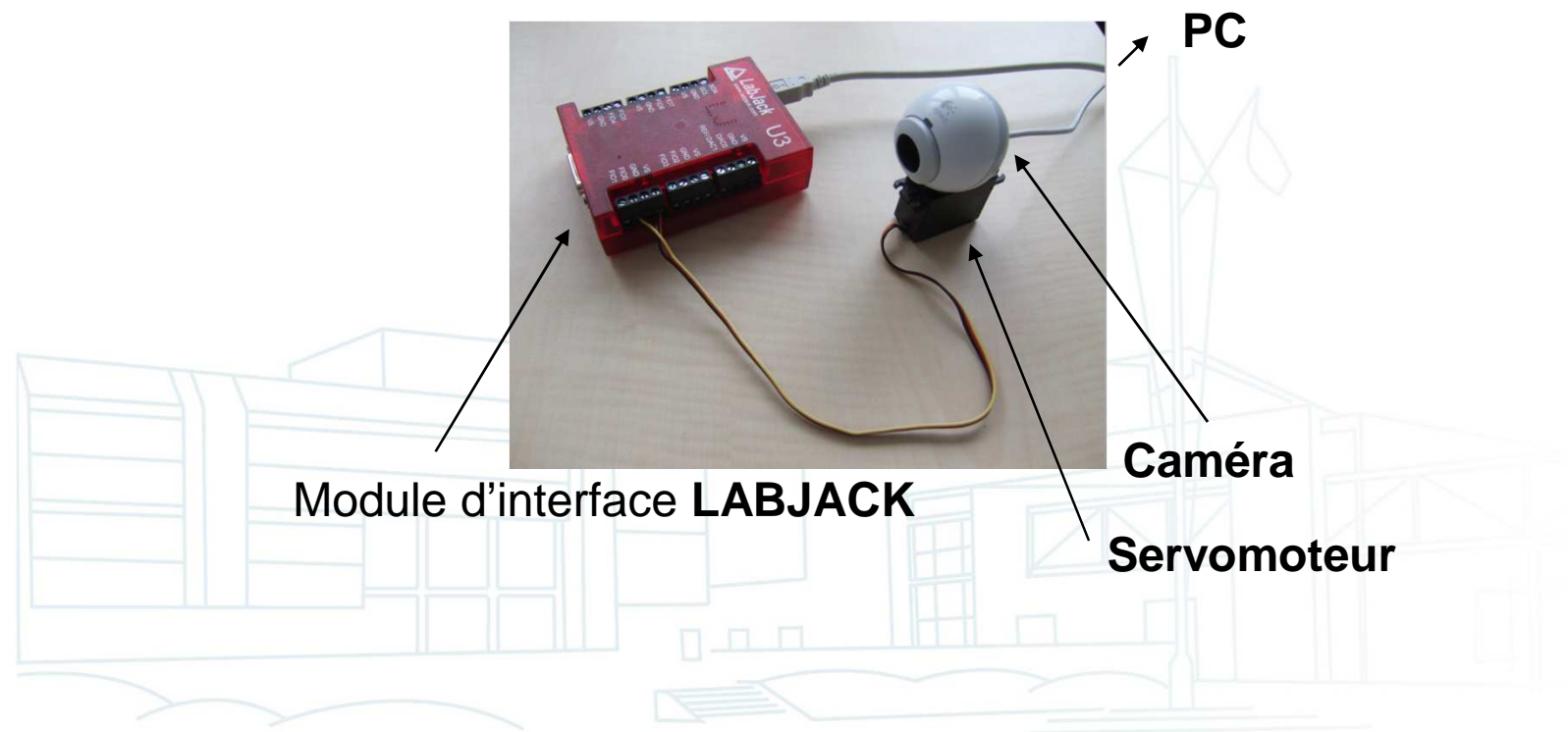




Introduction au langage C,
traitement d'image et
contrôle de moteur

But

- Suivi d'un objet coloré à l'aide d'une webcam montée sur un servomoteur



> Sommaire

1. Introduction au langage C

- a) Hello World!
- b) Fonctions
- c) Commentaires, types, structures, macros, constantes
- d) Tableaux statiques
- e) Chaînes de caractères
- f) Pointeurs
- g) Pointeurs et tableaux
- h) Allocation mémoire et tableaux dynamiques

2. Traitement d'image avec OpenCV

- a) OpenCV
- b) Utilisation rapide
- c) Images et C/C++
- d) Images et OpenCV


3. Contrôle de servomoteurs

- a) Carte de puissance
- b) Servomoteur
- c) Carte d'interface
- d) Contrôle des moteurs et servomoteurs via Labjack

Introduction au langage C

Hello World!

On souhaite utiliser des fonctions
déclarées dans ce fichier (ici printf())



```
#include <stdio.h>

int main(int argc, char* argv[])
{
    printf("Hello\n");
    return 0;
}
```



Fonctions

Déclaration de fonction : nécessaire ici car on souhaite utiliser `print_arg()` dans le `main()` mais on définit son contenu après le `main()`



```
#include <stdio.h>
L
void print_hello(void)
{
    printf("Hello\n");
}
void print_arg(char* name);
L
int main(int argc, char* argv[])
{
    if (argc > 1)
    {
        print_arg(argv[1]);
    }
    else
    {
        print_hello();
    }

    return 0;
}
void print_arg(char* name)
{
    printf("Hello %s!\n", name);
}
```



Commentaires, types, structures, macros, constantes

```
#include <stdio.h>
#include <string.h>

/*
C-style comments.
*/

// C++-style comments. Cannot be used on some C compilers.

// Floating types.
float float_number = -0.1f;
double double_number = -0.1;
long double long_double_number = -0.1;
```

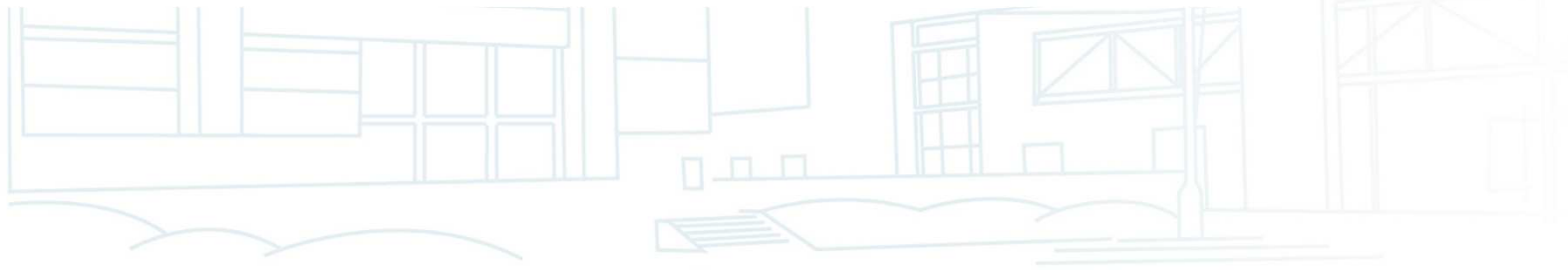
Commentaires, types, structures, macros, constantes

```
// Integer types.
int int_number = -1;
unsigned int unsigned_int_number = 1;
long long_number = -1;
short short_number = 32768;

// Character type. It is an integer, each alphanumerical character
// is in reality an integer code.
char char_number = 'A';

// Use typedef to define custom types from existing one.
// Note that there is no predefined boolean type, the int
// type is usually used : 0 -> false, other number -> true.
typedef int BOOLEAN;
```

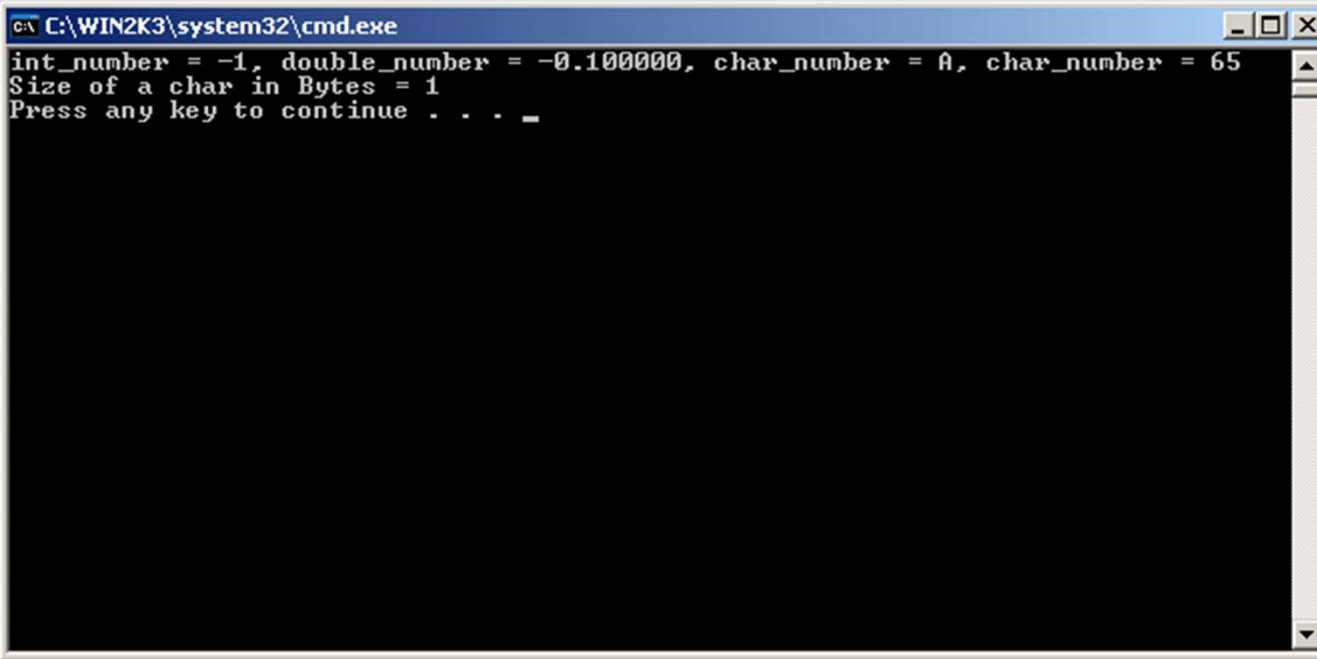
```
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```



Commentaires, types, structures, macros, constantes

```
int main(int argc, char* argv[])
{
    printf("int_number = %d, double_number = %f, char_number = %c, char_number = %d\n",
        int_number, double_number, char_number, (int)char_number);
    printf("Size of a char in Bytes = %d\n", sizeof(char));

    return 0;
}
```



```
C:\WIN2K3\system32\cmd.exe
int_number = -1, double_number = -0.100000, char_number = A, char_number = 65
Size of a char in Bytes = 1
Press any key to continue . . . _
```

Commentaires, types, structures, macros, constantes

```
// Structures.  
struct _STRUCT_EXAMPLE  
{  
    int num;  
    double value;  
};  
  
struct _STRUCT_EXAMPLE struct_example;  
  
typedef struct _STRUCT_EXAMPLE STRUCT_EXAMPLE;  
  
STRUCT_EXAMPLE struct_example_after_typedef;
```

Commentaires, types, structures, macros, constantes

```
// Macros.  
#define MACRO_EXAMPLE(number) (2*number+1)  
  
// Constants.  
const double constant_example = 3.1;  
#define CONSTANT_MACRO_EXAMPLE (2*3.0)
```



Tableaux statiques

```
#include <stdio.h>
#include <string.h>

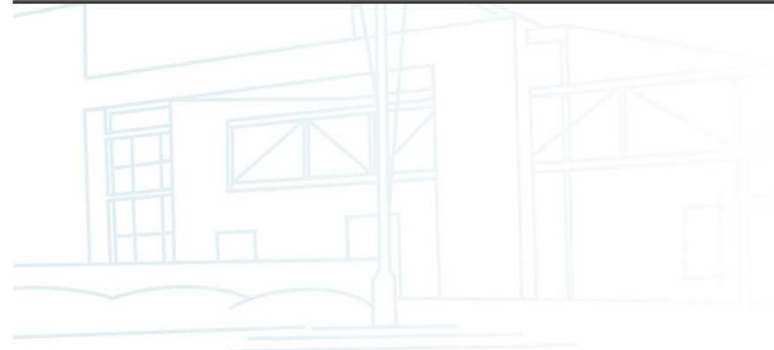
// Arrays.
int array_example[4][3] =
{
    { 1, 1, 1 },
    { 2, 2, 2 },
    { 3, 3, 3 },
    { 4, 4, 4 },
};

int main(int argc, char* argv[])
{
    int i = 0, j = 0;

    printf("array_example = \n");
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 3; j++)
        {
            array_example[i][j] = i*j;
            printf("%d ", array_example[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
C:\WIN2K3\system32\cmd.exe
array_example =
0 0 0
0 1 2
0 2 4
0 3 6
Press any key to continue . . .
```



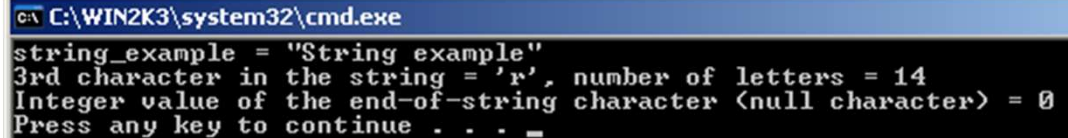
Chaînes de caractères

```
#include <stdio.h>
#include <string.h>

// Character string.
char string_example[256] = "String example";
char empty_string[1] = "";

int main(int argc, char* argv[])
{
    printf("string_example = \"%s\"\n", string_example);
    printf("3rd character in the string = '%c', number of letters = %d\n",
           string_example[2], strlen(string_example));
    printf("Integer value of the end-of-string character (null character) = %d\n",
           |(int)empty_string[0]);

    return 0;
}
```



```
C:\WIN2K3\system32\cmd.exe
string_example = "String example"
3rd character in the string = 'r', number of letters = 14
Integer value of the end-of-string character (null character) = 0
Press any key to continue . . . _
```



Pointeurs

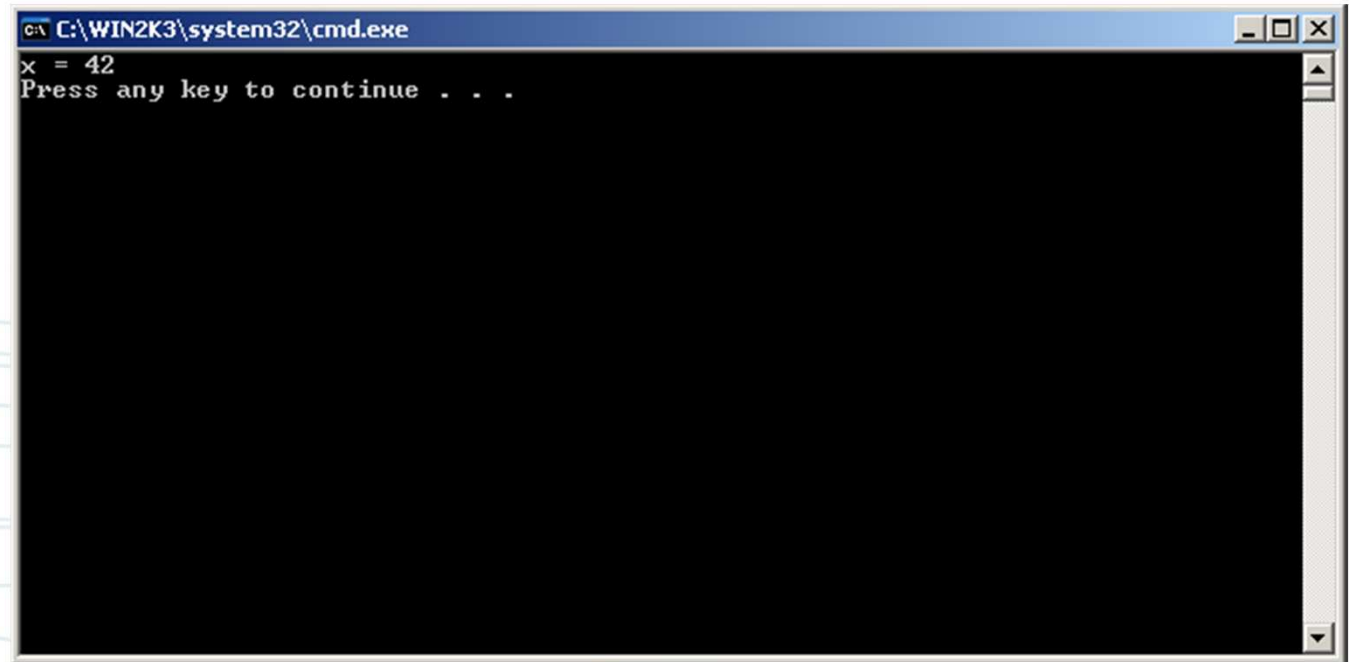
```
#include <stdio.h>
{
int main(int argc, char* argv[])
{
    int x = -4;
    int* address = NULL;

    // Put the memory address (place number in the memory where x is stored)
    // of x in address.
    address = &x;

    // Now we can modify x.
    *address = 42;

    printf("x = %d\n", x);

    return 0;
}
```



```
C:\WIN2K3\system32\cmd.exe
x = 42
Press any key to continue . . .
```

Pointeurs et tableaux

```
#include <stdio.h>
L
int main(int argc, char* argv[])
{
    int tab[42];
    int i = 0;

    for (i = 0; i < 42; i++)
    {
        *(tab+i) = 0; // tab[i] = 0;
    }

    return 0;
}
```

Allocation mémoire et tableaux dynamiques

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    int n = 42;
    int* tab = (int*)malloc(n*sizeof(int));
    int i = 0;

    for (i = 0; i < 42; i++)
    {
        *(tab+i) = 0; // tab[i] = 0;
    }

    free(tab);

    return 0;
}
```


Traitement d'image avec OpenCV

■ Présentation

- <http://opencv.willowgarage.com/wiki/>
- Bibliothèque open source
- Portable (fonctionne sous Linux, Windows, Mac OS)
- Fonctions en C/C++ ou Python
- Codes optimisés développés à l'origine par Intel
- Documentation et exemples de codes
- ...



- Chargement et affichage d'un fichier image

```
// Chargement d'une image au format bitmap
IplImage* image = cvLoadImage("image.bmp");

// Création d'une fenêtre
cvNamedWindow("Fenêtre");

// Affichage de l'image
cvShowImage("Fenêtre", image);

// Attend que l'utilisateur appuye sur une touche
cvWaitKey(0);

// Libération de la mémoire utilisée pour l'image chargée
cvReleaseImage(&image);

// Destruction de la fenêtre
cvDestroyWindow("Fenêtre");
```



- Récupération d'une image webcam

```
CvCapture* pCaptureFromCAM = cvCaptureFromCAM(0);  
frame = cvQueryFrame(pCaptureFromCAM);  
...  
cvReleaseCapture(&pCaptureFromCAM);
```



Images et C/C++

- Représentation courante d'une image en C/C++

```
unsigned char * img;  
int width;  
int height;  
  
imgsize = width * height * 3;  
  
// Pixel à ligne i, colonne j  
img[0+3*j+3*width*i] = 0; // Rouge  
img[1+3*j+3*width*i] = 0; // Vert  
img[2+3*j+3*width*i] = 255; // Bleu
```

	Pixel 1	Pixel 2	Pixel 3	Pixel 4				
Pixel 5	R	G	B	R	G	B		
Pixel 9								



Pixel 1

Pixel 2

- Utilisation d'OpenCV
 - Type IplImage défini par OpenCV

```
typedef struct _IplImage
{
    int nChannels; /* 3 composantes pour une image couleur (rouge, vert, bleu) */
    int depth; /* 8 bits (1 octet, taille d'un char) pour chaque composante */
    int width; /* Largeur de l'image en pixels */
    int height; /* Hauteur de l'image en pixels */
    char *imageData; /* Pointeur vers les données de l'image */
    ...
}
IplImage;
```

- Création d'une image couleur vierge

```
IplImage* im = cvCreateImage(
    cvSize(320, 240), // Taille de l'image
    8, // 8 bits (1 octet, taille d'un char) pour chaque composante
    3 // 3 composantes pour une image couleur (rouge, vert, bleu)
);
```

Images et OpenCV

- Utilisation d'OpenCV
 - Accès aux pixels d'une IplImage

```
IplImage* frame;  
  
...  
  
unsigned char* data = reinterpret_cast<unsigned char*>(frame->imageData);  
  
for (int i = 0; i < frame->height; i++)  
{  
    for (int j = 0; j < frame->width; j++)  
    {  
        data[0+3*j+3*frame->width*i] = 0; // Bleu  
        data[1+3*j+3*frame->width*i] = 0; // Vert  
        data[2+3*j+3*frame->width*i] = 255; // Rouge  
    }  
}
```



Contrôle de servomoteurs

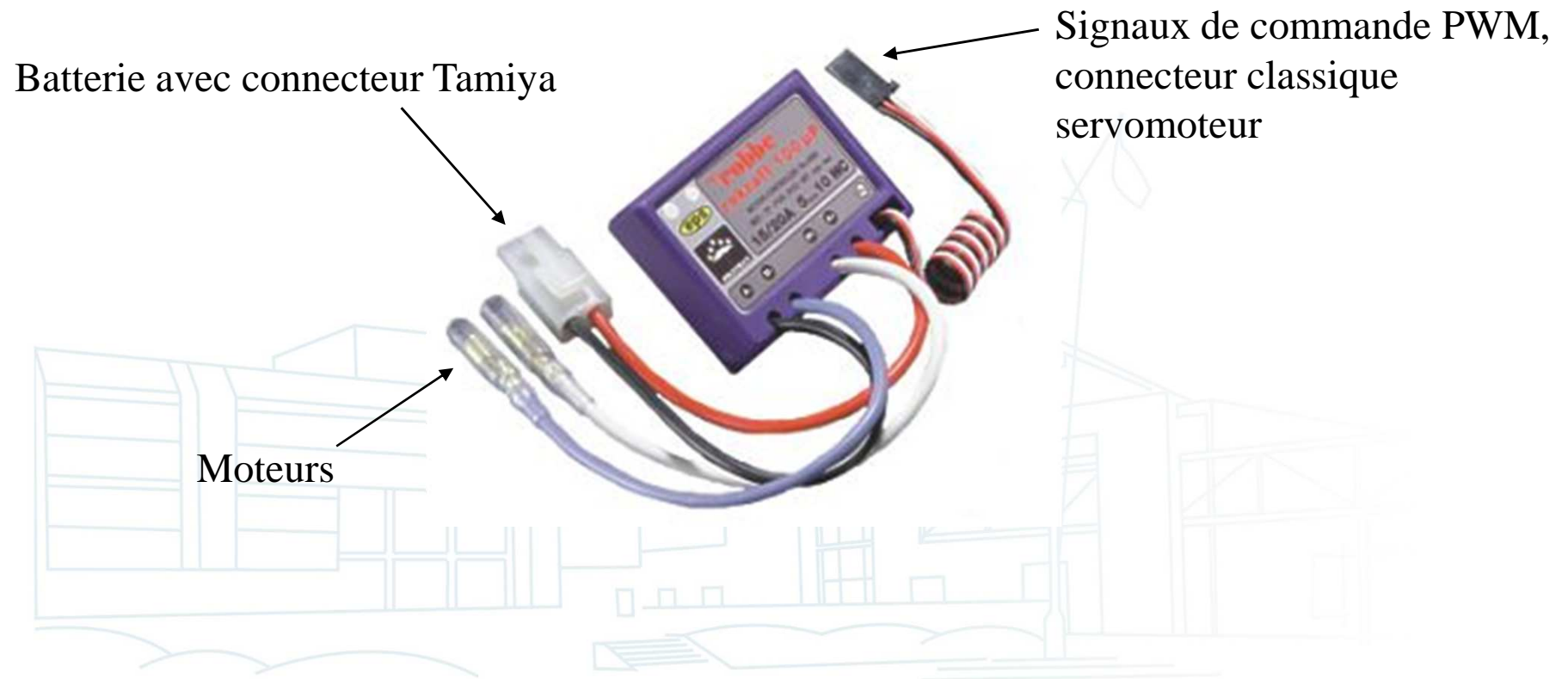
Carte de puissance

- Permet de contrôler les moteurs par des signaux de commande
 - Moteurs : tensions et courants élevés provenant des batteries
 - Signaux de commande : tensions et courants faibles venant directement ou indirectement du PC
 - Exemples : signaux PWM, I2C



Carte de puissance

- Exemple : Robbe Rokraft

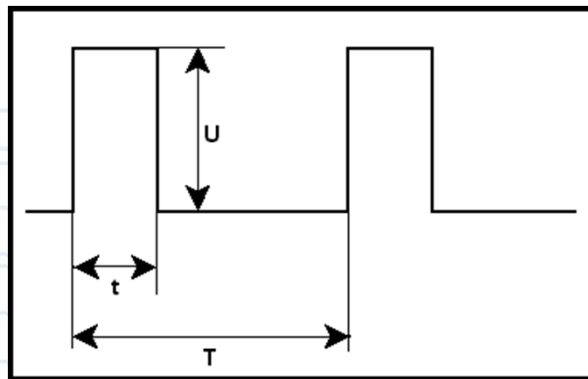


Carte de puissance

■ Exemple : Robbe Rokraft

– Fonctionnement

- La puissance envoyée aux moteurs (et donc leur vitesse) dépend du signal de commande PWM
- PWM = Pulse Width Modulation : modulation en largeur d'impulsion



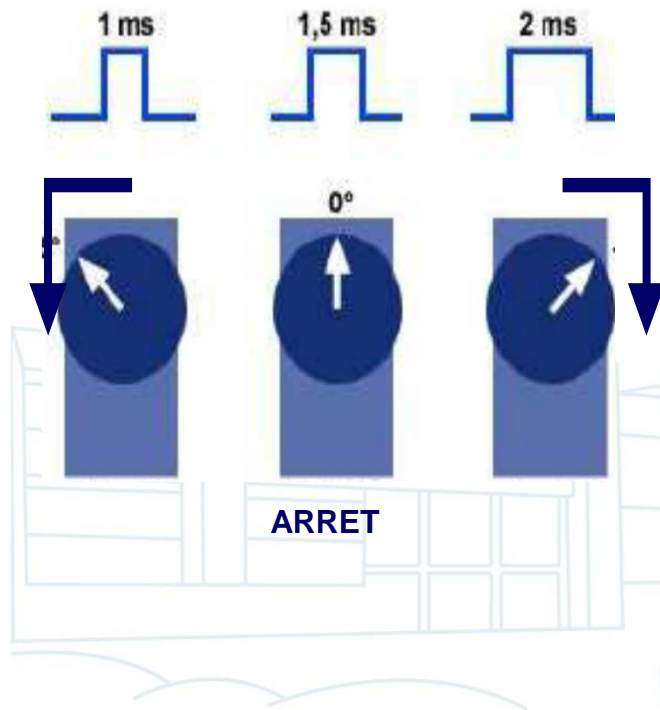
U : tension du PWM (5 V)

t : largeur d'impulsion (entre 1 et 2 ms)

T : période (20 ms)

Carte de puissance

- Exemple : Robbe Rokraft
 - Fonctionnement
 - Correspondance largeur d'impulsion / vitesse de rotation



État du moteur	Largeur d'impulsion
Moteur à l'arrêt	1.5 ms
Rotation dans un sens, en accélérant	1.5 à 2.0 ms
Rotation dans le sens inverse, en décélérant	1.0 à 1.5 ms

Servomoteur

- Servomoteur = petit moteur + carte de puissance : pour orienter la webcam
- Commandé par PWM
- 2 types de servomoteurs :
 - Asservis en position : tournent de -40 à $+40^\circ$ par exemple
 - Asservis en vitesse



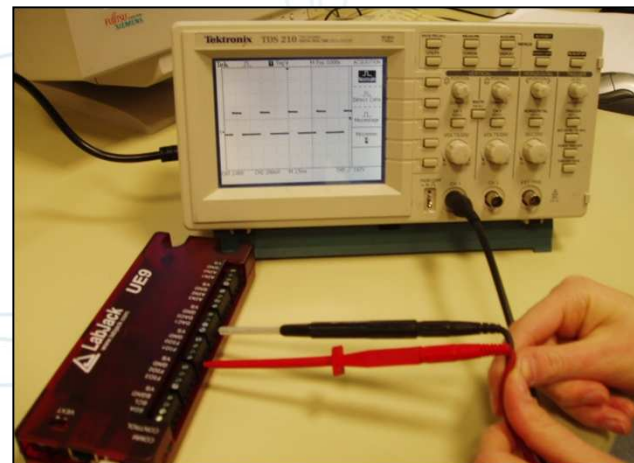
Carte d'interface

- Relie la partie informatique avec la partie électronique (capteurs, actionneurs)
 - Partie informatique : intelligence par le biais de programmes sur PC
 - Partie électronique : capteurs, actionneurs



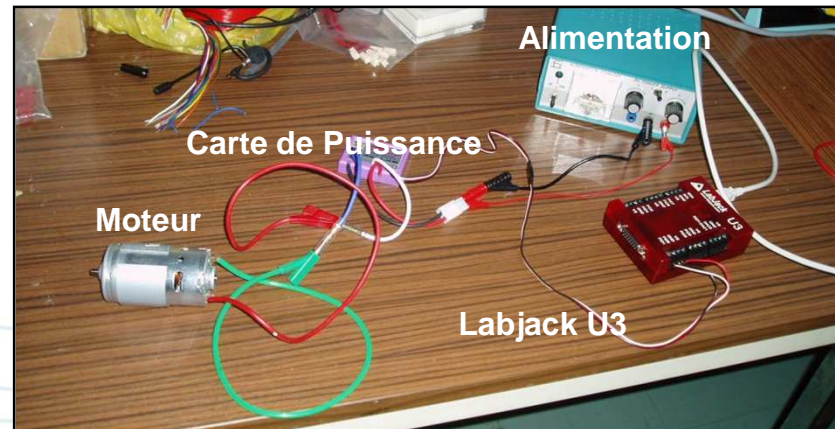
Carte d'interface

- Exemple : boîtier Labjack
 - Se branche sur l'ordinateur en USB et est contrôlé par des programmes exécutés sur l'ordinateur
 - Peut générer des signaux PWM, I2C
 - Peut générer des petites tensions
 - Peut lire des petites tensions (venant de capteurs analogiques tels que des télémètres, odomètres, boussoles...)



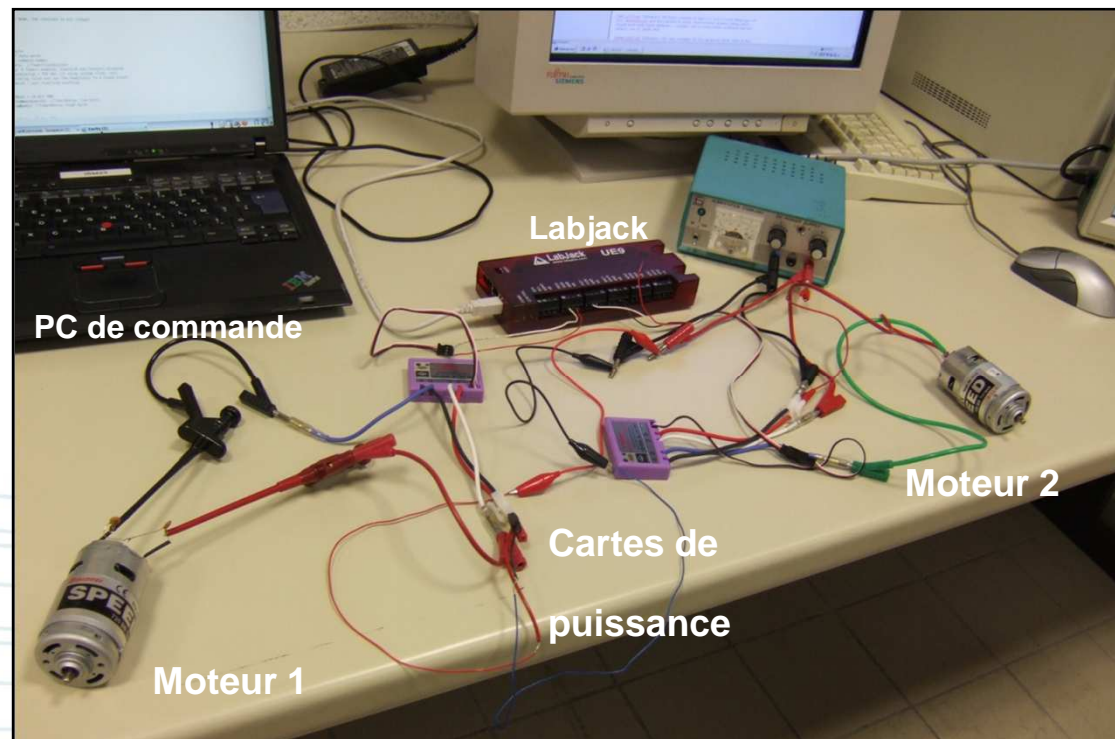
Carte d'interface

- Exemple : boîtier Labjack



Carte d'interface

- Exemple : boîtier Labjack



Dispositif de commande de moteurs avec la carte uE9

Contrôle des moteurs et servomoteurs via Labjack

- <http://www.labjack.com/>
- Une bibliothèque de fonctions et drivers fournis pour différents langages, OS
- Programmes exemples disponibles
- Pour contrôler les moteurs, nous utiliserons les fonctions « timer » du Labjack



Annexes

Passage C/C++

- Le C est inclus (à 99%) dans le C++ : quand on fait du C, on fait aussi du C++ mais l'inverse n'est pas forcément vrai
- Le C++ rajoute des notions de programmation orientée objet (classe, héritage, polymorphisme) ainsi que des facilités d'écriture



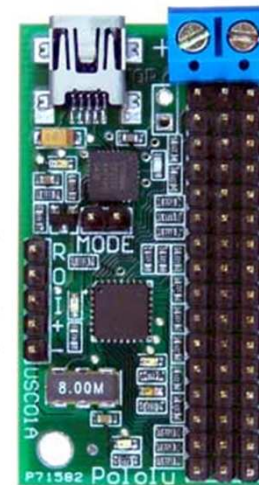
■ Présentation

- 4 parties principales :
 - CXCORE (types et fonctions de base : matrices, images, arbres, graphes, fonctions mathématiques, dessin de formes...)
 - CV (traitement d'image : détection d'objets, de mouvement, calibration...)
 - HIGHGUI (récupération et affichage d'images : lecture/enregistrement de fichiers images et vidéos, gestion des webcams, affichage dans des interfaces graphiques...)
 - MLL (arbres de décision, réseaux de neurones...)



Carte d'interface

- Autres exemples : Cartes Parallax et Pololu
 - Se branchent en série (ou USB via un convertisseur USB-série) et génèrent jusqu'à 16 PWM



Contrôle des moteurs et servomoteurs via Labjack

- Le Labjack peut générer jusqu'à 6 timers/PWM dont la fréquence est définie par

Fréquence finale du PWM
Doit être proche de 1/20ms
pour pouvoir contrôler un servo

Fréquence interne du Labjack

$$f_{PWM} = \frac{f_{sys}}{timer_clock_divisor * 2^{16}}$$

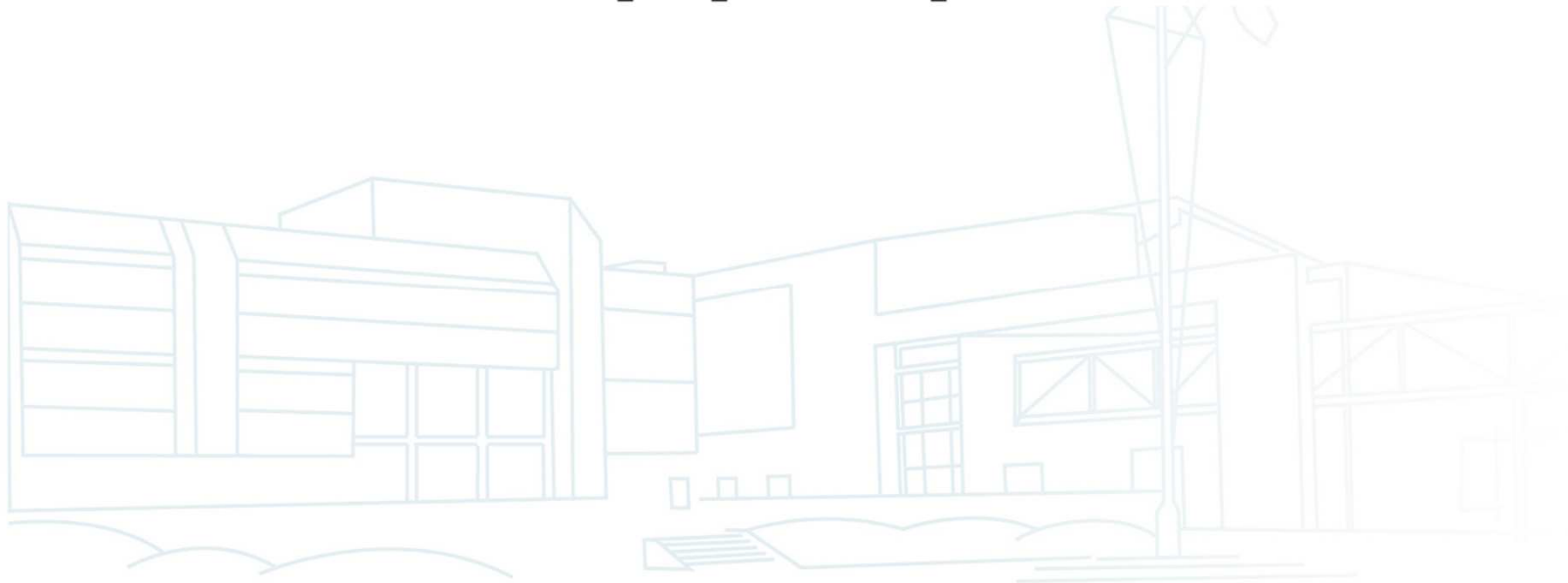
Paramètre pouvant être propre à chaque PWM
pour faire varier sa fréquence indépendamment des autres

Mode du PWM
=> précision de la largeur d'impulsion

Contrôle des moteurs et servomoteurs via Labjack

- Fonctions utiles

```
LJ_HANDLE lngHandle;  
  
// Ouvre le 1er Labjack U3. lngHandle contiendra son identifiant  
OpenLabJack(LJ_dtU3, LJ_ctUSB, "1", 1, &lngHandle);  
  
// Reset  
ePut(lngHandle, LJ_ioPIN_CONFIGURATION_RESET, 0, 0, 0);
```

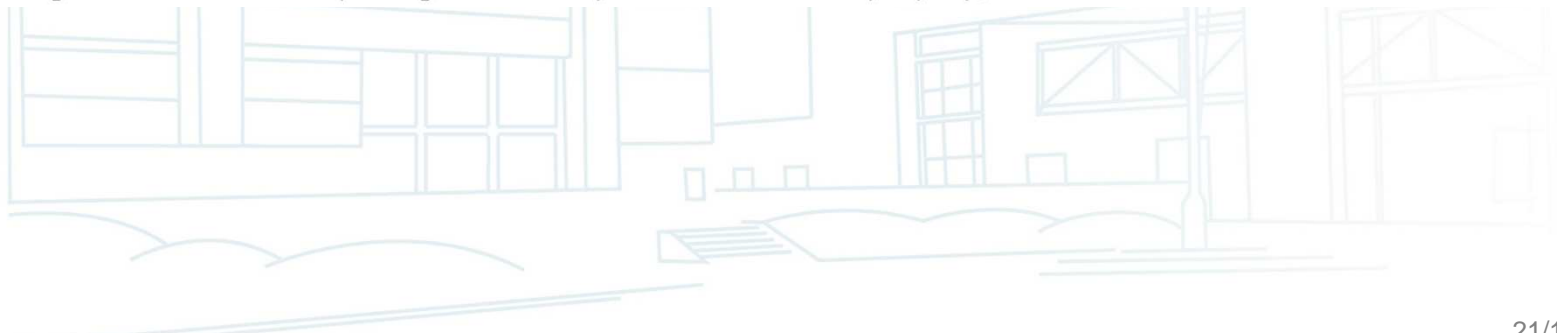


Contrôle des moteurs et servomoteurs via Labjack

■ Fonctions utiles

```
// Activation des timers et configuration de leurs fréquences
alngEnableTimers[0] = 1; // Active Timer0 (il sera sur FIO4, voir lngTCPinOffset)
alngEnableTimers[1] = 1; // Active Timer1 (il sera sur FIO5, voir lngTCPinOffset)
alngEnableCounters[0] = 0; // Désactive le 1er compteur
alngEnableCounters[1] = 0; // Désactive le 2ème compteur
lngTCPinOffset = 4; // Offset indiquant que le 1er timer sera sur FIO4.
lngTimerClockBaseIndex = LJ_tc48MHZ_DIV; // Fréquence interne de 48 MHz avec gestion du lngTimerClockDivisor
// pour pouvoir obtenir la fréquence de PWM voulue
lngTimerClockDivisor = 16; // La fréquence du PWM sera 45.78 Hz (voir formule)
alngTimerModes[0] = LJ_tmPWM16; // La largeur d'impulsion sera codée sur 16 bits :
alngTimerModes[1] = LJ_tmPWM16; // 0 -> largeur min, 65535 : largeur max
adblTimerValues[0] = 65535; // Largeur d'impulsion de départ de Timer0
adblTimerValues[1] = 0; // Largeur d'impulsion de départ de Timer1

eTCCConfig(lngHandle, alngEnableTimers, alngEnableCounters, lngTCPinOffset, lngTimerClockBaseIndex,
lngTimerClockDivisor, alngTimerModes, adblTimerValues, 0, 0);
```



Contrôle des moteurs et servomoteurs via Labjack

■ Fonctions utiles

```
// Modification des largeurs d'impulsion
alngReadTimers[0] = 0;
alngReadTimers[1] = 0;
alngUpdateResetTimers[0] = 1; // Mettre à jour Timer0
alngUpdateResetTimers[1] = 0; // Ne pas modifier Timer1
alngReadCounters[0] = 0;
alngReadCounters[1] = 0;
alngResetCounters[0] = 0;
alngResetCounters[1] = 0;
adb1TimerValues[0] = 32768; // Nouvelle largeur d'impulsion
adb1TimerValues[1] = 0;

eTCValues(lngHandle, alngReadTimers, alngUpdateResetTimers, alngReadCounters,
          alngResetCounters, adb1TimerValues, adb1CounterValues, 0, 0);
```



Rappels C/C++

- Le C est inclus (à 99%) dans le C++ : quand on fait du C, on fait aussi du C++ mais l'inverse n'est pas forcément vrai
- Du code C ou C++ peut être écrit dans un fichier .cpp mais seul du code C peut être écrit dans un fichier .c
- Dans un .h, on peut écrire du C ou du C++, mais il faut que les .c n'incluent que des .h avec du C

Rappels C/C++

- Le C++ rajoute des notions de programmation orientée objet (classe, héritage, polymorphisme) ainsi que des facilités d'écriture



Rappels C/C++

- Différences entre les compilateurs Windows et Linux
 - Linux
 - Le compilateur C le plus utilisé est GCC
 - Son équivalent C++ est G++
 - Windows
 - GCC/G++ existent avec Cygwin et MinGW
 - Différents IDE existent et fournissent leurs propres compilateurs
 - Microsoft Visual Studio avec CL
 - Borland C++ Builder / Turbo C++ / Borland Developer Studio avec BCC32
 - Code Blocks / Dev-C++ avec MinGW

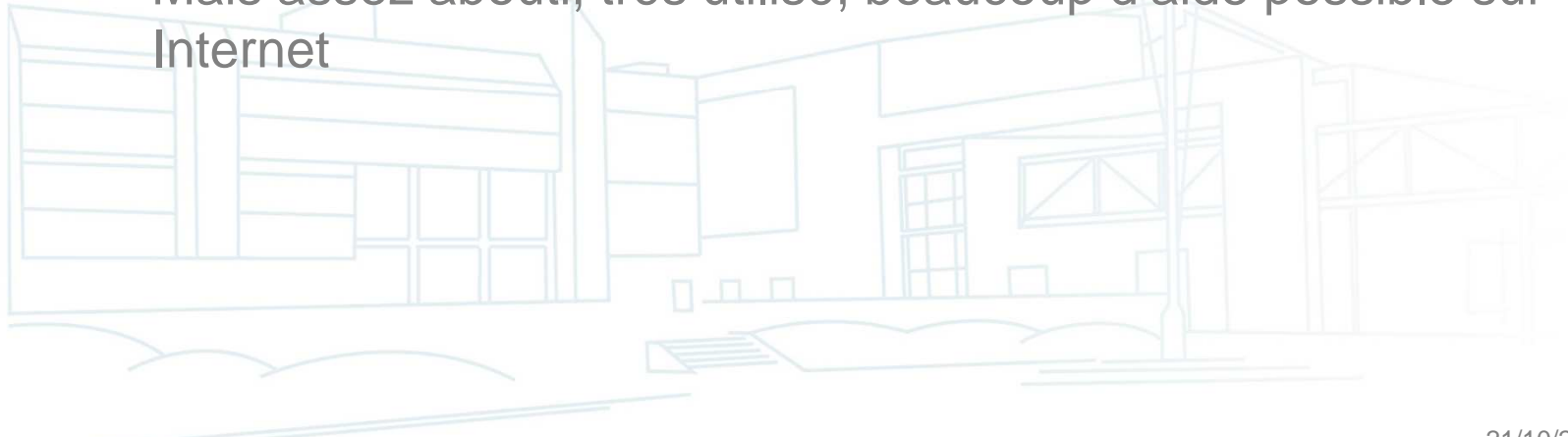


Rappels C/C++

- Différences entre les compilateurs Windows et Linux

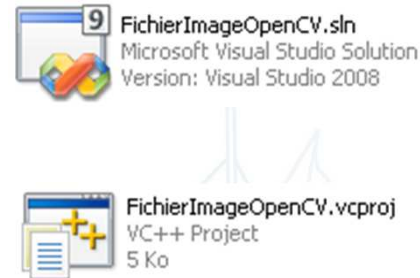
Equivalences Linux / Windows		
	Linux/GCC	Windows/Visual C++
Fichiers objets	.o	.obj
Bibliothèque statique	.a	.lib
Bibliothèque dynamique	.so	.dll
Exécutable	-	.exe

- Visual Studio
 - Versions
 - Visual C++ 6 : date de 1998
 - Visual Studio 2002 / 2003 ou .Net : refonte de l'IDE et ajout des projets .Net
 - Visual Studio 2005 / 2008 : quelques mises à jour
 - Difficile à prendre en main au début : nombreux types de projets, nombreuses options incompréhensibles
 - Mais assez abouti, très utilisé, beaucoup d'aide possible sur Internet



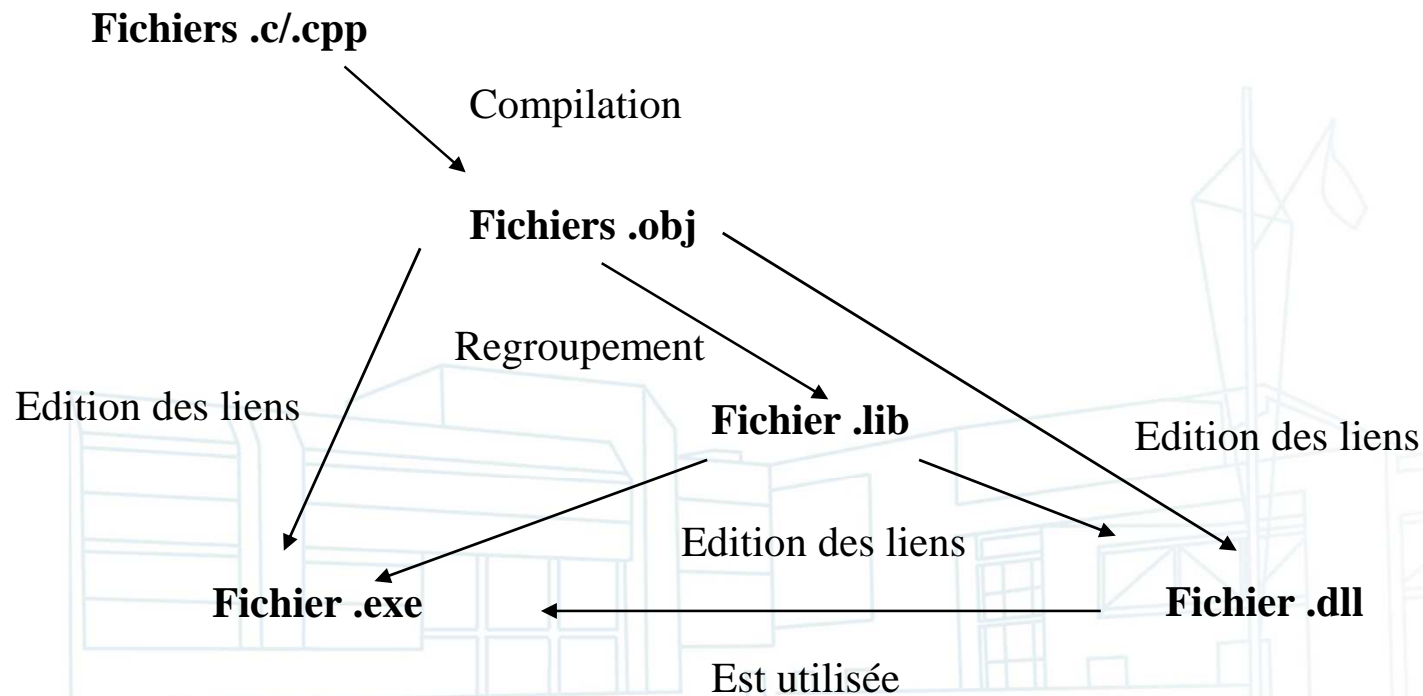
Rappels C/C++

- Visual Studio
 - Organisation
 - Workspace/Solution (fichier .dsw/.sln) : ensemble de projets
 - Projet (fichier .dsp/.vcproj) : ensemble de fichiers nécessaires à la génération d'un exécutable ou bibliothèque (.h, .c, .cpp...)



Rappels C/C++

- Utilisation de bibliothèques de fonctions externes



Utilisation d'OpenCV avec Visual C++ 6

- Les chemins suivants sont des chemins absolus considérant l'installation par défaut dans **C:\Program Files\OpenCV**
- Créer un projet **Win32ConsoleApplication**
- Dans le menu "**Project**", "**Settings**", "C/C++", catégorie "**Preprocessor**".
Ajouter les chemins suivants dans "**Additional include directories**":
 - **C:\Program Files\OpenCV\cv\include,**
 - **C:\Program Files\OpenCV\cvaux\include,**
 - **C:\Program Files\OpenCV\cxcore\include,**
 - **C:\Program Files\OpenCV\otherlibs\highgui**
- Dans le menu "**Project**", "**Settings**", "**Link**", catégorie "**Input**".
Ajouter les bibliothèques suivantes dans "**Object/library modules**":
 - **cv.lib cvaux.lib cxcore.lib highgui.lib**
- Ajouter le chemin suivant dans "**Additional library path**":
 - **C:\Program Files\OpenCV\lib**
- Modifier la variable d'environnement "PATH" de Windows en ajoutant:
 - **C:\Program Files\OpenCV\bin**
- Dans le code, ajouter
 - **#include "cvaux.h"**
 - **#include "highgui.h"**

Rappels C/C++

- Utilisation de bibliothèques de fonctions externes
 - Cas où on possède des fichiers .h et .c/.cpp
 - Exemple : on a Main.cpp qui doit appeler des fonctions déclarées dans Lib.h et définies dans Lib.cpp
 - On met `#include « Lib.h »` dans Main.cpp
 - On copie Lib.h et Lib.cpp dans le dossier de Main.cpp
 - On compile et lie Lib.cpp et Main.cpp en les ajoutant au projet



Rappels C/C++

- Utilisation de bibliothèques de fonctions externes
 - Cas où on possède des fichiers .h, .lib et .dll
 - Exemple : on a Main.cpp qui doit appeler des fonctions déclarées dans Lib.h et définies dans Lib.lib et Lib.dll
 - On met #include « Lib.h » dans Main.cpp
 - On ajoute le dossier de Lib.h dans les chemins de recherche de fichiers .h du projet
 - On ajoute le dossier de Lib.lib dans les chemins de recherche de fichiers .lib du projet
 - On compile Main.cpp et lie avec Lib.lib en les ajoutant au projet
 - On ajoute le dossier de Lib.dll à la variable d'environnement PATH du système

