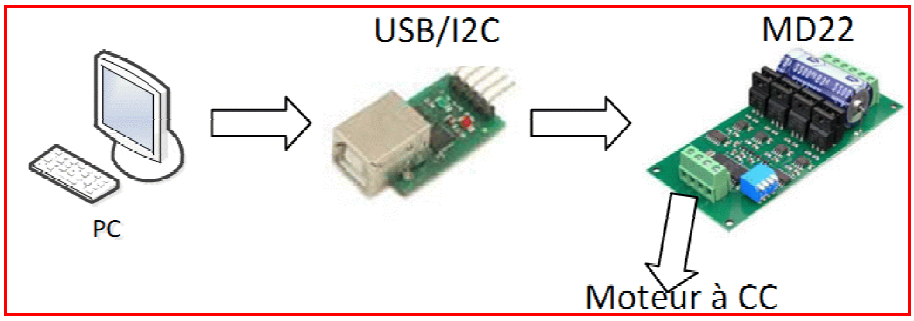
TP 5 - Pilotage, en mode I2C, d'un module moteur à courant continu

Tp logiciel C++
Durée: 6 h



Objectifs :

- Développer un programme, sous Qt, destiné à commander le module moteur MD22 .
- Etudier et mettre en oeuvre le protocole I2c.
- Etudier le module de communication Usb /I2c
- Etablir une nouvelle classe Usb_I2c.
- Analyser la notice technique constructeur du module MD22 afin de concevoir un interface permettant d'agir sur le sens et la vitesse d'un moteur à courant continu.

But :

- Mise en oeuvre des connaissances théoriques, sur le fonctionnement du bus I2C, les échanges d'informations sur les lignes Sda et Scl entre le module de communication et la carte MD22.
- Définir une nouvelle classe Usb_I2c pour générer le protocole de communication pour les composants I2c.
- Programmer une application, en langage C++, afin de piloter le sens et la vitesse d'un moteur CC.

Le TP nécessite la maîtrise des notions suivantes :

- Le bus I2c : Bus de communication très utilisé en domotique ou dans l'électronique grand public.
- La programmation d'un port série : Le convertisseur Usb/I2c est détecté par le PC comme un simple port série.
- Commande moteur : Le module MD22 est une unité de commande de moteur basée sur un pont H (H-Bridge), capable de fournir une puissance suffisante pour l'entraînement du moteur. Les commandes sont reçues par le bus I2c. Il est d'ailleurs possible de connecter plusieurs modules MD22 au bus afin de commander plusieurs moteurs.

Critères d'évaluation :

Vous serez évalués sur les critères suivants:

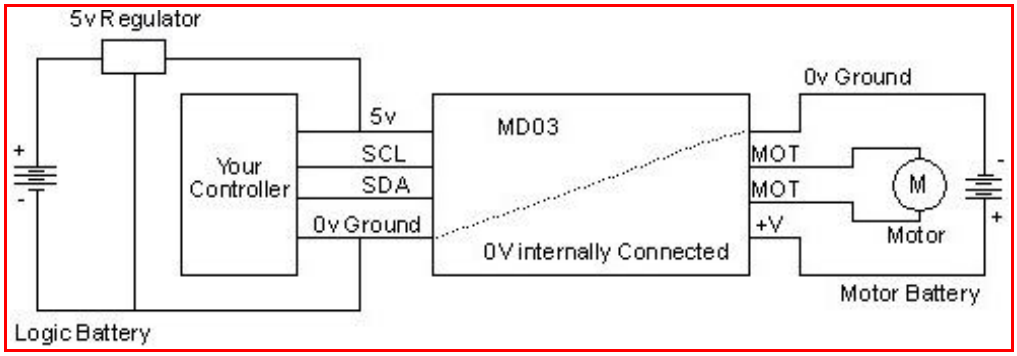
* Votre autonomie à résoudre les différents problèmes énoncés.
* Votre capacité à mettre en oeuvre le travail étudié.
* Votre capacité à utiliser le matériel mis à votre disposition.
* La rédaction de l'algorithme et du programme de mise en oeuvre.

1) Notice constructeur du module de commande moteur MD22 :

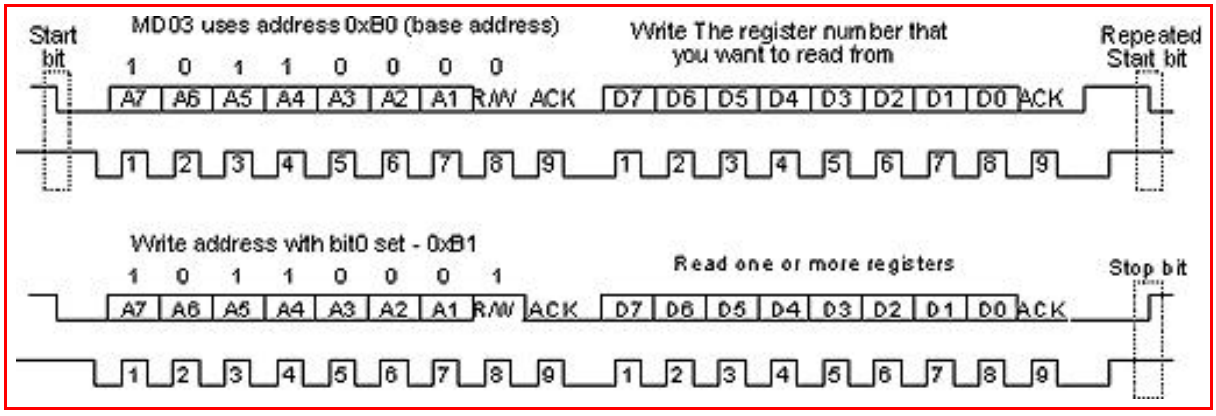
Cette carte est un module de puissance avec "pont en H" destiné à piloter des moteurs "cc" via de votre microcontrôleur, votre module PICBASIC, etc... Ne nécessitant que 2 sources d'alimentation (+5 V/50 mA pour la logique et 5 à 50 Vcc / 5A max. pour les moteurs), elle pourra être très facilement pilotée à partir de 5 types de signaux différents:

Types de signaux de commandes reconnus par la carte:

- Interface I2C™ (8 cartes pourront être ainsi pilotées et différents paramètres tels que la vitesse, l'accélération, la valeur indicative de la température et du courant de limitation, etc... peuvent être géré).



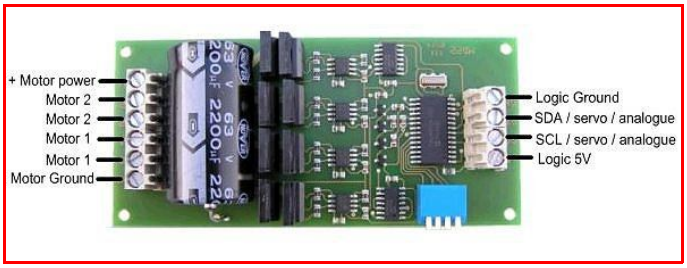
De part les courants importants pouvant être mis en œuvre, il sera impératif d'utiliser un fusible de protection externe et si possible d'utiliser 2 sources d'alimentation séparées (ne pas relier les masses entre elles - ceci est déjà fait au niveau de la carte).



- Signal analogique: 0 - 2,5 - 5 V
  (0v ➞ marche arrière pleine vitesse - 2,5v ➞ Stop - 5v ➞ Marche avant pleine vitesse)
- Signal analogique: 0 - 5V avec signaux de sélection de sens séparés
- Mode RC (pilotage via récepteur modélisme)
- Via un signal "PWM"

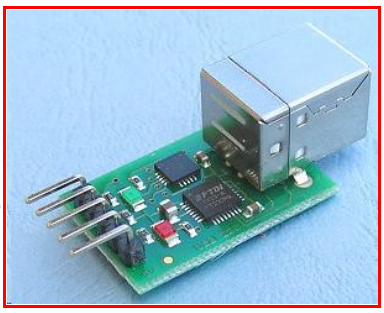
Voir la notice détaillée :

md22tech.htm



Le module MD22 sera piloté par le module de communication Usb/I2c.

2) Module de communication Usb/I2c



Spécifications techniques:

Le module Usb/I2c fournit une interface complète entre votre PC et le bus I2C. Le module est auto alimenté par le câble USB et peut fournir jusqu'à 70mA à 5V pour circuits externes à partir d'un port standard USB 100mA. Le module se comporte uniquement en maître I2C.

a) Première étape - installer les “drivers”

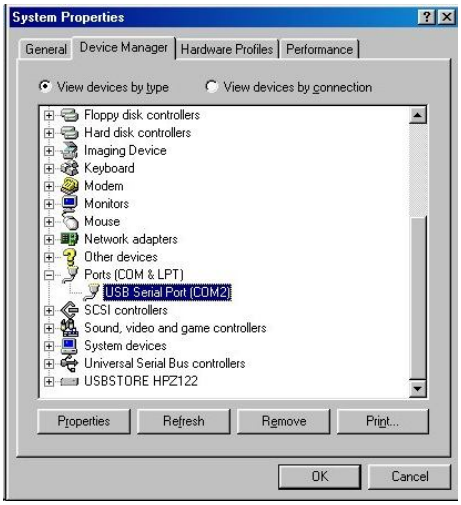
Le module Usb-I2c utilise la puce FTDI FT232R USB pour gérer tous les protocoles USB. Avant d'utiliser l'USB-I2C, Il faut installer les pilotes port COM virtuel FTDI (VCP).

Ces pilotes sont reconnus par le système comme un port COM supplémentaire. Le logiciel d'application accède au périphérique USB de la même manière que pour accéder à un port COM standard de Windows à l'aide d’appels de méthodes API ou en utilisant une bibliothèque de Port Com.

Les pilotes sont disponibles pour Windows, Apple, Linux et Open BSD systèmes directement à partir du site FTDI . Vous est nécessaire d’installer les pilotes, avant de brancher le module Usb/I2c à votre ordinateur.

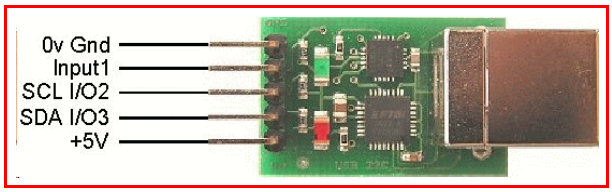
Quel port COM ?

Après avoir installé les pilotes, et brancher le module Usb/I2c sur un port USB libre, il faut connaître le port COM assigné. Cela varie d'un système à l'autre selon le nombre de ports COM que vous avez installée. Puis le port COM doit être mis en place pour une communication à 19200 bauds, 8 bits de données, pas de parité et deux bits d'arrêt.



b) Connexions

Le schéma ci-dessous montre les connexions I2C.



0v Gnd : La broche doit être connectée au 0V (Masse) sur votre périphérique I2C.

+5 V : L'alimentation +5 V du module USB-I2C peut fournir jusqu'à 70mA à des appareils externes. Si votre périphérique I2C nécessite plus que cela, ou possède sa propre source, laissez la broche +5 V non connecté.

Input 1 : La pin est en fait la ligne de réinitialisation du processeur et est utilisé dans notre atelier pour programmer le processeur après le montage final. La fonction de remise à zéro a été désactivé dans le logiciel de telle sorte que cette pin peut être utilisé en tant que broche d'entrée. Il a une résistance pull up de 47k sur le PCB, si l'entrée n'est pas nécessaire, vous pouvez simplement l'ignorer.

SCL et SDA : Ces broches sont les connexions du bus I2C. Ils doivent être connectés directement aux broches SCL et SDA sur votre périphérique I2C. Le module USB-I2C est toujours un maître de bus, et est équipé de résistances pull-up de 4,7K sur le PCB.

Commandes:

Commandes	Valeurs	Descriptions	Disponible en version Usb/i2c
I2C_SGL	0x53	- Lecture / écriture octet non enregistrés périphériques, tels que le Philips PCF8574 - I / O puce.	Tous
I2C_MUL	0x54	- Lire plusieurs octets sans mettre la nouvelle adresse (eeprom, les capteurs de pression Honeywell, etc.)	V5 et supérieur
I2C_AD1	0x55	Read / Write octets simples ou multiples avec 1 octet de registre pour les appareils adressés (la majorité des appareils utiliseront celui-ci)	Tous
I2C_AD2	0x56	Read / Write octets simples ou multiples pour 2 octet de registre des appareils adressé, (Circuits mémoires : EEPROM de 32kbit (4kx8) et plus)	V6 et supérieur
I2C_USB	0x5A	Une série de commandes pour le module USB-I2C, généralement pour améliorer les communications sélectionnées. (voir notice complète)	Tous

Le module Usb-I2c prend soin de tous les besoins du bus I2C comme le démarrage / redémarrage / arrêt, séquençage et gère les cycles de reconnaissance. Il suffit de fournir une chaîne d'octets pour dire au module quoi faire :

- une commande d'un octet propre au module,
- l' adresse du circuit I2c,
- 0,1 ou 2 octets pour les registre des adresses internes des circuit I2c ,
- 0 ou 1 octet, nombre d'octets de données,
- puis lors de l'écriture, les octets de données .

Dans sa forme la plus simple, la chaîne d'octets contient :
- 2 octets : - 0x53, 0x41, qui lit les entrées du circuit PCF8574 I / O et retourne 1 octet lu.
- 3 octets : - 0x53, 0x40, 0xD5 qui écrit sur les sorties du circuit PCF8574 I / O l'octet 0xD5

Écriture sur les dispositifs I2C avec un 1 octet de registre interne lié au circuit I2c:

Il inclut presque tous les périphériques I2C. Suite à la commande, I2C_AD1, on envoie l'adresse du périphérique I2c, puis l'adresse internes du circuit I2c ou l'on veut écrire, le nombre d'octets à écrire ainsi que les octets de données. Le nombre maximal d'octets de données ne doit pas dépasser 64 pour ne pas faire déborder le tampon Usb-I2c interne.

Voici un exemple d'écriture : une séquence de 8 octets pour initialiser le moteur commander par la carte MD22:

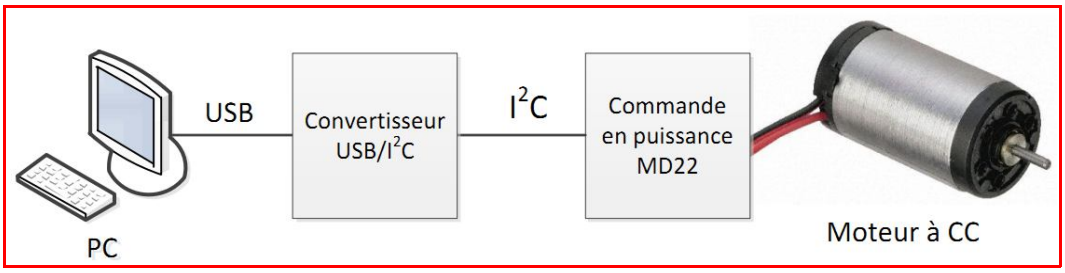
I2C_AD1	Adresse carte MD22 + R / W	Registre Mode	Nombre d'octets de données	Carte MD22 en mode 1	Moteur gauche arrêté	Moteur droit arrêté	L'accélération rapide
0x55	0xB0	0x00	0x04	0x01	0x00	0x00	0x02

le module Usb-I2c répond par non nulle, si l'écriture a réussi, et zéro si elle a échoué. Un échec signifie qu'aucun circuit ne reconnaît la commande.

- Voir la notice complète du module Usb/I2c pour une utilisation approfondie:

USB-I2C Communications Module.pdf

3) Travail demandé :



a) Etude du bus I2c

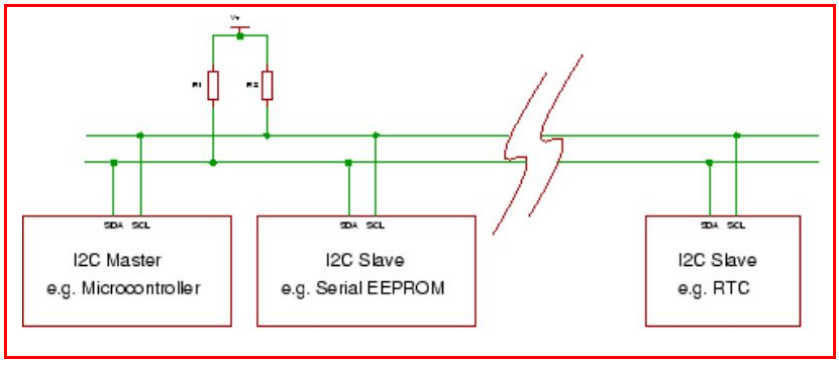
Question 1 : Comment appelle-t-on le périphérique qui génère le signal d'horloge SCL ?

Question 2 : Quelle est la structure de la trame I2c (Les éléments qui constituent la trame) ? Expliquez, dans un tableau récapitulatif, le nom, la taille en bits et la signification de chaque élément de cette trame.

Question 3 : Comment un périphérique peut-il communiquer avec un autre en lecture comme en écriture (Expliquez la différence entre lecture et écriture) ?

Question 4 : Dessinez les chronogrammes i2c (SCL et SDA) pour la transmission en écriture de l'octet $27 à l'adresse $B0 (On supposera la présence du ACK)

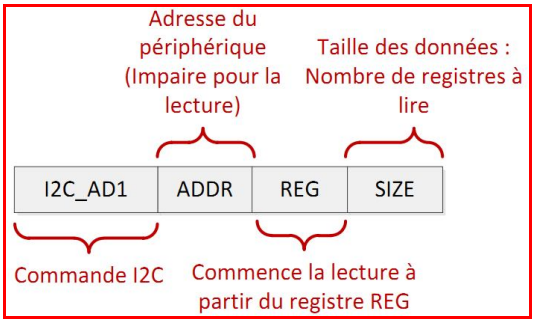
Question 5 : D'après le schéma électrique ci-dessous représentant un exemple de mise en œuvre du bus I2C. Quel est le rôle joué par les résistances de « pull-up » connectées sur SDA et SCL ?



b) Interface Usb/I2c

Le mode utilisé pour dialoguer avec le module MD22 sera le mode I2C_AD1 (Voir registres internes du MD22)

Exemple de trame de lecture envoyée en mode I2C_AD1 :



Question 6 : D'après la documentation, quelle serait la trame permettant de lire tous les registres du contrôleur MD22 ? Détaillez tous les champs de la trame.

c) Contrôleur MD22

Question 7 : Quel est le rôle d'un pont en H (H bridge) appliquée à la commande de moteur ? (Explications avec un graphique)

Question 8 : Quels sont les registres internes du MD22 et quels sont leurs rôles respectifs ? (Tableau récapitulatif)

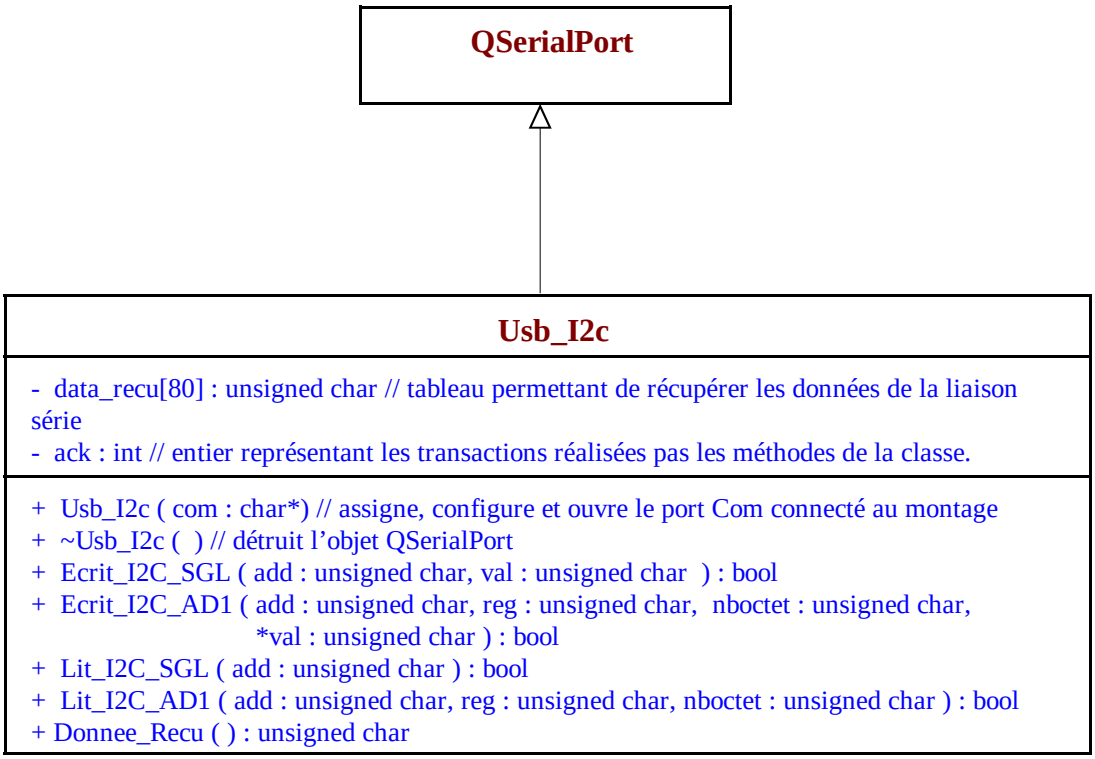
Question 9 : Comment peut-on modifier l'adresse I2C du module MD22 ? Quel paramétrage permet-il de configurer l'adresse 0xB0 ?

d) Codage du programme

Le programme doit être capable de :

- Assurer l'envoi et la réception de données via le module I2C/USB en configurant le port série associé de manière appropriée.
- Configurer le mode de fonctionnement du MD22 et commander la vitesse des moteurs.

1) En vous aidant du cours sur le bus **I2c** : fichier "**Cours_I2c.pdf** ", et de la documentation technique du module **Usb/I2c** et de la carte contrôleur **MD22**, développer une nouvelle classe "**Usb_I2c**" dérivant de la classe **QSerialPort** , en C++, qui génère le protocole des lignes Sda et Scl du module de communication tel que:



Programmation de la liaison COM d'un PC sous Qt :

La configuration de l'**UART** ,1'écriture, ou la lecture dans ses registres sont réalisées à l'aide des fonctions **API** de communications. On entend par **API** (**Application Programming Interface**), l'ensemble des fonctions systèmes de l'OS qui peuvent être appelées. Sous Qt5, la classe **QSerialPort** fournit des classes pour le contrôle des ports séries.

Manipulation, sous Qt5, de la classe QSerialPort:

Pour l'utilisation de la classe **QSerialPort**, il faut ajouter le composant **QtserialPort** lors de l'installation du système de développement Qt5.

Puis, dans le fichier **.pro** de votre projet, on ajoute la commande suivante :

```
...
QT +=  serialport
...
```

La manipulation de votre port série s'effectue par l'intermédiaire d'un objet **QserialPort**, ne pas oublier d'ajouter dans votre fichier .h :

```
#include <QtSerialPort/QSerialPort>
```

- Instancier l'objet
- ouvrir la connexion à l'aide de la méthode:

```
bool  open(OpenMode mode);
```

- Configurer la liaison à l'aide des méthodes:

```
bool  setPortName(const QString &name);
bool  setBaudRate(BaudRateType);
bool  setFlowControl(FlowControl flow)
bool  setDataBits(DataBitsType);
bool  setParity(ParityType);
bool  setStopBits(StopBitsType);
```

- Communiquer avec votre port série à l'aide des méthodes:

```
qint64 readData(char *data, qint64 maxSize);
qint64 writeData(const char *data, qint64 maxSize);
```

Pour des informations complémentaires, consulter l'aide interne de Qt5:

QSerialPort Class

Le port Com doit être mis en place pour une communication à 19200 bauds, 8 bits de données, pas de parité et deux bits d'arrêt.

Développer les méthodes:

+ **Usb_I2c (com : char *) ,** constructeur de la classe qui assigne et configure et ouvre le port **Com** connecté au montage.

+ **~Usb_I2c ( )** , Cet méthode ferme l'accès à la communication du port Com.

+ **Ecrit_I2C_SGL (add : unsigned char, val : unsigned char) : bool ,** méthode qui envoie, par l'intermédiaire du port Com , l'octet **val** au circuit I2c dont l'adresse est **add** à l'aide de la commande **I2C_SGL.** La fonction renvoie 0, si tout c'est bien déroulé.

+ **Ecrit_I2C_AD1 (add : unsigned char, reg : unsigned char, nboctet : unsigned char, *val : unsigned char) : bool** , méthode qui envoie, par l'intermédiaire du port Com , plusieurs octets (nboctet) contenu à l'adresse du pointeur val au circuit I2c (adresse add) au départ de l'emplacement reg à l'aide de la commande **I2C_AD1.** La fonction renvoie 0, si tout c'est bien déroulé.

2) A partir d'une nouvelle application, tester votre classe **Usb_I2c** en instanciant un objet moteur et réaliser les tests unitaires pour chaque méthode.

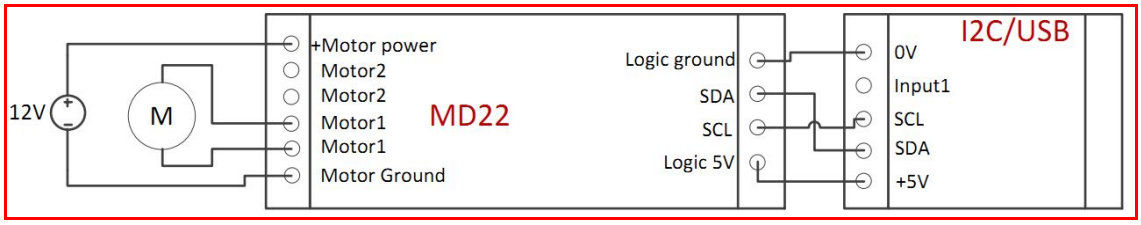
3) Déterminer, à partir de la notice constructeur, l'adresse ainsi que les données à envoyer pour actionner le moteur à 25% de sa vitesse maximum , l'arrêter, dans un sens puis dans l'autre, si vous effectuez une connexion **I2c** avec le module **MD22**.

4) Développer l'application du module MD22 pour piloter un moteur à courant continu comme dans l'exemple proposé.

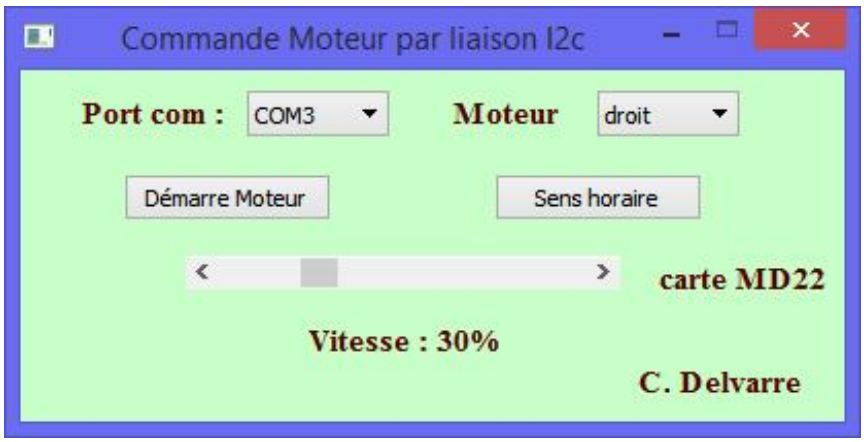
Fichier "Tp_Commande_Moteur_Del.exe"

5) Test sur un moteur à CC

L'objectif est de connecter le moteur afin de mettre celui-ci en mouvement. Voici le montage à réaliser :



Réaliser le câblage et effectuer les tests sur le PC dédié avec votre programme



Bon courage ...