Coding with Arduino to operate the prosthetic arm

Setup Board

Install FTDI Drivers

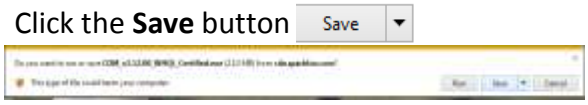
This is so that your RedBoard will be able to communicate with your computer. If you have Windows 8 or above you might already have the drivers.

1. **Download** the FTDI driver at one of the following links;

   Windows https://cdn.sparkfun.com/assets/learn_tutorials/7/4/CDM_v2.12.00_WHQL_Certified.exe

   Apple/Mac: http://www.ftdichip.com/Drivers/VCP/MacOSX/FTDIUSBSerialDriver_v2_2_18.dmg
2. Depending on the browser you are using, follow the instructions below to install the drivers.

Internet Explorer

Click the **Save** button 

Click the **Open Folder** button 

Google Chrome

Click the arrow and select **Show in Folder**

3. **Right-click** (on the 'CMD_v2.12.00_WHQL_Certified.exe' file and (Or for Apple/Mac Ctrl-click on the 'FTDIUSBSerialDriver_v2_2_18.dmg') then select Run as administrator

4. Click '**Extract**,' and continue through the installation until it finishes. Click the Extract, Next and install buttons and Accept any agreements.
5. Click the Finish button and all done…

The Interface

What do we write our code in and use to upload our code to the Redboard?

Codebender or Arduino application

Codebender (https://codebender.cc/)

- *Only works in Chrome and Firefox browsers. You have to add plugin to browser for it to work.*
- *This online application is a place you can get sample code and then manipulate/edit it to fit your uses.*
- *The code can be saved in the Cloud and as thus, you don't lose your code and also works for team environment coding.*
- *Arduino* application *is okay but saving to a single user account on the computer might not work at your school.*

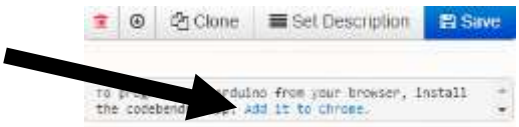
To get started with CodeBender

1. Go to https://codebender.cc
2. **Register** with a generic username and password for your students to use. Register
   a. Use an email address that you can assess to confirm registration.
3. Confirm registration by going to the email you used and click the **Confirm Account** link in the email from codebender

In codebender...

4. Click the **Create sketch** button 

5. Add the codebender plugin by clicking the **Add it to Chrome** link.
   a. Or Add it to Firefox if you are using Firefox browser



Connecting the CodeBender software to the Board

1. **Connect your RedBoard to your Computer**
   a. Use the USB cable provided in SIK kit to connect RedBoard to one of your computer's USB inputs.

In codebender...

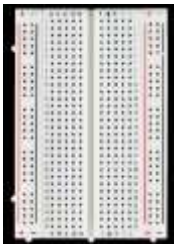
   b. Select the Arduino Uno or Sparkfun Redboard board
   c. Select a COM port
      - It is usually the highest port number.
   d. Speed should be 9600

The Objects

SparkFun Redboard

- A anti-static Prebuilt board with the following components
    2. Microcontroller
    3. USB connection
    4. Power input and outputs
    5. Digital and Analog input and outputs
    6. 32K flash memory

Breadboard



- Solderless way to connect components

What features do you see on the board?

- Power and ground, plus and minus
    7. Red generally used as positive, Black is negative or ground
    8. The entire column is connected
- Rows are grouped in 5, all connected with metal clip.
- The ditch divides the two halves. You can split it in half.
- Designed to be stuck on to something with adhesive on back.

Servo

Servo horns

- Servos have integrated gears and a shaft that can be precisely controlled through the use of coding
- Servo motors have three wires:
    9. Power - Typically red, and should be connected to the 5V pin on the Arduino board
    10. Ground - Typically black or brown and should be connected to a ground pin on the Arduino board
    11. Signal - Typically yellow, orange or white and should be connected to a digital pin on the Arduino board.

LED

- Light-emitting diode
- One leg is longer than the other.
- One directional
- Short leg needs to go to ground

Motors

- These are a pair of right angle, hobby gear motors.
- Can go up to 65 RPM.
- Have to solder the power and ground connectors to the motors.

Resistors

10K resistor
Orange-Black-Brown

330 resistor
Brown-Orange-Orange

- 10K and 330 ohms.
- Used as a current limiter in a circuit.

Temperature Sensor

- Low voltage, precision centigrade temp. sensor. It provides a voltage output that is linearly proportional to the Celsius temperature

Mini Photocell

- A very small light sensor
- Changes resistance depending on the amount of light it is exposed to.
- Great ambient light triggers (when light in the room turns on, do something).

Trimpot 10K with Knob

- Potentiometer

Momentary Pushbutton Switch

- Great for user input
- Large button head and good tactile feel (it 'clicks' really well).

H-Bridge Motor Driver 1A

- Capable of driving high voltage motors

The Arduino Code

Literacy component of coding is a basis for modeling and computational thinking more than memorizing how to write in Java.

> *Mother Jones- "the greatest contribution the young programmers bring isn't the software they write. It's the way they think. It's a principle called "computational thinking," and knowing all of the Java syntax in the world won't help if you can't think of good ways to apply it."*

The Arduino Language is a variant of C++, which supports Object Oriented Programming. The goal is to build a familiarity with the coding while working with Objects.

A lot of coding is repetitive and below is a list of the most common components of the Arduino code.

Comments

These are notes to ourselves. Comments are ignored by the Arduino when it runs the sketch

/* multiline comment */

// single line comment

This is like putting your name on your paper.

Braces or curly braces { }

These have to surround statements in the code

Copy and paste, ctrl+c and ctrl+v, (In Mac; command+c and command+v)

Ctrl Z to undo

Save often

Can't assume they have done it or use it.

Semicolon ;

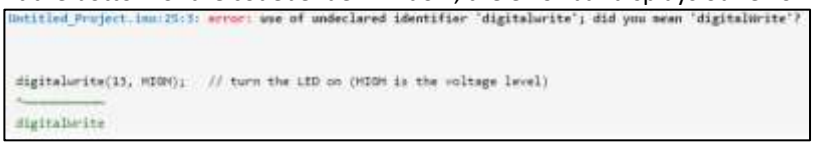
Used to end a statement

Syntax is important

Everything is as intended; it is intentional to have all CAPS or Camel case. The capitalization, semi colon, braces are required.

1. Change digitalWrite to digitalwrite by making the W lowercase.
2. Click the **Run on Arduino** button. 

At the bottom of the codebender window, the error bar displays our error.



- *There are specific commands that are reserved that Arduino has in it.*
- *There is a reference sheet to basic commands available on the nmmesa.org website.*

Putting the code into action

A *sketch* is the name that Arduino uses for a program. It's the unit of code that is uploaded to and run on an Arduino board. Make sure that your Redboard is connected to your computer using the Red USB cord.

1. Type **Blink** in the **search** box, click the **Examples** tab,
2. Select the **Blink** file that is in the **01 Basics** Library.
3. Click the **Clone & Edit** button. 

4. Click the **Run on Arduino** button. 

   The blue LED on the board will blink on and off

Read the code with comments, what do you think it is doing, what is it saying, and what features do you see?

In the **Setup** area

pinMode(13, OUTPUT)

This sets the output Pin on the board for output based on the code. Every pin you use must be configured in Setup.

In the **Void** area

Three instructions we are using in this sketch.

digitalWrite(13, HIGH);
delay (1000);
digitalWrite(13, LOW);

Digital is either off or on and in this case HIGH is on and LOW is off.
Delay is in milliseconds.

Saving your Codebender Sketch.

With the sketch open that you want to save…

1. Click on the sketch name
2. Type in a new descriptive name
3. Press the **Enter key** on the keyboard
4. Click the **Save** button
5. Verify by going back to the codebender main page.

You saved sketch should be visible in the sketch list.

Make sketch private.

To make your sketch private, Double-click the Icon to the left of the sketch name 

You can only make one sketch private per user account.

Putting it all together with Experiments

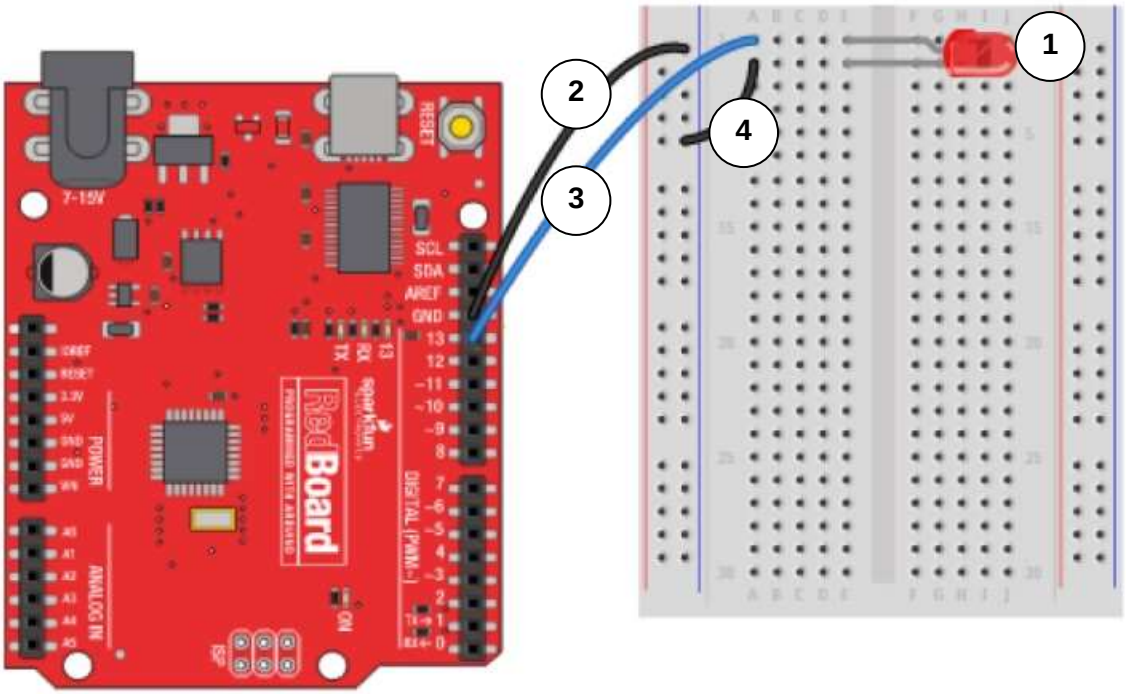
Simple LED circuit

Circuit, has the base word of circle.

- Our circle is; from power on the board to pin 13 through LED back to ground on the board.
- Arduino PINs are limited to 40 mA (Milliamps)
- The LED will pop if you put it into 5v and ground it will burn out the LED.
- The pins have a current limiter so they are safe.
- Short leg needs to go to ground

Wiring diagram and instructions for Simple LED circuit

1. Plug a LED into the Breadboard with the **long pin in Row 1** and the **short pin on the LED in Row 2**
2. Connect **Negative (-)** on the Breadboard to **GND** on the RedBoard
3. Connect **A1** on the Breadboard to **PIN 13** on the Redboard.
4. Connect **A2** on the Breadboard to **Negative (-)** on the Breadboard



The Dimmer Experiment

In this experiment, we will be Sensing using analog inputs.
Using the Trimpot, we will be showing how you can use an object as a sensor. The Trimpot is a Potentiometer; Variable resistor. Using code and the serial monitor we can observe the changes as it senses the value and sends the data back to the computer.

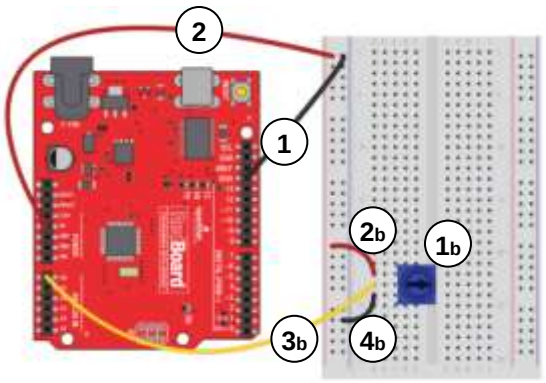
Build the Board

Power the BreadBoard

1a. Connect Negative (-) on the Breadboard to GND on the RedBoard
2a. Connect Positive (+) on the Breadboard to 5v on the RedBoard

Power the Trimpot

1b. Plug the Trimpot onto the Breadboard. It has 3 legs that come off of it and each should be in a separate row.
2b. Connect Positive (+) on the Breadboard to the row on the breadboard that the top leg of the Trimpot is connected to.
3b. Connect **A0** PIN on the Redboard to the row on the breadboard that the middle leg of the Trimpot is connected to.
4b. Connect Negative (-) on the Breadboard to the row on the breadboard that the bottom leg of the Trimpot is connected to.



Upload the Code

1. Go to https://codebender.cc/sketch:142450
   a. Or type in **MESA Analog** in the search box in Codebender.
2. Click the **Run on Arduino** button. 

View the Trimpot's output in the Serial Monitor section…

3. Make sure the baud rate is **9600** (data rate bits per second). This will establish what speed it will be.
4. Click **Connect** button.
5. You will be able to see the output from the Trimpot on the screen.
   a. Adjust the Trimpot by turning the knob and look at the results in the serial monitor.
6. Click **Disconnect** button when done *(was the Connect button)*.

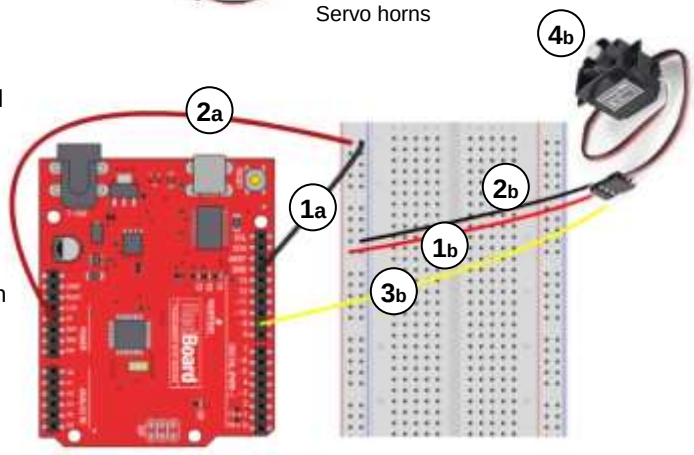
Read the code, what is it doing?

```
void setup()
{
/* Initialize the serial communication at 9600 bits per second. This allows you to send or receive sensor values and other data between the Arduino and another device (like your computer). 9600 is the standard speed most devices use.*/
        Serial.begin(9600);
}
void loop()
{
/* create an integer variable (int) called val make val equal to the sensor reading on pin A0 you should get a sensor reading between 0 and 1023*/
        int val = analogRead(A0);
//print the string "Val = " over the serial port
        Serial.print("Val = ");
//print the variable val and end the line with a carriage return
        Serial.println(val);
//wait a bit
        delay(100);
// Upload this code and open your serial Monitor to see the data.
}
```

Using the Servo experiment

In this project, we are going to introduce the use of a special type of motor called a Servo. These motors are great for precision movement of things like latches, grippers, or other linkages.

Servo horns



Build the Board

Power the BreadBoard

1a. Connect Negative (-) on the Breadboard to GND on the RedBoard

2a. Connect Positive (+) on the Breadboard to 5v on the RedBoard

Power the Servo

There is a 3-wire harness connected to the Servo One for Power (5V), One for Ground (GND), and the third for a Signal.

1b. Connect Positive (+) on the Breadboard to the red wire from the Servo.

2b. Connect Negative (-) on the Breadboard to the black wire from the Servo.

3b. Connect the ~9 PIN on the RedBoard to the white wire from the Servo.

a. The ~ on the PIN number means that this PIN uses PWM (Pulse width modulation)

4b. Attach a Servo horn (arm) to the Servo

Upload the Code

1. Go to https://codebender.cc/sketch:140905
   a. Or type in **MESA servo** in the search box in Codebender
2. Click the **Run on Arduino** button. 

Read the code, what is it doing?

```
//include the Servo library in your sketch
#include <Servo.h>

//create a servo object called myServo
Servo myServo;

void setup()
{
//attach myServo to pin 9
myServo.attach(9);

//write the angle of 90 to myServo once
myServo.write(90);
}

void loop()
{
 myServo.write(170);
 delay(500);
 myServo.write(10);
delay(500);
;
}
```

The first line includes the **`Servo.h`** library into the sketch. This provides access to new commands and functions to control special devices - like a servo motor.
Notice that this is one of the few instances where there is no semicolon at the end of the line.

Next, **Servo myServo** - this creates an object that is called myServo. An object is similar to a variable that has extra properties and commands. The two commands that we care about are: attach `()` and `.` write `()` .

The **myServo.attach(9)** is similar to a setup command. It links the `myServo` object to pin 9 on the Arduino.

Finally, **myServo.write(90)** is the command that tells the servo motor what angle to rotate to. The range of motion for a servo is roughly 180 degrees. You can use any value from 0 to 180 with this command. The placement in setup sets our starting angle at 90 degrees.

In the loop, we have the Servo going from 170 degrees to 10 degrees over and over again. The **delay**() commands tell the Arduino to do nothing for 500 milliseconds so the servo horn has time to move.

Notes:

- The servo has a limited range of motion. Roughly, how far does it rotate? It can go up to 180 degrees.
  Warning: Don't go 0 to 180 degrees, Avoid pushing the motors to these limits. It can damage the motors.
- To stop the servo, you can just pull one of the power lines in case the noise is bugging people.
- To clear the code on the board, you can simply upload a bare minimum sketch to the board.