
{
    int get  = 0;
    char out;
    keep_alive = 4;
    Hid_Read();
    PORTE.F1   = 1;
    start_timer1();

    while(1) {
      get = Hid_Read();
      if(get > 0 )
      {
           if(userWR_buffer[0].F6 == 1 && userWR_buffer[0].F7 == 1) { out =
4*(userWR_buffer[0] & 0b00111111); Usart_Write(out); PORTB = out; }
             if(userWR_buffer[0] == CMD_KEEPALIVE_CLIENT) { keep_alive = 4;
}
             if(userWR_buffer[0] == 30) {PORTE.F2 = ~PORTE.F2; }
             get=0;
      }
      if( keep_alive <= 0 ) { userWR_buffer[0] = 0; re_init(); return 0; }
    }
```

```
return 0;
}
```

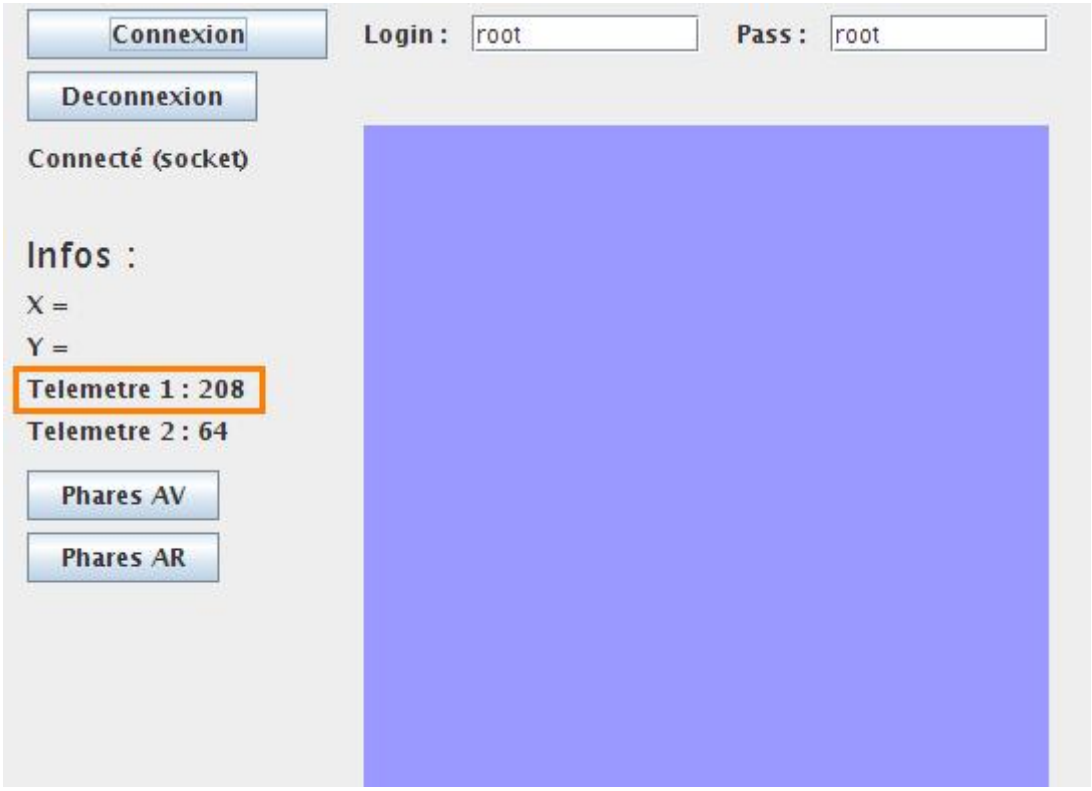
Cette fonction est le coeur du programme. Elle ré-émet les commande reçues du PC sur la carte moteur et les LEDs de statut. La fonction start_timer1() est d'abord appelée pour enclencher le compteur responsable du contrôle de la liaion avec le PC. Ensuite, le programme rentre dans une boucle d'éxecution « infinie » qui ne s'achêve que si le PC ne renvoit pas l'ordre `CMD_KEEPALIVE_CLIENT` pendant 200 milisecondes, ce qui peut signifier soit une déconnexion, soit un bog sur le PC.

```
void interrupt_low()
{
     count++;

     if(count >= TEMPS_KEEPALIVE)
     {
               keep_alive--;
               count = 0;

               I2C_Start();
               I2C_Wr(0xE2);
               I2C_Wr(2);
               I2C_Repeated_Start();
               I2C_Wr(0xE3);
               userRD_buffer[0] = I2C_Rd(0);
               I2C_Stop();

               I2C_Start();
               I2C_Wr(0xE2);
               I2C_Wr(3);
               I2C_Repeated_Start();
               I2C_Wr(0xE3);
               userRD_buffer[1] = I2C_Rd(0);
               I2C_Stop();

               I2C_Start();
               I2C_Wr(0xE2);
               I2C_Wr(0);
               I2C_Wr(81);
               I2C_Stop();

               Hid_Write(&userRD_buffer, 2);
     }
     PIR1.TMR1IF = 0;
}
```

Cette fonction est spéciale, elle est déclenchée par une interuption. Elle n'est pas volontairement appelée dans le programme mais s'éxecute lors du débordement du registre compteur, toutes les 50 milisecondes. Elle fonctionne, d'une certaine manière, en parallèle du reste du programme. Elle s'occupe de décrémenter la variable « keep_alive », qui, si elle tombe à zéro provoquera la réinitialisation et l'arrêt du PIC. Elle gère aussi la communication avec le télémetre suivant le schéma d'une communication I2C.

C) Résultats

La communication entre le micro-contrôleur et le PC fonctionne bien, le mécanisme de gestion d'erreur fonctionne également si un arrêt inattendu du programme de contrôle sur le PC survient. On peut observer les résultats des mesures du télémètre sur l'ordinateur :



4) CONCLUSIONS ET PERSPECTIVES

Nos objectifs de départ sont réalisés, la carte d'interfacage est fonctionnelle et le programme du micro-contrôleur est stable. Le programme situé sur l'ordinateur reçoit correctement les informations provenant du télémètre et n'a plus qu'à les afficher. L'ajout de nouveaux matériels sur la carte actuelle devrait être simple et ne nécessiter que la reprogrammation du micro-contrôleur. Néanmoins, nous aurions aimé trouver le matériel qui nous aurait permis de brancher et de débrancher facilement les périphériques de la carte. Ici, il faut souder ou déssouder les périphériques.

Concernant l'apport personnel du projet, nous pouvons dire que nous avons été très interessés par le sujet et que nous serions heureux de poursuivre les recherches sur les micro-contrôleurs. La gestion des membres et la répartition des tâches a aussi été un apprentissage qui nous a été bénéfique. Nous regrettons cependant d'avoir parfois été bloqués à certaines étapes de la fabrication de la carte pour des raisons telles que le manque de composants ou le délai de réalisation des circuits imprimés.

Les perspectives de poursuite du projet sont multiples. Nous utilisons actuellement notre carte d'interfacage dans le cadre d'un projet de robotique, mais nous pourrions l'utiliser par exemple pour des mesures de température, de pression, de luminosité dans une installation domotique. Aussi, nous pourrions utiliser des capteurs infrarouges pour permettre un contrôle de la carte par télécommande.

5) BIBLIOGRAPHIE

[1] lien internet : http://fr.wikipedia.org/wiki/Microcontr%C3%B4leur_PIC (valide à la date du 13/06/2009).

[2] lien internet : http://libhid.alioth.debian.org/ (valide à la date du 13/06/2009).

[3] lien internet : http://fribotte.free.fr/bdtech/cours/pic16f84/index.html (valide à la date du 13/06/2009).

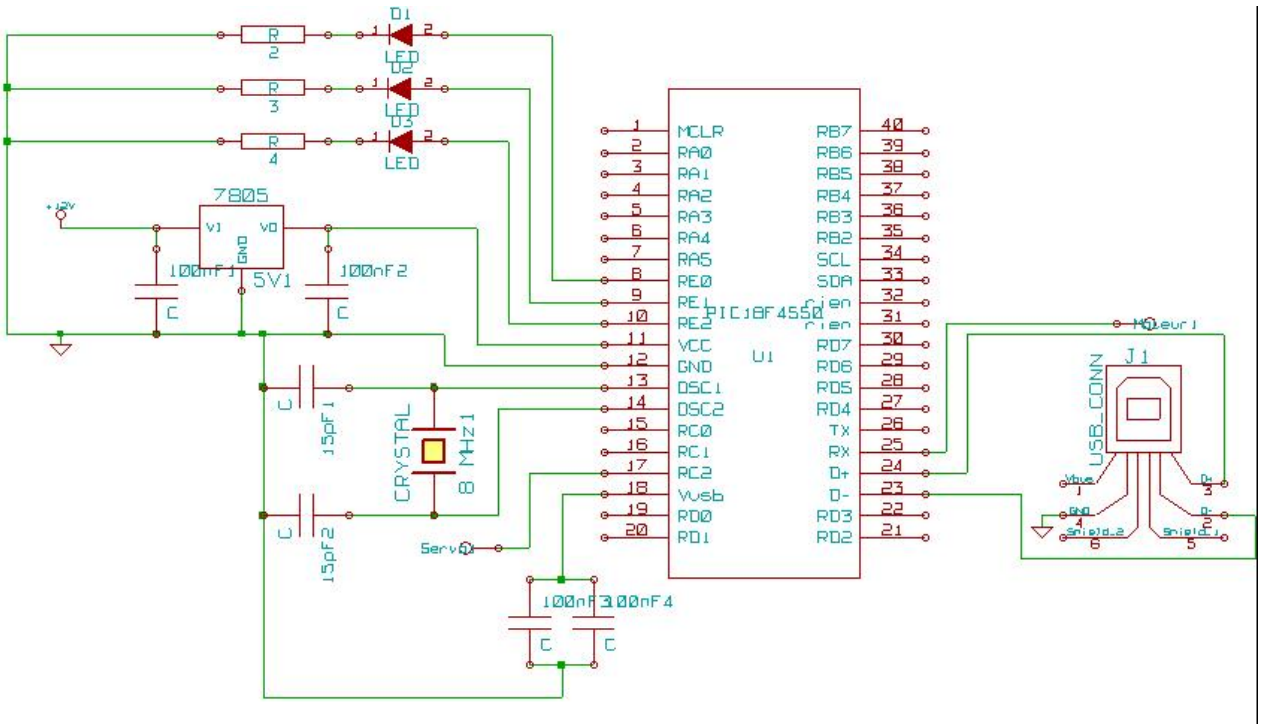
[4] lien internet : http://clubelek.insa-lyon.fr/joomla/fr/base_de_connaissances/electronique/developpement_de_cartes_electroni_2.php (valide à la date du 13/06/2009).

[5] lien internet : http://www.pobot.org/A-manger.html (valide à la date du 13/06/2009).

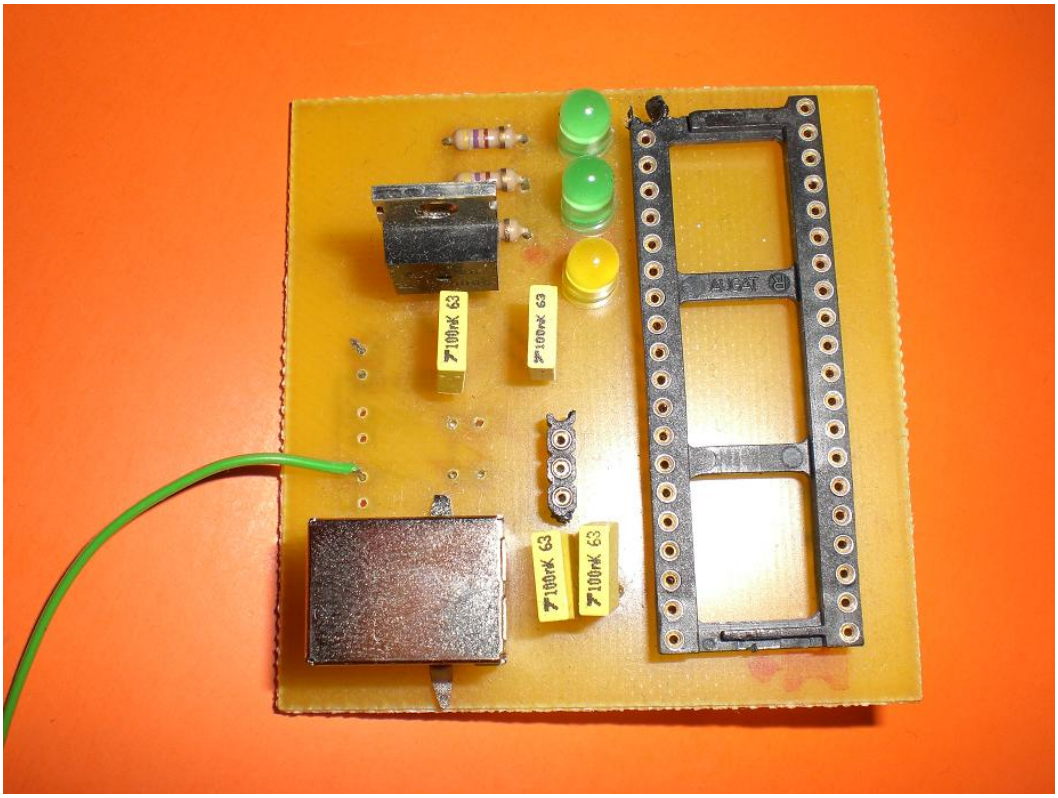
[6] lien internet : http://fr.wikipedia.org/wiki/I%C2%B2C (valide à la date du 13/06/2009).

6) ANNEXES

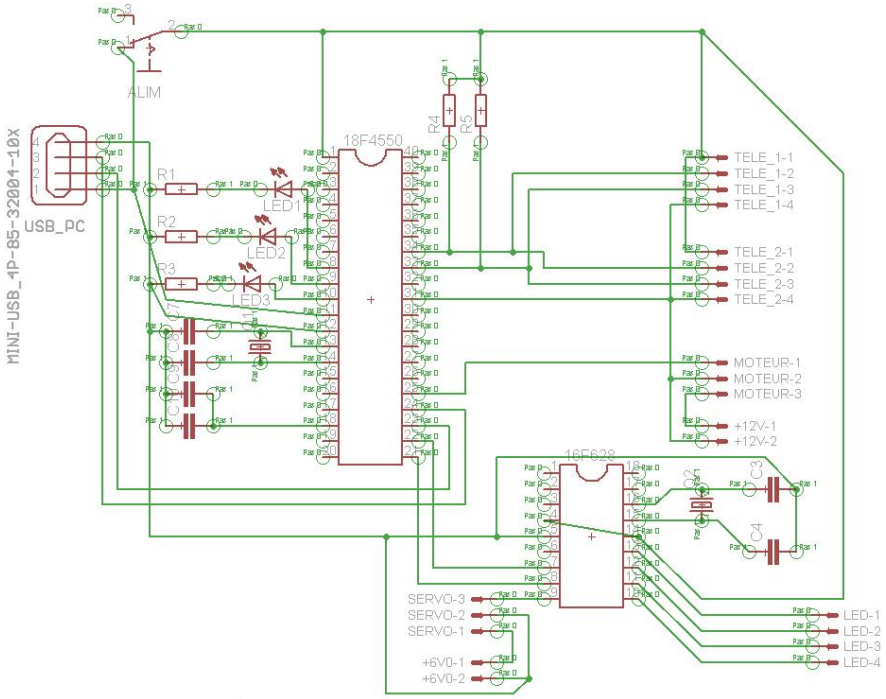
Premier schéma



Première carte



Deuxième shéma



Deuxième carte