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 opensource 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.

Design for Manufacturing Considerations

Ask yourself why you need an Arduino? While there are many different uses for an Arduino, the main one is to collect data using a sensor or control a motor. There are many different types of microcontrollers which are best suited for different purposes. Do you need an addon board? There are many “shields” that are made for the Arduino, these shields add some sort of functionality to the Arduino while passing through more I/O pins and mean you don’t have to install a separate circuit.

Make sure that there is a library for the purpose that you want. While it is possible to write almost any code, using a library is a lot faster than coding it yourself.

Plan your Arduino code without running the code on the Arduino itself. Using TinkerCAD circuits, you can set up and test a circuit before buying the materials.

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 4*AA battery holder
- 4 x AA battery
- 1 x Breadboard
- 4 x Male-Male jumper
- 8 x Male-Female jumper
- Arduino IDE Software
- Corresponding Arduino IDE Libraries (outlined below)
- 1 x small flathead screwdriver

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)

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.

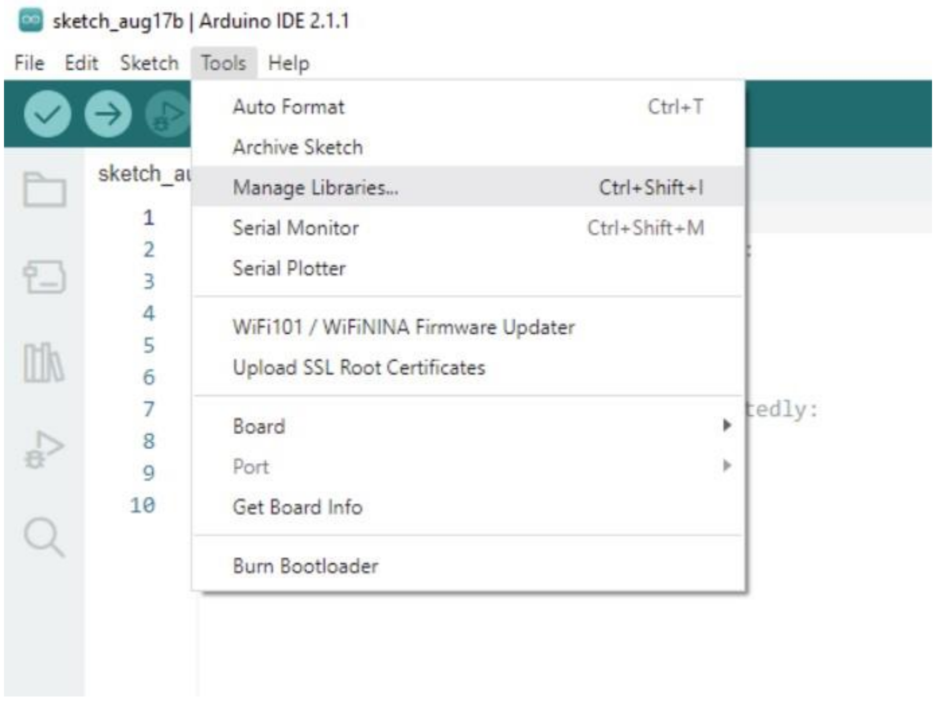
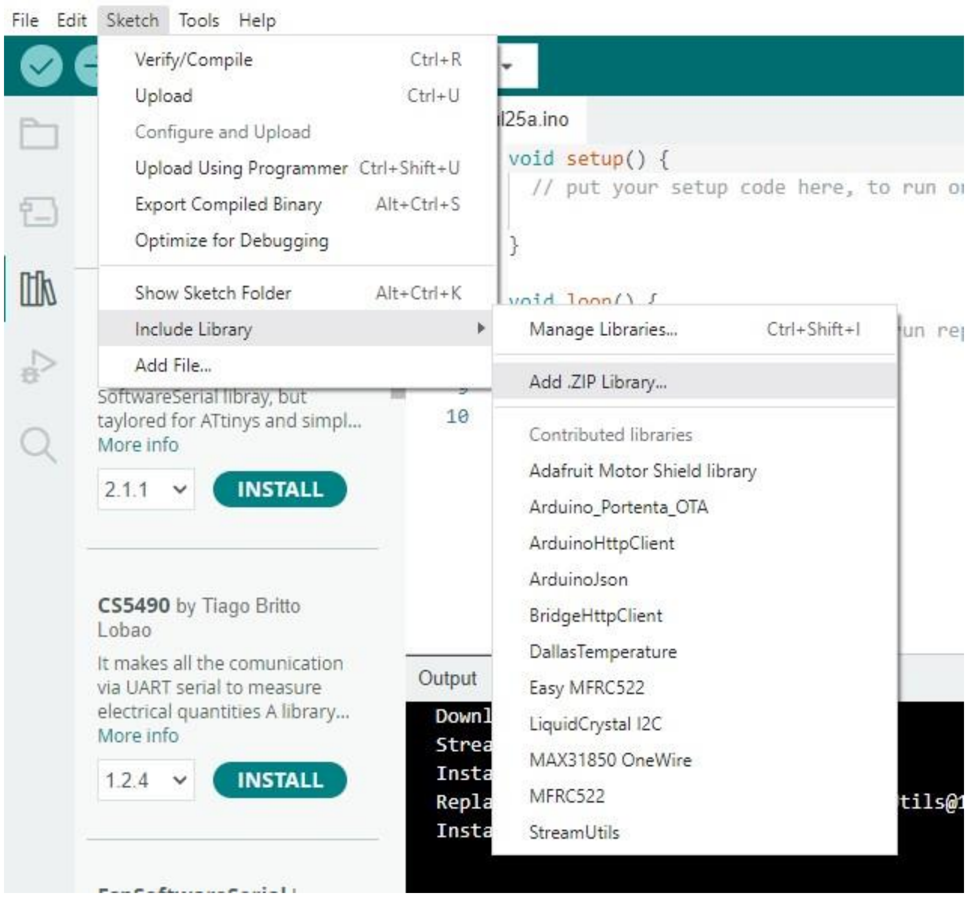

sketch_aug17b | Arduino IDE 2.1.1
File
Edit
Sketch
Tools
Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board
Port
Get Board Info
Burn Bootloader

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.



```
void setup() {
  // put your setup code here, to run o
}
```

Figure 2 : Arduino included libraries

Review Questions

What is the role of a microcontroller?

A microcontroller is a compact integrated circuit that controls specific tasks in embedded systems.

Which microcontroller are we using in this lab?

Arduino Uno

How can the ultrasonic sensor measure distance?

The sensor will use one speaker to emit an ultrasonic sound then the other sensor to receive the reflected sound from the object it bounces off.

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

The rotation direction of a DC motor can be changed by reversing the polarity of the voltage applied to its terminals.

What are the main sections of an Arduino program?

Void setup() and Void loop()

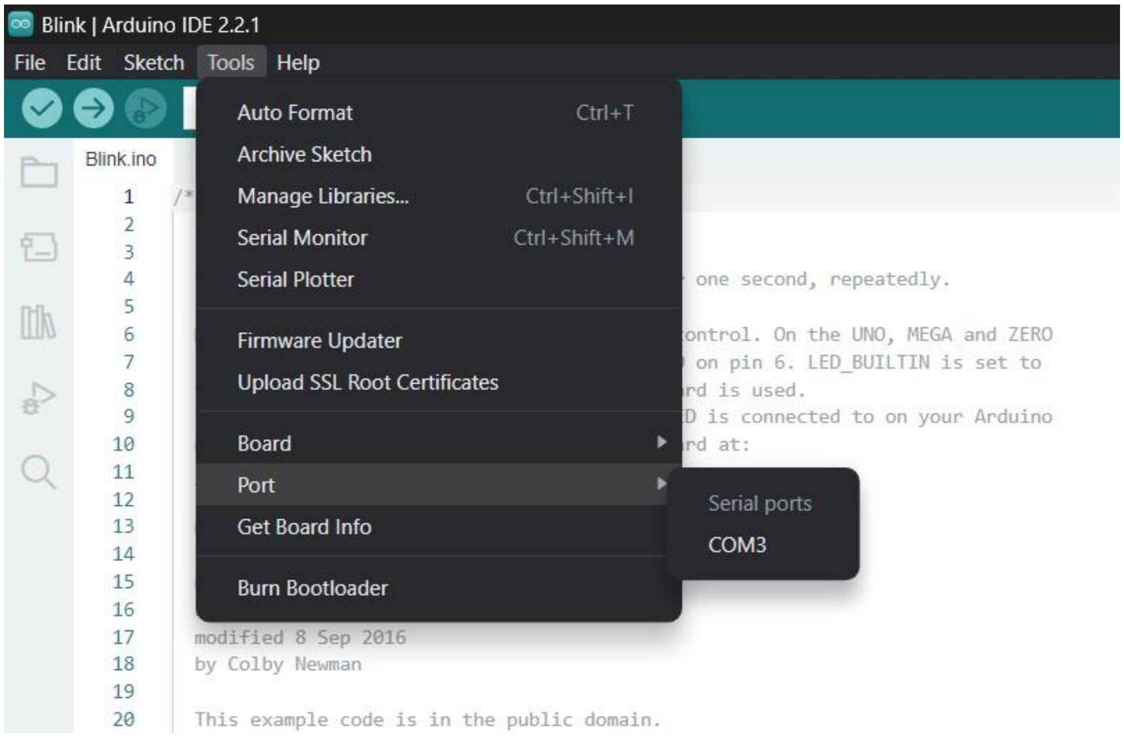
Procedure

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.

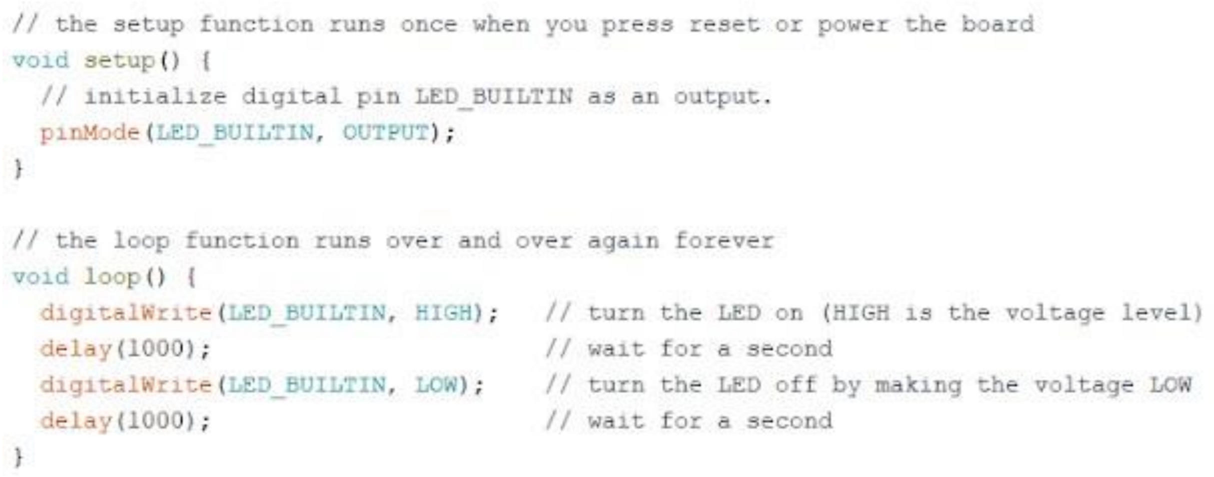

Blink | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Blink.ino
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board
Port
Get Board Info
Burn Bootloader
Serial ports
COM3

```
/*
                                   one second, repeatedly.

                                  ontrol. On the UNO, MEGA and ZERO
                                   on pin 6. LED_BUILTIN is set to
                                  rd is used.
                                  D is connected to on your Arduino
                                  rd at:

  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.
```

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.wchic.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.

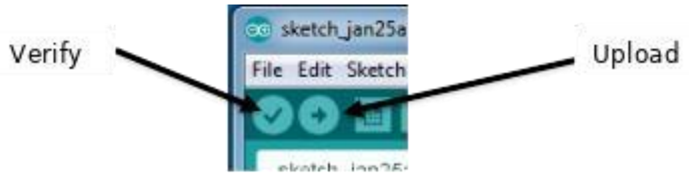


Figure 5 : Verify and upload buttons

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 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.

8. **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.

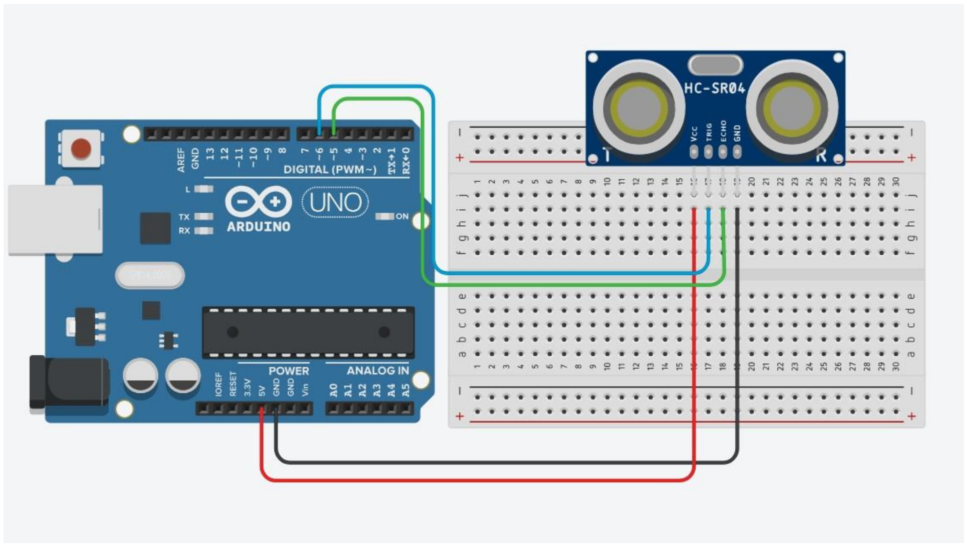


Figure 6 : Wiring of the ultrasonic sensor

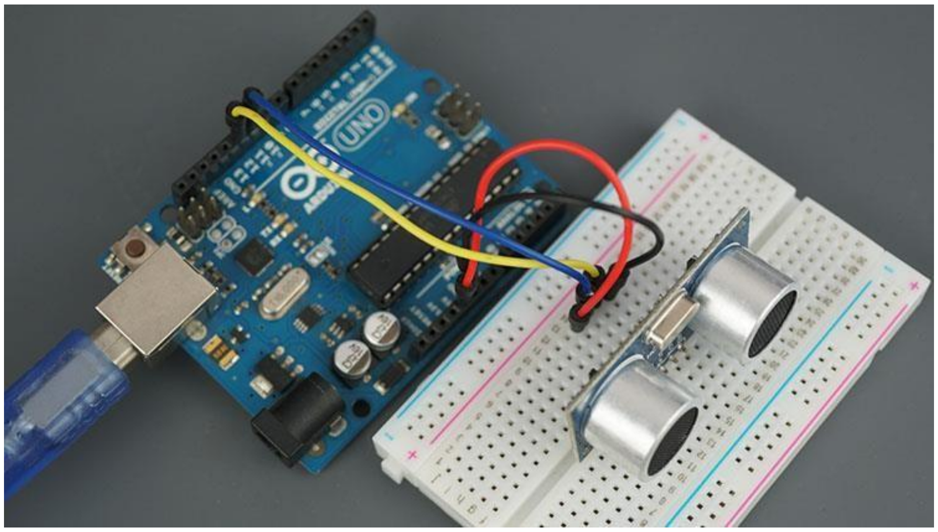


Figure 7 : Placement of the ultrasonic sensor in a breadboard

9. **Open** a new example sketch in the Arduino IDE from the "NewPing" library that should have been installed. **Go** to File→Examples→NewPing→NewPingExample.
10. 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 6 and receive an echo through pin 5.

Q1. What are the 2 steps that need to be completed to change the pin assignment (physical and code)?

Physical steps: Connect pins from 6 on the Arduino to the Trigger on the sensor, and from 5 on the Arduino to the Echo on the sensor.

Code: Set the TRIGGER_PIN on line 7 to a value of 6, and the ECHO_PIN on line 8 to a value of 5

11. **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.
12. Once the example has been uploaded, open the serial monitor to see the sensor data. **Click** the magnifying glass icon in the top right corner of the IDE, or **go** to Tools→Serial Monitor.
13. 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 right 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.**

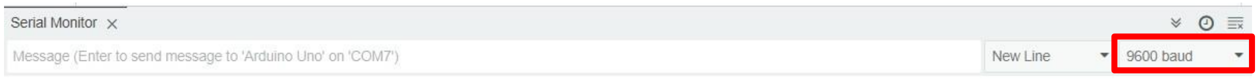


Figure 8 : Serial monitor options

Q2. What is the difference between the ECHO and the TRIG pins of the sensor?

Trig (Trigger) pin is used to trigger the ultrasonic sound pulses. Echo pin produces a pulse when the reflected signal is received.

Q3. What is the function sonar.ping_cm() calculating?

Calculating the distance in cm of the object that the sound pulse is reflecting off of.

Part C – DC Motor Control

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

14. **Disconnect** the sensor from the Arduino but don't take apart the breadboard. **Insert** the motor shield board into the Arduino headers.
15. **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.

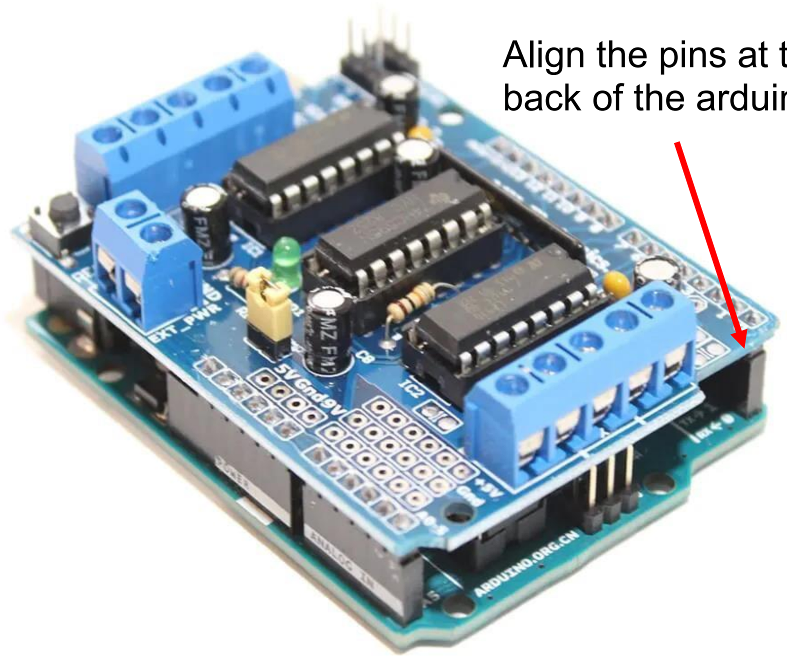

Align the pins at the back of the arduino

Figure 9 : Motor shield connected to arduino

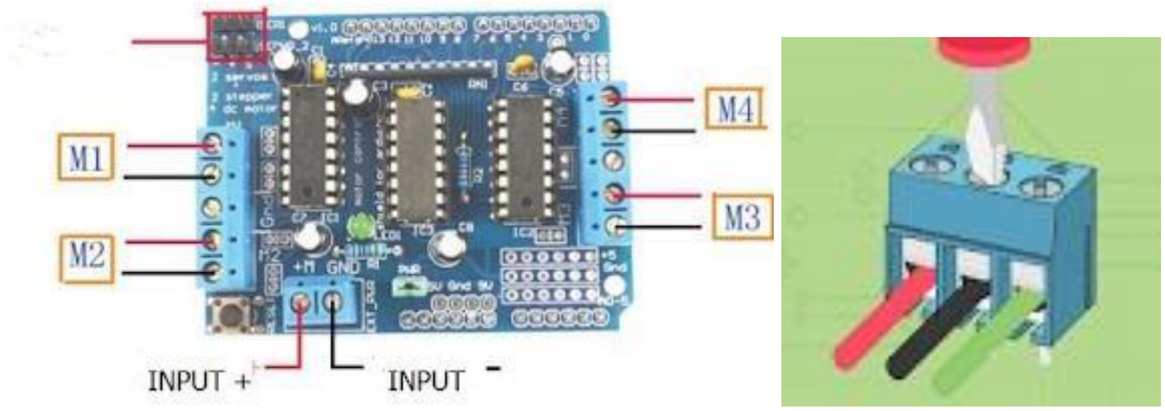
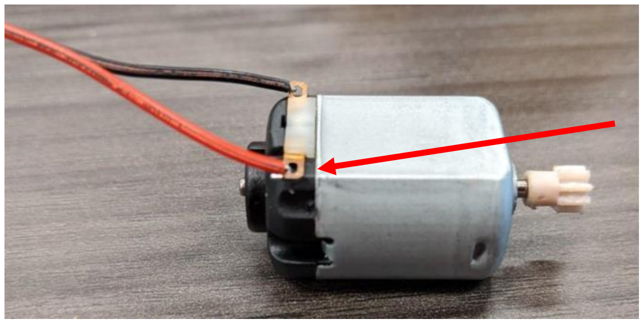


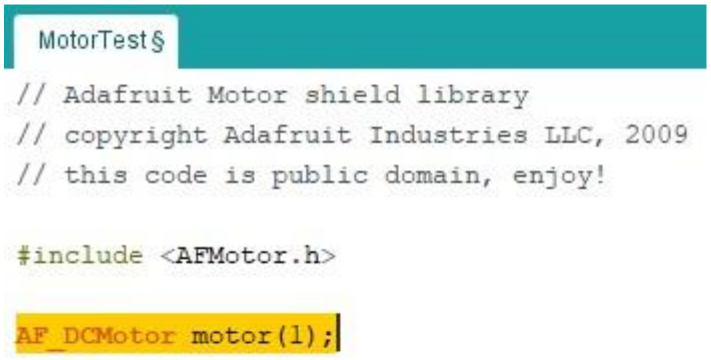
Figure 10 : Motor shield pins overview



*Be careful of the motor terminals, they are fragile

Figure 11 : Motor pin

16. **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.
17. The motor test example controls a motor connected to port M4 by default. Change this port to M1 by **changing** the line shown below. The new line should read "AF_DCMotor motor(1);".



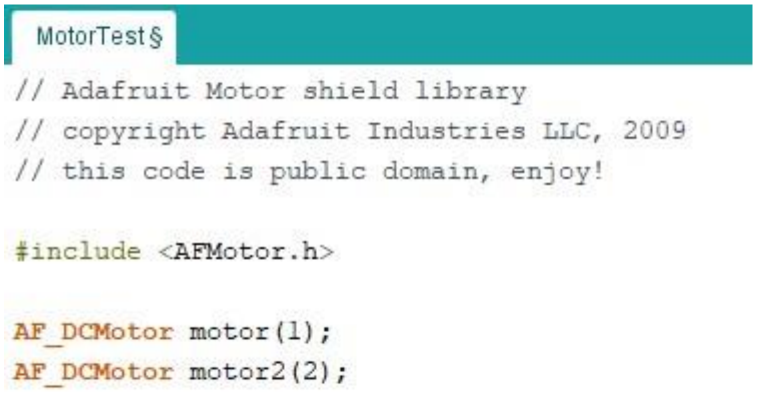
```
// Adafruit Motor shield library
// copyright Adafruit Industries LLC, 2009
// this code is public domain, enjoy!

#include <AFMotor.h>

AF_DCMotor motor(1);
```

Figure 12 : Change of motor pin from M4 to M1

18. **Compile** and **upload** this sketch to the Arduino board. **Verify** that the DC motor spins in conjunction with the programming.
19. **Swap** the connections in the M1 port. **Q4. What happens? Why?**

The rotation switches directions because the polarity is inverted.

20. 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.
21. **Add** command lines to the sketch to control the second motor.
    a. 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 17. Both AF_DCMotor statements should name their corresponding motor a unique name (see below).



```
// 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

b. 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 in order to control the second motor.

22. **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.

Q5. What happens when the function 'run' is given the parameter RELEASE?

The motor is given no direction to run, essentially put into it's defaults state instead of the forward or backward direction.

Q6. How does the 'for' loop work?

The 'run' command will set the direction of either forward or backward. Then the first for loop incrementally increases the speed from 0 to 255, then the next loop incrementally decreases the speed back down to 0.

23. **Modify** the code so that each set of messages in one loop (ticktocktech) is on a separate line. (think about how the messages were displayed in the ultrasonic example) **Q7. What is the modification you made?**

Added a 'Serial.print("\n");' statement to the end of loop function.

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.

24. On your smartphone, **download** the application called LightBlue on the AppStore or Google Play Store.
    a. https://play.google.com/store/apps/details?id=com.punchthrough.lightblueexplorer&hl=en&pli=1
    b. https://apps.apple.com/us/app/lightblue/id557428110
25. Using 4 female-female jumper wires, **connect** the AT-09 Bluetooth module to the motor shield by making the following connections:

    a. Using 4 female-female jumper wires, **connect** the ultrasonic sensor to the motor shield by making the following connections:
    b. AT-09 module GND to the SER1 (-) pin in the motor shield
    c. AT-09 module VCC to the SER1 (+) pin in the motor shield
    d. AT-09 module TXD to the SER1 (S) pin in the motor shield
    e. AT-09 module RXD to the SER2 (S) pin in the motor shield

26. Using 4 female-female jumper wires, **connect** the ultrasonic sensor to the motor shield by making the following connections:
    a. Ultrasonic sensor VCC to the SER2 (+) pin in the motor shield
    b. Ultrasonic sensor Trig to the A0 pin in the motor shield
    c. Ultrasonic sensor Echo to the A1 pin in the motor shield
    d. Ultrasonic sensor GND to the SER2 (-) pin in the motor shield

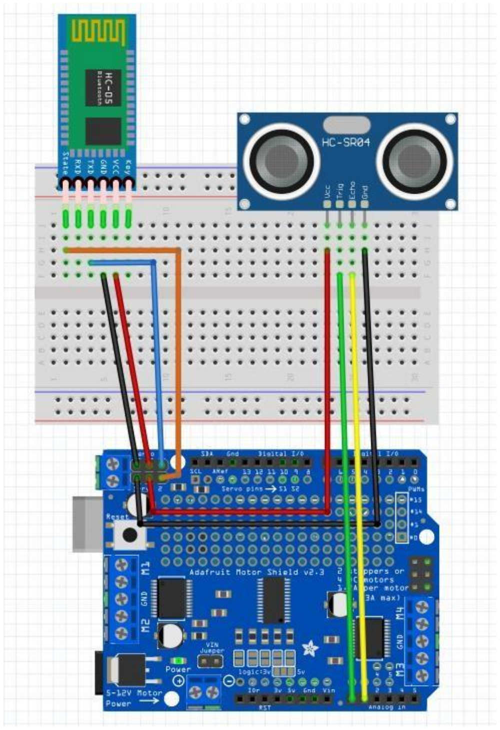


Figure 14 : Motor shield circuit

27. Now you will write the code that will allow you to control the motors with the app you just downloaded. **Start** with the Bluetooth control. Include the SoftwareSerial library as shown in the picture below.

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

28. **Initialize** the Bluetooth module pins.

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

29. In the setup function, **initialize** both the serial monitor and the serial communication.

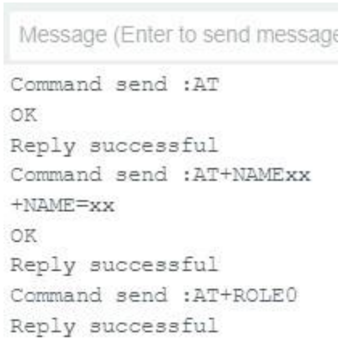
```
void setup() {
  // put your setup code here, to run once:
  mySerial.begin(9600);
  Serial.begin(9600);
}
```

30. **Write** a new function that will allow us to send commands to the Bluetooth module.

```
void sendCommand(const char* command) {
  Serial.print("Command send :");
  Serial.println(command);
  mySerial.println(command);
  //wait some time
  delay(500);

  char reply[100];
  int i = 0;
  while (mySerial.available()) {
    reply[i] = mySerial.read();
    i += 1;
  }
  //end the string
  reply[i] = '\0';
  Serial.print(reply);
  Serial.println("Reply successful");
}
```

31. Now we can configure the module by sending AT commands. In the setup, **type** these lines (you should get the response 'OK' in the monitor):
    a. Test the connection: `sendCommand("AT");`
    b. To set a new name use the +NAME command but replace xx with your name: `sendCommand("AT+NAMExx");`
    c. To set the module as a slave (0) use the +ROLE command: `sendCommand("AT+ROLE0");`
    d. **Note: Other AT commands exist and may be used to evaluate other input/output information from the system. You can type AT+HELP to see the full list of commands.**

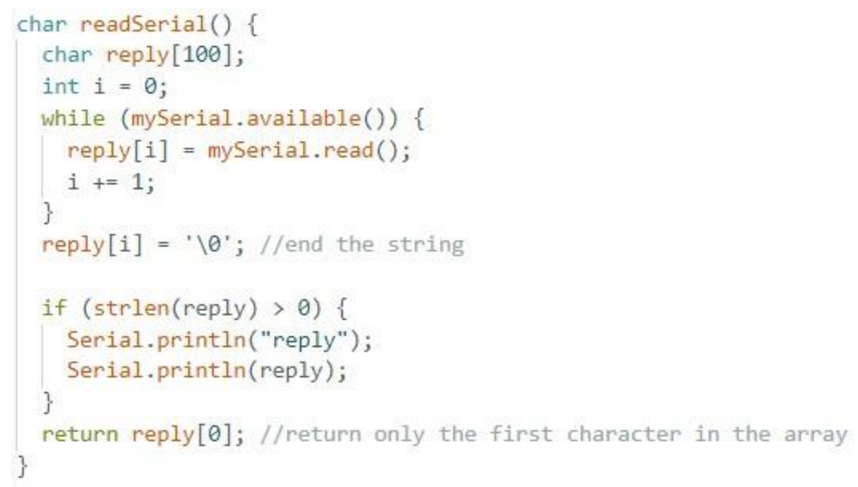
32. **Upload** the code, your serial monitor should look something like this:



Message (Enter to send message

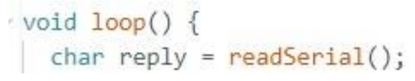
```
Command send :AT
OK
Reply successful
Command send :AT+NAMExx
+NAME=xx
OK
Reply successful
Command send :AT+ROLE0
Reply successful
```

33. **Write** a new function that will allow the Bluetooth module to read any information that is sent from the smartphone and return the first character when the function is called.



```
char readSerial() {
  char reply[100];
  int i = 0;
  while (mySerial.available()) {
    reply[i] = mySerial.read();
    i += 1;
  }
  reply[i] = '\0'; //end the string

  if (strlen(reply) > 0) {
    Serial.println("reply");
    Serial.println(reply);
  }
  return reply[0]; //return only the first character in the array
}
```

34. In the loop function, **add** the read function to monitor for data.



```
void loop() {
  char reply = readSerial();
```

35. In the app, **connect** to the Bluetooth device and go down to the ‘Characteristic’ option. Then change the data format to ‘UTF-8 String’, type a message and send it to the arduino. It should appear in the serial monitor.
36. Now based on what you learned in the previous sections, **include** the motor control in your sketch so that you can type ‘f’, ‘b’ and ‘s’ to control the motors. **Include** instructions so that the motors stop if an obstruction is detected as well. You can start from the code below and replace the missing sections (***).
    a. Check the circuit to see which pins the sensor is connected to.

```
#include <SoftwareSerial.h>


SoftwareSerial mySerial(10, 9);  // RX, TX
***                              // initialize all the ultrasonic sensor and the motor
values


//two variables to be used in the
program bool prox; int distance;
```

```
 void setup()
{
  // put your setup code here, to run once:
  mySerial.begin(9600);
  Serial.begin(9600);
   sendCommand("AT");
sendCommand("AT+NAMExx");
sendCommand("AT+ROLE0");

  ***                            //define motor speed
}  void
sensor_read() {
  distance=***;                  //read sensor value
Serial.println(distance);
  if (distance > 10 or distance == 0) {
Serial.println("No obstruction");     prox
= false;
  } else {
    Serial.println("Obstruction");
prox = true;
  }
}
void sendCommand(const char* command) {
  Serial.print("Command send :");
Serial.println(command);   mySerial.println(command);
  //wait some time
delay(500);
   char reply[100];   int i = 0;
while (mySerial.available()) {
reply[i] = mySerial.read();
i += 1;
  }
  //end the string
reply[i] = '\0';
Serial.print(reply);
  Serial.println("Reply successful");
}  char
readSerial() {
  char reply[100];   int i = 0;
while (mySerial.available()) {
reply[i] = mySerial.read();
i += 1;
  }   reply[i] = '\0'; //end the
string

  if (strlen(reply) > 0) {
Serial.println("reply");
    Serial.println(reply);
  }   return reply[0]; //return only the first character in the
array }  void loop() {   char reply = readSerial();
sensor_read(); //read the sensor value
   if (prox == false) { //check if there is an
obstruction     if (reply == 'f') {  // Go forward
      Serial.println("forward");
      ***                       //turn motors forward
    } else if (reply == 'b') {  // Go backward
      Serial.println("backward");
      ***                       //turn motors backward
    } else if (reply == 's'){
      Serial.println("stop");
      ***                       //stop the motors
    }
  } else {
    ***                         //stop the motors
  }
delay(500);
}
```

Q8. Based on the code, explain the expected behavior of the motors.

This code will spin the motors forward or backward, or stop the motor for moving depending on the input character received from the bluetooth app.
The code can also use the sensor_read() function and the prox variable to determine if there is space for teh motors to move forward or not. The prox variable is continuously updated and will act as a condition to stop the motors from moving forward if there is an obstruction.

37. Finally **compile** the sketch and upload it. Open the serial monitor and make sure that everything is working perfectly.
38. With your phone, control the arduino by using the LightBlue App. **Select** your Bluetooth module and try typing ‘f’, ‘b’ and ‘s’ in the app and see what happens to the motors. **Make sure** the motors are moving.

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/.