# Manuel de Laboratoire de Programmation Arduino

Introduction à Développement de Produits et Gestion pour Ingénieurs et Informaticiens
GNG 2501
Faculté de Génie
Université d'Ottawa
Hiver 2019

Vérifications requises par le TA:

| Membre du groupe | Partie A | Partie B | Partie C | Partie D |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Dr. Patrick Dumond

## Objectif

Utiliser la plate-forme 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 »e 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 DC dans un corps de Micro Servo
- Logiciel Arduino 1.8.2 IDE
- Bibliothèques correspondantes Arduino IDE (expliqué plus bas)
- 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 lab sont :

- AFMotor.h (téléchargé de https://learn.adafruit.com/adafruit-motor-shield/library-install)
- NewPing.h (téléchargé de http://playground.arduino.cc/Code/NewPing)
- Adafruit Bluefruit LE Master library (téléchargé de https://learn.adafruit.com/introducingthe-adafruit-bluefruit-le-uart-friend/software)

Pour installer les bibliothèques Arduino, navigue vers Sketch>Add .ZIP Library… et trouve le dossier qui contient la bibliothèque ou bibliothèques qui ont besoin d'être ajoutées. Pour une option plus intégrée (et permanente), bouge le dossier qui contient la bibliothèque au dossier bibliothèque (librairies) dans les fichiers système d'Arduino. Pour la plupart des utilisateurs windows le chemin défaut est va être Program Files (x86)>Arduino>libraries, mais ceci peut changer avec des systèmes d'opération différent, différentes versions de windows ou si le Arduino IDE a été installé dans une place différente su l'ordinateur. Pour les deux méthodes le Arduino IDE devrait être recommencé pour que les bibliothèques deviennent utilisables. Adafruit.com a un tutoriel utile pour installer des bibliothèques qui peut être trouvé ici : h$ps://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use

**Questions Pré-lab**

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 plaques 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. Connecte la plaque Arduino à un port USB ouvert dans l'ordinateur avec le câble USB. La plaque Arduino devrait s'allumer car il tire de la puissance à travers la connexion USB.
2. Ouvre le logiciel Aduino IDE et clique Tools>Board>Arduino/Genuino Uno dans le menu déroulant. De retour dans l'onglet 'Tools', sélectionne 'Port' et choisis le port de série à laquelle le Arduino est connecté. Si le bon port n'est pas apparent, débranche l'Arduino et observe quel port disparait. Rebranche-le et sélectionne ce port.
3. Ouvre File>Examples>01.Basics>Blink. Ceci va ouvrir un programme qui ressemble à celle-ci-dessous.

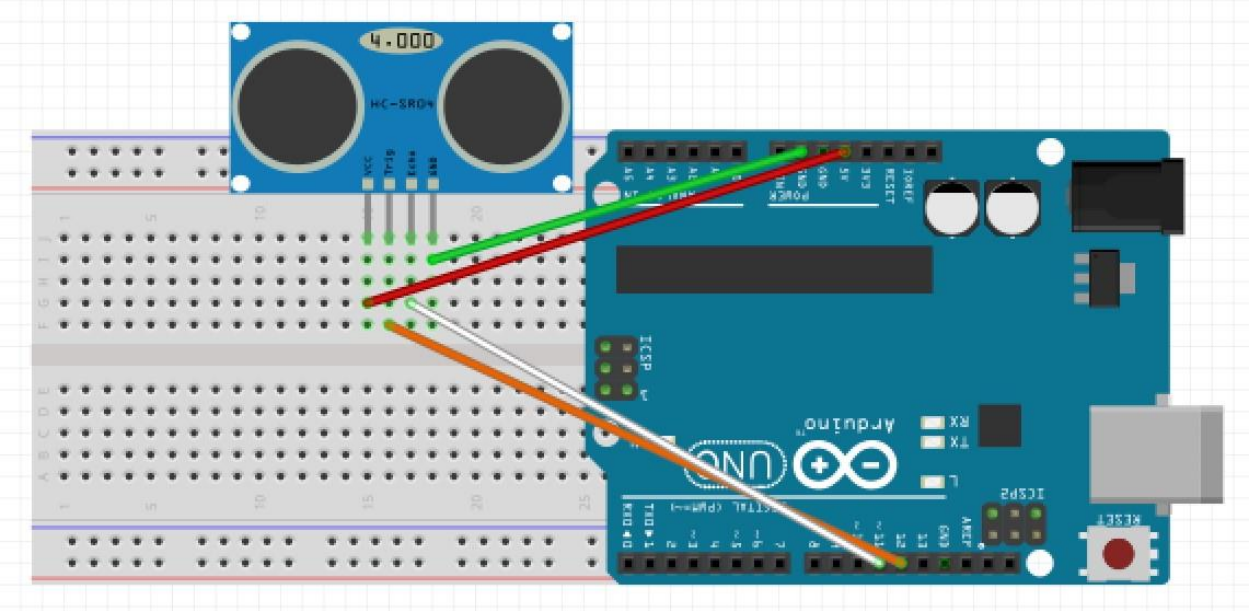
4. Maintenant vérifie (verify) et télécharge (upload) le sketch à la plaque Arduino. Pour vérifier, clique le bouton dans le coin en haut à gauche. Le Arduino IDE va essayer de compiler le programme (sans télécharger à la plaque) et t'avertir de n'importe quelles erreurs de syntaxe dans la programmation. Quand ceci est terminé, clique le bouton télécharger. Le programme va recompiler et télécharger à la plaque.
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. Montre ton travail au TA.

```
  modified 2 Sep 2016
  by Arturo Guadalupi

  modified 8 Sep 2016
  by Colby Newman
*/


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

**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. Connecte un capteur ultrason à l'Arduino en utilisant le diagramme de câblage ci-dessous. La plaque Arduino devrait encore être connectée à l'ordinateur via le câble USB.




2. Ouvre 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. Aller à File>Exmples>NewPing>NewPingExample.
3. Compile et télécharge cet exemple sur la plaque Arduino. Soit certain de vérifier que la bonne plaque est choisie (Arduino/Genuino Uno) et que le bon port de série (comm port) est utilisé.
4. Une fois que l’exemple est téléchargé, ouvre le moniteur de série pour voir les données du capteur. Clique l’icône dans le coin en haut à droite du IDE ou dans Tools>Serial Monitor.
5. Fixe le baud rate du moniteur de série pour correspondre avec le baud rate spécifié dans l’exemple. Sélectionne 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 apparaitre dans la fenêtre du moniteur. Montre ton travail au TA.
6. Après avoir vérifié que le capteur fonctionne correctement, change les pins qui sont utilisés sur l’Arduino pour envoyer et recevoir des données du capteur. Après les deux lignes au début du programme qui définissent les pins écho (echo) et déclenche (trigger) (les pins de défaut sont 11 et 12 respectivement) pour envoyer une déclenche à travers pin 13 et recevoir un écho à travers pin 2 (voir ci-dessous).

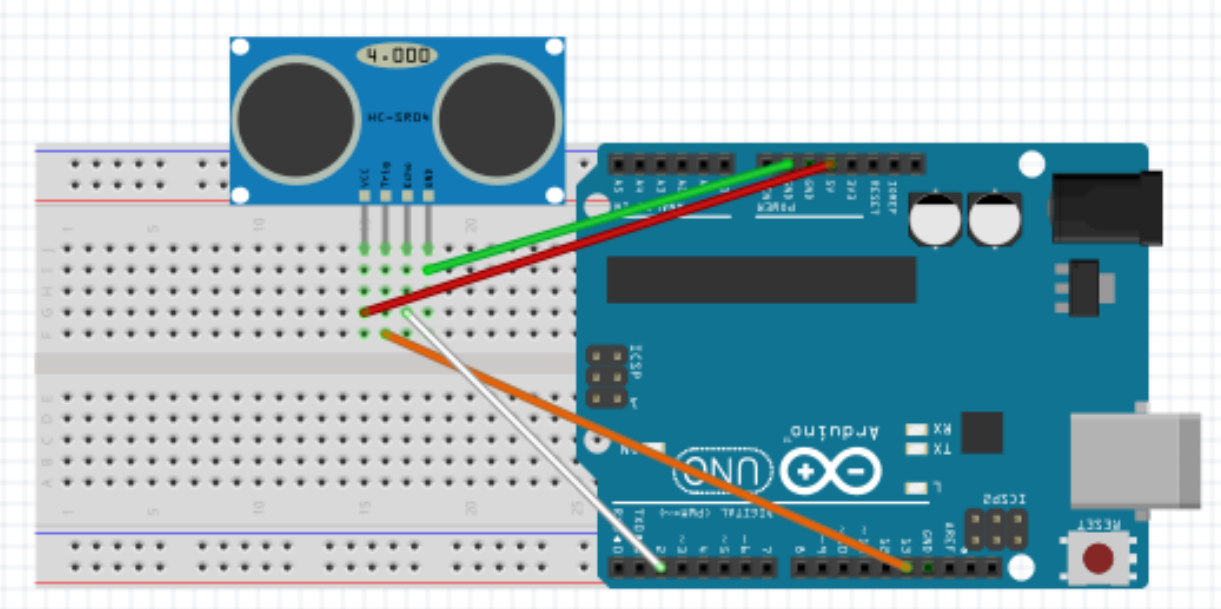
NewPingExample

```
// ---------------------------------------------------------------------------
// Example NewPing library sketch that does a ping about 20 times per second.
// ---------------------------------------------------------------------------

#include <NewPing.h>

#define TRIGGER_PIN  12  // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     11  // 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.
```

NewPingExample §

```
// ---------------------------------------------------------------------------
// Example NewPing library sketch that does a ping about 20 times per second.
// ---------------------------------------------------------------------------

#include <NewPing.h>

#define TRIGGER_PIN  13  // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     2  // 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.
```

7. Rebranche le capteur à l'Arduino pour refléter le changement de pins qu'on vient de faire.




8. Compile et télécharge de nouveau le programme, Ouvre le moniteur de série de la même façon et vérifie que le capteur utilise les nouveaux pins correctement. Montre ton travail au TA.

## Partie C – Contrôle de Moteurs DC

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

1. Déconnecte le détecteur de l'Arduino. Met le blindage contrôleur de moteur (motor shield) sur l'Arduino et connecte un moteur DC au port M1 sur le blindage.
2. Ouvre l'exemple trouvé dans Files>Examples>DC Motor Shield Library>Motor Test. Cet exemple allume un moteur DC connecté au port M1 sur le blindage.
3. L'exemple motor test contrôle un moteur connecté au port M4 par défaut. Change ce port au port M1 en changeant la ligne montré ci-dessous. La nouvelle ligne devrait 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(4);

void setup() {
```

4. Compile télécharge le programme à la plaque Arduino. Vérifie que le moteur DC tourne en conjonction avec la programmation, Montre ton travail au TA.
5. Maintenant ajoute des commandes pour contrôler un deuxième moteur DC en conjonction avec le premier moteur. Connecte un moteur DC au port M2 sur le blindage (voir ci-dessous).
6. Ajoute des lignes de commandes au programme pour contrôler le deuxième moteur. Commence par initialiser le nouveau moteur et le port sur le blindage qui est utilisé. Écrit 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 un nom unique (voir ci-dessous).

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

#include <AFMotor.h>

AF_DCMotor motorA(1);
AF_DCMotor motorB(2);

void setup() {
  Serial.begin(9600);           // set up Seria
```

7. Chaque déclaration de moteur dans le programme va maintenant avoir besoin d'être modifié pour refléter le nom du nouveau moteur. Ajoute une deuxième déclaration en dessous de chaque déclaration existante de moteur (pas seulement celles montrées) pour contrôler le deuxième moteur (voir ci-dessous).

MotorTest §

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

#include <AFMotor.h>

AF_DCMotor motorA(1);
AF_DCMotor motorB(1);

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Motor test!");

  // turn on motor
  motorA.setSpeed(200);
  motorB.setSpeed(200);

  motorA.run(RELEASE);
  motorB.run(RELEASE);
}

void loop() {
  uint8_t i;

  Serial.print("tick");

  motorA.run(FORWARD);
  motorB.run(FORWARD);
  for (i=0; i<255; i++) {
    motorA.setSpeed(i);
    motorB.setSpeed(i);
    delay(10);
 }

  for (i=255; i!=0; i--) {
    motorA.setSpeed(i);
    motorB.setSpeed(i);
    delay(10);
 }

  Serial.print("tock");

  motorA.run(BACKWARD);
  motorB.run(BACKWARD);
  for (i=0; i<255; i++) {
    motorA.setSpeed(i);
```

8. Compile et télécharge le programme. Vérifie que les deux moteurs maintenant tournent ensemble. Montre 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.

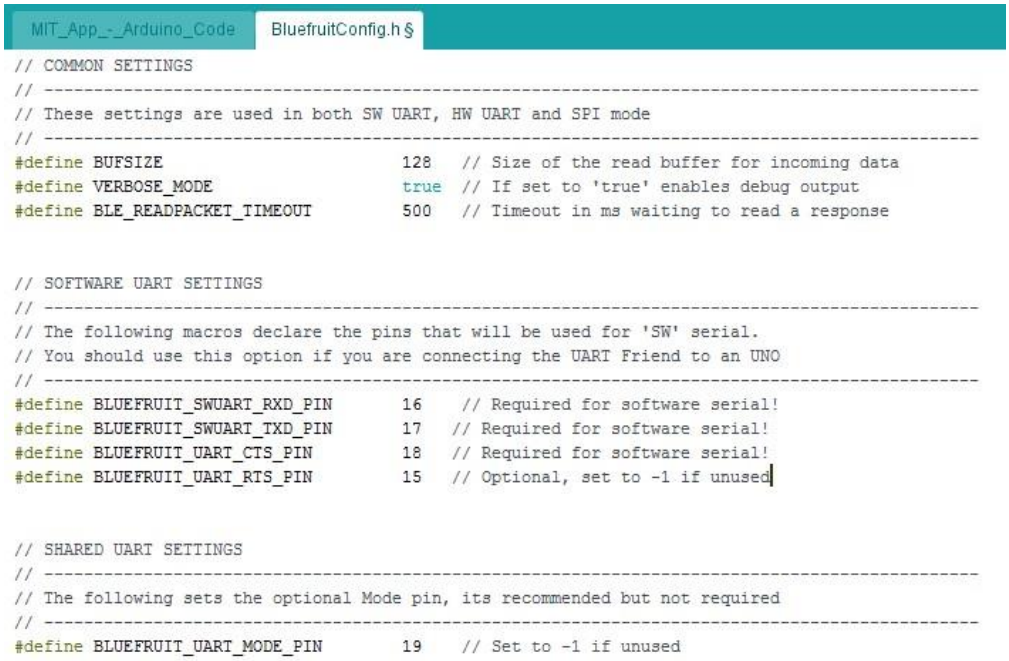
## Partie D – Contrôle de Moteur via Bluetooth

**SVP** identifie si ton dispositif utilise BLE (Bluetooth 4.0 Low Energy) ou Blutooth régulier (demande un TA si tu n'es pas certain). Tu devrais utiliser un dispositif BLE.

### Contrôle de Moteur Bluetooth pour iPHONES (4s et plus récent) ET ANDROID (LE seulement)

Ce programme va utiliser les données entrantes d'un dispositif compatible BluetoothLE pour contrôler les moteurs avec la chip Bluetooth bluefruitBLE. Le capteur va agir comme une vérification qui va limiter quand les moteurs peuvent être utilisés. Ceci va être fait en modifiant un programme open-source de Adafruit Industries.

1. En t'assurant que les bibliothèques Aduino appropriées sont installées, navigue vers File>Examples>Adafruit BluefruitLE nRF51 et ouvre l'exemple bleuart_cmdmode. Ceci va ouvrir un programme avec deux onglets disponible en haut; bleuart_cmdmode et Bluefruit_Config.h.
2. Change l'onglet Bluefruit_Config.h pour utiliser une connexion logiciel UART. C'est un type de connexion en série (similaire à la communication utilisé dans un dispositif USB) utilisé pour communiquer entre le module BluetoothLE sur l'auto et la plaque Arduino Uno. Les sections de code requises pour configurer une connexion logiciel UART sont les paramètres communs, les paramètres de logiciel UART et les paramètres communs de UART. Toutes les autres sections peuvent être supprimées. Les pins utilisés pour la connexion doivent aussi être modifiés pour refléter quels pins à lesquels le module est attaché sur la plaque Arduino. Utilise la photo ci-dessous pour fixer les pins appropriés et assurer que le fichier Bluefruit_Config.h est bien configuré.




3. Maintenant modifie l'onglet bleuart_cmdmode. Le paramètre du logiciel UART ne doit pas être un commentaire (supprime les caractères "/*" avant les lignes et "*/" après les

lignes) et les lignes du UART matériel doivent être enlevés (ou commentés en ajoutant "//" avant la ligne).

```
// Create the bluefruit object, either software serial...uncomment these lines
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN, BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);


/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
// Adafruit_BluefruitLE_UART ble(Serial1, BLUEFRUIT_UART_MODE_PIN);

/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);

/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
//Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
//                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
//                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

Avant la section de configuration (setup), inclus toutes les bibliothèques et initialise toutes les composantes et variables requises pour faire fonctionner les moteurs et capteurs. Ils vont inclure les bibliothèques AFMotor.h et NewPing.h, définir les pins du capteur (déclencheur/trigger et écho/echo) et la distance maximale de mesure (maximum measuring distance) et initialiser le capteur, les deux moteurs, une variable boolean et une variable long. Voir la section de code ci-dessous comme référence.

```
/**************************************************************************/
/*!
    @brief  Sets up the HW an the BLE module (this function is called
            automatically on startup)
*/
/**************************************************************************/

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

#define TRIGGER_PIN  13  // Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN     2 // 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.


NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.

AF_DCMotor rightmotor(3, MOTOR34_64KHZ);
AF_DCMotor leftmotor(4, MOTOR34_64KHZ);

boolean prox;
int distance;

void setup(void)
{
```

4. Modifie la section de configuration du programme (setup). Enlève la déclaration while et la déclaration delay avant la ligne Serial.begin(115200). Insert des lignes pour fixer la vitesse (speed) des moteurs immédiatement après la déclaration Serial.begin (utilise la commande "rightmotor.setSpeed(200)").

```
void setup(void)
{
  Serial.begin(115200);
  rightmotor.setSpeed(200);
  leftmotor.setSpeed(200);
  Serial.println(F("Adafruit Bluefruit Command Mode Example"));
  Serial.println(F("---------------------------------------"));

  /* Initialise the module */
  Serial.print(F("Initialising the Bluefruit LE module: "));

  if ( !ble.begin(VERBOSE_MODE) )
  {
    error(F("Couldn't find Bluefruit, make sure it's in CoMmanD mode & check wiring?"));
  }
  Serial.println( F("OK!") );
```

5. Après la déclaration “ble.readline()” appelle la fonction “sensor_read()” qui va être construite dans la prochaine étape. Ensuite supprime la prochaine condition if qui contient “strcmp”. Après la déclaration “Serial.print(F(“[Recv] ”))” ajoute la ligne “String received = String(ble.buffer);”. Maintenant tu peux ajouter des déclarations pour contrôler les moteurs et lire le capteur à l’intérieur de la section de boucle (loop). Immédiatement après la déclaration “String received = String(ble.buffer);” (proche de la fin de la section de boucle) ajoute des déclarations if et if else pour contrôler la direction dues moteurs basé sur les données reçues via Bluetooth. Voir la section de code ci-dessous comme référence.

Noubliez pas ceci!

```
  // Check for incoming characters from Bluefruit
  ble.println("AT+BLEUARTRX");
  ble.readline();

  sensor_read();

  // Some data was found, its in the buffer
  Serial.print(F("[Recv] ")); Serial.println(ble.buffer);
  String received = String(ble.buffer);
  Serial.print("prox is: ");
  Serial.println(prox);
  if (prox == false || received == "backward") {
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
    if (received == "backward") {
      rightmotor.run(BACKWARD);
      leftmotor.run(BACKWARD);
      delay(3000);
    }
  }
  if (received == "forward") {
    rightmotor.run(FORWARD);
    leftmotor.run(FORWARD);
  } else if (received == "backward") {
    rightmotor.run(BACKWARD);
    leftmotor.run(BACKWARD);
  } else if (received == "left") {
    rightmotor.run(FORWARD);
    leftmotor.run(BACKWARD);
    delay(250);
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  } else if (received == "right") {
    rightmotor.run(BACKWARD);
    leftmotor.run(FORWARD);
    delay(250);
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  } else if (received == "stop") {
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  }

  ble.waitForOK();
}
```

6. Maintenant ajoute des déclarations pour contrôler le capteur. Crée une nouvelle fonction et appelle-le “sensor_read()”. Fait en sorte que la fonction ping le capteur, imprime la distance au port de série et fixe la variable boolean qui a été initialisée plus tôt à “true” si la distance lue est plus grande que 10. Souviens-toi qu’une valeur lue de “0” veut dire que la distance est hors de portée. S’il y a quelque chose à moins de 10 cm du capteur, fixe la variable boolean à “false”. Voir l’exemple ci-dessous comme référence.

```
    delay(250);
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  } else if (received == "right") {
    rightmotor.run(BACKWARD);
    leftmotor.run(FORWARD);
    delay(250);
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  } else if (received == "stop") {
    rightmotor.run(RELEASE);
    leftmotor.run(RELEASE);
  }
  ble.waitForOK();
}

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;
  }
}
/**************************************************************************/
/*!
    @brief  Checks for user input (via the Serial Monitor)
*/
/**************************************************************************/
```

7. Finalement télécharge le programme à l’Arduino et test en utilisant l’application mobile qui va être développée dans un lab plus tard. Montre ton travail au TA.

*NOTE: Ce lab utilise des bibliothèques Arduino IDE qui ne viennent pas dans la version standard dans l’environnement de Arduino IDE. Pour que ces programmes fonctionnent bien, ils ont besoin d’être téléchargées et insérées dans le dossier de bibliothèques dans les fichiers système. Une liste de bibliothèques requises peut être trouvée dans la section de téléchargements. Elles sont installées sur les ordinateurs de lab mais doivent être installées si un étudiant utilise son ordinateur personnel.

**Ressources Additionnels**

- Arduino est open-source qui veut dire qu’il y a beaucoup d’information qui existe. Pour commencer tu peux 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/.