Manuel de Laboratoire de Programmation Arduino

Objectif

Utiliser la plateforme Arduino IDE Software pour écrire des commandes logiques de contrôle simples. Ce code sera utilisé pour contrôler une variété d'entrées et de sorties connectées à un microcontrôleur Arduino Uno. Ce laboratoire exposera également les élèves à des codes de contrôle plus compliqués et des codes de communication ainsi que des ressources en ligne disponibles pour le logiciel Arduino.

Contexte

En conjonction avec les différents modèles de microcontrôleurs, la société Arduino fournit un environnement de développement intégré (IDE) Source Ouverte Arduino. Cet environnement de programmation est écrit en Java et a été conçu pour introduire la programmation à un public qui est moins familier avec le codage. Les programmes écrits dans cet environnement sont appelés «croquis» et le langage utilisé simple avec un support pour les langages de programmation C et C++.

Un croquis Arduino de base se compose d'au moins deux fonctions: une fonction 'setup' et une fonction 'loop'. La fonction « setup » ne s'exécute qu'une fois au début de l'exécution du programme. Il exécute toutes les actions qui sont initialement nécessaires pour exécuter le reste du programme, comme l'initialisation de tous les composants externes et le réglage de la fréquence de communication entre la carte Arduino et le PC. La fonction « loop » agit en tant que pilote de programmes et s'exécute dans une boucle continue (c'est-à-dire "boucle pour toujours"). Cette boucle spécifie l'ordre des opérations que le microcontrôleur exécutera sur une base continue.

Les cartes Arduino, ainsi que de nombreux composants externes compatibles Arduino, contiennent du matériel pour la communication série. Cette forme de communication peut envoyer des données à une vitesse élevée, mais transmet des bits de données en série (c'est-à-dire l'un après l'autre, un bit unique à la fois). Cela permet aux composants du système (ou d'un PC) de communiquer des séquences de données plus grandes et plus complexes avec beaucoup plus de facilité, avec moins de contraintes sur les broches physiques d'entrée ou de sortie.
En raison de la nature d'open-source d'Arduino, les sociétés tierces, ainsi que les utilisateurs finaux sont libres de prendre le logiciel et de le personnaliser selon leurs besoins individuels. Cela signifie que pour presque n'importe quelle application, les croquis sont disponibles pour téléchargement en ligne. Les entreprises fabriquant des périphériques externes (comme un écran de moteur ou des

capteurs) à utiliser avec les cartes Arduino ont souvent des croquis d'exemple et des bibliothèques de croquis disponibles pour une utilisation spécifique pour leurs produits. Ces ressources peuvent être utilisées pour créer un croquis personnalisé qui peut utiliser plusieurs périphériques à la fois, en combinant des aspects de nombreuses esquisses d'exemples différents.

Aperçu des Appareils et Équipement

L'équipement qui va être utilisé pour ce lab inclus:

- 1 x Microcontrôleur Arduino Uno R3
- 1 x Cable USB 2.0 Type A à USB 2.0 Type B
- 1 x Ordinateur de Lab ou Personnel (avec Windows, Macintosh, ou Linux OS)
- 1 x Capteur Ultrason (HC - SR04)
- 1 x Blindage Contrôleur de Moteur Adafruit V1 (L293D)
- 2 x Moteur CC
- 1 x plaque de prototypage
- 4 x cables Male-Male
- Logiciel Arduino 1.8.2 IDE
- Bibliothèques correspondantes Arduino IDE (expliqué plus bas)
- 4 x Fil 22 Ga.

Préparation Pré-Lab

Avant d'arriver dans le lab, les étudiants devraient réviser le manuel de lab et se familiariser avec les appareils du lab et les procédures. Les étudiants peuvent utiliser leur ordinateur pour ce lab ou s'ils veulent, à condition qu'ils téléchargent Arduino IDE ainsi que les bibliothèques requises, souligné dans la section téléchargements. La révision du syntaxe d'Arduino IDE ainsi que d'essayer d'écrire certains programmes l'avance est encouragé. Une liste utile de commandes utilisées dans Arduino peut être trouvée ici: https://www.arduino.cc/en/Reference/HomePage.

Téléchargements

Le Arduino IDE peut être téléchargé ici : https://www.arduino.cc/en/Main/Software
Les bibliothèques utilisées dans ce laboratoire peuvent être installées via le gestionnaire de bibliothèques de l'IDE Arduino.

- Adafruit Motor Shield V1 (AFMotor.h)
- NewPing (NewPing.h)
- SoftwareSerial (SoftwareSerial.h- celui-ci est pré-installé dans Arduino)

Pour installer les bibliothèques nécessaires, vous devez accéder au gestionnaire de bibliothèques dans l'IDE Arduino. Ouvrez l'IDE Arduino et naviguez vers Tools>Manage libraries.

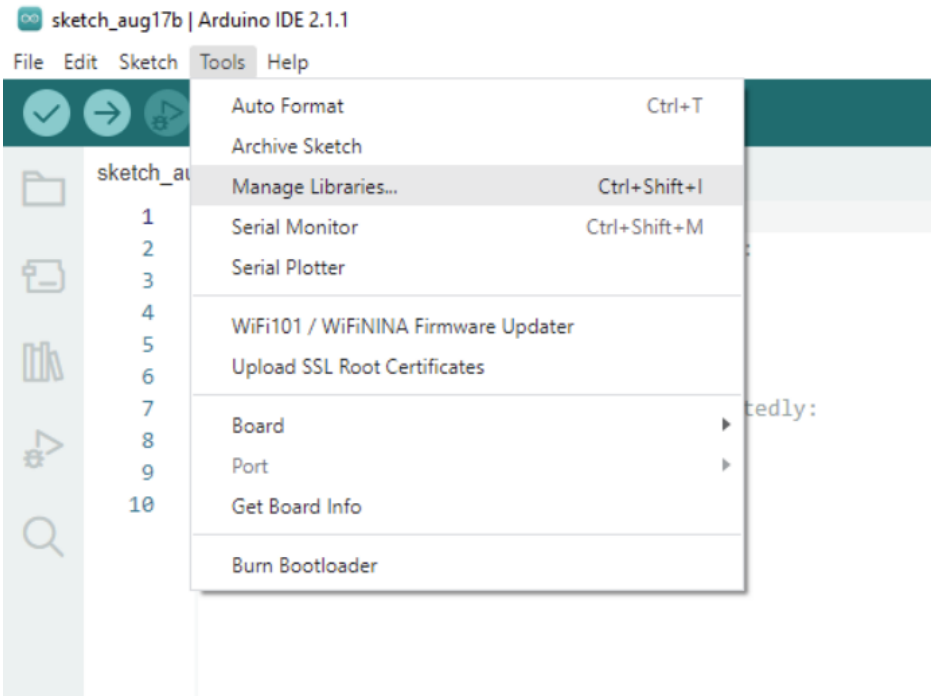


Figure 1: Bibliothèque d'Arduino

L'onglet du gestionnaire de bibliothèques devrait s'ouvrir sur le côté gauche de la fenêtre. Recherchez maintenant les bibliothèques requises énumérées ci-dessus et installez-les. Procédez ainsi pour chaque bibliothèque. Il se peut que vous deviez redémarrer l'IDE.

Si vous avez des bibliothèques qui doivent être importées dans l'IDE, il est possible d'utiliser le fichier ZIP. Allez dans Sketch>Inclure une bibliothèque>Ajouter une bibliothèque .ZIP... et

recherchez et sélectionnez le dossier qui contient la ou les bibliothèques qui doivent être ajoutées.

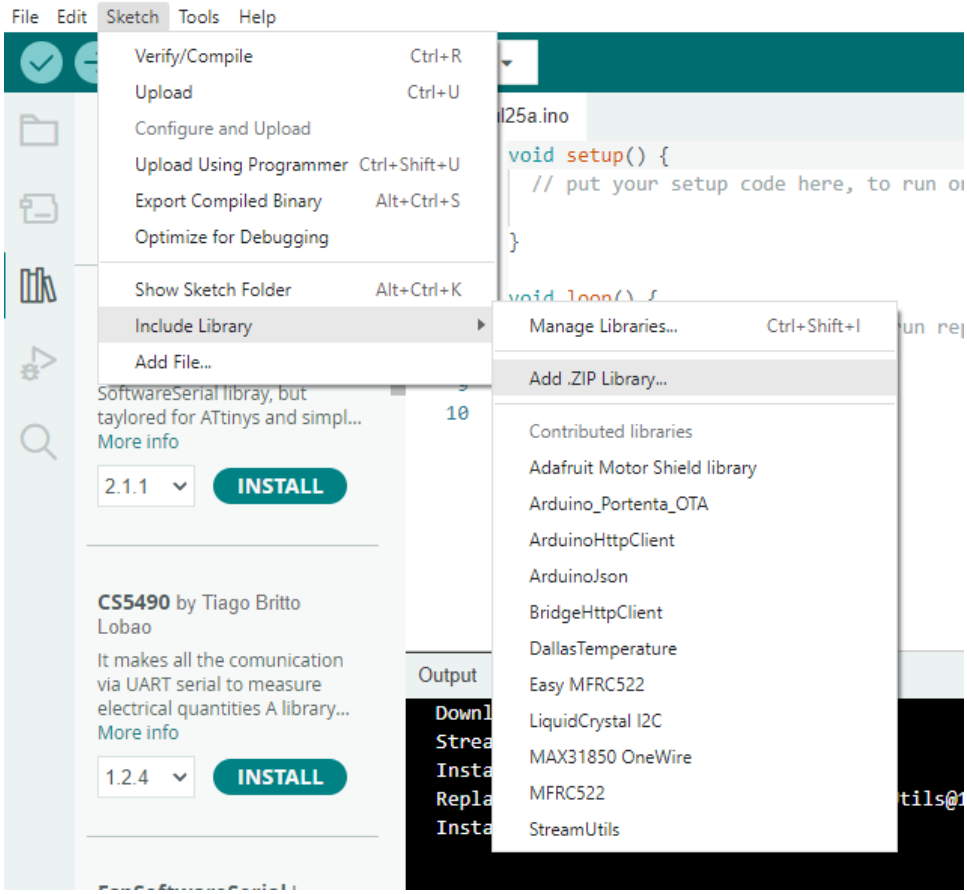


Figure 2 : Bibliothèque inclus à Arduino

Questions de Révision

C’est quoi le rôle d’un microcontrôleur?

Quel microcontrôleur est-ce qu’on utilise dans ce lab?

Comment un capteur ultrason peut mesurer une distance?

_____

_____

Comment nos moteurs vont être contrôlés pour tourner dans les deux sens?

_____

_____

Quelle méthode de communication sans-fil est-ce qu'on utilise?

_____

_____

Procédure

Des composantes électriques sensibles comme des cartes de contrôle et capteurs utilisés dans ce lab peuvent être endommagés ou brisés d'une décharge statique provenant d'une mauvaise manipulation pendant assemblage ou usage. Avant de préparer ce lab. Les étudiants devraient s'assurer que leur espace de travail est préparé de manière statique (espace de travail non-métal, pas de tapis/autres matériaux fibreux) et ils devraient se 'mettre à terre' en touchant un objet en métal large avant de manipuler les composantes. Il y a des accessoires pour améliorer l'espace de travail d'assemblage pour la résistance statique, comme un tapis antistatique et des bandes statiques de poignet/cheville, qui devraient être utilisés si disponible.

Partie A – Programme DEL clignotante

Ce programme va courir la méthode de base pour se connecter et programmer un microcontrôleur Arduino. Il utilise une programme exemplaire inclus dans une des bibliothèques par défaut de l'Arduino IDE, et c'est principalement utilisé pour vérifier que le microcontrôleur et la connexion fonctionnent bien.

1. **Connectez** la carte Arduino à un port USB ouvert dans l'ordinateur avec le câble USB. La carte Arduino devrait s'allumer car il tire de la puissance à travers la connexion USB.
2. **Ouvrez** le logiciel Arduino IDE et cliquez Tools>Board>Arduino Uno dans le menu déroulant. De retour dans l'onglet 'Tools', **sélectionnez** 'Port' et choisissez le port de série

à laquelle le Arduino est connecté. Si le bon port n'est pas apparent, débranchez l'Arduino et observez quel port disparaît. **Rebranchez-le** et **sélectionnez** ce port.

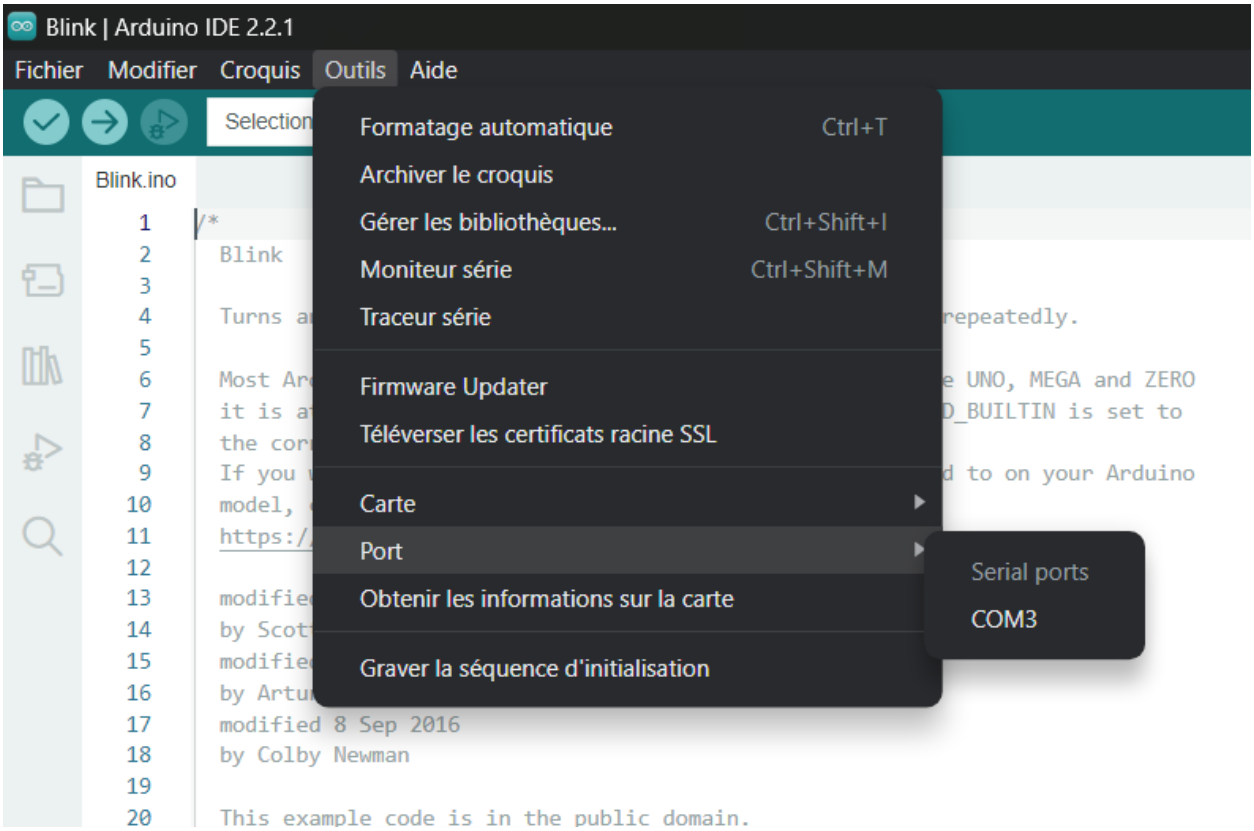


Figure 3 : Sélection du port Arduino

Si le port n'est pas détecté par l'IDE, il peut s'agir d'une erreur de pilote. Essayez de télécharger et d'installer le pilote CH340 suivant sur votre ordinateur : http://www.wch-ic.com/downloads/CH341SER_ZIP.html

3. **Ouvrez** File>Examples>01.Basics>Blink. Ceci va ouvrir un programme qui ressemble à celle-ci-dessous.

```
// la fonction de configuration s'exécute une fois lorsque vous appuyez sur reset ou allumez la
void setup() {
  // initialiser la broche numérique LED_BUILTIN comme une sortie.
  pinMode(LED_BUILTIN, OUTPUT);
}

// la fonction de boucle tourne encore et encore pour toujours
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // allumer le LED (HIGH est le niveau de tension)
  delay(1000);                       // attendez une seconde
  digitalWrite(LED_BUILTIN, LOW);    // éteindre le LED en rendant la tension BASSE
  delay(1000);                       // attendez une seconde
}
```

Figure 4 : Exemple de clignotement

4. Maintenant vérifiez (verify) et téléchargez (upload) le croquis à la carte Arduino. Pour vérifier, **cliquez** le bouton dans le coin en haut à gauche. Le Arduino IDE va essayer de compiler le programme (sans télécharger à la carte) et t'avertir de n'importe quelles erreurs de syntaxe dans la programmation. Quand ceci est terminé, cliquez le bouton télécharger. Le programme va recompiler et télécharger à la carte.

Figure 5 : Vérifier et télécharger les boutons

5. Une fois que le téléchargement est complet, le programme va automatiquement exécuter sur le microcontrôleur. Dans ce cas, la DEL avec l'étiquette 'L' sur le Arduino devrait clignoter lentement. **Montrez** votre travail au TA.
6. **Naviguez** jusqu'à esquisse → Inclure une bibliothèque → Gérer les bibliothèques, une liste de bibliothèques s'affiche.
7. **Recherchez** et **installez** (en passant la souris sur l'élément et en cliquant sur installer) les bibliothèques suivantes :
    a. NewPing
    b. Adafruit Motor Shield Library V1

Partie B – Lecture de Capteur

Ce programme va automatiquement sonder un capteur ultrason, qui cause le capteur à émettre une impulsion ultrasonore. Le programme va lire le temps écoulé entre l'envoi de l'impulsion et la réception d'un écho, convertis le temps en distance en utilisant la vitesse du son et imprime la distance dans sur l'écran d'ordi via la connexion au port de série.

1. **Connectez** un capteur ultrason à l'Arduino en utilisant le diagramme de câblage ci-dessous. La carte Arduino devrait encore être connectée à l'ordinateur via le câble USB.
    a. **Ne suivez pas aveuglément le schéma, assurez-vous que vous connectez GND à GND et Vcc à 5V.**
    b. Si les lumières de l'arduino s'éteignent, cela signifie que vous avez un court-circuit! Débranchez rapidement les fils et vérifiez vos connexions.

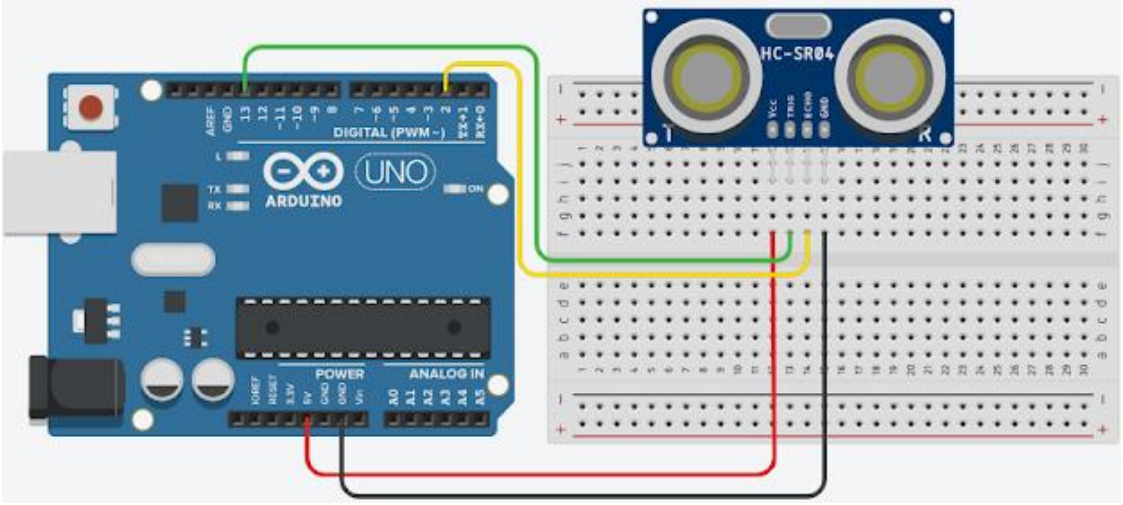


Figure 6 : Câblage du capteur à ultrasons

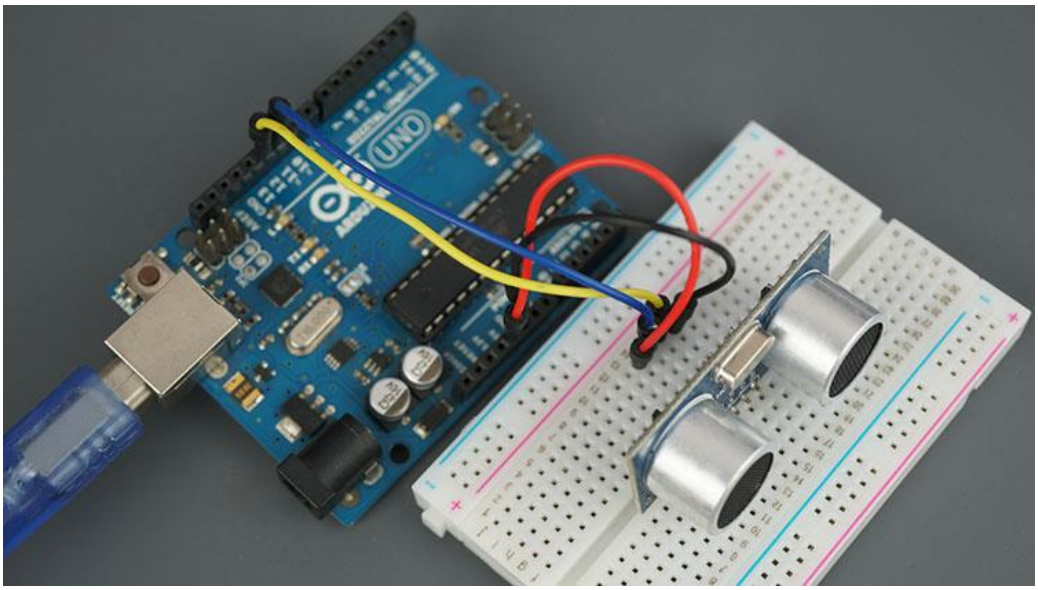


Figure 7 : Placement du capteur dans une plaque de prototypage

2. **Ouvrez** un nouveau programme exemplaire dans le Arduino IDE. Cet exemple spécifique va être inclus dans la bibliothèque "NewPing" qui devrait avoir été ajouté à l'Arduino IDE. **Allez** à File>Exmples>NewPing>NewPingExample.
3. Modifiez les broches de l'Arduino qui sont utilisées pour envoyer et recevoir des données au capteur. **Modifiez** les deux lignes au début du sketch qui définissent les broches d'écho

et de déclenchement (les broches par défaut sont respectivement 11 et 12) pour envoyer un déclenchement par la broche 13 et recevoir un écho par la broche 2 (voir ci-dessous).

```
#include <NewPing.h>

#define TRIGGER_PIN  12
#define ECHO_PIN     11
#define MAX_DISTANCE 200
```

Originel

```
#include <NewPing.h>

#define TRIGGER_PIN  13
#define ECHO_PIN     2
#define MAX_DISTANCE 200
```

Nouveau

Figure 8 : Modification de la broche de déclenchement et d'écho

4. **Compilez** et **téléchargez** cet exemple sur la carte Arduino. Vérifiez que la bonne carte est sélectionnée (Arduino Uno), et que le bon port de communication est utilisé.
5. Une fois que l'exemple est téléchargé, ouvrez le moniteur de série pour voir les données du capteur. **Cliquez** l'icône dans le coin en haut à droite du IDE ou dans Tools>Serial Monitor.
6. Fixez le baud rate du moniteur de série pour correspondre avec le baud rate spécifié dans l'exemple. Sélectionnez le bon baud rate en **utilisant** le menu déroulant dans le coin en bas à gauche de la fenêtre du moniteur de série. Dans ce cas le baud rate devrait être 115200. Après quelques instants, des mesures de distance devraient commencer à apparaître dans la fenêtre du moniteur.
    a. **Si vous voyez 0cm, il est possible que les connexions trig et écho aient été inversées.**

Q1. Quelle est la différence entre les broches ECHO et TRIG du capteur?

_____

_____

Q2. Qu'est-ce que la fonction sonar.ping_cm() calcule?

_____

_____

Partie C – Contrôle de Moteurs CC

Ce programme va tester et contrôler deux moteurs CC en utilisant une séquence de commandes prédéfinie.

1. **Déconnectez** le capteur de l'Arduino. **Insérez** la carte de protection du moteur dans les têtes de l'Arduino.
2. **Connectez** un moteur à courant continu au port M1 du blindage du moteur en insérant les fils du moteur dans le bornier bleu et en utilisant un tournevis pour les serrer en place.

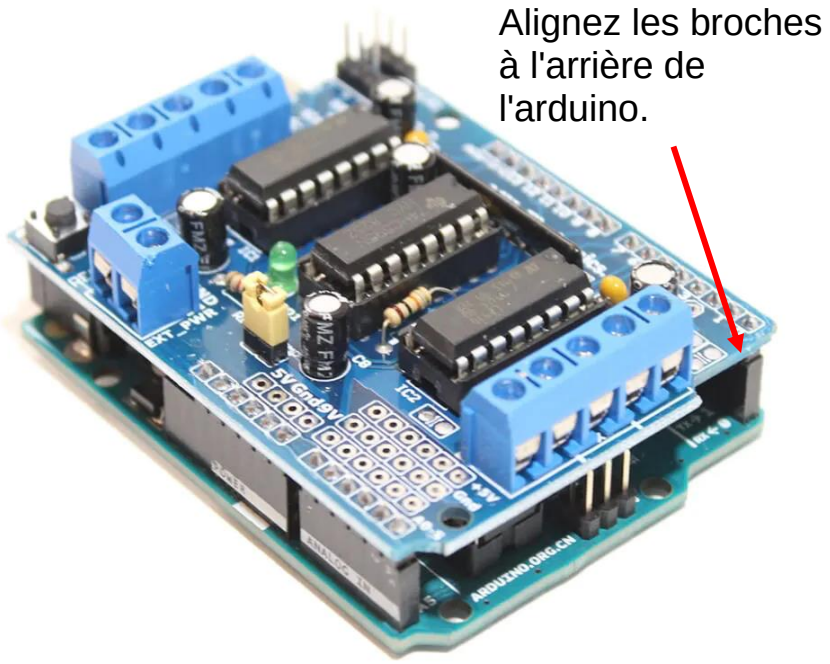


Figure 9 : Blindage moteur connecté à l'arduino

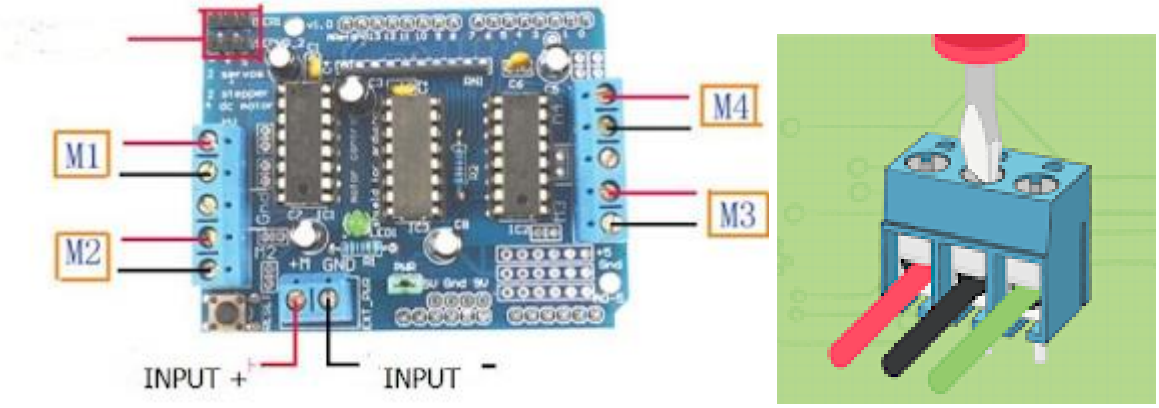
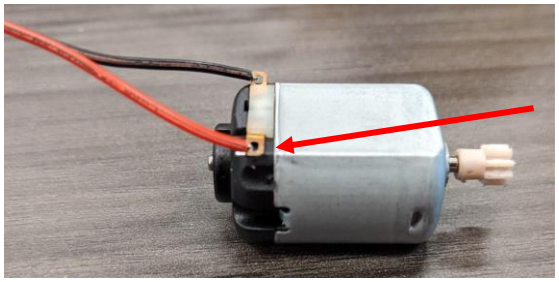


Figure 10 : Aperçu des broches du blindage du moteur



Vous pouvez utiliser du ruban adhésif pour maintenir les fils en place.
*Faites attention aux bornes du moteur, elles sont fragiles.

Figure 11 : Broche du moteur

3. **Ouvrez** l'exemple qui se trouve dans Fichiers→Exemples→Adafruit Motor Shield Library→MotorTest. Ce sketch d'exemple fait fonctionner un moteur à courant continu connecté au port M1 du bouclier moteur.
4. L'exemple de test de moteur contrôle un moteur connecté au port M4 par défaut. Changez ce port en port M1 en **changeant** la ligne montrée ci-dessous. La nouvelle ligne doit se lire "AF_DCMotor motor(1) ;".

```
MotorTest §

// Adafruit Motor shield library
// copyright Adafruit Industries LLC, 2009
// this code is public domain, enjoy!

#include <AFMotor.h>

AF_DCMotor motor(1);
```

Figure 12 : Changement de la broche du moteur de M4 à M1

5. **Compilez** et **téléchargez** ce sketch sur la carte Arduino. **Vérifiez** que le moteur à courant continu tourne en fonction de la programmation. **Montrez** votre travail au TA.
6. **Intervertissez** les connexions dans le port M1.

Q3. Que se passe-t-il? Pourquoi?

---

7. Maintenant, **ajoutez** des commandes pour contrôler un second moteur DC en conjonction avec le premier moteur. **Connectez** un moteur DC au port M2 de la protection moteur (voir ci-dessous).
8. **Ajoutez** des lignes de commandes au programme pour contrôler le deuxième moteur. Commencez par **initialiser** le nouveau moteur et le port sur le blindage qui est utilisé. **Écrivez** une déclaration additionnelle de AF_DCMotor dessous celle qui a été modifiée

dans l'étape 3. Les deux lignes AF_DCMotor devraient nommer leur moteur correspondant à un nom unique (voir ci-dessous).

```
// Bibliothèque Adafruit Motor shield
// copyright Adafruit Industries LLC, 2009
// ce code est du domaine public, profitez-en !

#include <AFMotor.h>

AF_DCMotor motor(1);
AF_DCMotor motor2(2);
```

Figure 13 : Initialisation d'un deuxième moteur

9. Chaque instruction de moteur dans le sketch doit maintenant être modifiée pour refléter le nouveau nom du moteur. **Ajoutez** une deuxième instruction de moteur sous chaque instruction de moteur existante (pas seulement celles qui sont illustrées) afin de contrôler le deuxième moteur (voir ci-dessous).

```
void setup() {
  Serial.begin(9600);           // configurer la bibliothèque série à 9600 bps
  Serial.println("Motor test!");

  // turn on motor
  motor.setSpeed(200);
  motor2.setSpeed(200);

  motor.run(RELEASE);
  motor2.run(RELEASE);
}

void loop() {
  uint8_t i;

  Serial.print("tick");

  motor.run(FORWARD);
  motor2.run(FORWARD);

  for (i=0; i<255; i++) {
    motor.setSpeed(i);
    motor2.setSpeed(i);
    delay(10);
 }


  for (i=255; i!=0; i--) {
    motor.setSpeed(i);
    motor2.setSpeed(i);
    delay(10);
 }
```

Figure 14 : Configuration du deuxième moteur

10. **Compilez** et **téléchargez** le programme. **Vérifiez** que les deux moteurs maintenant tournent ensemble. **Montrez** ton travail au TA.
    a. Parfois l'ordinateur n'est pas capable de donner assez de puissance pour les deux moteurs, si c'est le cas demande au TA pour des piles AA.

Q4. Qu'est-ce qui se passe quand la fonction 'run' est donnée le paramètre RELEASE?

Q5. Comment est-ce que la boucle 'for' fonctionne?

_____

_____

Partie D – Contrôle de Moteur via Bluetooth

Ce programme utilisera les données entrantes d'un appareil compatible BluetoothLE pour contrôler les moteurs. Le capteur agira comme un contrôle qui limitera le moment où les moteurs peuvent être utilisés.

1. Sur votre smartphone, **téléchargez** le l'application Simple Ble Joystick sur l'AppStore.
    a. Vous pouvez trouver la version Android de l'application ici : https://apkcombo.com/simple-ble-joystick/ru.experementy.simpleblejoystick/
2. Maintenant vous allez écrire le code qui va vous permettre de contrôler le chariot avec l'application que vous venez de télécharger. Premièrement, inclure toutes les bibliothèques nécessaires montées dans la figure ci-dessous.

```
#include <AFMotor.h>
#include <NewPing.h>
#include <SoftwareSerial.h>

#define TRIGGER_PIN  A0  // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     A1  // Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters).
                         //Maximum sensor distance is rated at 400-500cm.
```

3. Initialiser le capteur ultrason et les moteurs.

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.
AF_DCMotor leftmotor(2);
AF_DCMotor rightmotor(1);
```

4. Initialiser les broches du capteur Bluetooth.

```
SoftwareSerial mySerial(10, 9); // RX, TX
```

5. Définissez deux variables qui vont être utilisés dans le programme.

```
boolean prox;
int distance;
```

6. Dans la fonction de configuration, réglez la vitesse du moteur, démarrez le moniteur série et initialisez les commandes du module Bluetooth.

```
void setup() {
  // put your setup code here, to run once:
  leftmotor.setSpeed(200);
  rightmotor.setSpeed(200);

  mySerial.begin(9600);
  Serial.begin(9600);
}
```

7. Dans la fonction loop, écrivez le code qui va permettre le téléphone intelligent de contrôler le chariot via l'Arduino et le module Bluetooth.

```
void loop() {
  char reply[100];
  int i = 0;
  while (mySerial.available()) {
    reply[i] = mySerial.read();
    i += 1;
  }
  sensor_read();
  //end the string
  reply[i] = '\0';
  if(strlen(reply) > 0){
      Serial.println(" pressed");
      Serial.println(reply);
      if (reply[0] == 'C') { // Go backward! Yes, I know that it says run forward
        if (prox == true) {
        rightmotor.run(FORWARD);
        leftmotor.run(FORWARD);
        Serial.println(reply);
          }
        else {
         rightmotor.run(RELEASE);
         leftmotor.run(RELEASE);
          }
        }
```

```
    if (reply[0] == 'A') { // Go forward! Yes, I know that it says run backward
      rightmotor.run(BACKWARD);
      leftmotor.run(BACKWARD);
      Serial.println(reply);
      }
    if (reply[0] == 'D') { //turn left
      rightmotor.run(BACKWARD);
      leftmotor.run(FORWARD);
      delay(250);
      rightmotor.run(RELEASE);
      leftmotor.run(RELEASE);
      Serial.println(reply);
      }
    if (reply[0] == 'B') { //turn right
      rightmotor.run(FORWARD);
      leftmotor.run(BACKWARD);
      delay(250);
      rightmotor.run(RELEASE);
      leftmotor.run(RELEASE);
      Serial.println(reply);
      }
  } else {
    Serial.println(" released");
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  }
}
```

8. Ajoutez maintenant des instructions pour contrôler le capteur. **Créez** une nouvelle fonction et appelez-la "sensor_read()". Demandez à la fonction d'envoyer un ping au capteur, d'imprimer la distance au port série et de définir la variable booléenne initialisée plus tôt sur "true" si la distance de lecture est supérieure à 10. N'oubliez pas qu'une valeur de lecture de "0" signifie que la distance est hors d'intervalle. S'il y a quelque chose à moins de 10 cm du capteur, réglez la variable booléenne sur "faux". Voir l'exemple ci-dessous pour référence.

```
void sensor_read() {
  delay(250);
  distance=sonar.ping_cm();
  Serial.println(distance);
  if (distance > 10 or distance == 0) {
    Serial.println ("No obstruction");
    prox=true;
  } else {
    Serial.println("Obstruction");
    prox=false;
  }
}
```

Q6. Basé sur le code, expliquez le comportement attendu du chariot.

_____

_____

_____

9. Finalement **compilez** le croquis et téléversez-le sur une des voitures de test disponible dans le lab. Ouvrir le moniteur de série et assurez-vous que tout fonctionne correctement. Le moniteur de série devrait ressembler à la photo suivante.
10. Avec votre téléphone, contrôlez la voiture en utilisant l'application Simple Joystick. **Sélectionnez** votre module Bluetooth, choisissez la manette sur la droite et **commencez** à appuyer sur les boutons. **Assurez-vous** que les moteurs tournent.

Assurez-vous de sauvegarder ce code quelque part que vous pourrez le retrouver plus tard, pour l'utiliser dans un laboratoire plus tard avec votre chariot.

Voici le code complet comme référence :

```
/* Chariot */

#include <AFMotor.h>
#include <NewPing.h>
#include <SoftwareSerial.h>

#define TRIGGER_PIN  A0  // Broche Arduino attaché à la broche trigger sur le capteur ultrason.
#define ECHO_PIN     A1  // Broche Arduino attaché à la broche echo sur le capteur ultrason.
#define MAX_DISTANCE 200 // Distance maximale pour le ping (en centimetres).
                         //Distance maximale du capteur est 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // Initialisation de broches NewPing et distance maximale.
AF_DCMotor leftmotor(2);
AF_DCMotor rightmotor(1);

SoftwareSerial mySerial(10, 9); // RX, TX

boolean prox;
int distance;
```

```
void setup() {
  // initialiser le code ici, qui execute une fois:
  leftmotor.setSpeed(200);
  rightmotor.setSpeed(200);

  mySerial.begin(9600);
  Serial.begin(9600);
}

void sensor_read() {
  delay(250);
  distance=sonar.ping_cm();
  Serial.println(distance);
  if (distance > 10 or distance == 0) {
    Serial.println ("No obstruction");
    prox=true;
  } else {
    Serial.println("Obstruction");
    prox=false;
  }
}

void loop() {
  char reply[100];
  int i = 0;
  while (mySerial.available()) {
    reply[i] = mySerial.read();
    i += 1;
  }
  sensor_read();
  //fin de chaine
  reply[i] = '\0';
  if(strlen(reply) > 0){
      Serial.println(" pressed");
      Serial.println(reply);
      if (reply[0] == 'C') { // Allez en arrière! Oui je sais que ça dit d'aller en avant
        if (prox == true) {
        rightmotor.run(FORWARD);
        leftmotor.run(FORWARD);
        Serial.println(reply);
```

```
      }
     else {
      rightmotor.run(RELEASE);
      leftmotor.run(RELEASE);
       }
     }
   if (reply[0] == 'A') { // Allez en avant! Oui je sais que ça dit d'aller en arrière
     rightmotor.run(BACKWARD);
     leftmotor.run(BACKWARD);
     Serial.println(reply);
     }
   if (reply[0] == 'D') { //tourner à gauche
     rightmotor.run(BACKWARD);
     leftmotor.run(FORWARD);
     delay(250);
     rightmotor.run(RELEASE);
     leftmotor.run(RELEASE);
     Serial.println(reply);
     }
   if (reply[0] == 'B') { //tourner à droite
     rightmotor.run(FORWARD);
     leftmotor.run(BACKWARD);
     delay(250);
     rightmotor.run(RELEASE);
     leftmotor.run(RELEASE);
     Serial.println(reply);
     }
  } else {
    Serial.println("released");
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  }
}
```

Ressources Additionnelles

- Arduino est open-source qui veut dire qu'il y a beaucoup d'information qui existe. Pour commencer vous pouvez regarder (en anglais) https://www.arduino.cc/en/Tutorial/HomePage pour trouver des tutoriels et de l'information d'extra.

- Adafruit a aussi beaucoup de tutoriels sur plusieurs types d'Arduinos et capteurs différents (en anglais), https://learn.adafruit.com/.