Introduction

Dans le cadre des "Samedis Techniques", le Radio club de la Haute Île, F5KFF-F6KGL présente une solution simple et accessible à tous pour comprendre et mettre en œuvre la technique des microcontrôleurs PIC en général et le PIC18F25K20 en particulier. Cet article est la synthèse de ce qui a été présenté par Vladimir F4FNA lors de la réunion du 11/12/10.

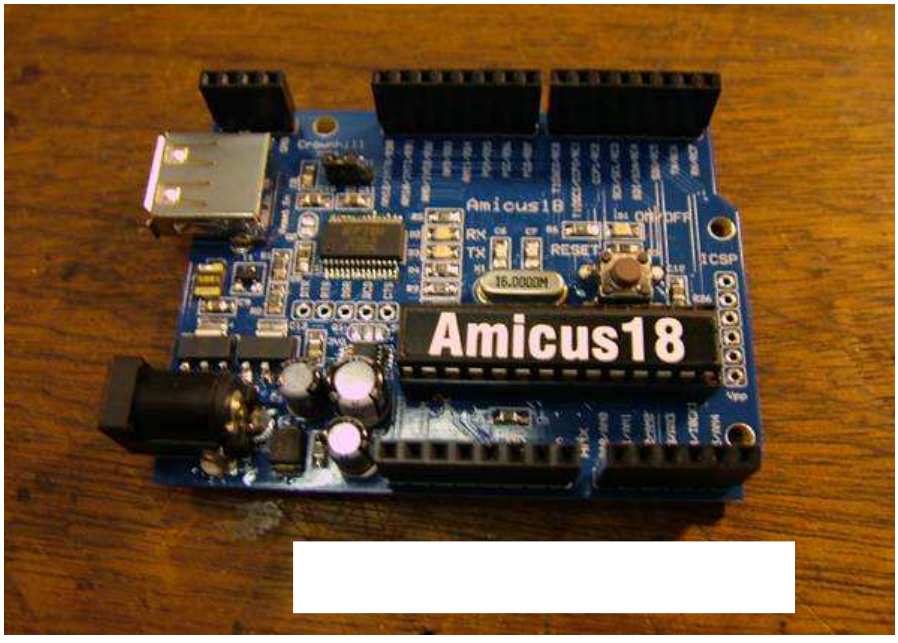
1) présentation de la platine AMICUS 18

Les microcontrôleurs sont des circuits intégrés programmables. Le PIC18F25K20 dispose d'une mémoire programmable de type flash, ce qui lui permet d'être écrite et effacée environ 10000 fois, soit l'équivalent d'une programmation quotidienne durant 30 ans. Le programme mémorisé dans le PIC est une description des actions à effectuer par celui-ci selon des conditions définies ou des états des entrées. Pour écrire ce programme on doit :

- Connaître un langage de programmation
- Avoir un environnement de développement
- Avoir un compilateur
- Avoir un projet et ... du temps (! ! !)

Actuellement, il existe des solutions très abordables pour commencer à développer des programmes pour microcontrôleurs. L'une des ces solutions se nomme Amicus18 et c'est l'objet de cet article

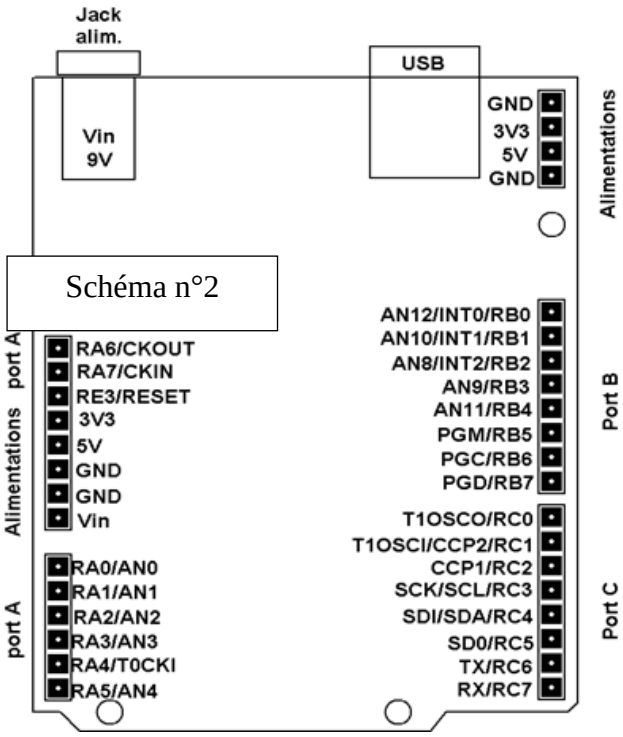
Amicus18 est un système de développement en langage Basic pour un circuit PIC18F25K20. Ce système complet est composé d'un environnement de développement intégré (IDE) gratuit, d'un compilateur Basic très performant et gratuit, le tout accompagné d'une documentation complète téléchargeable gratuitement.



Coté hardware, une platine de développement avec le circuit PIC18F25K20 cadencé à 16MHz avec ses connecteurs d'entrées/sorties, une interface USB et un régulateur d'alimentation.

La platine possède également, un connecteur pour PicKit2, le programmateur /débogueur fabriqué par Microchip (ce module est illustré par la photo n°1).

Les entrées /sorties du PIC sont groupées sur 3 ports identifiables par une simple lettre (A, B, C). Un port, peut avoir au maximum 8 bits identifiables à leur tour par un chiffre de 0 à 7. Si bien que les 24 pins des ports de la platine sont notées de RA0 à RC7 et sont toutes reliées à un connecteur .Une autre caractéristique est le fait que chaque pin du PIC peut avoir plusieurs fonctions possibles selon ce qui a été défini dans le programme.



De plus, la platine Amicus18 possède deux alimentations : +3,3V, +5V générées à partir d'une tension variable de +9V à +12V injectée sur la prise Jack 2,1mm.

Tous les connecteurs se présentent sous forme d'embases au pas de 2,54 mm (1 pouce), ce qui nous donne la possibilité d'enficher des platines avec nos applications spécifiques.

Le schéma 2 représente l'ensemble des connexions disponibles sur la platine

Le minimum de matériel pour démarrer les montages décrits ci-après est constitué de la platine Amicus 18 et d'un câble USB/USB qui la relie au PC. Il faut d'abord installer les logiciels (gratuits). Bien sûr, il faut un PC assez récent, avec Windows XP ®, Vista ® ou Win7 ®.

Pour information, la société Farnell France vous propose ce microcontrôleur pour le prix TTC de 46,25 € + frais de port (http://fr.farnell.com/amicus/amicus18/board-pic18f25k20-proton-amicus18/dp/1818281). D'autres revendeurs proposent aussi ce circuit. Par exemple : http://www.picmicro.nl/buy_now.html. En tapant "Amicus 18" sur votre moteur de recherche favori, il n'y a que l'embarras du choix...

2) téléchargement de la partie logiciel

Côté logiciel, il n'y a plus besoin de sortir sa carte bancaire : tout est disponible gratuitement sur Internet. Le site de Microchip (http://www.microchip.com/) permet de télécharger la dernière version de l'environnement de programmation baptisé MPLAB IDE (http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469#P171_5058). L'image n°3 est la copie d'écran de cette page. Si vous avez déjà installé MPLAB IDE, la version 8.54 ou supérieure vous sera nécessaire. Le lien de téléchargement est au bas de la page. Attention, cette installation prend du temps. On télécharge le fichier MPLAB_IDE_v0_00.zip (0_00 indique la version du fichier qui pèse environ 100 Mo), il faut décompresser le fichier puis l'installer en cliquant sur Setup.exe

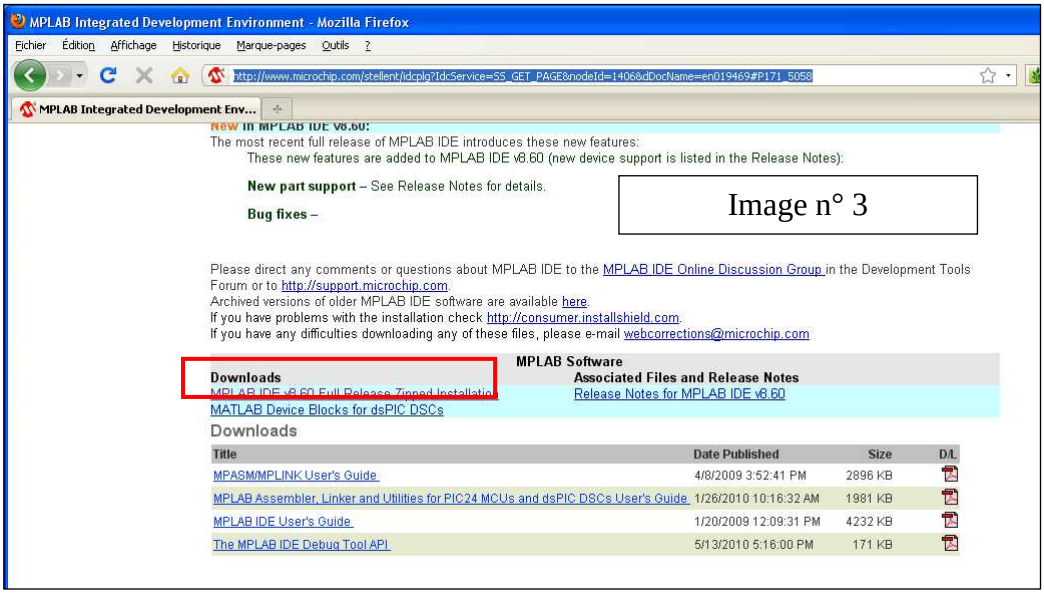
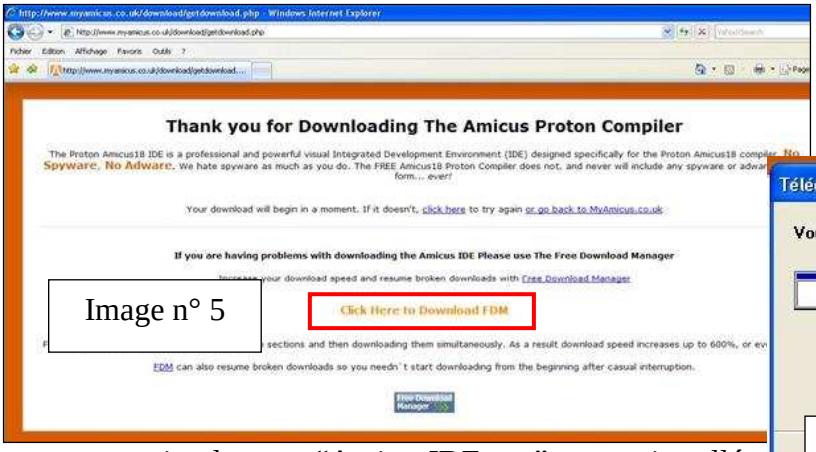


Image n° 3

Ensuite, aller sur la page http://www.myamicus.co.uk/content.php?245-Free-AMICUS18-Compiler cliquer sur "Download", la page de l'image n°4 s'affiche. Cochez la case "I have downloaded and installed MPLAB IDE" (voir image n°4)

La page suivante apparait : elle vous propose de télécharger "Free Download Manager" (FDM) (voir image n° 5) si vous le souhaitez (pour accélérer les téléchargements sur certains sites, si vous le souhaitez. Attention, FDM ne fonctionne qu'avec Internet Explorer mais pas avec les autres navigateurs. Après avoir cliqué sur "I have downloaded and installed MPLAB IDE" (image n° 4), le téléchargement démarre, cliquer sur "Executer" ou "Enregistrer" puis installer le ficher Amicus18 Compiler Setup.exe (image n° 6). Le logiciel

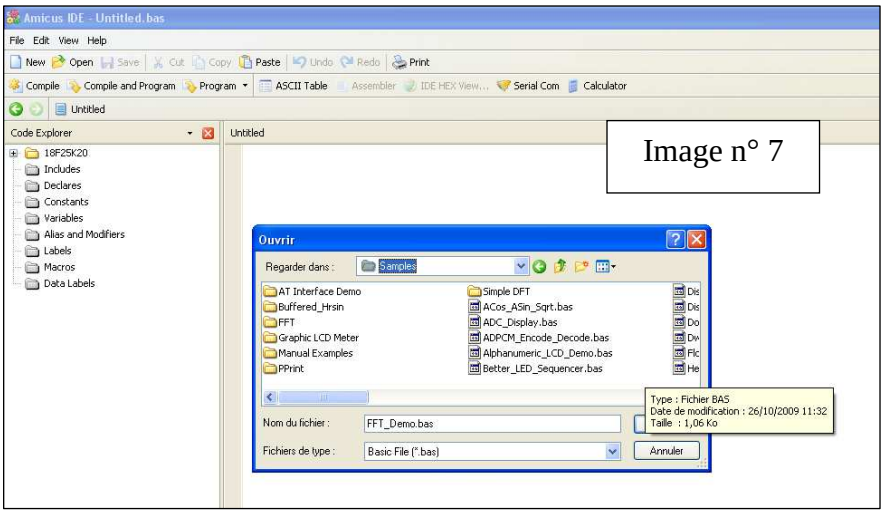


se nomme simplement "AmicusIDE.exe" et est installé par défaut dans "C:\Program Files\AmicusIDE"

3) le langage BASIC pour PIC

Le langage BASIC est un de plus répandu et des plus connu des langages de programmation. Écrit en anglais, les commandes sont aussi dans la même langue. Mais cela reste d'un niveau compréhensible par tous.

Pour le microcontrôleur Amicus18 le langage "Proton Amicus18 BASIC Compiler Language" est très complet avec 133 commandes différentes pour les opérations d'entrées/sortie, de calcul mathématique, d'opérations logiques et bien sûr analogiques.



Une documentation complète (en Anglais) est téléchargeable sur le site officiel d'Amicus18.

Le logiciel Amicus IDE est livré avec une large bibliothèque d'exemples

documentés en anglais (en cliquant simplement sur “File” puis “Open” et “Samples” (image n° 7)

Pour commencer, réalisons quelque chose de simple :

3-1) Exemple n°1 : le classique "Allumer un LED"

Pour la bonne réussite d'un projet, on doit définir ce que l’on doit faire, quel ports employer, etc. Puis, on dessine un schéma. Ensuite on commence à écrire le programme.

Dans notre exemple, on doit allumer un LED pendant une seconde, puis l’éteindre.

On choisit par exemple le Port B, bit 0 sur lequel on relie la cathode de la LED au travers d’une résistance de 2 kΩ pour limiter le courant. L'anode de la LED est relié au +5V (voir image n° 8)

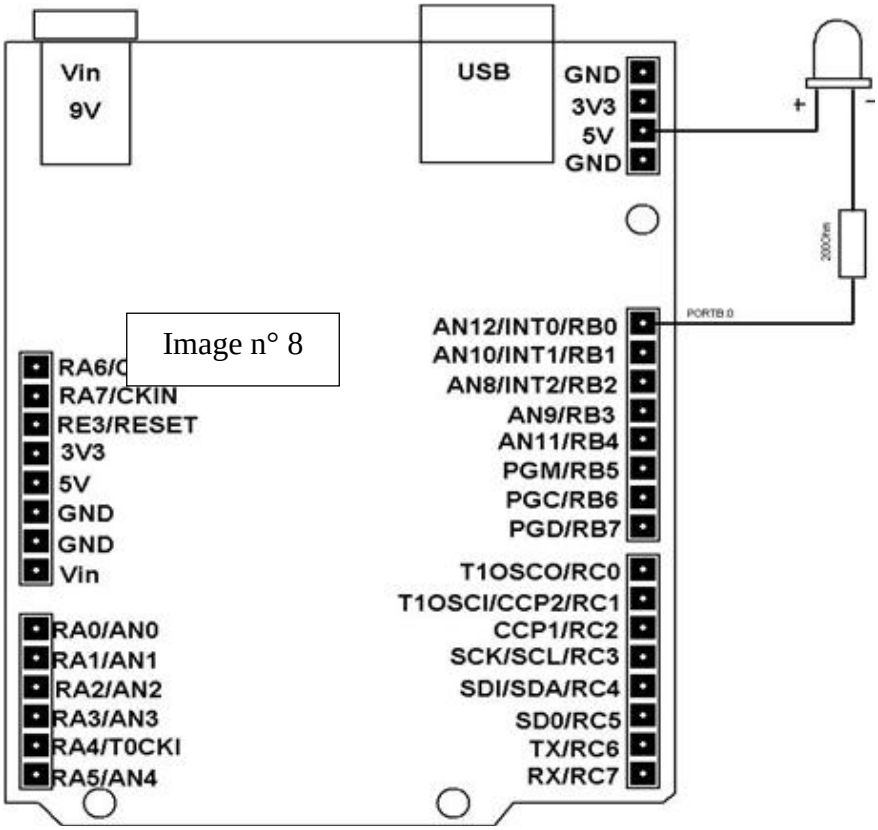


Image n° 8

Tout d'abord on doit dire au compilateur la signification des sorties. Il est fortement conseillé de le faire pour une bonne lisibilité du programme :

```
Symbol Sortie_LED = PORTB.0
```

Ensuite on va définir la variable “t_1” pour l'intervalle de temps :

```
Dim t_1 As Word
```

On va lui donner une valeur en ms :

```
t_1=1000
```

Et on écrit les actions à faire : mettre sur le port l’état logique bas (0V). Alors, la cathode de la LED étant reliée au port B.0 et l'anode étant au +5V, la LED s'allume.

```
Low Sortie_LED
```

On fait attendre l'intervalle de temps défini plus haut :

```
DelayMS t_1
```

On va mettre l'état logique haut sur le port (+5V) et donc sur la cathode de la LED qui s’éteindra :

```
High Sortie_LED
```

Et on finit le programme :

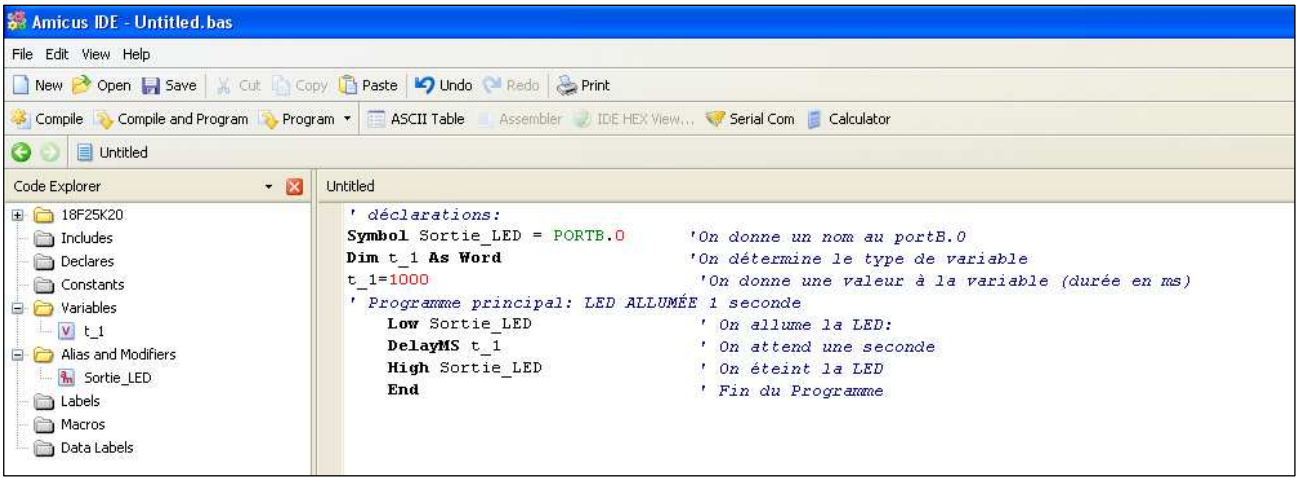
```
End
```

Enfin, on peut écrire notre programme d'une manière cohérente :

```
' déclarations:
Symbol Sortie_LED = PORTB.0   'On donne un nom au port B.0
```

```
Dim t_1 As Word                 'On détermine le type de variable pour t_1
t_1=1000                        'On donne une valeur à la variable t_1 (durée en ms que l'on peut modifier si on le souhaite)
' Programme principal: LED ALLUMÉE 1 seconde
      Low Sortie_LED     ' On allume la LED (le port B.0 est mis au niveau bas)
      DelayMS t_1        ' On attend ou 1000 ms, soit une seconde
      High Sortie_LED    ' On éteint la LED (le port B.0 est mis au niveau haut)
      End                ' Fin du Programme
```

Ce qui donne l'écran suivant sur Amicus IDE (image n° 9) :



On peut remarquer que dans le texte du programme, l'écriture a des couleurs différentes. Tout écriture précédée d'un apostrophe ou d'un point virgule est un commentaire (ici en bleu et en italique) et n'est pas prise en compte par le compilateur. Elle sert à nous rappeler ce qu'on a voulu faire (et croyez-moi, après 6 mois, dans des programmes de plus des 50 lignes on se demande pourquoi on a écrit ça...). Plus on écrit de commentaires, moins on perd du temps au débogage (erreur dans la programmation du logiciel, ce qui le fait "planter" ou qui lui fait faire une action non souhaitée et encore moins prévue). De plus, à gauche (zone Code Explorer), un récapitulatif des différentes variables utilisées dans le programme est affiché, ce qui facilite grandement les modifications (un clic sur ces variables renvoie automatiquement aux déclarations du programme (dans la partie droite de l'écran)

Une fois le programme écrit, un simple clic sur "Compile and Program" permet de vérifier que le programme est sans erreur et le sauvegarde en basic (extension du fichier = .bas) puis génère un fichier binaire (extension du fichier = .hex) qui est transféré vers le microcontrôleur via le câble USB. Le fait de transférer le programme dans le microcontrôleur génère l'action immédiatement (voir image n° 10).

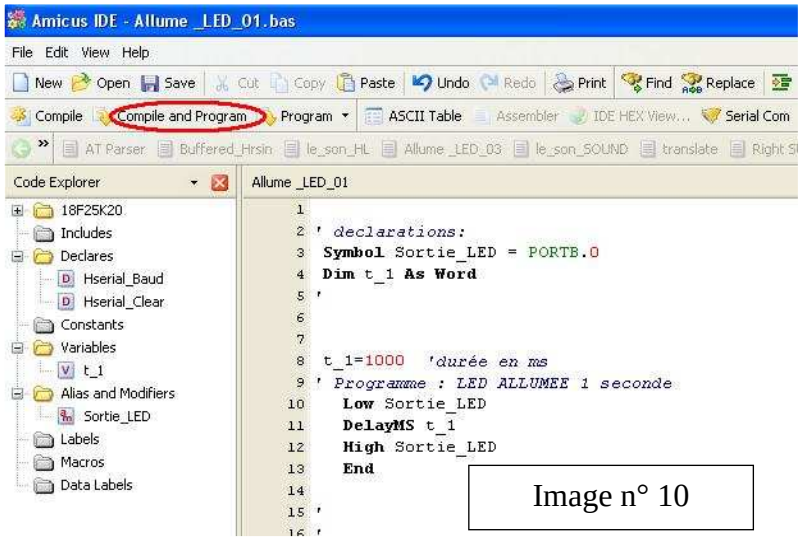


Image n° 10

Le bouton "Compile" permet de compiler (et de vérifier) sans programmer et le bouton "Program" génère et transfère le fichier binaire seulement si le programme a déjà été compilé. L'avantage de décomposer ces commandes devient évident quand on a des

programmes de grande taille : le débogage (recherche d'erreurs de programmation) prend moins de temps. Les exemples n° 2 et 3 mettent en évidence le fait que, sans modifier la partie hardware, on donne de nouvelles fonctionnalités en changeant uniquement la programmation.

3-2) Exemple n°2 : la diode clignote 10 fois penda nt 1 seconde

```
' déclarations:
 Symbol Sortie_LED = PORTB.0 'On donne un nom au port B.0
 Dim t_1 As Word             'On détermine le type de variable
 Dim i As Byte               'On rajoute une nouvelle variable (i)
  t_1=1000                   ' On donne une valeur à la variable t_1
' Programme : LED ALLUMEE pendant 1 seconde 10 fois
   For i = 0 To 9            ' pour dix cycles (numérotés de 0 à 9)
   Low Sortie_LED            ' On allume la LED
   DelayMS t_1               ' On attend 1 seconde
   High Sortie_LED           ' On éteint la LED
   DelayMS t_1               ' On attend 1 seconde
   Next                      ' On passe au cycle suivant en ajoutant 1 à i
   End                       ' Après dix cycles… La Fin
```

Le cycle du programme se répète dix fois entre la commande "For" et la commande "Next"

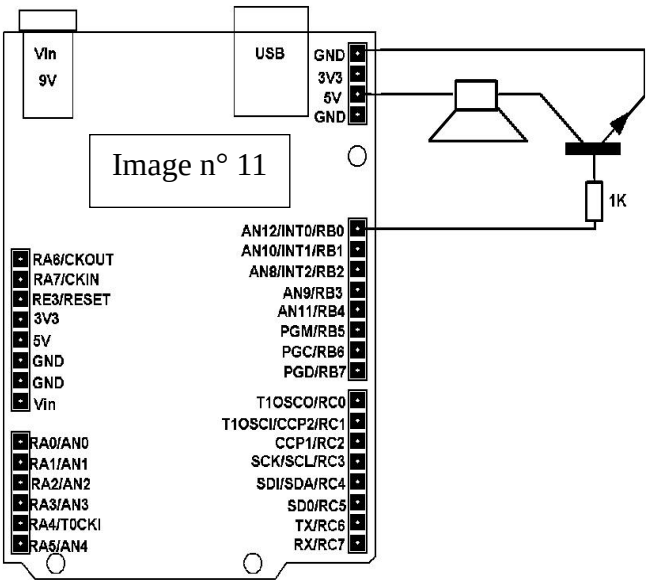
3-3) Exemple n°3 : la diode clignote 10 fois avec u ne durée de plus en plus longue

```
' déclarations:
Symbol Sortie_LED = PORTB.0 'On donne un nom au port B.0
 Dim t_1 As Word            'On détermine le type de variable
' Programme : LED ALLUMEE  10 fois  pour période de plus en plus longues
   For t_1 = 1 To 1000 Step 100  'pour 10 cycles de plus en plus longs (de 1 à
1000 en ajoutant 100 à chaque cycle)
   Low Sortie_LED                ' On allume la LED
   DelayMS t_1                   ' On attend 1 durée variable selon le cycle (1
ms à 1 seconde)
   High Sortie_LED               ' On éteint la LED
   Délayas 1000                  ' On attend 1 seconde
   Net                          ' On passe au cycle suivant
   End                           ' Après dix cycles... La Fin
```

Donc, de point de vue hardware, dans le premier exemple on peut remplacer le microcontrôleur par un transistor, un condensateur chimique et quelques résistances, pour le deuxième il faut mettre au moins 2 transistors et pour le troisième il nous faut beaucoup plus que ça... Avec notre microcontrôleur, en changeant quelques lignes du programme et avec le même schéma, on met en œuvre de nouvelles fonctionnalités. Ceci dit, il vaut mieux faire chauffer les méninges que le fer à souder!

3-4) Générer un son de 1000 Hz pendant 1 seconde.

Pour générer du son, il faut déplacer l'air à la fréquence désirée, le dispositif le plus répandu étant le haut-parleur. Sa bobine se déplace selon l'intensité du courant qui la traverse et le sens du dit courant. Pour le moment, il est suffisant d'appliquer à la bobine du haut parleur un signal carré pour produire cette vibration. Dans les exemples qui suivent, on va explorer les possibilités mises à notre disposition par le langage Basic.



Le schéma doit être modifié : on emploie toujours la sortie du port B.0 que nous connaissons bien maintenant. Un transistor est nécessaire pour commander le haut-parleur. On relie sa base à la sortie d'Amicus 18 au travers d'une résistance de 1 kΩ (image n° 11). Nous utiliserons les tensions disponibles sur la platine Amicus 18 pour alimenter le transistor.

On doit prendre soin que la sortie de l'Amicus18 soit à un niveau bas (0V) au repos pour éviter une consommation inutile du transistor.

Le programme va commander le niveau du port B.0 par les commandes High et Low que nous avons vues précédemment. Dans les exemples précédents, nous avions utilisé la commande DelayMS pour déterminer un temps une milliseconde, nous allons maintenant utiliser la commande DelayUS pour déterminer un temps en microsecondes :

```
' déclarations:
 Symbol Sortie_son = PORTB.0
  Dim i As Word
' Programme : son d'une seconde à 1000 Hz
    Low Sortie_son          'on bloque le transistor
    For i=0 To 1000         'pour 1000 cycles de 1mS
    High Sortie_son         'Le courent passe
    DelayUS 500             'pendant 500 µs
    Low  Sortie_son         'on bloque le transistor
    DelayUS 500             'on attend 500 µs
    Next                    'on passe au cycle suivant
    Low Sortie_son          'on bloque le transistor
    End                     'Fin du programme
```

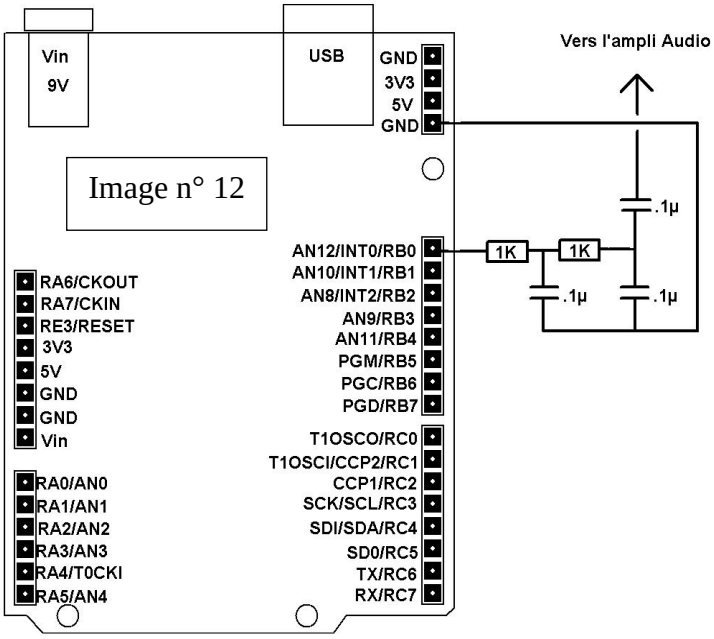
Si on veut changer la fréquence, on doit modifier les chiffres après "DelayUS", de manière à avoir une demi-période en µs. Par exemple, pour une fréquence de 2.000 Hz, la période dure 500 µs et une demi-période (ou une alternance) dure 250 µs. Si bien que la valeur DelayUS devient 250.

Mais le Langage Basic met à notre disposition une autre commande pour générer des sons avec une forme d'onde rectangulaire : c'est la commande "Sound". Le programme se simplifie et devient :

```
' déclarations:
 Symbol Sortie_son = PORTB.0    'Sortie son
 Dim ton As Byte                'Variable Tonalité
```

```
 Dim t_1 As Byte                  'Variable Durée
 ton=118                          'valeur de la tonalité pour environ 1000 Hz
 t_1=100                          'durée = t_1 X 10ms
' Programme : son d'une seconde à 1000 Hz
   Low Sortie_son                 'on bloque le transistor
   Sound Sortie_son, [ton,t_1]    'on génère le son
   Low Sortie_son                 'on bloque le transistor
   End                            'Fin du programme
```

Ce type de son produit beaucoup d'harmoniques (signal carré) donc de distorsions et ce n'est pas idéal pour de la modulation dans nos applications radio. Heureusement, il est possible de produire des sons avec une forme d'onde sinusoïdale. C'est ce que l'on peut voir dans l'exemple suivant avec la commande "FreqOut". En revanche, on doit changer le schéma, car on doit monter un filtre en sortie d'Amicus18 (image n° 12). Le filtre est composé de deux résistances de 1 kΩ et de trois condensateurs non polarisés de 100 nF (0,1 µF). La sortie du filtre sera branchée sur un ampli audiofréquence (enceinte active de PC par exemple) car le niveau du son est trop faible pour faire fonctionner un simple haut-parleur.



Le programme devient :

```
' déclarations:
 Symbol Sortie_son = PORTB.0      'Sortie son
 Dim ton As Word                  'variable fréquence en Hz
 Dim t_1 As Word                  'variable durée en ms
 ton=1000                         'fréquence en Hz
 t_1=1000                         'durée en ms
' Programme : son de 1000 Hz pendant une seconde
   FreqOut Sortie_son, ton,t_1    'son de 1000 Hz pendant une seconde
   End                            'Fin du programme
```

Concernant les applications pratiques spécifiques aux radioamateurs, nous présenterons dans une des séances suivantes les divers générateurs audio: 1750 Hz, 5 Tons, DTMF, etc.

(Fin de la séance n°1)

Vladimir F4FNA pour le Radio-Club de la Haute Île F5KFF/F6KGL.

Rappel : les « Samedis Techniques » ont lieu tous les 2ème samedis de chaque mois de 14h00 à 17h00 dans les locaux du Radio-Club (Port de Plaisance, 93330 Neuilly sur Marne). Toutes les informations sur notre radio-club et sur ces réunions sont disponibles sur notre site : http://f6kgl/f5kff.free.fr . Tout le monde peut participer à ces réunions. N'hésitez pas à pousser la porte de notre radio-club : vous serez toujours les bienvenus !