Mise en œuvre d'un capteur de température par bus I2C

Eric Pommier, Vincent Creuze, Didier Crestani, Sophie Dupuis, Bertrand Gelis, Isabelle Pinon, Vincent Thomas

Mise en œuvre d'un capteur de température par bus I2C

Eric Pommier, Vincent Creuze, Didier Crestani, Sophie Dupuis,
Bertrand Gelis, Isabelle Pinon et Vincent Thomas.
prenom.nom@univ-montp2.fr
IUT de Montpellier, Département GEII, 99 Avenue d'Occitanie, 34296 Montpellier

RESUME : Cet article présente une séance de travaux pratiques permettant à des étudiants d'IUT GEII (Génie Electrique et Informatique Industrielle) de découvrir le fonctionnement du bus de communication I2C. Elle fait partie des enseignements relatifs à la programmation avancée (en langage C) des microcontrôleurs (2ème semestre de la première année). Nous décrivons les principales étapes du TP et nous présentons également les résultats d'un sondage illustrant comment ce TP est perçu par les étudiants.

Mots clés : informatique industrielle, microcontrôleur, bus série, langage C.

1 INTRODUCTION

De nombreuses applications des microcontrôleurs reposent sur la lecture de capteurs. Les échanges de données entre ces derniers et les microcontrôleurs peuvent être analogiques (via les convertisseurs analogiques intégrés aux microcontrôleurs) ou numériques. Dans le cas de liaisons numériques, le nombre limité de broches des microcontrôleurs a favorisé le développement de liaisons "série", qu'elles soient asynchrones, telles l'USART (Universal Serial Asynchronous Receiver Transceiver), ou synchrones, telles le bus SPI (Serial Peripheral Interface) ou le bus I2C (Inter-Integrated Circuit).

Il nous a donc semblé intéressant de proposer aux étudiants de première année de l'IUT GEII de Montpellier une séance de travaux pratiques dans laquelle ils mettraient en œuvre une liaison I2C entre un microcontrôleur et un capteur de température.

Cette séance de travaux pratique, d'une durée de trois heures, s'intègre dans le module II2 d'informatique industrielle et vient compléter une série de travaux pratiques consacrée à la programmation des microcontrôleurs PIC 18F452 en langage C et à l'utilisation de leurs diverses fonctionnalités intégrées : USART, Convertisseurs Analogiques Numériques, Timers, Module CCP (Capture, Compare, and PWM), Interruptions...

Dans cet article, nous présentons tout d'abord les objectifs du TP, puis nous détaillons les caractéristiques techniques des composants et de la maquette. Nous rappelons ensuite le principe de communication du protocole I2C. Nous fournissons l'algorithme et le programme en langage C. Enfin, nous présentons les résultats d'un sondage effectué en mars 2014 auprès des étudiants de plusieurs groupes à l'issue de cette séance de TP.

2 OBJECTIFS ET STRUCTURE DE LA SEANCE DE TRAVAUX PRATIQUES

L'objectif de la séance de TP est d'effectuer, à l'aide d'un microcontrôleur PIC 18F452, la lecture de la température ambiante au moyen d'un capteur numérique Maxim Integrated DS1621. La donnée lue sur le capteur est convertie en caractères ASCII par le microcontrôleur qui sont envoyés vers un afficheur LCD (cette dernière partie fait appel à des fonctions d'affichage développées lors d'une séance de TP précédente). La communication entre le capteur de température et le microcontrôleur se fait par bus série de type I2C.

Outre la découverte du protocole I2C, ce TP permet de développer les capacités de programmation en langage C et d'approfondir les mécanismes de configuration des registres spéciaux (SFR, Special Function Registers) du PIC (dans le cas de ce TP : SSPCON1, SSPCON2, SSPSTAT, SSPADD).

La séance de TP dure trois heures et est accomplie en binôme. Les étudiants doivent d'abord dessiner l'organigramme du programme (les algorithmes sont fournis dans l'énoncé), puis codent ce dernier fonction après fonction à l'aide de la documentation du microcontrôleur et des notions détaillées par le professeur en début de séance (fonctionnement du bus I2C, du module I2C du PIC et des registres associés).

3 PRESENTATION DES COMPOSANTS ET DE LA MAQUETTE

3.1 Microcontrôleur PIC 18F452

Le microcontrôleur Microchip PIC 18F452 est un microcontrôleur 8 bits, doté de 32Ko de mémoire flash programmable, de 1546 octets de mémoire RAM et de 256 octets de mémoire EEPROM non volatile. Il peut fonctionner jusqu'à 40MHz avec une horloge externe. Il compte 40 broches et offre de nombreuses fonctionnalités, parmi lesquelles : 2 niveaux de priorité d'interruption, 4 timers, 2 modules CCP (Capture/Compare/PWM), un module USART (RS232, RS485), un convertisseur analogique numérique 10 bits et un module MSSP (Master Synchronous Serial Port) fonctionnant en mode SPI ou I2C. C'est ce dernier module que nous exploitons aujourd'hui. Le brochage du PIC18F452, extrait de la datasheet du composant [1], est représenté sur la figure 1.

L'environnement de développement MPLAB fourni par le constructeur Microchip est performant et gratuit [2]. Le compilateur XC8, également gratuit en version limitée (code non optimisé lors de la compilation), est suffisant pour les applications envisagées. Il est intéressant de noter que cette suite logicielle permet le débogage in-situ des microcontrôleurs PIC18.

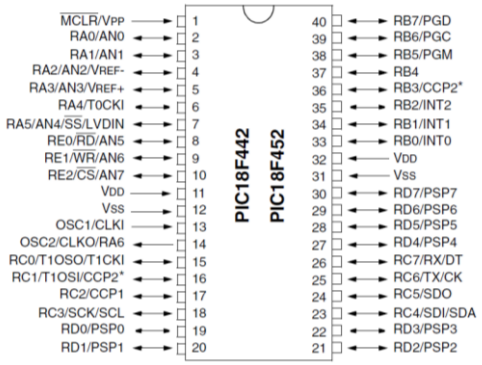


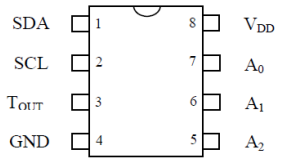
Fig. 1. Brochage du microcontrôleur Microchip PIC 18F452 (extrait de la datasheet du composant).

3.2 Capteur de température DS1621

Le capteur de température DS1621 permet de mesurer des températures comprises entre -55°C et +125°C, avec une résolution de 0.5°C. La donnée mesurée est transmise en I2C sur 9 bits (un octet contenant les 8 MSB et un octet contenant le LSB). Dans ce TP, nous ne travaillerons que sur 8 bits, donc avec une résolution de 1°C.

La mesure de température est effectuée en moins d'une seconde. Le composant peut être configuré en mesure répétée (ce que nous utiliserons afin d'avoir toujours une donnée de température disponible) ou en mesure unique (one shot). Une sortie numérique T_{out} et deux registres internes définissant des seuils haut et bas permettent d'utiliser également ce composant de façon autonome en tant que régulateur thermostatique. Nous n'utiliserons pas cette fonctionnalité.

Le composant compte 8 broches, peut être alimenté entre 2.7V et 5.5V et communique par bus I2C au choix à 100kHz ou à 400kHz. Son brochage, extrait de la datasheet du composant [3], est reproduit sur la figure 2.



DS1621 8-PIN DIP (300mil)

PIN DESCRIPTION

SDA	- 2-Wire Serial Data Input/Output
SCL	- 2-Wire Serial Clock
GND	- Ground
T_{OUT}	- Thermostat Output Signal
A0	- Chip Address Input
A1	- Chip Address Input
A2	- Chip Address Input
V_{DD}	- Power Supply Voltage

Fig. 2. Brochage du capteur de température Maxim Integrated DS1621 (extrait de la datasheet du composant).

3.3 Maquette de travaux pratiques

La maquette (fig. 3) a été conçue et assemblée au sein du département GEII de l'IUT de Montpellier. Elle est utilisée en TD et TP pour l'apprentissage de la programmation d'un microcontrôleur, aussi bien en C qu'en assembleur. La maquette a pour base un microcontrôleur PIC18F452 cadencé par un quartz de 10MHz. Elle possède :

- Son propre programmateur/debugger compatible ICD2 permettant une liaison simple (USB) avec un PC,
- Un afficheur LCD 2*16 caractères (mode 4 bits),
- 8 LEDs de visualisation (Port D du PIC),
- 4 interrupteurs connectés sur 2 entrées d'interruptions externes et sur 2 entrées externes de 2 compteurs,
- 2 entrées analogiques (CAN interne) adaptées pour pouvoir convertir du 0/+5V ou du -5V/+5V (sur fiche BNC),
- 2 sorties analogiques (CNA externe série I2C MAX517) adaptées pour pouvoir convertir en 0/+5V ou en -5V/+5V (sur fiche BNC),
- Un capteur de température intégré DS1621 avec protocole I2C,
- Une mémoire EEPROM externe 24LC256,
- Un connecteur BD37 permettant la liaison (25 lignes entrée/sortie sur les 34 du µC) avec diverses cartes filles.

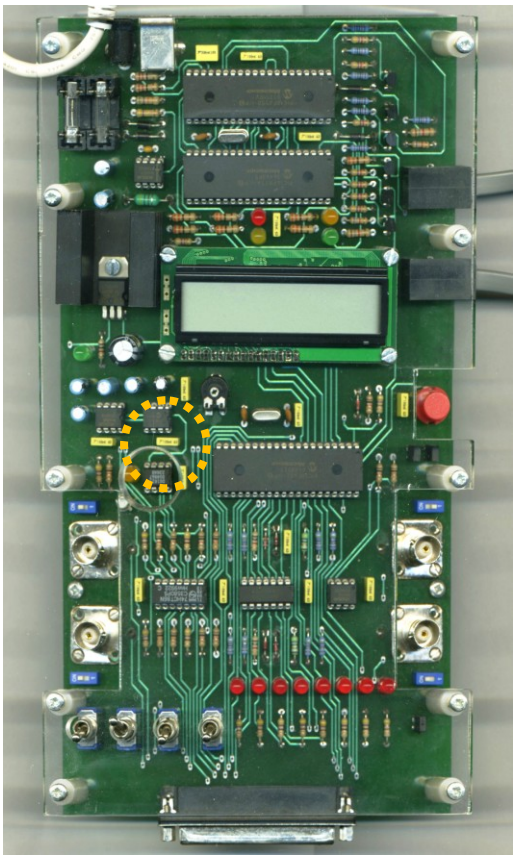


Fig. 3. Maquette de travaux pratiques. Le PIC 18F452 est au centre. A sa gauche, entouré de pointillés jaunes, on voit le capteur de température Maxim Integrated DS 1621.

La maquette peut fonctionner soit en mode Debugger, sous MPLAB (en maintenant la liaison entre le programmateur ICD2 et le PIC18F452), soit en mode Programmateur (en supprimant, après programmation, la liaison entre l'ICD2 et le PIC18F452).
Tous les composants intégrés du kit sont sur supports DIP pour faciliter la maintenance. Le circuit imprimé du kit est inséré entre deux plaques de plexiglas afin de prévenir toutes mauvaises manipulations. Au-dessus du capteur de température, un trou dans la protection en plexiglas permet à l'étudiant de poser son doigt sur le capteur pour en faire varier la température.

4 RAPPELS SUR LE PROTOCOLE I2C

Le protocole I2C a été développé par Philips pour la communication entre les circuits intégrés avec seulement trois fils : SDA (données), SCL (horloge) et masse. C'est un système de transfert bidirectionnel de données synchrone série, à architecture « maître-esclave », pouvant piloter jusqu'à 128 circuits intégrés.

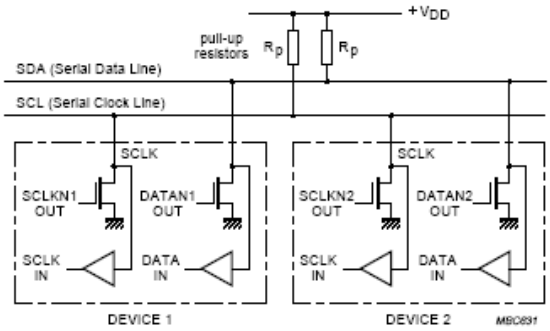


Fig. 4. Support physique du bus I2C (extrait des spécifications Philips [4])

Chaque circuit a une adresse sur 7 bits (0x00 à 0x7F en câblage externe et interne). Un circuit peut recevoir ou envoyer des données, mais c'est le maître qui gère les transferts. Ces derniers sont sécurisés par une procédure d'accusé de réception (ACK) qui confirme la bonne réception du message par le destinataire.
Un échange de données entre le maître et un des circuits esclaves se déroule comme suit. Tout d'abord le maître initie la transmission (condition de départ : Start, la ligne SDA passe de 1 à 0 pendant que SCL est maintenue à 1), puis il sélectionne l'esclave par son adresse. Ensuite le maître échange des données (SDA) avec l'esclave. Chaque octet est suivi d'un acquittement (ACK) émis par le circuit qui reçoit la donnée (que ce soit le maître ou un esclave). A la fin de l'échange de données, le maître met fin à la transmission (condition de fin : Stop). Durant toute la durée de l'échange, le maître génère l'horloge de synchronisation de la communication (SCL).
La figure 5 montre la façon dont se déroule une écriture du maître vers l'esclave. On remarque que l'adresse émise est complétée par un 0 (LSB), permettant de signaler à l'esclave qu'il s'agit d'une écriture (de commandes ou de données). On observe que lorsque le maître émet ses données vers l'esclave, après chaque octet émis, l'esclave produit un bit d'acquittement (ACK).
La figure 6 montre la façon dont se déroule une lecture du maître sur l'esclave (par exemple pour aller lire la température mesurée par le capteur). On remarque que l'adresse émise est complétée par un 1 (LSB), permettant de signaler à l'esclave qu'il s'agit d'une lecture. On observe que lorsque l'esclave émet en retour des données vers le maître, après chaque octet, c'est le maître qui produit le bit d'acquittement (ACK). Le dernier bit d'acquittement émis par le maître est de type NACK (not acknowledge).
Une description plus complète du bus I2C peut être trouvée sur le site web de NXP (ex-Philips) [4].
La figure 4 montre le support physique du bus I2C

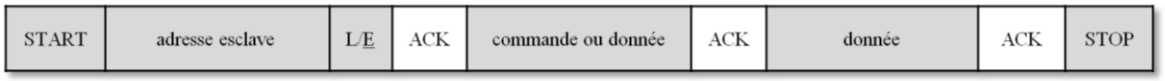


Fig. 5. Trame I2C d'écriture du maître vers l'esclave. Dans ce cas, le bit L/E est à 0 (écriture). Les données en gris sont émises par le maître. Celles en blanc représentent les acquittements émis par l'esclave pour signaler au maître de que chaque donnée a bien été reçue.

Fig. 6. Trame ^{I2C} de lecture par le maître sur l'esclave. Dans ce cas, le bit L/E est à 1 (lecture). Les données en gris sont émises par le maître. Celles en blanc sont émises par l'esclave. C'est le maître qui clôt l'échange par l'émission d'un NACK (Not-Acknowledge) et d'un STOP.

5 CONFIGURATION DU MODULE MSSP DU PIC

Le microcontrôleur PIC 18F452 est équipé d'un module MSSP (Master Synchronous Serial Port) lui permettant de communiquer directement avec des périphériques I2C. La configuration de ce module est assez simple et se fait par l'intermédiaire de registres.

Le module I2C du PIC18 comprend 6 registres :

- SSPCON1 et SSPCON2 → registres de contrôle
- SSPSTAT → registre de l'état de la transmission
- SSPBUF → registre des données en lecture ou écriture (accessible par l'utilisateur)
- SSPSR → registre de conversion parallèle série pour la transmission ou la réception(non accessible à l'utilisateur)
- SSPADD → registre des adresses ou de la vitesse de transmission (accessible par l'utilisateur)

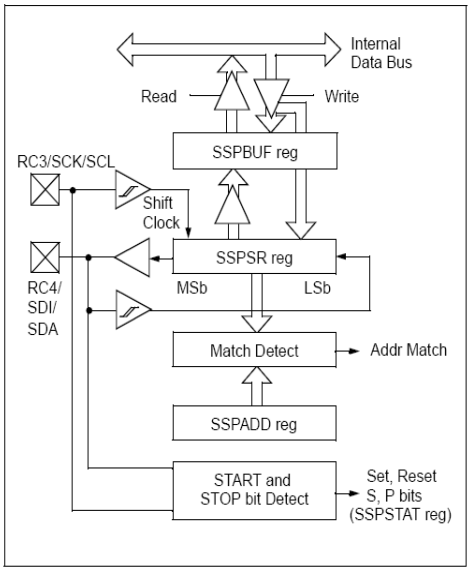


Fig. 7. Schéma du module I2C du PIC18F452 (extrait de la datasheet du composant [1])

La vitesse de transmission (100kHz dans ce TP) est gérée automatique par le Baud Rate Generator (BRG) et elle est réglable au moyen des 7 bits de poids faible du registre SSPADD. Lorsqu'une écriture est réalisée dans le registre SSPBUF (registre d'émission du module MSSP), le module BRG (pré-chargé avec la valeur contenue dans SSPADD) est décrémenté à chaque cycle machine jusqu'à atteindre 0. Le compteur du module BRG est rechargé automatiquement avec le contenu de SSPADD à la fin de la transmission série de chaque bit. La fréquence de transmission dépend donc de la fréquence F_{osc} de l'oscillateur cadençant le PIC (quartz à 10MHz dans notre cas) et est définie par la relation suivante :

$$F_{transmission} = F_{osc}/(4*(SSPADD + 1))$$

Ainsi dans notre cas, afin d'obtenir une transmission à 100kHz, SSPADD doit contenir le nombre 24 (0x18 en hexadécimal).

Les broches SCL (RC3) et SDA (RC4) doivent être configurées en sortie grâce au registre TRISC.

La configuration du mode maître et l'activation de l'I2C se fait au moyen du registre SSPCON1 (voir datasheet du composant pour plus de détails).

Le bit SMPT du registre SSPSTAT permet en mode maître de régler l'échantillonnage en fin de donnée (désactivation du slew-rate control).

6 ALGORITHME

Une fois la configuration des ports et du module I2C effectuée, le programme suit l'algorithme suivant :

- Configuration du module MSSP du PIC.
- Initialisation du capteur de température.
- Boucle infinie
  {
  - Demande d'échantillon de température
  - Lecture de la température
  - Conversion température en code ASCII
  - Affichage de la température sur le LCD
  }

L'adresse de notre capteur est 0x90 en écriture et donc 0x91 en lecture.

L'initialisation du capteur consiste à programmer son registre de configuration. Pour cela, il suffit d'envoyer la commande 0xAC (Access Config, voir

datasheet du composant), puis d'envoyer la valeur à écrire dans le registre (0x02 dans notre cas, pour définir la polarité positive pour l'échange de données et pour régler la mesure de température en mode continu).
La demande d'échantillon de température est faite en envoyant la commande 0xEE au capteur (en mode écriture).
La lecture de la température mesurée se fait en envoyant la commande 0xAA au capteur.

7 PROGRAMME ET FONCTIONS

L'algorithme ci-dessus correspond au programme en langage C suivant:

```
TRISCbits.TRISC3 = 0;       // SCL en sortie
TRISCbits.TRISC4 = 0;       // SDA en sortie
SSPCON1 = 0x28;             // SSPEN = 1 + Master mode
SSPADD = 24;                // Fréquence 100KHz
SSPSTATbits.SMP = 1;        // slew rate ctrl disabled

//Initialisation du capteur
ecrire_donnee_i2c(0x90,0xAC,0x02);

while (1)
{
//demande d'échantillon de température
ecrire_command_i2c(0x90,0xEE);

//Lecture de la température
Temp = lecture_donnee_i2c(0x90,0xAA) ;

Affichage_LCD(Temp) ;
}

return;
```

Sur le code ci-dessus, on constate qu'un certain nombre de fonctions a été défini pour la gestion du protocole I2C. A titre d'exemple, voici l'algorithme ainsi que la définition de la fonction permettant d'envoyer une commande à l'esclave (capteur) :
Algorithme de la fonction :

ecrire_command_i2c(adresse, command)
Générer un « START »
** Placer le bit SEN du registre SSPCON2 à '1'*
** Tant que SEN = '1' : **Attendre***
Envoyer l'adresse du circuit concerné
** Charger le registre de transmission SSPBUF avec l'adresse du circuit*
** Valider la transmission I2C*
Envoyer la commande du CI concerné
** Charger le registre de transmission SSPBUF avec la commande du circuit*
** Valider la transmission I2C*
Générer un « STOP »
** Placer le bit PEN du registre SSPCON2 à '1'*
** Tant que PEN = '1' : **Attendre***

Définition de la fonction

```
void ecrire_command_i2c(char adresse, char command)
{
        SSPCON2bits.SEN = 1;        //Start
        while(SSPCON2bits.SEN!=0);
        SSPBUF = adresse;  //Adresse capteur
        valid_I2C();
        SSPBUF = command;  //init du capteur
        valid_I2C();
        SSPCON2bits.PEN = 1;        //Stop
        while(SSPCON2bits.PEN!=0);
        return ;
}
```

Avec la fonction de validation (attente par le maître de l'acquittement de l'esclave) **`valid_I2C( )`** définie comme suit :
Algorithme de la fonction

valid_I2C( )
** Tant que le transfert en écriture ou en lecture n'est pas terminé (R_W): **Attendre***
** Tant que l'acquittement n'est pas constaté (ACKSTAT) : **Attendre***
Définition de la fonction

```
void valid_I2C(void)
{
        while(SSPSTATbits.R_W != 0);
        while(SSPCON2bits.ACKSTAT != 0);
}
```

Les fonctions permettant d'écrire une donnée ou de lire une donnée sont les suivantes:
Algorithme de la fonction :

ecrire_donnee_i2c(adresse, command, donnee)
Générer un « START »
** Placer le bit SEN du registre SSPCON2 à '1'*
** Tant que SEN = '1' : **Attendre***
Envoyer l'adresse du circuit concerné
** Charger le registre de transmission SSPBUF avec l'adresse du circuit*
** Valider la transmission I2C*
Envoyer la commande du CI concerné
** Charger le registre de transmission SSPBUF avec la commande du circuit*
** Valider la transmission I2C*
Envoyer la donnée du CI concerné
** Charger le registre de transmission SSPBUF avec la donnée du circuit*
** Valider la transmission I2C*
Générer un « STOP »
** Placer le bit PEN du registre SSPCON2 à '1'*
** Tant que PEN = '1' : **Attendre***

Définition de la fonction

```
void  ecrire_donnee_i2c(char  adresse,  char  command,
char donnee)
{
        SSPCON2bits.SEN = 1;        //Start
        while(SSPCON2bits.SEN!=0);
        SSPBUF = adresse;  //Adresse du capteur
        valid_I2C();
        SSPBUF = command;  //Commande
        valid_I2C();

        SSPBUF = donnee;            //Donnée
        valid_I2C();
        SSPCON2bits.PEN = 1;        //Stop
        while(SSPCON2bits.PEN!=0);
}
```

Algorithme de la fonction :

lecture_donnee_i2c(adresse, command)
Générer un « START »
** Placer le bit SEN du registre SSPCON2 à '1'*

** Tant que SEN = '1' : **Attendre***
Envoyer l'adresse du circuit concerné
** Charger le registre de transmission SSPBUF avec l'adresse du circuit*
** Valider la transmission I2C*
Envoyer la commande de lecture du CI concerné
** Charger le registre de transmission SSPBUF avec la commande du circuit*
** Valider la transmission I2C*
Générer un « STOP »
** Placer le bit PEN du registre SSPCON2 à '1'*
** Tant que PEN = '1' : **Attendre***
Générer un « START »
** Placer le bit SEN du registre SSPCON2 à '1'*
** Tant que SEN = '1' : **Attendre***
Envoyer l'adresse (en mode lecture) du circuit concerné
** Charger le registre de transmission SSPBUF avec l'adresse+1 du circuit*
** Valider la transmission I2C*
Attendre une réception
** Placer le bit RCEN du registre SSPCON2 à '1'*
** Tant que RCEN = '1' : **Attendre***
** Lire la donnée présente dans le registre de transmission SSPBUF*
** Valider la transmission I2C*
Générer un « STOP »
** Placer le bit PEN du registre SSPCON2 à '1'*
** Tant que PEN = '1' : **Attendre***
Retourner la donnée lue

Définition de la fonction

```
char lecture_donnee_i2c(char adresse, char command)
{
        char donnee = 0x00;
        SSPCON2bits.SEN = 1;//Start
        while(SSPCON2bits.SEN!=0);
        SSPBUF = adresse; //Adresse du capteur en
mode écriture
        valid_I2C();
        SSPBUF = command; //Commande de lecture
        valid_I2C();
        SSPCON2bits.PEN = 1;//Stop
        while(SSPCON2bits.PEN!=0);
        SSPCON2bits.SEN = 1;//Start
        while(SSPCON2bits.SEN!=0);
        SSPBUF = adresse+1;//Adresse du capteur en
mode lecture
        valid_I2C();
        SSPCON2bits.RCEN = 1;//Attente réception
        while(SSPCON2bits.RCEN!=0);
        donnee = SSPBUF;//lecture buffer de réception
        valid_I2C();
        SSPCON2bits.PEN = 1;//Stop
        while(SSPCON2bits.PEN!=0);
        return(donnee);
}
```

8 RETOUR D'EXPERIENCE

A l'issue de cette séance de travaux pratiques, nous avons interrogé les étudiants au moyen de la plate-forme pédagogique en ligne Claroline. Il s'agit d'un sondage anonyme effectué auprès de 34 binômes. Les résultats (Fig. 8 à 11) montrent que les étudiants ont été majoritairement (79.4%) intéressés ou très intéressés par ce TP, mais que plus de la moitié (58.8%) regrettent de n'avoir pas eu assez de temps pour le terminer. Comme l'on pouvait s'y attendre, le fait qu'il s'agisse d'une application concrète (mesure de température) a séduit 88.2% des étudiants. Enfin 82.3% des étudiants estiment avoir augmenté leurs connaissances grâce à ce TP. Ces résultats nous encouragent à envisager une préparation de ce TP par un TD préalable. En outre, il nous conforte dans notre volonté de donner aux étudiants des exemples concrets à étudier.

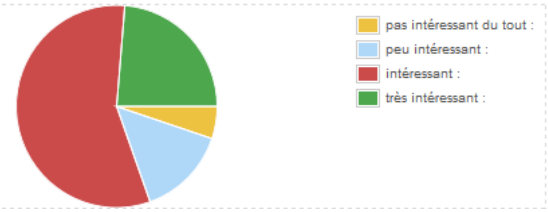


Fig. 8. Réponses à la question : "Avez-vous trouvé ce TP intéressant?"

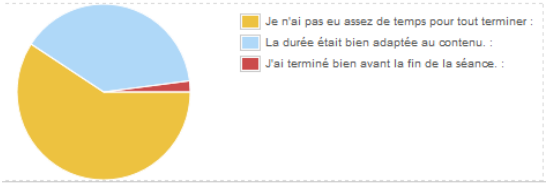


Fig. 9. Réponses à la question : "Pensez-vous que la durée de ce TP soit adaptée à son contenu?"

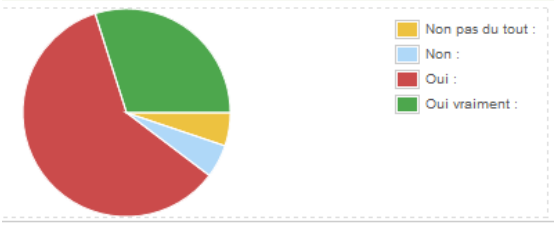


Fig. 10. Réponses à la question : "Etes-vous satisfait(e) d'avoir travaillé sur une application concrète pour étudier le bus I2C"

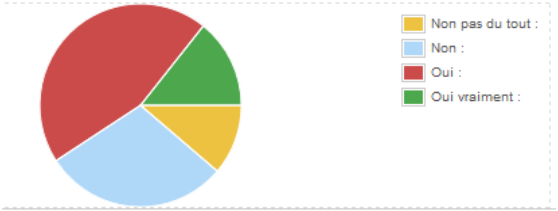


Fig. 11. Réponses à la question : "Avez-vous l'impression d'avoir augmenté vos connaissances avec ce TP?"

9 CONCLUSION

Nous avons présenté une séance de Travaux Pratiques consacrée à la mise en œuvre d'un bus I2C entre un microcontrôleur PIC et un capteur de température la suite de ce TP étant la gestion de l'afficheur LCD du Kit afin de pouvoir lire la

température. Plusieurs autres extensions de ce TP sont possibles. Ainsi, on peut envisager de lire la température sur 16 bits ou avec une résolution supérieure (voir datasheet), d'envoyer la valeur lue vers un autre périphérique en utilisant l'USART du PIC (pour une liaison RS232 avec un PC par exemple), d'assurer une régulation de température. A la suite de ce travail, on peut également étudier le bus SPI dont le protocole est très proche de celui du bus I2C, ou encore, lors de séances d'étude et réalisation, expérimenter la communication I2C entre le PIC et un autre composant, par exemple une EEPROM. A cette fin, la maquette présentée dans cet article est également équipée d'une mémoire Microchip 24LC256 et d'un CNA MAX517 communiquant par bus I2C. Enfin, Il et à noter que l'étude du bus I2C est une bonne approche avant l'étude de bus plus complexes comme le bus CAN.

10 REFERENCES

[1] Datasheet du PIC 18F452, Microchip, http://ww1.microchip.com/downloads/en/devicedoc/39564c.pdf

[2] Téléchargement de MPLAB X www.microchip.com/mplabx

[3] Datasheet du capteur de température DS1621, Maxim Integrated, http://datasheets.maximintegrated.com/en/ds/DS1621.pdf

[4] NXP, UM10204, I2C-bus specification and user manual, rev 5, oct 2012, http://www.nxp.com/documents/user_manual/UM10204.pdf