Utilisation de la carte d'acquisition/commande ARDUINO
Connexion à un/des moteur/s de type MCC

Arduino est un circuit imprimé qualifié de libre et open-source (les plans sont consultables et tout le monde peut fabriquer une copie exacte de l'Arduino, contrairement à la plupart des objets manufacturés de notre environnement comme les machines à laver, les téléphones, les ordinateurs). Seuls le nom et le logo sont réservés. Sur Arduino se trouve un microcontrôleur (calculateur) qui peut être programmé pour analyser et produire des signaux électriques.

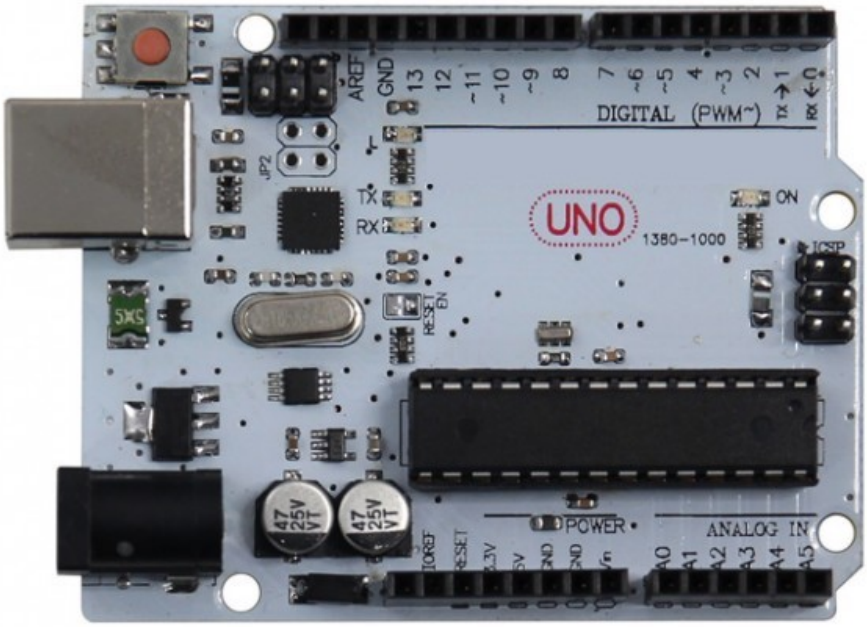
1. Installation du PC de la salle de TP sous Windows 7, 10 ou 11 ou autre

Ne sachant pas si le disque a été réinstallé avec ou sans les programmes, drivers et bibliothèques nécessaires à l'utilisation de la carte ARDUINO, il convient peut-être de commencer par :

1. 1 câbler la carte sur le PC en USB
1. 2 charger le driver de la carte
1. 3 déposer le répertoire de l'IDE Arduino sur le Bureau (`https://www.arduino.cc`)

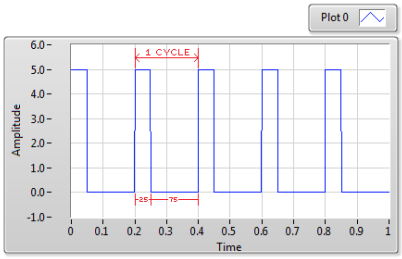
2. Carte ARDUINO

Préliminaires : cette carte est une carte contrôleur qui permet des entrées analogiques ou numériques et des sorties numériques en tension.
Entrées analogiques : ANALOG IN, 6 entrées numérotées A0 à A5
Entrées/Sorties numériques : DIGITAL, 14 entrées notées 0 à 13, masse notée GND, les bornes 0 et 1 étant réservées. Six de ces entrées/sorties peuvent assurer une sortie PWM (Pulse Width Modulation - Modulation de Largeur d'Impulsion, une astuce pour modifier le courant de sortie). Les 6 PWW sont les numéros 3, 5, 6, 9, 10, 11. Les 14 bornes reçoivent ou envoient des signaux numériques (donc 0 ou 1) et ces signaux se traduisent par 0V ou 5V. Leur puissance est limitée à 40 mA par broche pour un total de 200mA consommé.
Le fonctionnement (entrée ou sortie) est fixé dans le programme (INPUT, OUTPUT).



L'alimentation et la connexion au PC se fait à l'aide de la borne USB-B.
On peut également alimenter avec une source externe comme une pile.
Une fois un programme chargé dans la (petite) mémoire de la carte, elle devient autonome.

3. Fonction PWM



Les sorties numériques de l'ARDUINO peuvent, pour certaines, délivrer un signal créneau dissymétrique PWM (Pulse Width Modulation) ou MLI (Modulation de la Largeur d'Impulsion).

```
analogWrite(brochePWM, sensorValue);
```

On a ici la même forme de signal que ce qui est obtenu par hachage d'une tension continue.

4. Contrôle de l'ARDUINO : réalisation de "croquis"

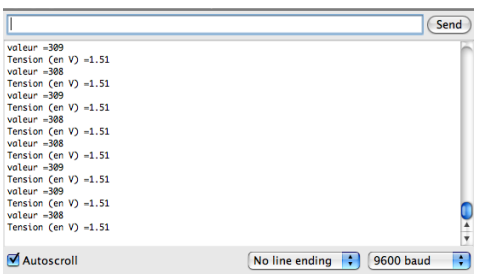
Les croquis ("sketches") sont des programmes en C++ qui sont compilés en chargés dans la mémoire de la carte ARDUINO et exécutés immédiatement.
La structure d'un programme est la suivante :

```
void setup() {
        // on met ici le code qui ne doit être exécuté qu'une fois:
}
void loop() {
        // on met ici le code qui  doit être exécuté en boucle :
}
```

Un exemple maintenant :

```
/*
AnalogReadSerial : Reads an analog input on pin 0, prints the result to the serial monitor.
*/
// the setup routine runs once when you press reset:
void setup() {
        // initialize serial communication at 9600 bits per second:
        Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
        // read the input on analog pin 0:
        int sensorValue = analogRead(A0);
        // print out the value you read: retour à la ligne avec "ln"
        Serial.println(sensorValue);
        delay(1);        // delay in between reads for stability
}
```

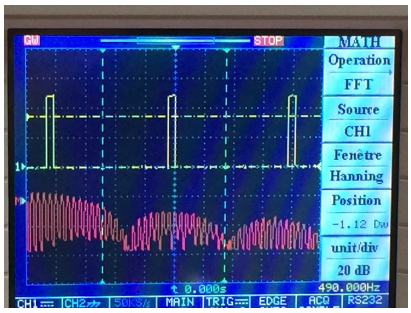
La sortie des données se fait alors sur le port série vers le PC. Sur ce dernier, on peut afficher facilement le traffic de données en ouvrant le "Moniteur Série" de l'IDE Arduino.



Dans cet exemple, on lit une valeur numérique de tension sur la borne $A0$.
Attention, par rapport au Python, ce langage est "typé statique" donc on doit déclarer le type de variable utilisé, comme le "int" utilisé pour dire que la variable "sensorValue" sera un INTEGER

5. Contrôle du α du PWM

Le α correspond au pourcentage de la période T qui est dans un état "haut" ($5\,V$ en tension logique).
Avec un hacheur, on contrôle le α à l'aide d'un signal à haute fréquence qui commande la base de l'interrupteur commandé.
Avec la carte Arduino, les bornes numériques peuvent délivrer du "tout ou rien" (donc 0 ou $5\,V$) , mais aussi un signal "PWM sur certaines bornes, à haute fréquence, de valeur moyenne réglable informatiquement avec `digitalWrite(pin, value)`
La variable `value` admet une donnée 8 bits entre 0 et 255.
Visualiser sur un oscilloscope la forme du signal récupéré sur la borne 3 par exemple.




```arduino
//étape 1 = lire le signal PWM sur l'oscilloscope
const int brochePWM = 3;
// valeur par défaut
int sensorValue = 50;
void setup() {
// ouverture du port de communication à la vitesse de 9600 bauds
Serial.begin(9600);
// configuration broche 3 en sortie
pinMode(brochePWM, OUTPUT);
}
void loop() {
//  réglage logiciel fixe de la valeur de $\alpha$
 analogWrite(brochePWM,sensorValue);
}
```

Changer dans le programme la valeur `int sensorValue = 100;` et observer les modifications.

Quelle valeur donner à `sensorValue` pour avoir $\alpha = 0,75$? Pour avoir $\alpha = 0,10$?
Quelle remarque quand au spectre de Fourier de ce signal ? Expliquer bien sa forme attendue.

6. Génération d'un signal carré

Il existe une fonction *tone*(*borne*, *frequence*, *duree*).
Allez voir la documentation Arduino : `https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/`
Visualiser son spectre de Fourier. Comment obtenir une tension sinusoïdale à partir de cela ?

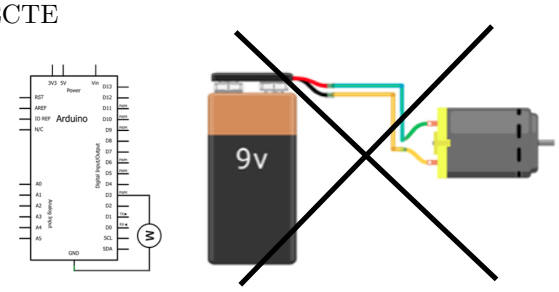
7. Modification de la fréquence du PWM

```arduino
int e; // Le signal d'entrée
int s=0; // Le signal de sortie
int entree = A0; // La broche d'entrée
int sortie = 9; // La broche de sortie
void setup () {
TCCR1B = 1; // Initialisation (PWM a 31KHz sur broches 9,10)
}
```

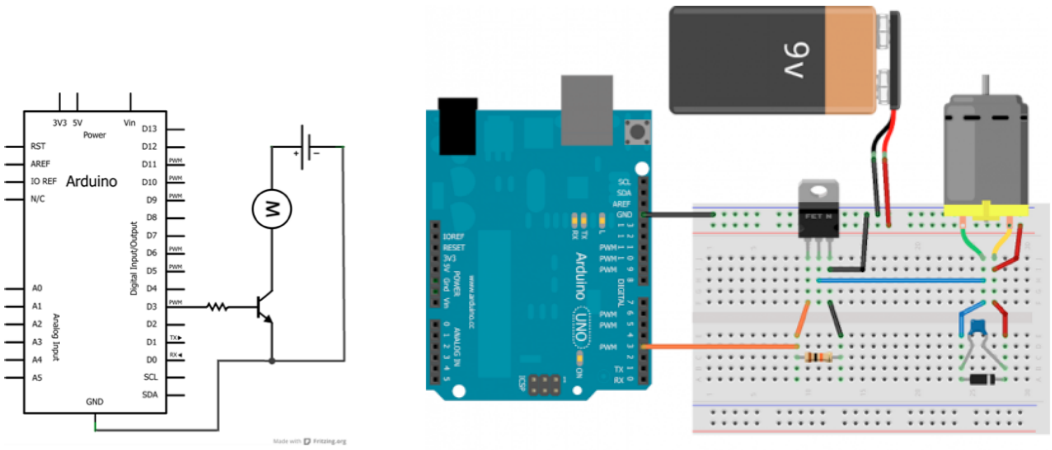
```
8  void loop() {
9  e=analogRead(entree)/4; // Entree ramenée sur 8 bits
10 s=s+101*(e-s)/256; //
11 analogWrite(sortie ,s);
12 delay(1); // Te=1ms (fe=1kHz)
13 }
```

8. Application à la commande du ω d'un moteur

PAS DE CONNEXION DIRECTE



Il faut un circuit extérieur commandé par un interrupteur.

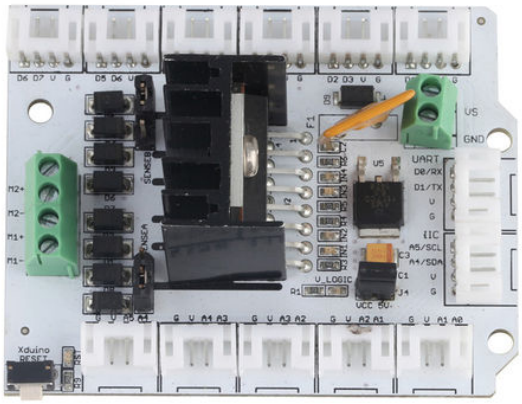


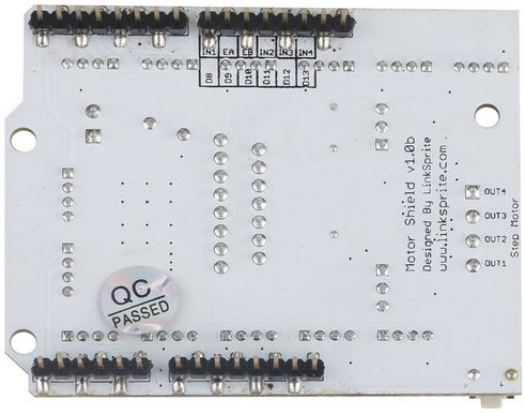
Expliquer comment ce montage permet de commander une MCC.

UTILISATION DE CARTES SPÉCIFIQUES :

Il existe aussi des "Shields" qui viennent s'emboiter sur la carte Arduino et permettant de connecter et contrôler des moteurs plus facilement.

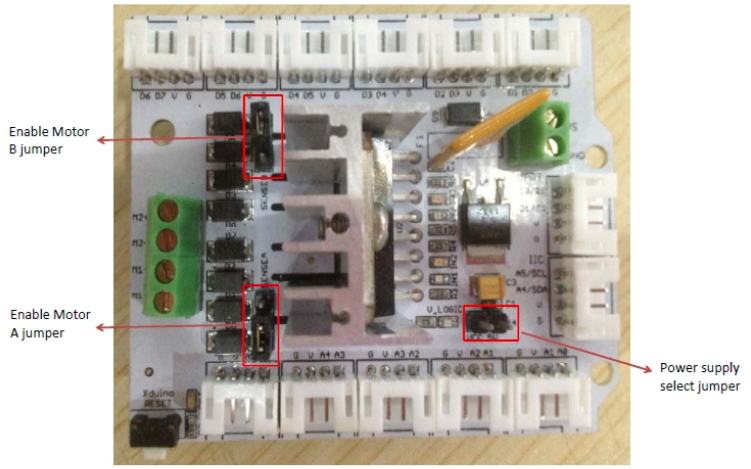
```
int speedpinA=9;//active motor A
int speedpinB=10;//active le motor B
int speeed =127;//definit la vitesse du moteur 0 à 255
analogWrite(speedpinA,speeed);//AnalogWrite pour générer PWM
analogWrite(speedpinB,speeed);
```





Item	Min	Typical	Max	Unit
Logic Control Voltage	4.5	5	5.5	V
Motor Supply Voltage	6	/	15	V
Output Voltage	0	/	Vinput -1	V
Output Current(For Each Channel)	/	/	2000 *	mA
Output Duty range	0%~100%			/
Dimension	68.5x54.5x29.5			mm
Net Weight	37			g

Pin Name	Direction	Connection to Arduino	Description
VCC	/	VCC	Power supply selector
VS, GND	/	/	Power Supply for Motor, 6V - 15V
EA	Input	D9	TTL Compatible Enable Input of Bridge A
EB	Input	D10	TTL Compatible Enable Input of Bridge B
IN1	Input	D8	TTL Compatible Inputs of the Bridge A
IN2	Input	D11	TTL Compatible Inputs of the Bridge A
IN3	Input	D12	TTL Compatible Inputs of the Bridge B
IN4	Input	D13	TTL Compatible Inputs of the Bridge B
M1-, M1+	Output	/	Outputs of the Bridge A
M2-, M2+	Output	/	Outputs of the Bridge B



Enable Motor B jumper: Connect above two interfaces with a jumper cap when driving the DC motor A.

Enable Motor A jumper: Connect below two interfaces with a jumper cap when driving the DC motor B.

LIEN vers le site du fabricant + code :

• Lien général :

http://linksprite.com/wiki/index.php?title=Motor_Shield

• Câblage et code Arduino :

http://learn.linksprite.com/pcduino/arduino-ish-program/pwm/run-motor-shield-on-pcduino/

9. Moteur synchrone

Cas des "stepper motor" = moteurs pas-à-pas

Commentez les documents suivants.
https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/747_le-mouvement-g
3439_a-petits-pas-le-moteur-pas-a-pas/

https://osoyoo.com/2017/07/10/arduino-lesson-stepper-motor/

https://elec13.wordpress.com/2016/03/19/controler-un-moteur-pas-a-pas-a-5-fils-avec-arduino/

http://www.geeetech.com/wiki/index.php/Stepper_Motor_5V_4-Phase_5-Wire_%26_ULN2003_Driver_Board_
for_Arduino

But : faites le lien avec la nécessité de créer un champ tournant. Expliquer la commande des bobines des exemples cités

10. MLI

Cas dans le sujet 2016

MLI : Modulation de largeurs d'impulsion

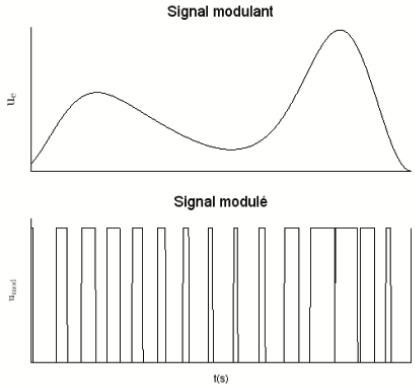
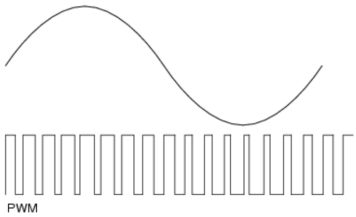
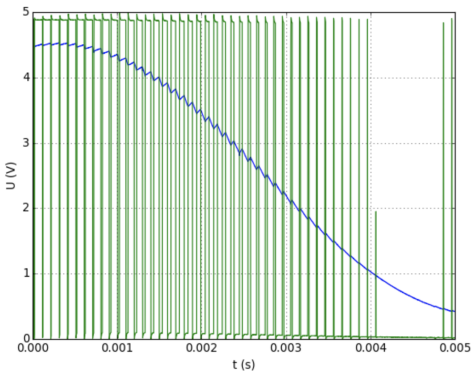


Figure A.1

Le principe consiste à générer des impulsions électroniques à intervalles régulières mais dont la largeur temporelle va dépendre d'un signal « modulant ». C'est la valeur de tension du ce signal qui va déterminer la largeur de l'impulsion (cf. Figure A.1).

On veut ici "simuler" puis reformer un signal sinusoïdal à partir d'un signal MLI avec la fonction `AnalogWrite()` donc :





Expliquer les différentes méthodes, y compris en expliquant pourquoi on a parfois des codes avec sinus "précalculé"