Arduino Programming Laboratory Manual

Objective

To utilize the Arduino IDE Software platform to write simple control logic commands. This control code will be used to make effective use of a variety of inputs and outputs connected to an Arduino Uno microcontroller. This lab will also expose students to more complicated control and communication codes, and online resources available for the Arduino software.

Background

In conjunction with different models of microcontrollers, the Arduino company provides the open-source Arduino Integrated Development Environment (IDE). This programming environment is written in Java and has been designed to introduce programming to an audience that is less familiar with coding. Programs written in this environment are called ‘sketches’, and the language used is based on the Processing language with support for C and C++ programming languages.

A basic Arduino sketch consists of at least two functions: a ‘setup’ function and a ‘loop’ function. The setup function performs any actions that are initially required to run the rest of the program, such as initializing any peripheral components and setting the communication frequency between the Arduino board and PC. The loop function acts as the programs driver; it runs in a continuous loop, and specifies the order of operations that the microcontroller will perform.

Arduino control boards as well as many Arduino-compatible peripherals contain hardware for serial communication. This form of communication transmits data one bit at a time over the serial connection. This allows the components of the system (or a PC) to communicate larger and more complex sequences of data much more easily, without physical input or output pin limitations.

Because of Arduino’s open-source nature, third party companies, as well as end users are free to take the software and tailor it to their individual needs. This means that for almost any application, sketches are available for download online, and companies making peripherals (such as a motor shield or sensors) for use with Arduino boards often have example sketches and sketch libraries available for specific use for their products. These resources can be used to create a custom sketch that can use many different peripherals at once, by combining aspects of many different example sketches.

Apparatus and Equipment Overview

The equipment that will be used in this lab includes:

- 1 x Arduino Uno R3 Microcontroller
- 1 x USB 2.0 Type A Plug to USB 2.0 Type B Plug Cable
- 1 x Lab or Personal Computer (running Windows, Macintosh, or Linux OS)
- 1 x Ultrasonic ranging module (HC - SR04)
- 1 x Adafruit Motor Shield V1 (L293D chipset)
- 2 x DC Motor
- 1 x Breadboard
- 4 x Male-Male jumper
- Arduino IDE Software
- Corresponding Arduino IDE Libraries (outlined below)
- 4 x 22 Gauge Wire

Pre-Lab Preparation

Before arriving at the lab, students should review the lab manual and familiarize themselves with the lab setup and procedures. Students may use their own computer for this lab if they wish, provided that they have downloaded and installed the Arduino IDE, as well as the required libraries outlined in the downloads section. Reviewing the Arduino IDE syntax, as well as trying to write some of the programs beforehand is also encouraged. A useful list of commands used in the Arduino IDE can be found here: https://www.arduino.cc/en/Reference/HomePage.

Downloads

The Arduino IDE can be downloaded from here: https://www.arduino.cc/en/Main/Software
The libraries used in this lab can be installed through the library manager in Arduino IDE

- Adafruit Motor Shield V1 (AFMotor.h)
- NewPing (NewPing.h)
- SoftwareSerial (SoftwareSerial.h- this one is pre-installed in Arduino)

To install the required libraries, you will need to access the library manager in the arduino IDE. Open the Arduino IDE and navigate to Tools>Manage libraries.

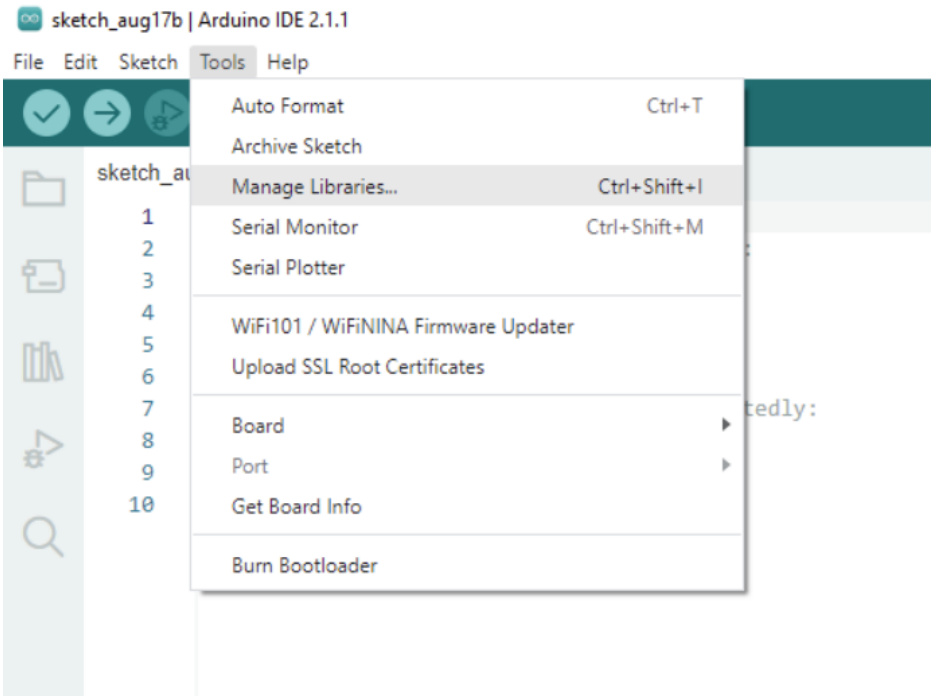


Figure 1: Arduino library manager.

The library manager tab should open up from the left side of the window. Now, search for the required libraries listed above and install. Do this for each library. You might need to reboot the IDE.

If you have libraries that need to be imported into the IDE it is possible by using the ZIP file, go to Sketch>Include library>Add .ZIP Library... and browse to and select the folder that contains the library or libraries that need to be added.

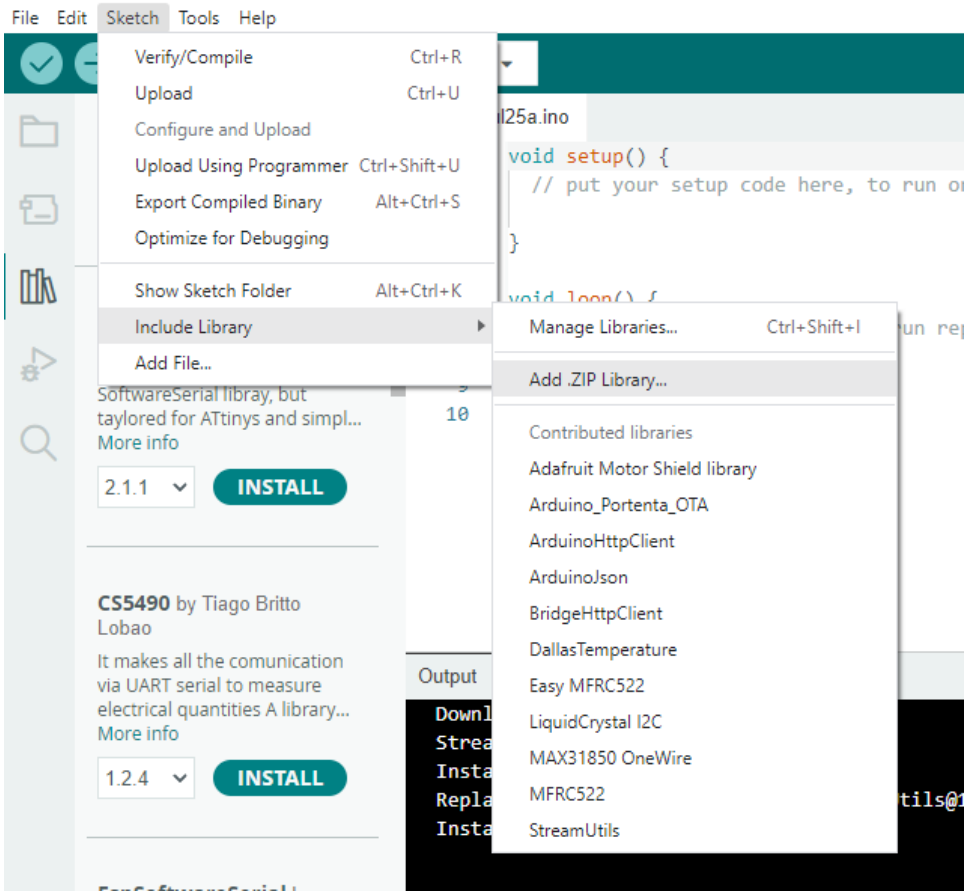


Figure 2 : Arduino included libraries

Review Questions

What is the role of a microcontroller?

Which microcontroller are we using in this lab?

How can the ultrasonic sensor measure distance?

_____

_____

How will the motors be controlled to rotate in both directions?

_____

_____

Which wireless communication method are we using?

_____

_____

Procedure

Sensitive electrical components like the control boards and sensors used in this lab can be damaged or destroyed from static discharge from mishandling during assembly and use. Before setting up this lab, students should ensure that their workstation is statically prepared (non-metal workstation, no carpet/other fibrous materials), and they have grounded themselves by touching a large metal object before handling the components. There are accessories to improve the assembly workstation for static resistance, such as an antistatic mat and static wrist/ankle straps, which should be used if available.

Part A – LED Blink Program

This program will go over the basic method to connecting and programming an Arduino microcontroller. It utilizes a sample sketch included in one of the Arduino IDE's default libraries, and is primarily used for verifying that the microcontroller and the connection are both functioning properly.

1. **Connect** the Arduino board to an open USB port on the computer with the USB cable. The Arduino board should light up, as it draws power through the USB connection.
2. **Launch** the Arduino IDE Software, and **open** Tools→Board→Arduino Uno from the drop-down menu. Back in the 'Tools' tab, **select** 'Port' and **choose** the serial port that the Arduino is connected to. If the correct serial port is not apparent, unplug the Arduino, and see which port disappears. **Re-connect** the board and **select** that port. If you don't select the correct serial port, then you won't be able to see any printed output.

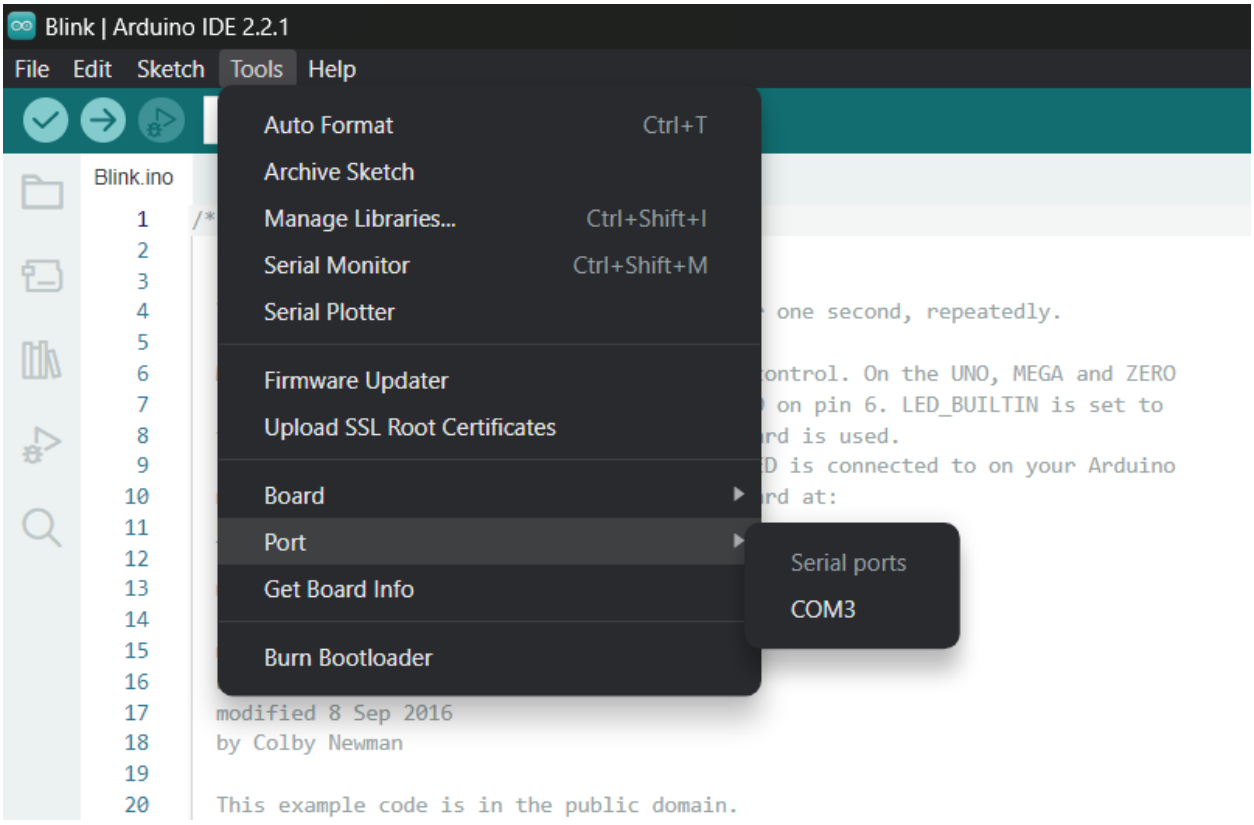


Figure 3 : Arduino port selection

If the port is not detected by the IDE it may be a driver error. Try to download and install the following CH340 driver on your computer: http://www.wch-ic.com/downloads/CH341SER_ZIP.html

3. **Open** File→Examples→01.Basics→Blink. This will open a sketch that looks like the one below.

```
// 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
}
```

Figure 4 : Blink example

4. Now verify and upload the sketch to the Arduino board. To verify, **click** the check mark in the upper left hand corner. The Arduino IDE will try to compile the sketch (without uploading to the board), and warn of any syntax errors in the programming. Once this is complete, **press** the upload button (arrow beside the check mark). The sketch will recompile, and upload to the board.

Figure 5 : Verify and upload buttons

5. Once the upload is complete, the program will automatically run on the microcontroller. In this case, the LED labeled 'L' on the Arduino should flash slowly. **Show** the TA your work.
6. **Navigate** to sketch → Include a library → Manage libraries, a list of libraries will appear.
7. **Search** and **install** (by hovering to the item and clicking on install) the following libraries:
    a. NewPing
    b. Adafruit Motor Shield Library V1

Part B – Sensor Reading

This program will automatically poll an ultrasonic sensor, causing the sensor to emit an ultrasonic sound pulse. The program will read the elapsed time between sending the pulse and receiving an echo, convert the time to distance using the speed of sound, and print the distance to the computer screen via the serial port connection.

1. **Connect** an ultrasonic sensor to the Arduino board by using the wiring diagram below. The Arduino board should still be connected to the computer via USB cable.
    a. Don't just blindly follow the diagram, make sure you are connecting GND to GND and Vcc to 5V. Make sure the echo and trig pins are connected properly as well.
    b. If the lights on the Arduino turn off it means you have a short circuit! Unplug the wires quickly and check your connections.

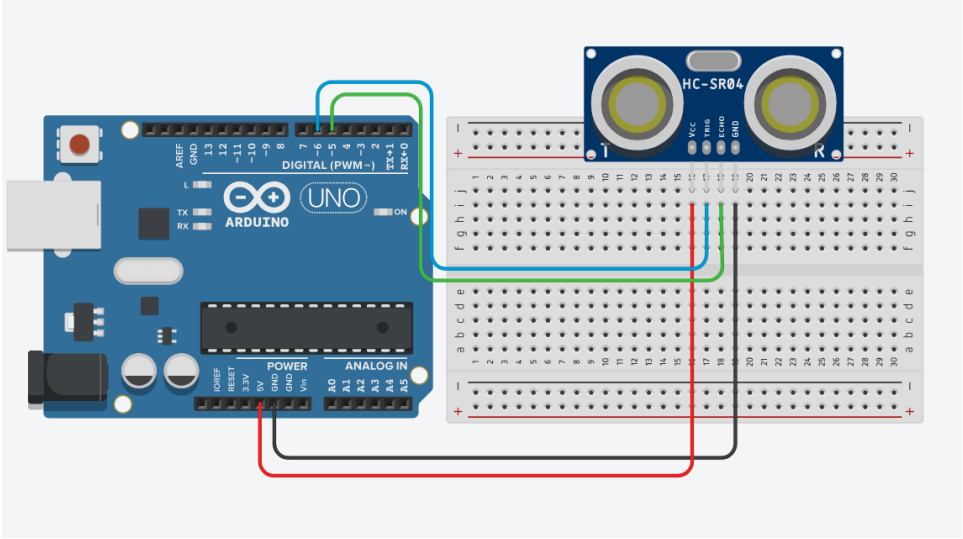


Figure 6 : Wiring of the ultrasonic sensor

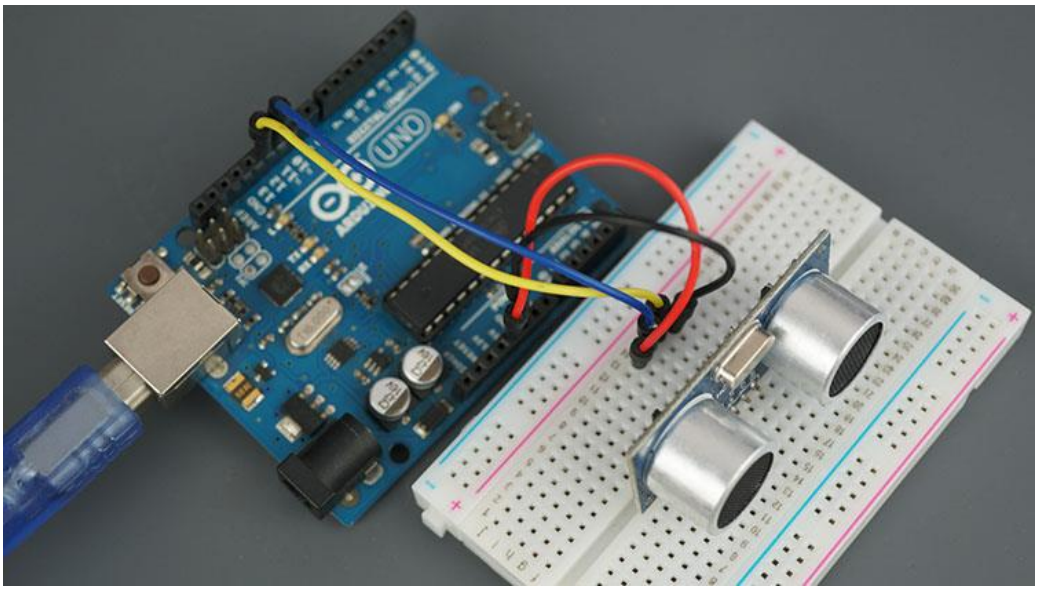


Figure 7 : Placement of the ultrasonic sensor in a breadboard

2. **Open** a new example sketch in the Arduino IDE. The specific example will be included in the "NewPing" library that should have been added to the Arduino IDE. **Go** to File→Examples→NewPing→NewPingExample.
3. Change the pins on the Arduino that are used to send and receive data to the sensor. **Alter** the two lines in the beginning of the sketch that define the echo and trigger pins (the default pins are 11 and 12 respectively) to send a trigger through pin 13 and receive an echo through pin 2 (see below).

```
#include <NewPing.h>

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

Original

```
#include <NewPing.h>

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

New

Figure 8 : Modification of trigger and echo pin

4. **Compile** and **upload** this example to the Arduino board. Be sure to verify that the correct board is selected (Arduino Uno), and the proper com port is being utilized.
5. Once the example has been uploaded, open the serial monitor to see the sensor data. **Click** the icon in the top right corner of the IDE, or **go** to Tools→Serial Monitor.
6. Set the baud rate of the serial monitor to correspond to the baud rate specified in the example sketch. Select the correct baud rate by **utilizing** the drop down menu in the lower left hand corner of the serial monitor window. In this case the baud rate should be 115200. After a few moments distance measurements should start to appear in the monitor window.
    a. **If you see 0cm, it is possible that the trig and echo connections have been reversed.**

Q1. What is the difference between the ECHO and the TRIG pins of the sensor?

_____

_____

Q2. What is the function sonar.ping_cm() calculating?

_____

_____

Part C – DC Motor Control

This program will test and control two DC motors using a pre-set sequence of commands.

1. **Disconnect** the sensor from the Arduino. **Insert** the motor shield board into the Arduino headers.
2. **Connect** a DC motor to port M1 on the motor shield by inserting the motor wires in the blue terminal block and using a screwdriver to clamp them in place.

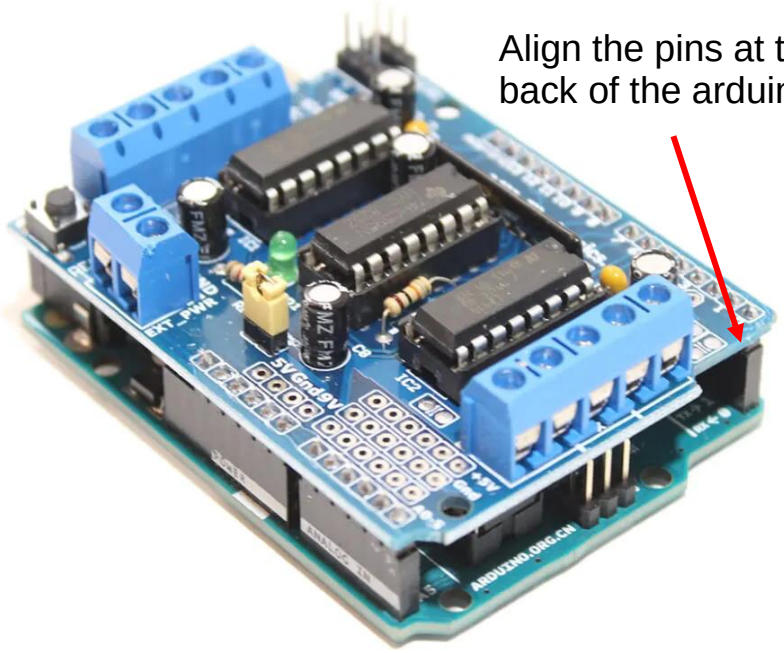


Figure 9 : Motor shield connected to arduino

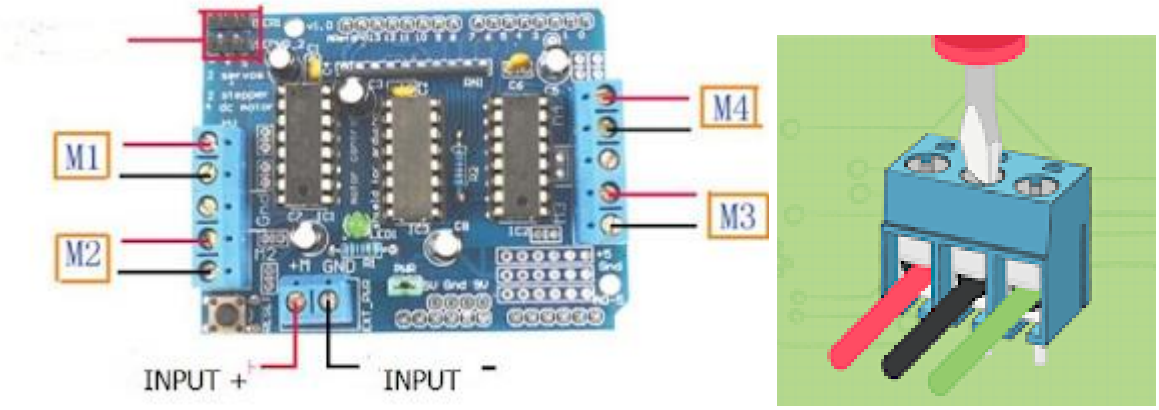
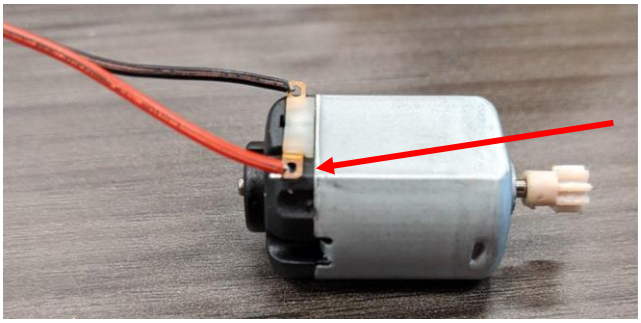


Figure 10 : Motor shield pins overview



You can use tape to hold the wires in place
*Be careful of the motor terminals, they are fragile

Figure 11 : Motor pin

3. **Open** the example found in File→Examples→Adafruit Motor Shield Library→MotorTest. This example sketch runs a DC motor connected to port M1 on the motor shield.
4. The motor test example controls a motor connected to port M4 by default. Change this port to port M1 by **changing** the line shown below. The new line should read "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 : Change of motor pin from M4 to M1

5. **Compile** and **upload** this sketch to the Arduino board. **Verify** that the DC motor spins in conjunction with the programming. **Show** your work to the TA.
6. **Swap** the connections in the M1 port.

Q3. What happens? Why?

_____

_____

7. Now **add** commands to control a second DC motor in conjunction with the first motor. **Connect** a DC motor to port M2 on the motor shield (see below).
8. **Add** command lines to the sketch to control the second motor. Start by **initializing** the new motor and the port on the motor shield being used. **Write** an additional AF_DCMotor line below the one that was altered in Step 4. Both AF_DCMotor statements should name their corresponding motor a unique name (see below).

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

#include <AFMotor.h>

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

Figure 13 : Initializing a second motor

9. Each motor statement within the sketch will now have to be modified to reflect the new motor name. **Add** a second motor statement below every existing motor statement (not just the ones shown) in order to control the second motor (see below).

MotorTest§

```
void setup() {
  Serial.begin(9600);           // set up Serial library at 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 : Configuring the second motor

10. **Compile** and **upload** the sketch. **Verify** that both motors now spin in unison. **Show** the TA your work.
    a. Sometimes the computer isn't capable of giving enough power for both motors, if this is the case you might need AA batteries as external power for the shield.

Q4. What happens when the function 'run' is given the parameter RELEASE?

_____

_____

Q5. How does the 'for' loop work?

_____

_____

Part D – Bluetooth Motor Control

This program will use incoming data from a BluetoothLE compatible device to control the motors. The sensor will act as a check that will limit when the motors can be used.

1. On your smartphone, **download** the application called Simple Ble Joystick on the AppStore.
    a. You can find the Android version of the app here: https://apkcombo.com/simple-ble-joystick/ru.experementy.simpleblejoystick/
2. Now you will write the code that will allow you to control the chariot with the app you just downloaded. First, include all the necessary libraries as shown in the picture below.

```
#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. Initialize the ultrasonic sensor and the motors.

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

4. Initialize the Bluetooth sensor pins.

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

5. Define two variables that will be used in the program.

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

6. In the setup function, setup the motor speed, start the serial monitor and initialize the Bluetooth module commands.

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

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

7. In the loop function, write the code that will allow the smartphone to control the chariot via the Arduino and the Bluetooth module.

```
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. Now add statements to control the sensor. **Create** a new function and call it "sensor_read()". Have the function ping the sensor, print the distance to the serial port, and set the Boolean variable initialized earlier to "true" if the read distance is greater than 10. Remember that a read value of "0" means the distance is out of range. If there is something less than 10 cm away from the sensor, set the Boolean variable to "false". See the example below for reference.

```
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. Based on the code, explain the expected behavior of the chariot.

_____

_____

_____

9. Finally **compile** the sketch and upload it to one of the test cars available in the lab. Open the serial monitor and make sure that everything is working perfectly.
10. With your phone, control the car by using the Simple Joystick App. **Select** your Bluetooth module, choose the controller pad on the right and **start** pressing the buttons. **Make sure** the motors are moving.

Make sure to save this code somewhere you will be able to find later, to use in a later lab with your chariot.

Here is the complete code for reference:

```
/* Chariot */

#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.

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

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

boolean prox;
int distance;

void setup() {
```

```
  // put your setup code here, to run once:
  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();
  //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);
   }
}
```

Additional resources

- Arduino is open source which means there is lots of information out there. To start you can browse https://www.arduino.cc/en/Tutorial/HomePage to find tutorials and extra information.
- Adafruit also has lots of tutorials on many different types of Arduinos and sensors, https://learn.adafruit.com/.