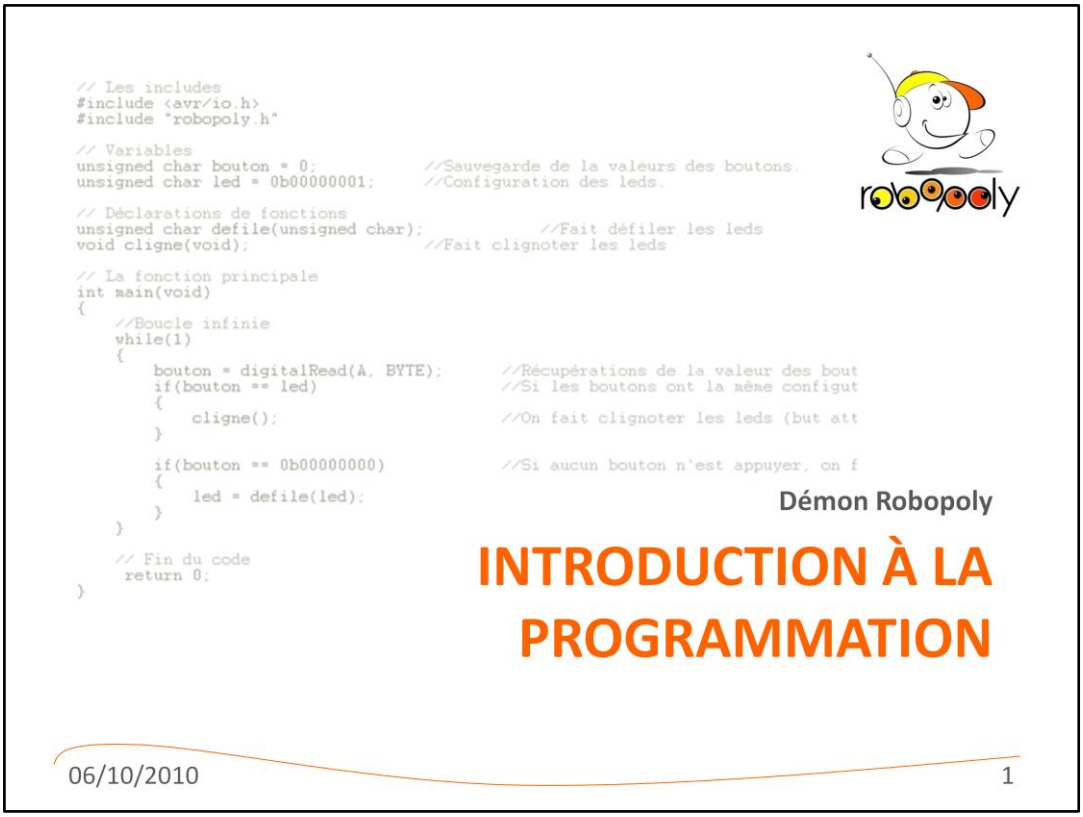
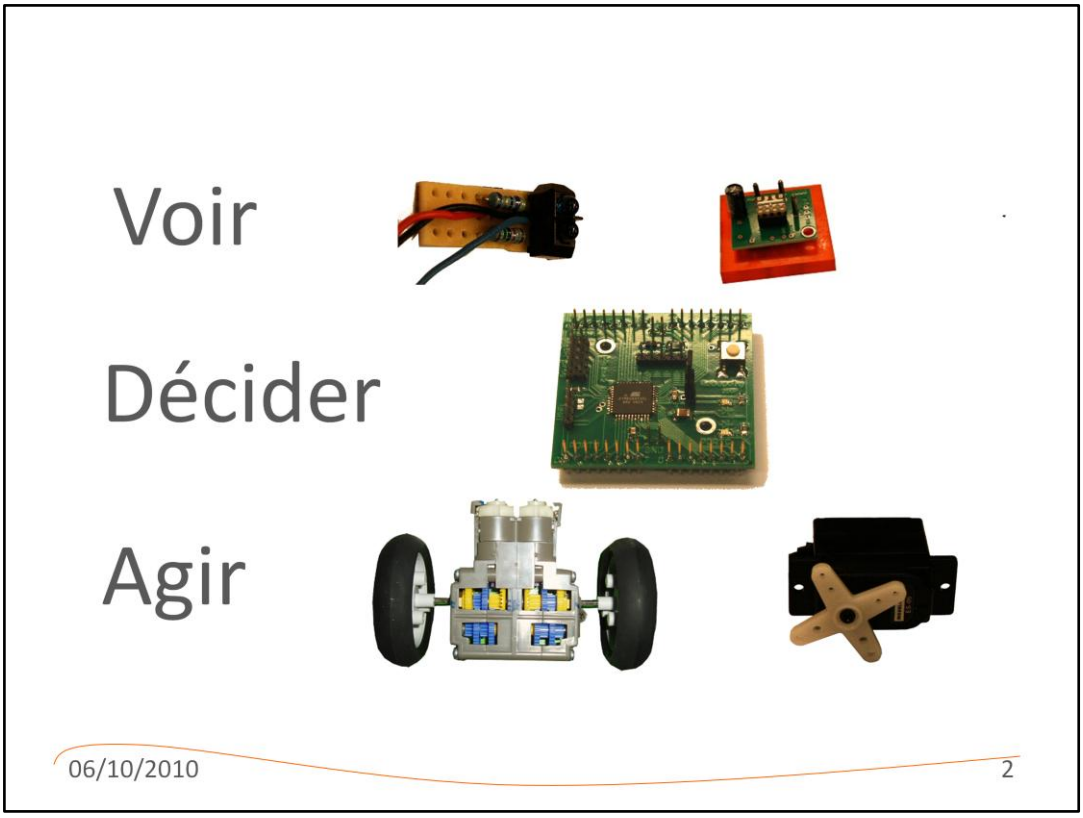


Ce démon contient des informations sur l'alimentation et donne des notions de bases sur la programmation en robotique.



Dans un robot, on a des capteurs pour voir et obtenir des informations pour ensuite agir sur l'environnement.

La programmation est l'étape qui permet de décider comment agir en fonction de ce que l'on voit.

But du démon

06/10/2010 3

A la fin du démon vous serez capables de:

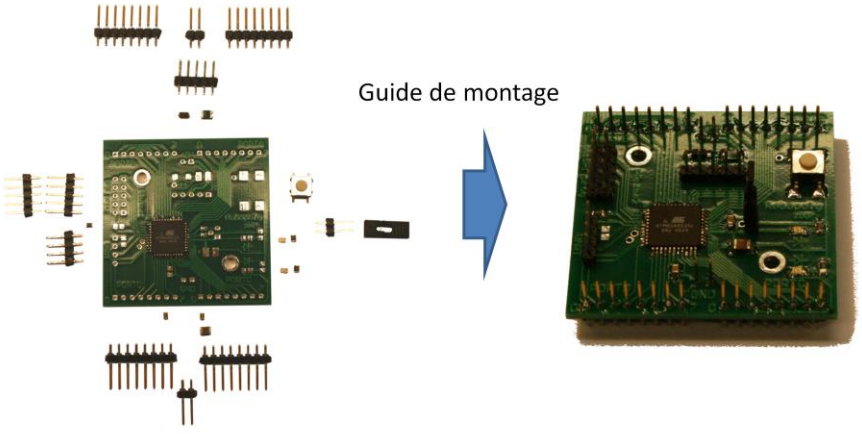
Lire des boutons, c'est-à-dire obtenir des informations de l'environnement.
Allumer des leds, ce qui correspond à agir sur l'environnement.

Démo : Lorsqu'on appuie sur un bouton, les lumières s'allument les unes après les autres

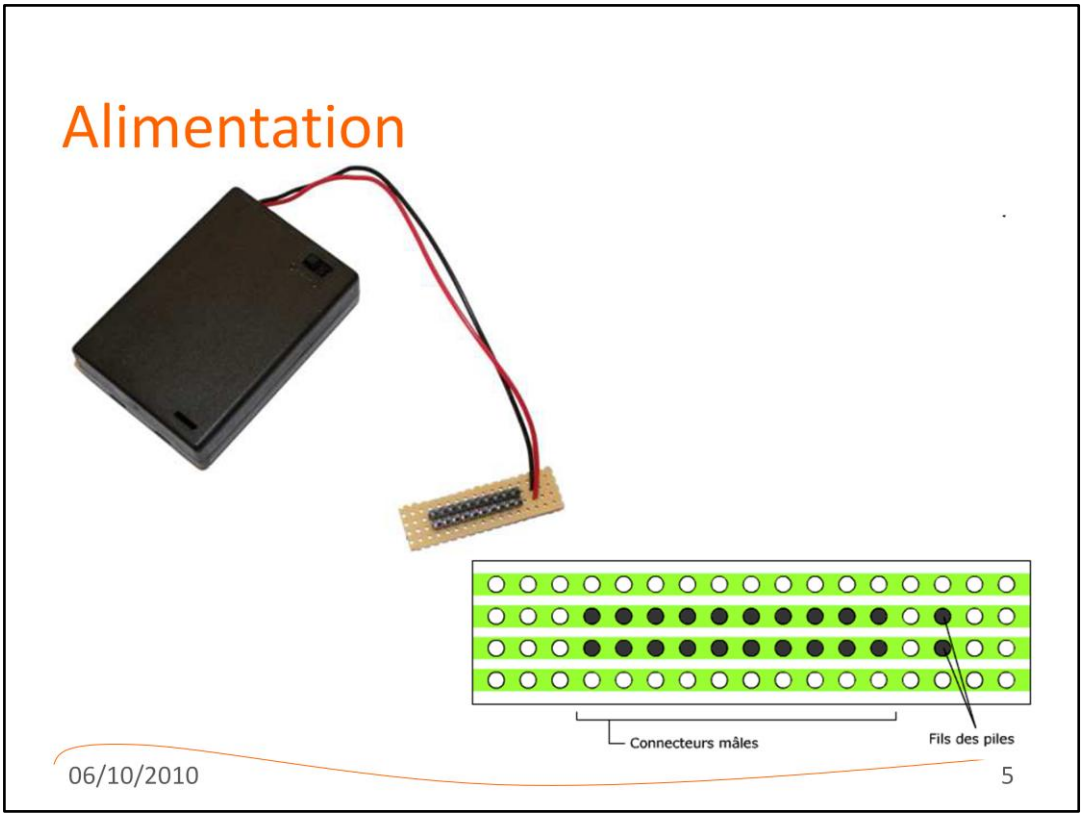
Code :

```
#include "robopoly.h"

//commentaires….

//brancher les maxileds sur le port A et les maxi boutons sur le port C

//fonction de défilement des lumières
void lumiere_defile(void)
{
	int lumiere = 0;	//déclaration de la variable lumière
	int i = 0;		//déclaration de la variable i pour la boucle for

	for(i=0;i<8;i++)	//boucle for
	{
		lumiere |=  1<<i;		//masquage : 0b00000001 -> 0b00000011 -> ... -> 0b11111111
		digitalWrite(A,BYTE,lumiere);		//écriture du lumière sur le port A
		waitms(500);				//il faut attendre sinon on voit rien!
		//on recommence jusqu'à ce que i vaut 8.
	}
}

// le main
int main()
{
	int bouton = 0;				//déclaration de la variable bouton

	while(1)
	{
		bouton = digitalRead(C,BYTE);		//lecture des boutons
		if(bouton != 0)				//si un bouton est appuyé
			lumiere_defile();		//on allume les lumières
	}
	return 0;
}
```

Rappel du démon précédent

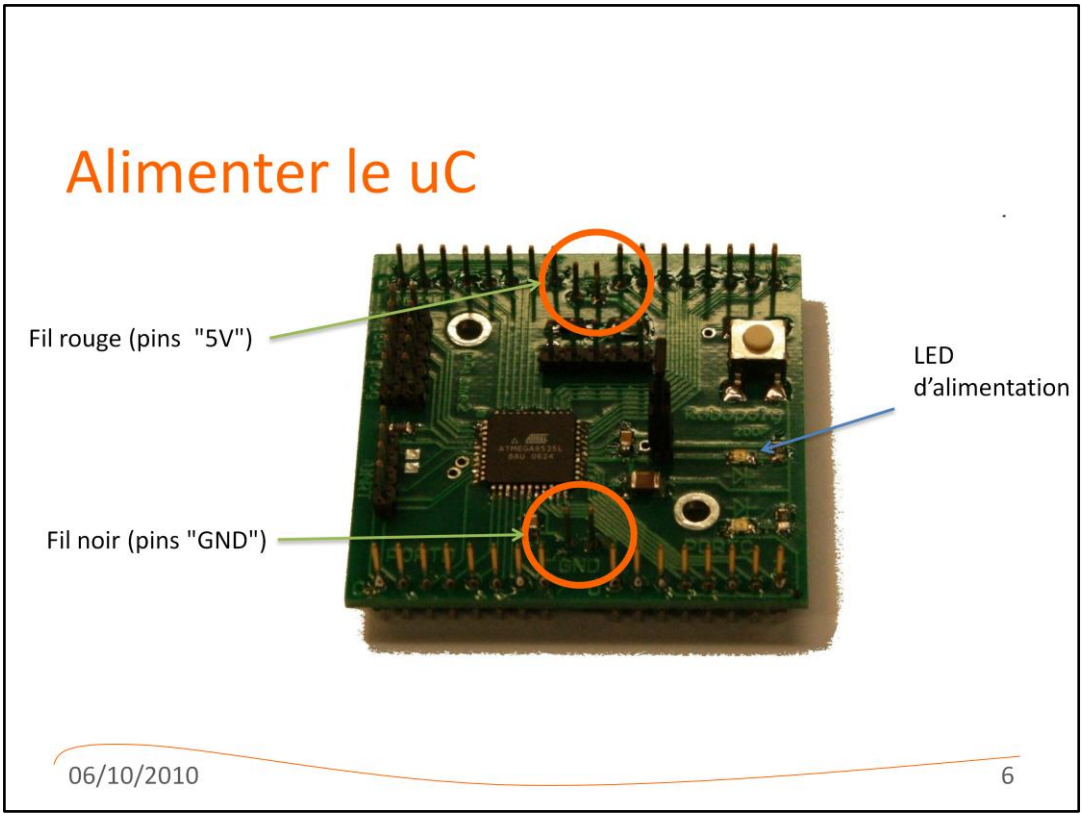


Le démon précédent explique les composants du PRisme, et donne quelques notions de soudure. Nous avons soudé le microcontroleur et ce démon apprend comment l'utiliser.



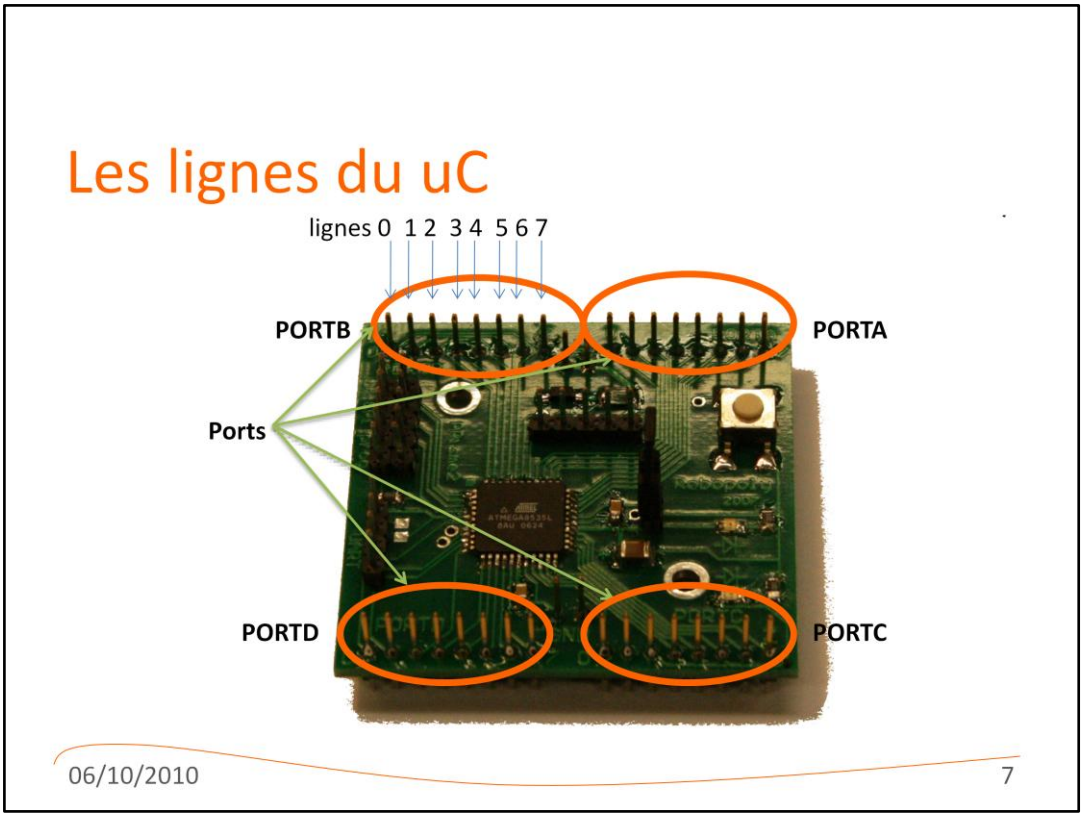
Tout d'abord, il faut alimenter le microcontroleur.
Pour cela nous avons le bloc de piles et une rampe d'alimentation.

La rampe d'alimentation est simplement deux lignes de pins, une qui est la masse, soit le – des piles (fils noirs), et l'autre la tension d'alimentation soit le + des piles (fils rouges). C'est là-dessus qu'on va brancher tous les composants pour les alimenter. La tension d'alimentation est à 3.6V (3 piles de 1.2V)



Pour alimenter notre microcontroleur, il faut le brancher à l'alimentation avec un fil rouge et un fil noir comme indiqué ci-dessus.

Une des deux leds verte devrait s'allumer pour indique que ça marche.

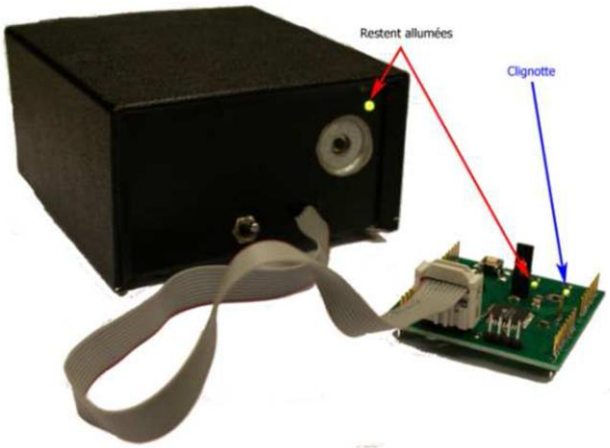


Voyons maintenant ce qu'on a disposition sur le microcontroleur :

Il y a 4 ports (A,B,C et D) composés chacun de 8 pins, numérotés de 0 à 7. Sur chacun de ces pins, on peut brancher des capteurs et des moteurs.

Dans les démos, les maxileds étaient branchées sur le port A et les maxiboutons sur le port C.

Le programmateur de table



06/10/2010 8

Avant la première utilisation, il faut flasher (mettre un code) sur le microcontroleur avec le programmateur de table. C'est un premier code qui permet par la suite de flasher à l'aide de l'ordinateur.

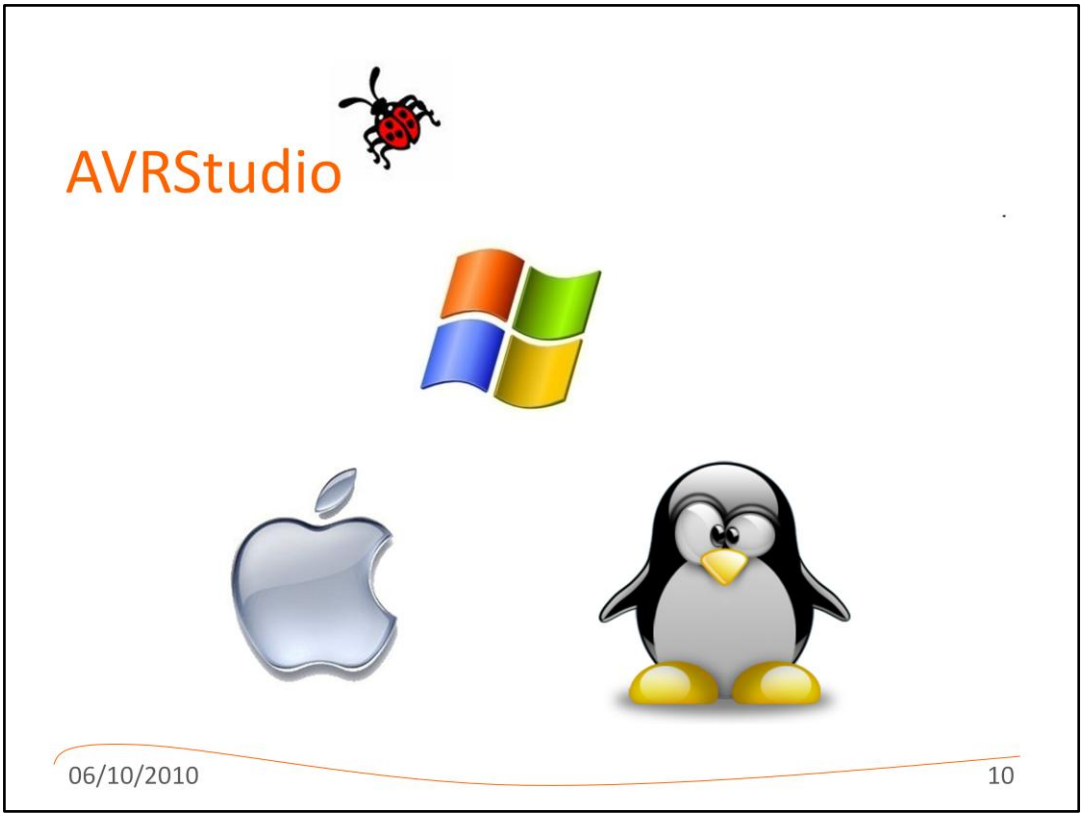
Passons à la programmation :
Tous les outils dont vous avez besoin sont sur le site robopoly.epfl.ch, sur la droite comme indiqué ci-dessus
Il faut télécharger et installer les deux programmes qui sont entourés :

AVRStudio

Et setupRobopoly.exe.

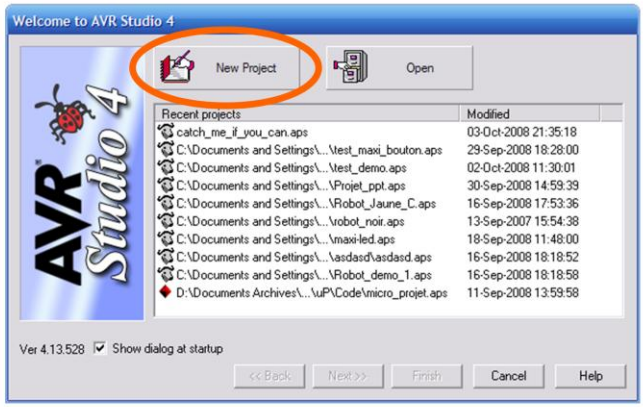
qui contient :

- WinAVR
- Plug-in Robopoly
- Driver USB
- Hterm
- Libraires Robopoly



L'éternel débat…

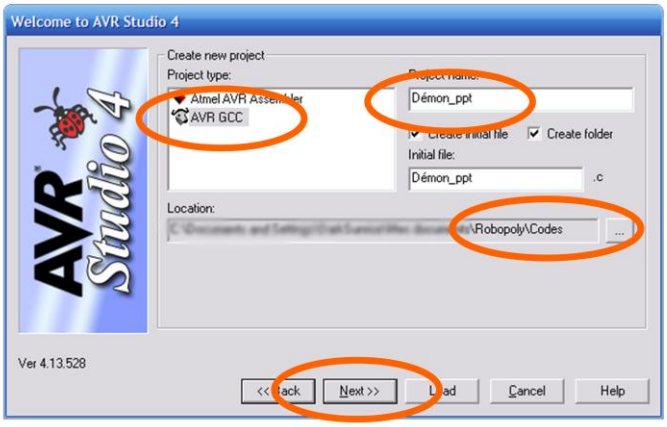
Mais plus sérieusement :
AVR Studio est l'environnement de développement que nous utilisons. Pour Windows, il n'y a pas de soucis.
Pour linux, les librairies linux sont sur le site avec toute la documention.
Pour mac, nous vous conseillons de passer par VirtualBox ou parallels.



Tutoriel pour créer un nouveau projet :

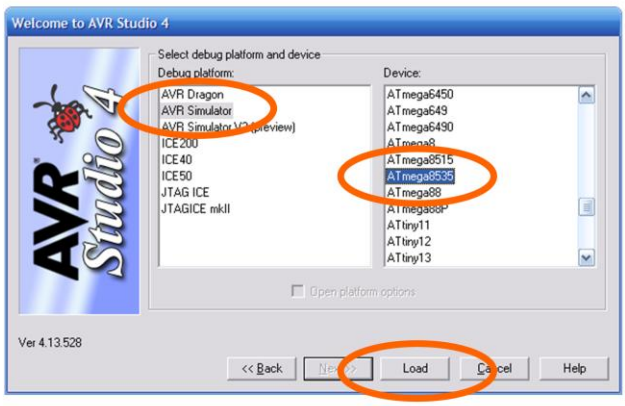
Quand on ouvre AVR studio, cette fenêtre s'affiche.

Pour créer un nouveau projet
Cliquer sur "New Project"



Sélectionner "AVR GCC"
C'est le compilateur. C'est lui qui s'occuper de transformer le code en C en langage machine.

Choisir une localisation
Donner un nom au projet
Cliquer sur "Next >>"

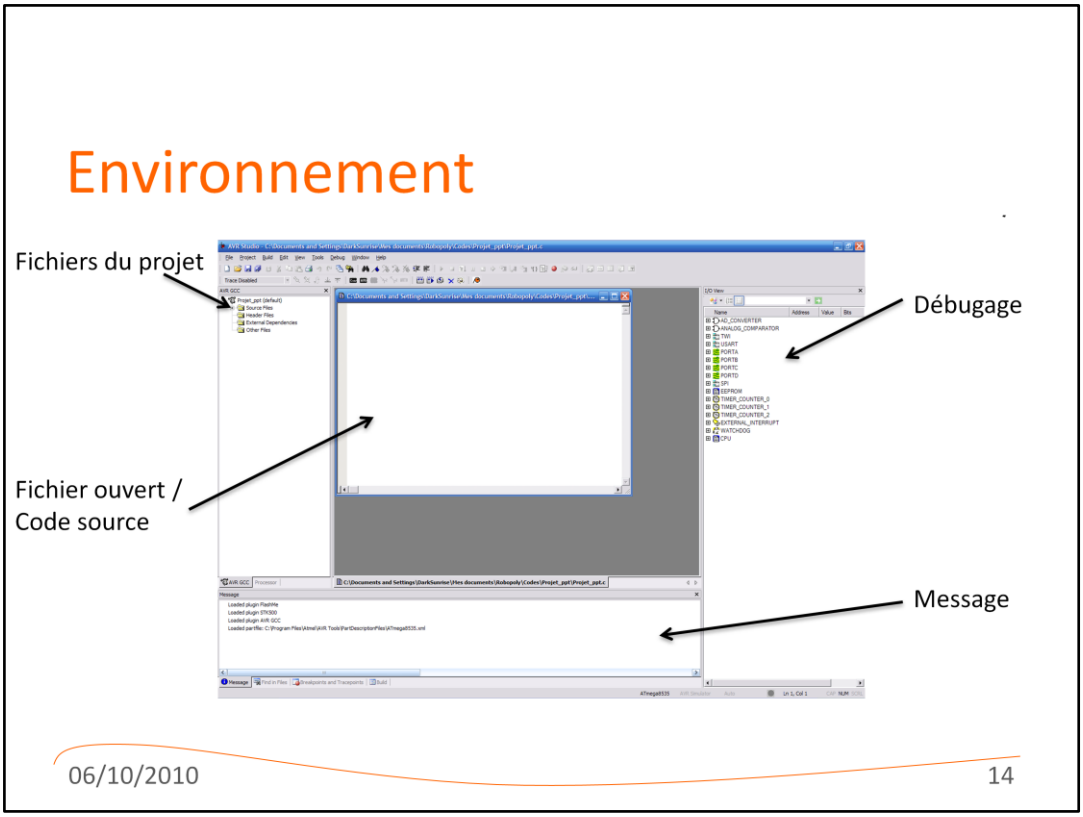


Nouveau projet (suite)

Sélectionner "AVR Simulator" à gauche pour la simulation

Sélectionner "ATmega8535" à droite, c'est le nom de notre microcontroleur

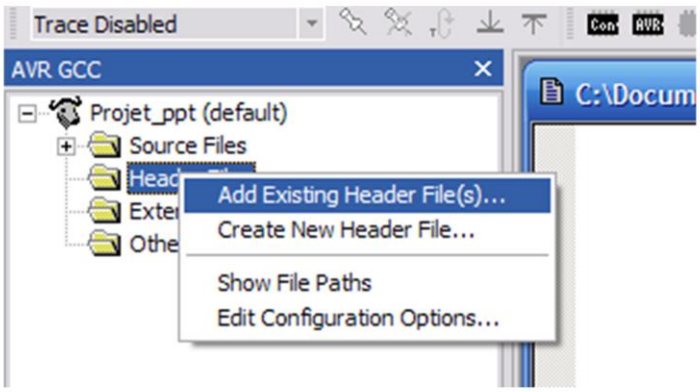
Cliquer sur "Load"



L'environnement d'AVR studio :

La fenêtre centrale est l'endroit où on écrit le code. A gauche, on a la liste des fichiers du projet. En bas, les messages de compilation et à droite la fenêtre de débuggage.

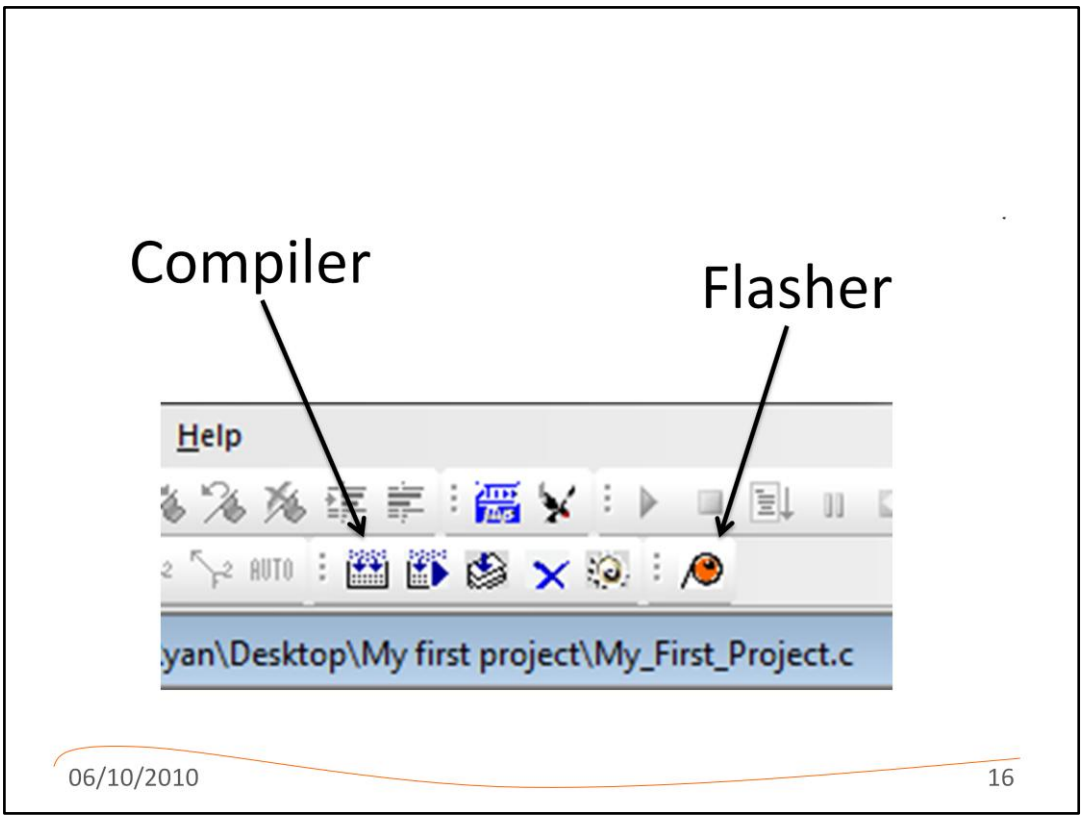
Inclure les fichiers



Dans la fenêtre de gauche, il faut ajouter les librairies de robopoly.

1. Copier "robopoly.c" et robopoly.h" qui se trouvent dans "C:\Program Files\Robopoly\" dans le dossier du projet en cours.
2. Ajouter "robopoly.h" dans "Header Files"
3. Ajouter "robopoly.c" dans "Source Files"

On peut maintenant commencer à coder.



Compiler : transformer le code en instructions compréhensible pour le robot.

Flasher : c'est mettre le code machine sur le robot.

Une fois qu'on a notre code prêt, il faut le flasher, le mettre sur le microcontroleur.
Pour cela on commence par le compiler avec le bouton compiler.

Il faut ensuite brancher le programmateur USB et cliquer sur le "p" de Robopoly. Une fenêtre noire s'affiche et demande un reset du PRisme -> le petit bouton.

IMPORTANT: il faut mettre sur le COM4 le programmateur USB.

Panneaux de configuration
Système
Onglet matériel
Gestion des périphérique
Arborescence Ports (COM et LPT)
Arborescence USB Serial Port (COMx)
Onglet Port Settings
Bouton Advanced
Sélectionner COM4, même s'il est déjà utilisé

```
 Entrée : état des leds actuel (unsigend char)
 Sortie : état des leds après la fonction(unsigned char)

////////////////////////////////////////////////////////////////////////////*/
unsigned char defile(unsigned char now)
{
    //Static == la variable n'est pas éffacée quand je sort de la fonction
    static unsigned char dir = 0;           //direction actuel de défilement

    /*
    On fait allume la maxi-led d'avant ou d'après en fonction de la valeur de dir

    exemples de l'astuce :
    -> 0b00000001 * 2 = 0b00000010
    -> 0b00010000 / 2 = 0b00001000
    */

    if(!dir)
        now = now * 2;
    else
        now = now / 2;

    //Si on est au bout des maxi-leds, on change la direction
    if(now == 0b00000001 || now == 0b10000000)
        dir = ~dir;

    return now;                         // Retour la nouvelle valeur des leds
}

/*////////////////////////////////////////////////////////////////////////////

 Nom : cligne
 Description : fait clignoter 4 fois toutes les leds à la férquence de 1Hz
 Entrée : rien
 Sortie : rien

////////////////////////////////////////////////////////////////////////////*/
void cligne(void)
{
    int i=3;                            //i+1 = nombre de fois que l'on veut faire cli
```

PROGRAMMATION

Les outils de la programmation!

Lorsqu'on programme, il faut toujours commenter son code.
Le robot ne va pas tenir compte des lignes de commentaire.

Il y a deux façons de faire des commentaires. La première ne commente qu'une ligne, et la seconde commente un paragraphe.

Les librairies

```
#include <avr/io.h>
#include "robopoly.h"
```

06/10/2010 19

Librairie c'est du code déjà fait pour faciliter la programmation.

La première chose à faire dans le code est d'inclure les librairies. On écrit tout en haut du code ces deux lignes.

io.h = apprendre au robot qu'il a des sorties et des entrées
robopoly.h = la librairie robopoly pour vous simplifiez grandement la vie

Le main

```
int main(void)
{
  // votre code...
  return 0;
}
```

06/10/2010 20

Ensuite, il faut un endroit où on écrit ce que va faire le robot, ça s'appelle le main.
C'est là que on codera le comportement du robot.

Il faut savoir que le robot commence à la première ligne de code du main, puis éxécute ligne par ligne les instructions suivantes.

Le robot fini à « return 0 ; ». Il ne fera plus rien après cette ligne. On remarque qu'il y a un « ; » à la fin du « return ». Cet ponctuation est nécessaire après chaque ligne d'instruction pour indiquer la fin de la ligne.

La boucle infinie

```
while(1)
{
  // Votre code...
}
```

Pour éviter que le robot s'arrête de fonctionner une fois qu'il arrive à la fin du main, il faut mettre une boucle infinie, dans lequel il répètera le code.

Une fois que le robot arrive à la fin de la boucle, il recommence en haut de la boucle.

Structure d'un code

```
// Les includes
#include <avr/io.h>
#include "robopoly.h"
// La fonction principale
int main(void)
{
    while(1)
    {
      // Votre code
    }
    // Fin du code
    return 0;
}
```


Voilà le code initial qu'il faut toujours avoir avant de commencer son propre code. Il faut bien veiller à espacer et indenter son code, pour qu'il reste lisible. Si vous programmez quelquechose sans structurer le code ni le commenter, et que vous revenez dans 2 mois, vous aurez aucune idée de ce que vous avez fait.

La variable

```
int ma_variable;
ma_variable = 0;
```

06/10/2010 23

Tout d'abord, nous avons besoin de variables, c'est-à-dire un paramètre dont on peut définir la valeur.

On peut s'imaginer une boite dans laquelle on met une information.

Une variable doit avoir un nom, dans cet exemple « ma_variable ». Elle doit aussi avoir un type, la nature de ce qui est stocké, ici « int » qui veut dire entier.
On commence par déclarer la variable, et ensuite lui donne la valeur 0.

Les bases

- binaire
- hexadécimale
- décimale

dec hex bin

Exemple : 162 = 0xA2 = 0b10100010

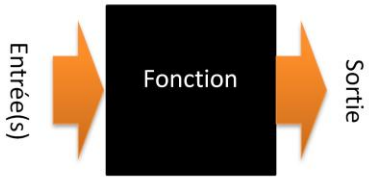
06/10/2010 24

Il y plusieurs façons de représenter un nombre. Nous utilisons tous les jours la base décimale, de 0 à 9. Mais en programmation, on trouve de l'hexadécimale, de 0 à 15 et du binaire, de 0 à 1.
Les différentes bases sont simplement une différente représentation du même nombre.

Une façon simple de faire la conversion est d'utiliser la calculatrice windows avec l'option « programmation » ou « scientifique »

Nous utiliserons beaucoup le binaire pour simplifier la chose. Voir dans quelques slides.

La fonction




```
int ma_fonction(int entree)
{
        //code
}
```

06/10/2010 25

Une fonction est simplement une boite noire, à laquelle on donne des entrées, éxécute quelquechose et renvoie une sortie.

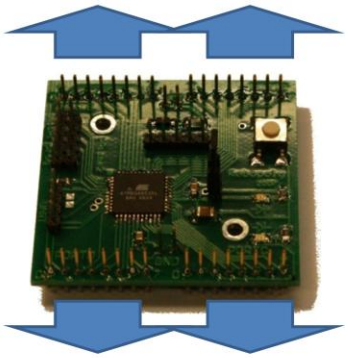
On peut faire l'analogie avec une fonction mathématique.

Exemple d'une fonction :

```
Int somme(int a, int b)
{
        int resultat;
        resultat = a + b;
        return resultat;
}
```

Ecrire une valeur sur un port

```
digitalWrite(port, bit, valeur);
```



06/10/2010 26

Voici une fonction de la librairie robopoly qui permet d'écrire une valeur sur un port.

```
digitalWrite(port, bit, valeur);
```

port = {A, B, C, D}
bit = {0, 1, 2, 3, 4, 5, 6, 7, BYTE}
valeur = {0, 1, 0b00000000.. 0b11111111}

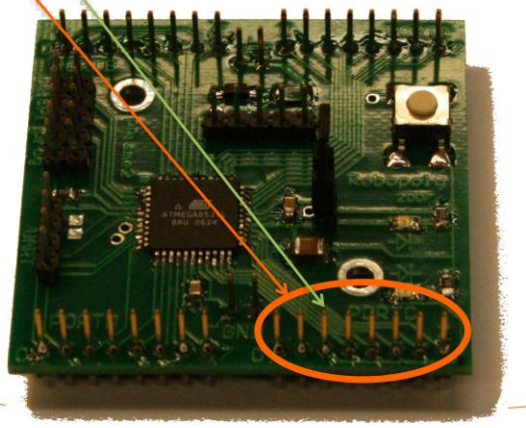
Exemples :

```
digitalWrite(A, 7, 1);
digitalWrite(A,BYTE, 0b11001100);
```

Les lignes du uC

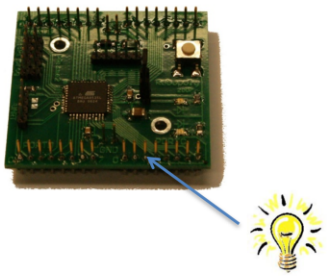
0 ou 1

```
digitalWrite(port, bit, valeur);
digitalWrite(C, 2, 1);
```



Exemple

```
digitalWrite(C, 2, 1);
```



Sur les autres ports / pins, rien ne se passe!

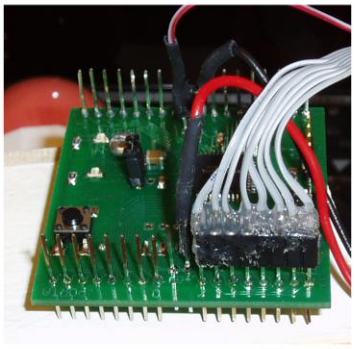
06/10/2010 28

Ici on a connecté les leds sur le port C, et on a vu que la 3ème led s'allume. Rien ne se passe sur les autres pins.

Allume l'ampoule numéro 2 (= 3ième) sur les maxi-leds.

Branchement des ampoules

Sur le port désiré, dans le sens désiré.



06/10/2010 29

Que se passe-t-il si je tourne le connecteur sur le port ?

C'est la lumière opposée qui s'allume.

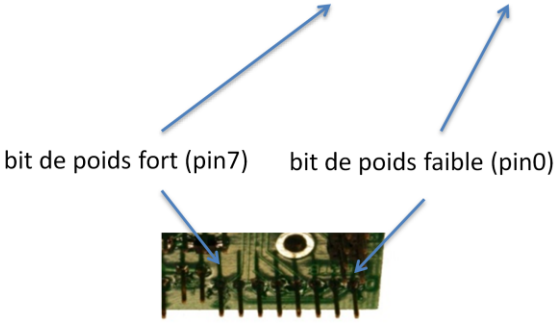
```
digitalWrite(C, 1, 1);
digitalWrite(C, 3, 1);
digitalWrite(C, 5, 1);
digitalWrite(C, 7, 1);
digitalWrite(B, 2, 1);
```

06/10/2010 30

Que font ces lignes de codes?

Elles allument une ampoule sur deux sur le PORTC. Comme il n'y a rien de branché sur le PORTB, on ne voit rien, mais si on branche les maxileds sur le PORTB, alors on vera que la 3ème lumière s'allume.

```
digitalWrite(C, BYTE, 0b10101010);
digitalWrite(B, BYTE, 0b00000010);
```



Une autre façon d'écrire un port est de mettre BYTE à la place du pin, et on peut écrire sur tout le port d'un coup. Pour ça, on utilise le binaire, où chaque bit, correspond à un pin.

Ca permet de compacter les 5 lignes de la slide précédente en 2.

Lire une valeur depuis un port

```
resultat = digitalRead(port,bit);
```

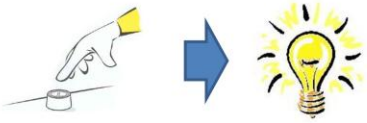
06/10/2010 32

Voici maintenant la fonction inverse qui permet de lire un port.
Resultat est une variable qu'on a déjà déclaré. La valeur retournée par cette fonction est comprise entre 0 et 1, ou si on met BYTE, donne un valeur en binaire.

```
result = digitalRead(port,bit);
```

port = {A, B, C, D}
bit = {0, 1, 2, 3, 4, 5, 6 ,7 , BYTE}
result = {0, 1, 0b0000000..0b11111111}

Attention : il ne faut pas oublier la déclaration de la variable result !!!



```
// code...
int bouton;
// code...
bouton = digitalRead(A, BYTE);
digitalWrite(C, BYTE, bouton);
// code...
```

06/10/2010 33

Voyons maintenant un exemple simple :

Le code allume l'ampoule si le bouton correspondant est enfoncé (bouton sur PORTA et ampoule sur PORTC)

Code :

```
#include <avr/io.h>
#include "robopoly.h"

void demo1()
{
	int bouton ;		//déclaration de la variable bouton
	bouton = digitalRead(C, BYTE);		//lecture des boutons sur le port C
	digitalWrite(A, BYTE, bouton);		//on allume les lumières
correspondantes
}

int main()
{
	while(1)
	{
		demo1();
	}
	//fin du code
	return 0;
}
```

Test logique

```
// Si le verre est plein
if(verre_plein == true)
{
   // alors
   bois();
}
else
{
   // sinon
   va_au_bar();
}
```


Un autre outil dont vous aurez besoin est le test logique.
On a le test if…else… Qui veut dire que si une condition est remplie, on fait une certain action, sinon une autre!

Opérandes

==	égal
> (>=)	plus grand que (ou égal)
< (<=)	plus petit que (ou égal)
!=	(différent de)

Pour la condition on peut utiliser plusieurs opérandes, donc, simplement : vérifier une égalité, une inégalité ou simplement une différence.

Exemple d'écriture (2)

```
// code...
int bouton = 0;

// code...
bouton = digitalRead(A, BYTE);
if (bouton == 0b00000010)
{
   digitalWrite(C, BYTE, 0b11111111);
}
else
{
   digitalWrite(C, BYTE, 0b00000000);
}

// code...
```


Voici un exemple avec un test logique qui, allume toutes les ampoules seulement si le deuxième bouton est appuyé.

Expliquer quand on appuie sur plusieurs boutons

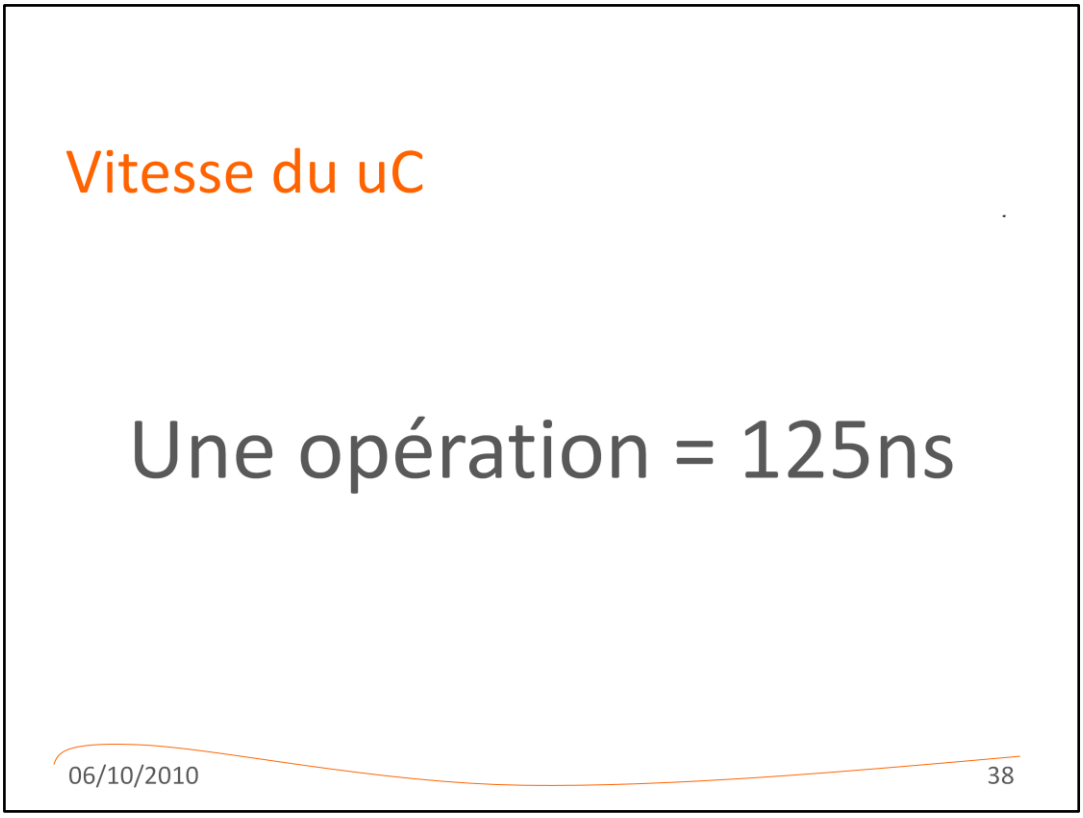
Code :

```
#include <avr/io.h>
#include "robopoly.h«

void demo2()
{
	int bouton = 0; //déclaration de la variable bouton

	bouton = digitalRead(C, BYTE);		//lecture des boutons sur le port C
	if (bouton == 0b00000010)		//test si les boutons ont exactement cette valeur
	{//alors
		digitalWrite(A, BYTE, 0b11111111);
	}
	else //sinon
	{
		digitalWrite(A, BYTE, 0b00000000);
	}

}

int main()
{
	while(1)
	{
		demo2();
	}
	//fin du code
}
```

Clignotement

```
// code...

digitalWrite(C, BYTE, 0b11111111);

digitalWrite(C, BYTE, 0b00000000);

// code...

// Ben ça marche pas...
```


Ce code ne marche pas pour faire clignoter des lumières



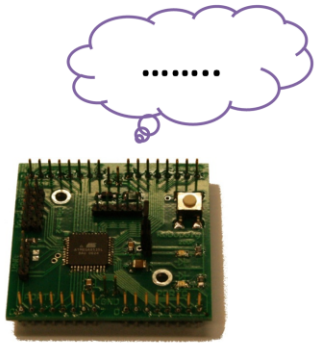
Il faut savoir que le microcontroleur exécute 8 millions d'opérations à la seconde, soit une opération toute les 125 ns. C'est bien trop rapide pour nous.

Cela correspond à une opération toutes les 0,000 000 125 s

Il faut donc faire attendre le microcontroleur.

Fonction d'attente

```
waitms(temps); // millisecondes

waitus(temps); // microsecondes
```



Pour cela on a deux jolies fonctions d'attentes waitms et waitus.

```
waitms(temps); // millisecondes
```
temps = 0..65536
```
waitus(temps); // microsecondes
```
temps = 0..255

Clignotement

```
// code...

digitalWrite(C, BYTE, 0b11111111);
waitms(500);

digitalWrite(C, BYTE, 0b00000000);
waitms(500);

// code...

// Là ça marche!
```

06/10/2010 40

Démo : faire clignoter toutes les lumières 1 fois par seconde :

Code :

```
#include <avr/io.h>
#include "robopoly.h"

void demo3()
{
	digitalWrite(A,BYTE, 0b11111111);
	waitms(500);
	digitalWrite(A,BYTE, 0b00000000);
	waitms(500);
}

int main()
{
	while(1)
	{
		demo3();
	}
	//fin du code
}
```

Démo

06/10/2010 41

Voici un exemple de jeu que vous pouvez faire avec un peu d'imagination.

Ca s'appelle Attrape-moi si tu peux!

Il faut appuyer sur le bouton au moment ou la lumière passe devant. Si ça clignote rapidement, c'est que vous avez perdu, et si ça clignote lentement, c'est gagné!

Vous avez tous les outils en main pour faire ce code!!!

La prochaine fois

Vous allez apprendre bouger votre robot grâce aux moteurs.

Pré-Fireday
16 octobre

06/10/2010 43

Date du pré-Fireday changé au 16 octobre à cause de la magistrale de l'EPFL.
Inscription toujours ouverte
Vente des Kits
Les moteurs seront distribués au fireday I.