Let's Code Blacksburg's Arduino Cookbook

Version .1 11/2013

Blink Recipe

What:

Arduino Software and driver should be installed. (See install page).

Select appropriate port
“Tools” “Serial port”

Select Arduino Uno.
“Tools” “Board” “Arduino Uno”
Load blink onto the arduino.

Look for the light blinking here.

Change both lines that say delay(1000); to delay(100); and reupload the program.

Look for the light to blink faster. This shows that the program changes you made are actually uploaded to the Arduino.

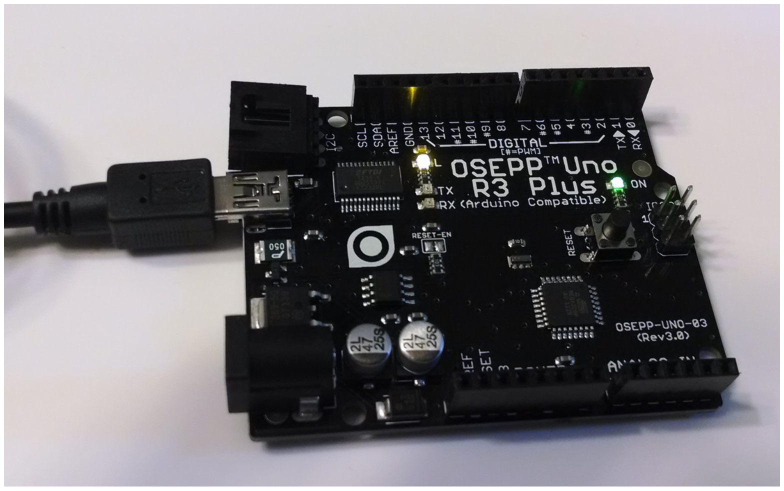


Illustration 1: Arduino Clone with power and "blink" light

How:

This process compiles and uploads code to the Arduino for execution.

There is a small LED (the light) connected to pin 13 of the Arduino.   When that pin is 'high' (meaning +5 volts for this Arduino clone), the LED lights. This is done by the command *digitalWrite(led, HIGH);*

When pin 13 is 'low' (meaning connected to ground) the LED (light) is off. This is done by the command *digitalWrite(led, LOW);*

Note that anything following “//” on a line is considered comments and ignored by the compiler.

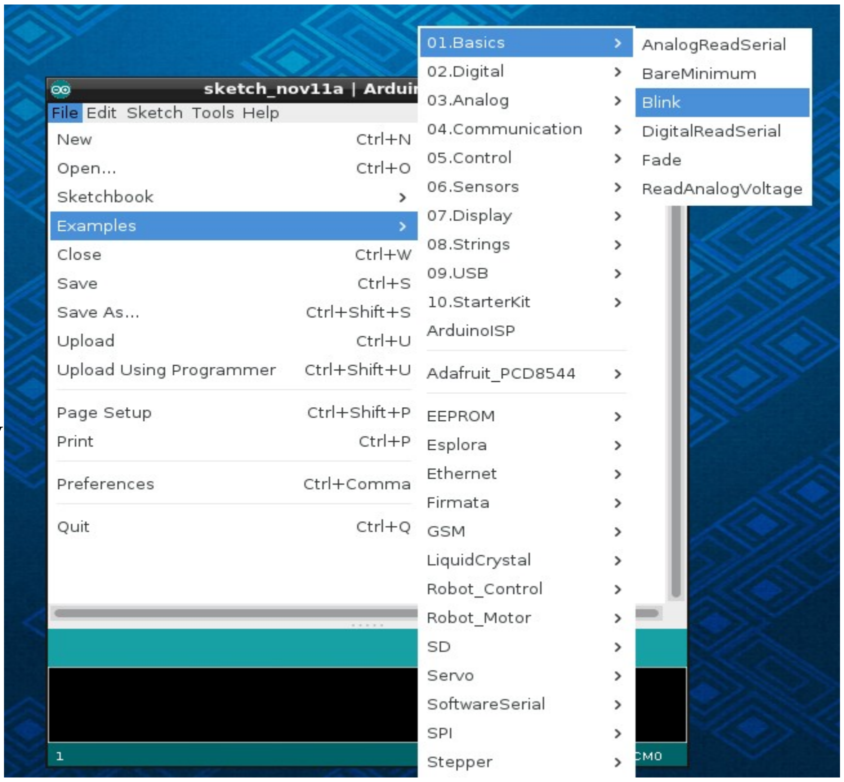


Illustration 2: Loading the "blink" program in the Arduino IDE

Fail:

Verify the correct serial port - tools menu
Unplug Arduino and list serial ports,
then plug up Arduino and llist serial ports again. An additional serial port should appear. That new serial port should be the Arduino port.
Try pluging Arduino into a different port USB port
Reboot your computer.
Test with a different cable and Arduino board.

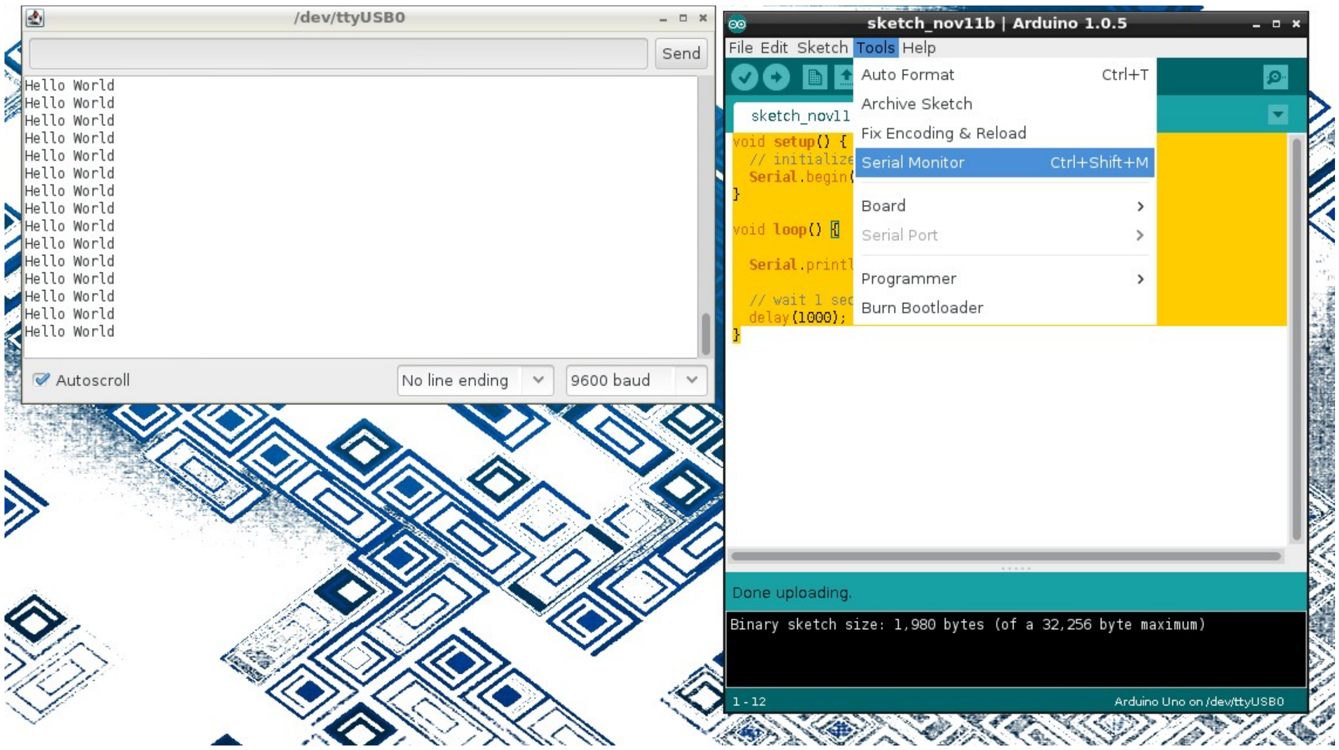
Serial Monitor Recipe

What:

Upload and execute the following sketch on the Arduino.

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {

  Serial.println("Hello World");

  // wait 1 second before the next loop
  delay(1000);
}
```

Then open the Serial Monitor by selection “Tools” and “Serial Monitor”.
You should see the line “Hello World” appear repeatedly in the window.



```
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

Autoscroll — No line ending — 9600 baud

```
Done uploading.
Binary sketch size: 1,980 bytes (of a 32,256 byte maximum)
```

How:

Serial Begin opens the serial port at on the Arduino and sets its speed to 9600 baud
The Serial.println statement prints a line followed by a newline.
When the Serial Monitor is opened it looks for characters appearing on the laptop serial port and prints them in the window.

Fail:

Look for RX and TX lights to blink on Arduino rapidly during upload.
If program is running successfully look for the TX light to blink once per second, showing that it is "trying" to transmitting data back to the laptop.
Double check your Serial Monitor settings for both port and speed.

Note: you have to restart the Serial Monitor after each upload of a new program, because the upload process uses the same serial port connection.

Breadboard or Protoboard Recipe

How does a breadboard work?
Wire pins, pushed into the breadboard are connected together as shown in the schematic below. This allows the quick building and testing of circuits.

Note: Unplug the Arduino from the laptop (and any other power supply while making and verifying connections).

Connection wire colors do not matter; but traditionally power (+) wires are red and ground (-) wires are black and those choices can help make troubleshooting a little easier.

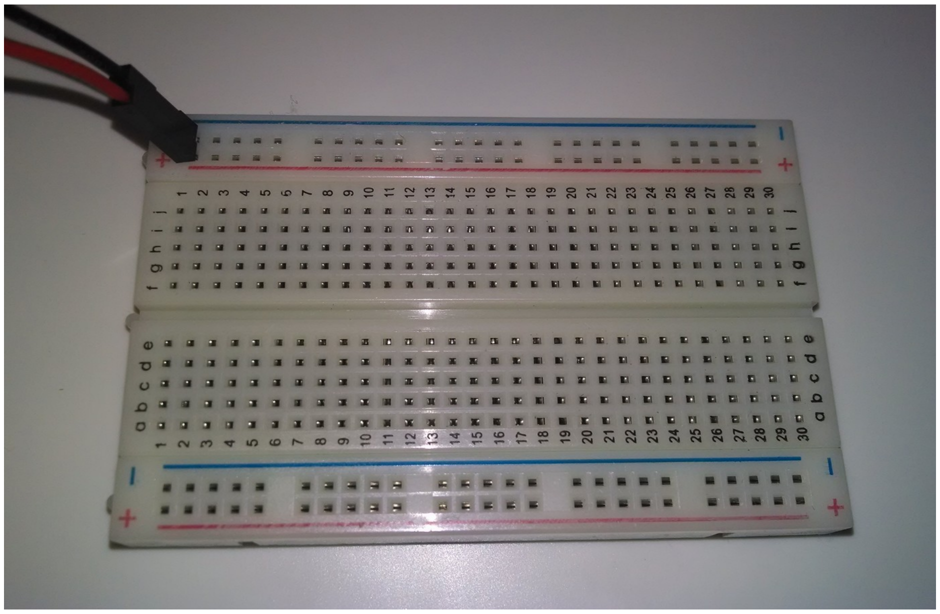


Illustration 3: "breadboard" or "protoboard"

Protoboard schematic from Wikipedia

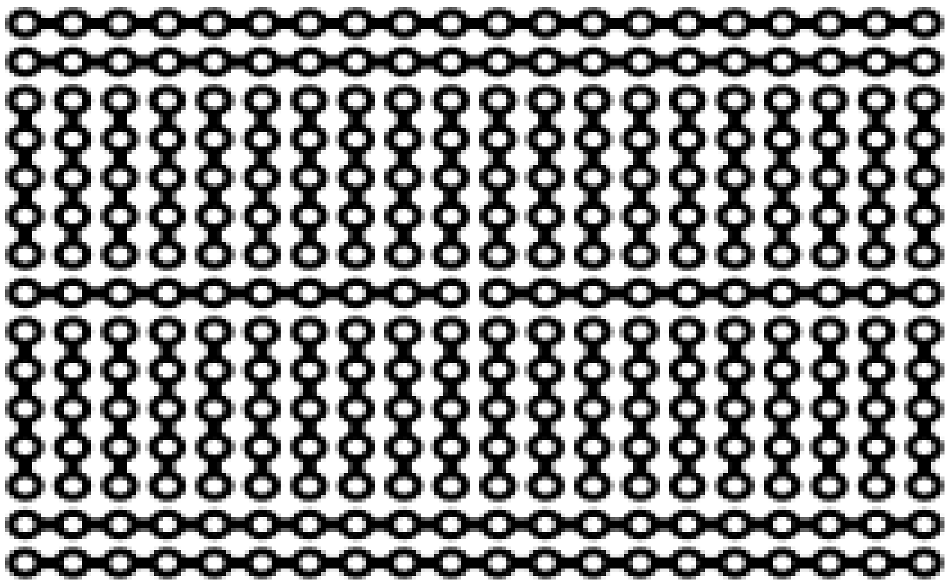
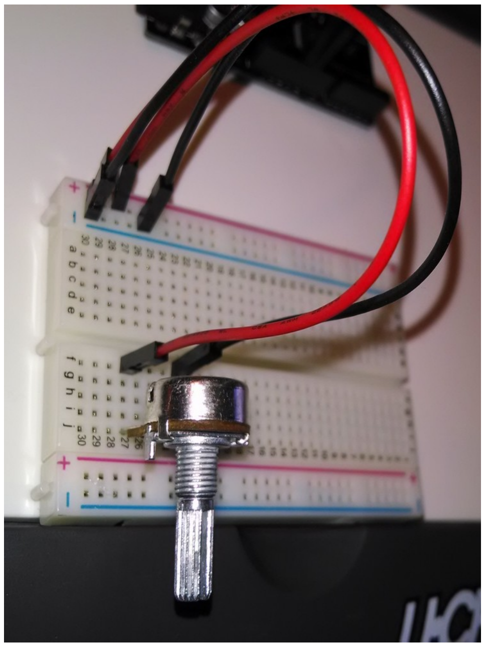


Illustration 4: Protoboard schematic. -Wikipedia

Potentiometer Recipe

What:
Firmly insert the potentiometer (also called a "pot") into the breadboard.
Connect the leftmost side to power and the rightmost side to ground.
The middle connector is the output of the pot in this case.
Connect it to an analog in pin on the Arduino.



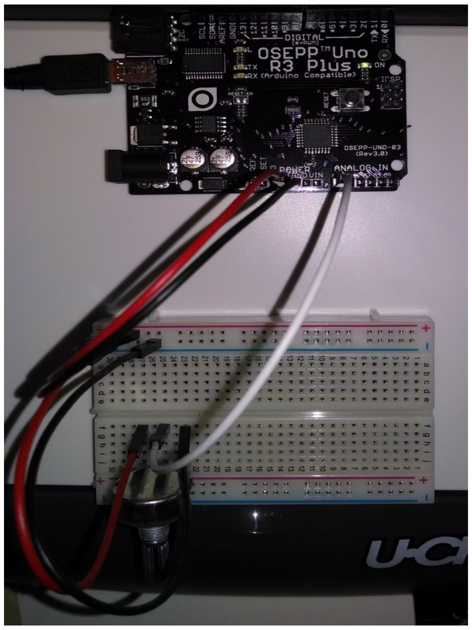
A0 is a good place to start.

Verify the circuit by reading and printing the pot value.

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  int sensorValue0;
  sensorValue0 = analogRead(A0);  // read the analog input
  Serial.print ("Pot 0 value=");
  Serial.println(sensorValue0);

  // wait 1 second before the next loop
  delay(1000);
}
```

Turn the pot left and right, the printed value should go from (near) 0 to (near) 1023

How:
The potentiometer implements a voltage divider circuit with "two" variable resistors. The out pin should vary as the potentiometer knob is turned from 0 volts (ground) to 5 volts. That is the value of the voltages applied on the outside pins. /*/



Fail:

Verify the pot is seated firmly in the protoboard
Verify 5 volts across the Pot with a multi-meter
ground one lead of the multi-meter and look for a varying voltage on the output pin of the pot.
Are you reading the input value into a variable?
Are you printing the correct variable for testing?
/*/ Discuss 'deadband' and "print on change"

Servo Recipe

Connect a servo to the Arduino.

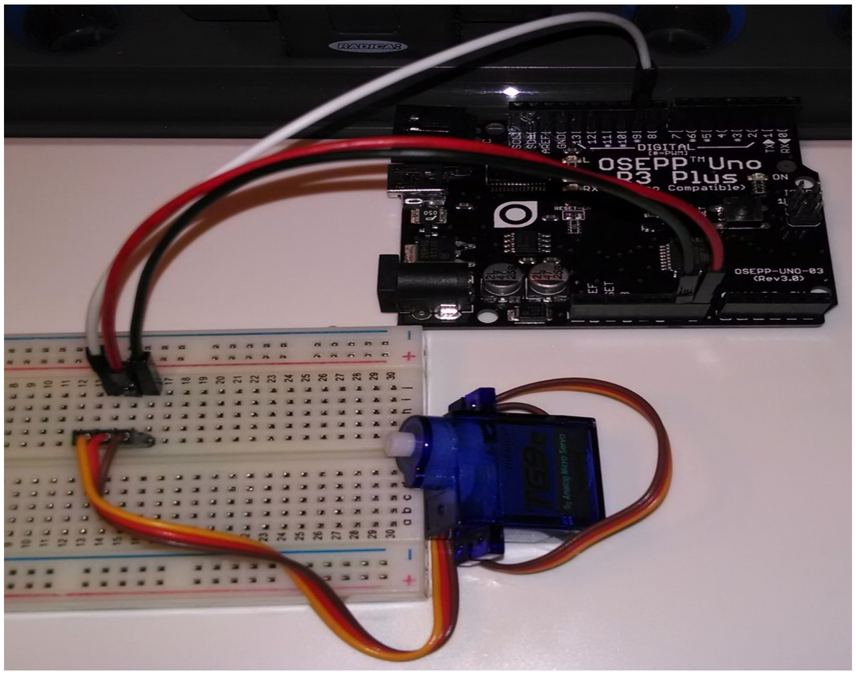
Use a 3 pin header to connect the servo to the protoboard.

- Connect the brown wire on the servo to ground.
- Connect the red wire on the servo to the Vin pin on the Arduino.
- Connect the orange wire to pin 9 on the Arduino.

Use the following code to test the servo.

```
#include <Servo.h>
// create a servo object
  Servo servo0;

void setup() {
  servo0.attach(9);  // servo is attached to pin 9
}

void loop() {

  servo0.write(60);  // tell servo to go to the 60 degree position
  delay(1000);       // wait 1 second
  servo0.write(120); // tell servo to go to the 120 degree position
  delay(1000);       // wait 1 second

}
```

Look for the preceding code to move the servo to the “60 degree” position, wait 1 second, then move the servo to the “120 degree” position. The actual movement degree may vary somewhat depending on the servo. The initial position of the servo horn may be changed by gently pulling it up, off the servo, turning it and then pressing it back down, re-engaging the “teeth” in a different rotational position.



How:
The position of a servo is set by sending it a 1 – 2 millisecond pulse. A 1ms pulse represents approximately 0 degrees. A 1.5 ms pulse represents approximately 90 degrees. A 2 ms pulse represents a servo position of 180 degrees.

This pulse should be sent every 20 ms or so. The exact timing between pulses is not critical.

While we could easily write code to pulse the servo control the proper time (between 1 and 2 ms) every 20 ms, the Servo library used above takes care of that for us and can control multiple servos simultaneously.

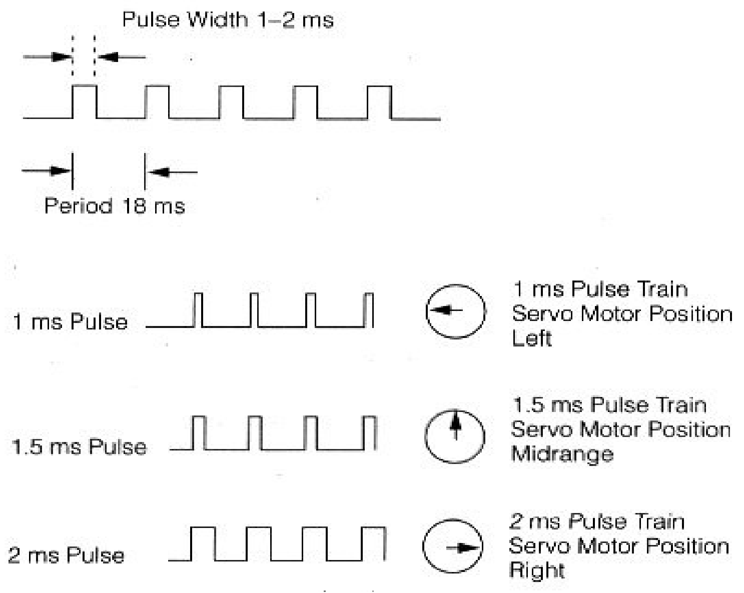


Illustration 5: Source: seattlerobotics.org

Note, servo range may vary; not all servos have a full 180 degree range.

Fail:

Unplug servo power line an plug in back in. Listen for servo to make a small move if power is connected properly.

Servo's can consume more power than available from the Arduino and from laptop USB port. Try connection the battery pack for additional power.

Check that the values printed to the Serial Monitor make sense as the pot is moved.

Make sure the orange wire is connected to the correct pin in the Arduino, especially if there is more than one servo connected. Change to servo control pin defined in the software if necessary.

If the electrical connections are suspect, try replacing the 3 pin headers with 3 jumper wires

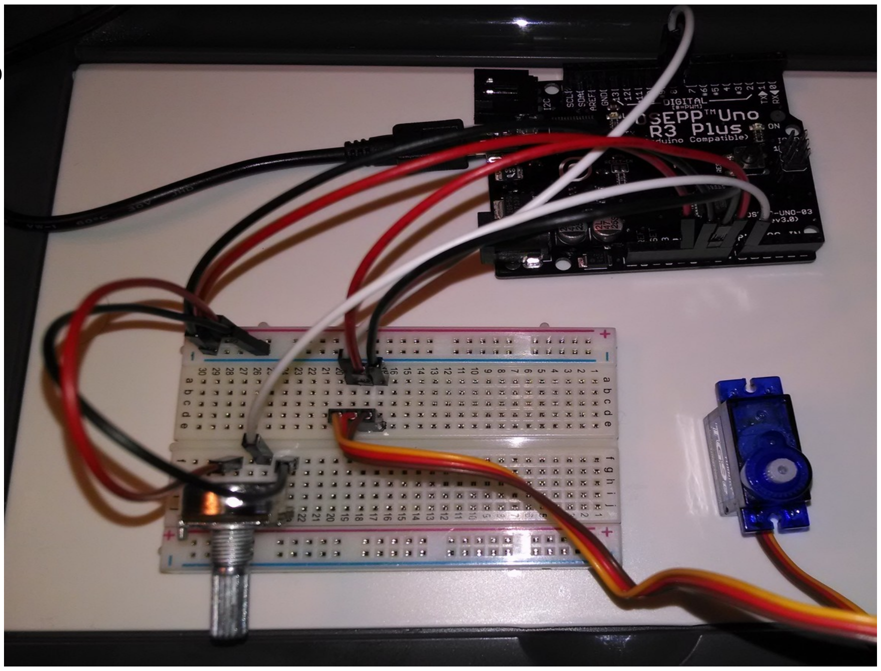
Note: placing a 330 ohm series resistor on the servo input line is a good idea to help protect the circuit in case of mis-wiring

Potentiometer Control of a Servo's Position Recipe

In order to use a potentiometer to control a servo:

- Build the potentiometer circuit connecting the pot output to the Arduino pin A0 (analog 0).
- Build the Servo circuit connection the servo input wire to Pin 9 of the Arduino.
- Use the program below

Here is a sample of how it could be wired up.



After testing the Pot and servo separately use the following code to test Servo control by a potentiometer.

Turning the pot left and right should cause a corresponding rotation in the servo.

The servo may move "faster" or "slower" than the pot depending on the map in the code below.

(We are using the values 30 to 150 for the degree settings, because some servo's have difficulties at the extremes of their movement.)

```
#include <Servo.h>  // servo library
// create a servo object
  Servo servo0;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  servo0.attach(9);  // servo is attached to pin 9
}

void loop() {
  int servo0Setting=90;  // for the servo position later

  // Read Pot value
  int sensorValue0;

  sensorValue0 = analogRead(A0);
  Serial.print ("Pot 0 value=");
  Serial.println(sensorValue0);


  // Map the pot reading to the servo degree setting
  //    we'll use 30 to 150 degrees for the servo

  servo0Setting=map(sensorValue0,0,1023,30,150);
  Serial.print ("  Servo 0 setting=");
  Serial.println (servo0Setting);


  // Set Servo position
  servo0.write(servo0Setting);  // tell servo to go to the designated position


}
```

Fail:
You can change the direction of the servo to potentiometer mapping by swapping the pot input values in the map statement.
e.g. change this:

```
  servo0Setting=map(sensorValue0,0,1023,30,150);
```

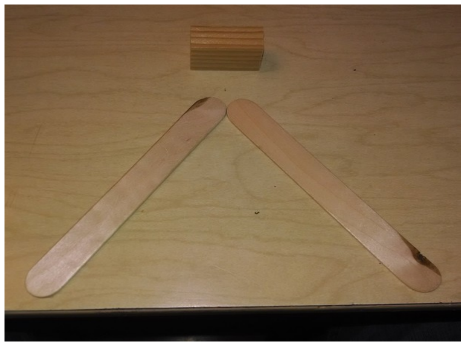
to this:

```
 servo0Setting=map(sensorValue0,1023,0,30,150);
```

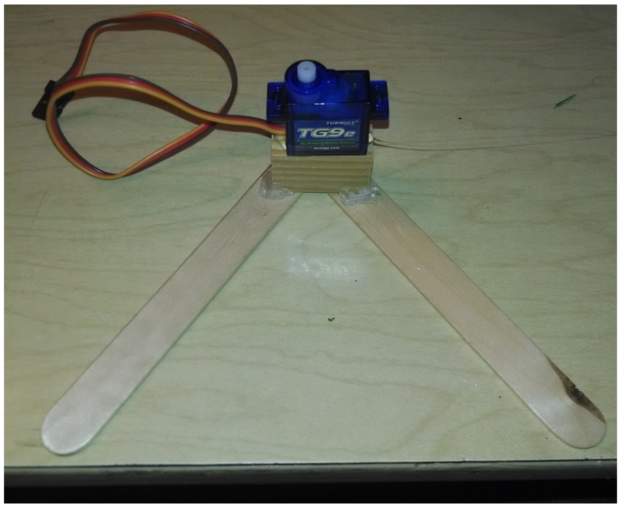
Of course different pins for the pot and servo may be used, but the test program will have to be modified to match.

Building Monta's Robot Drawing Arm Recipe

1. Glue together servo base
   1. position 2 touching craft sticks at approximately 90 degrees
   2. **see picture to obtain the proper orientation of the spacer block** (“narrow edge on top”)



   3. hot glue the 2 craft sticks together using the wooden spacer block
   4. hot glue servo1 to spacer block



   5. cover center of the flat side of both servo horns with a corner of tape to make sure the hot glue will not go through the center hole. (I'll repeat this step below for those that didn't see it here :))

Note: when building the arms, be sure to leave a little space at each end of the craft stick (perhaps .5 centimeter or so). If the arms are too long, the pencil may leave the edges of the paper when drawing and catch on the paper edge.

2. Build Arm 1
    1. cover the centers of the flat side of the servo horns with a corner of blue tape
    2. hot glue servo horn on one end of the craft stick
    3. hot glue servo2 on the other end **with shaft toward the middle of the stick**

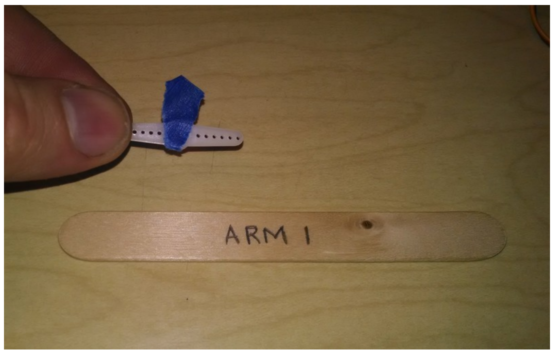


Illustration 7: Arm 1

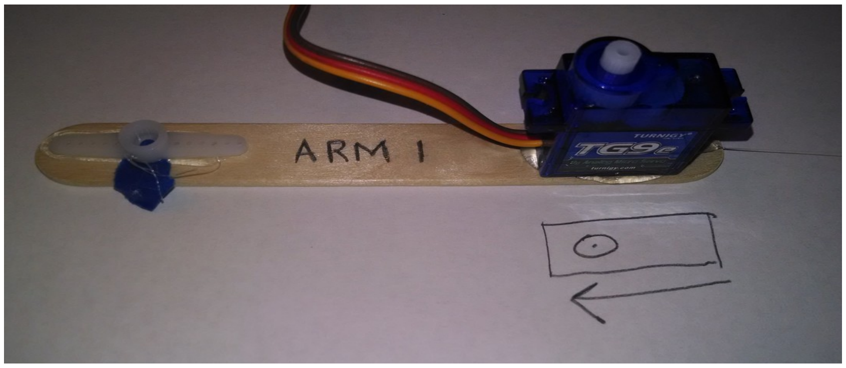


Illustration 6: Arm 1

3. Install cable servo extension cable to servo 2 noting the polarity
    1. Remove the 3 pin header from servo wire (if attached)- leave it in the protoboard
    2. The black wire from the extension cable should be on the same side as the brown wire from the servo (and the white wire should be directly across from the orange wire)

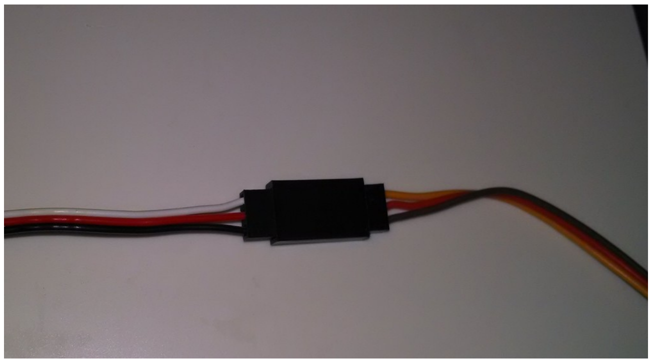


Illustration 8: Servo extension cable

4. Build Arm 2
   1. Cover the center of the flat side of the servo horn with a corner of blue tape
   2. Attach servo horn to arm 2 using hot glue (pencil will be attached later)

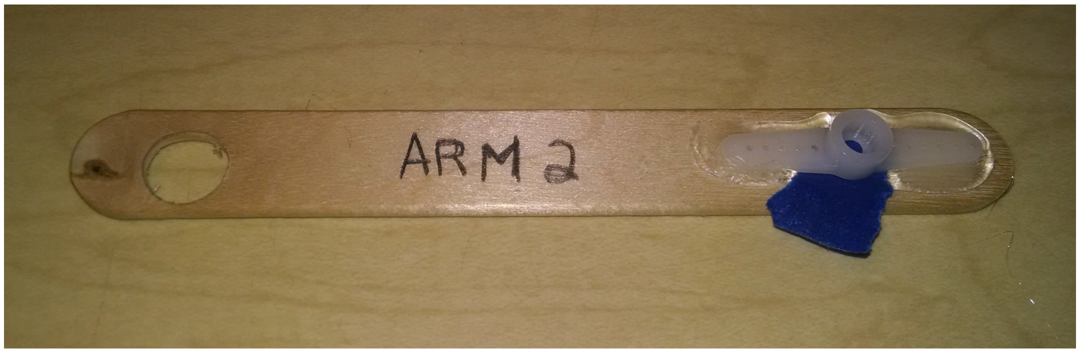
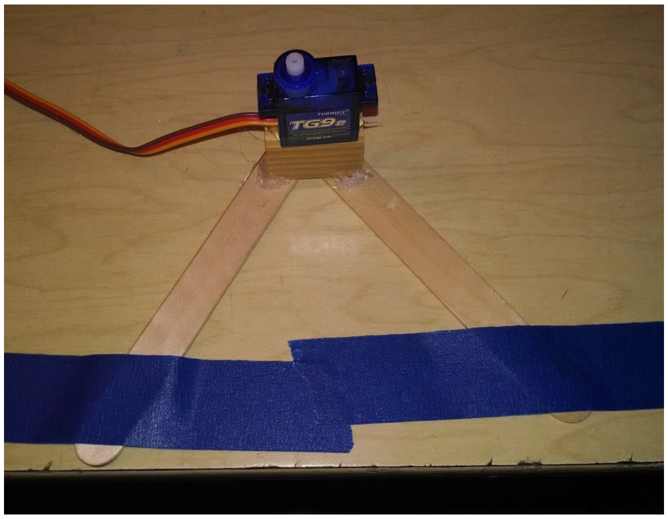


Illustration 9: arm 2 construction

5. Use blue tape to tape the servo base (with servo) to the table, leave some space near the servo for the drawing paper to slide under later (perhaps 5 cm).



6. Fit the arms on the servos
   1. **Turn both servos to their rightmost position using the potentiometers**
   2. Press the arms and servos together **in the position as seen in picture** (to maximize drawing area of the arm). The arms are pressed onto the servos, with the servo horn fitting on the toothed servo shaft. Exact placement isn't critical, and the arms may have to be turned slightly in order for the teeth on the servo to line up with the servo horns. The hole on Arm 2 should be as close to servo 1 as practical without binding.
   3. A small dot of hot glue, or piece of tape directly over the shaft of servo 1 can be used to hold servo 2's wire in place during arm movement.

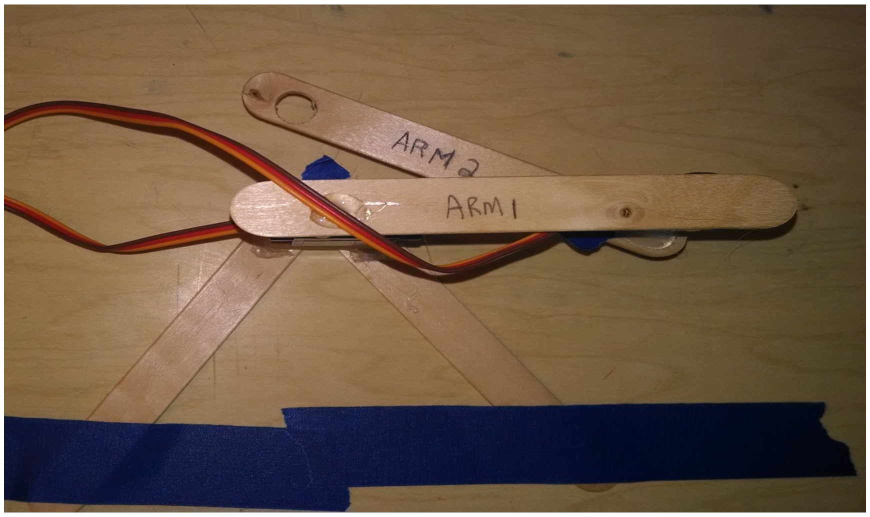


Illustration 10: Robot Arm Fit photo 1

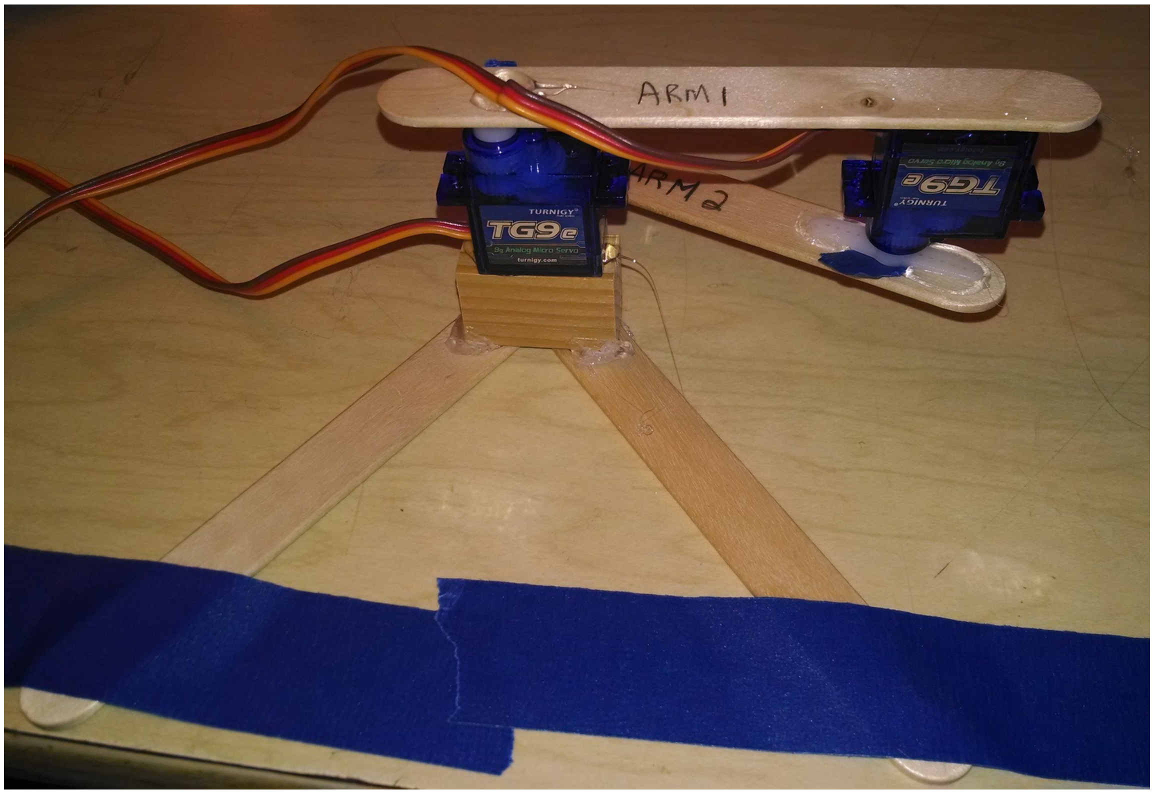
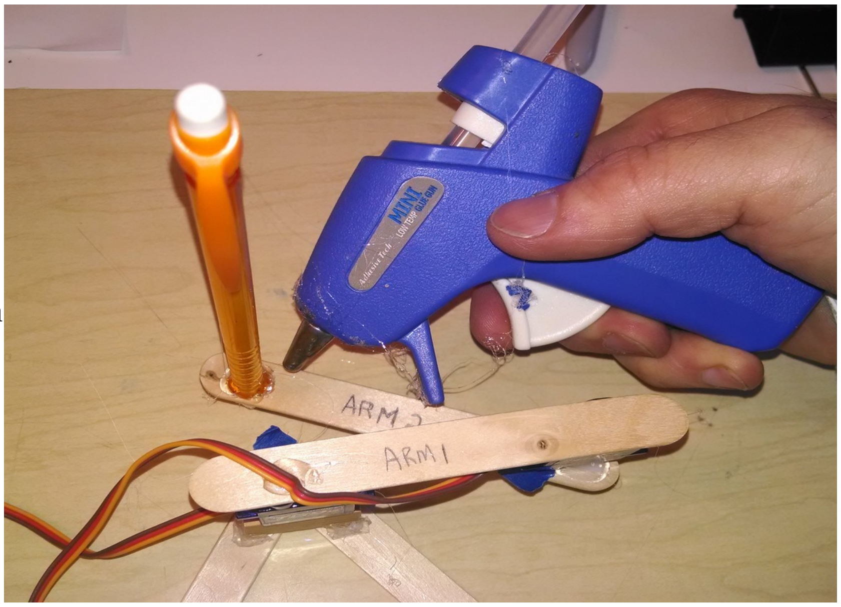


Illustration 11: Robot arm fit, photo 2

7. Fit Pencil
    1. **First retract pencil lead fully**
    2. Hot glue pencil in place, **lifting the arm slightly until the hot glue cools.** *This is to make sure that there is some pressure on the pencil tip when it is finished.*
    3. After glue has cooled extend pencil lead.
    4. You may want to change the direction of the servo(s) to match the expected potentiometer direction. See the Servo Recipe, fail notes for details.



FAIL
Extend the pencil lead further than normal
Strategically place a penny under the "base" to level arm 1
Hot glue some pennies near the pencil on arm 2 to increase downward force
Pry apart pieces that are glued together and re-glue them to level arm 2
Tape servo wires to the table to keep them out of the way and prevent them from being pulled loose

<u>FutureRecipes to Add</u>

Installing Arduino Software and drivers
- linux
- mac
- windows

Arduino Coding
- program sections
- reference commands
- compile and upload

Light and external LEDs
- 20ma limits
- resistor
- pin modes
- dim light means port wasn't set to output
- pwm

Drive a motor / transistor
- 20ma limits
- biasing a transistor
- pin modes
- pwm

Resistors and Ohms law recipe
Using a multi-meter
Ultrasonic Distance Sensor
Infrared detectors
Passive Infrared detectors
bluetooth
soldering skills

far future
GPS
9 DOF GYRO
USB hids devices