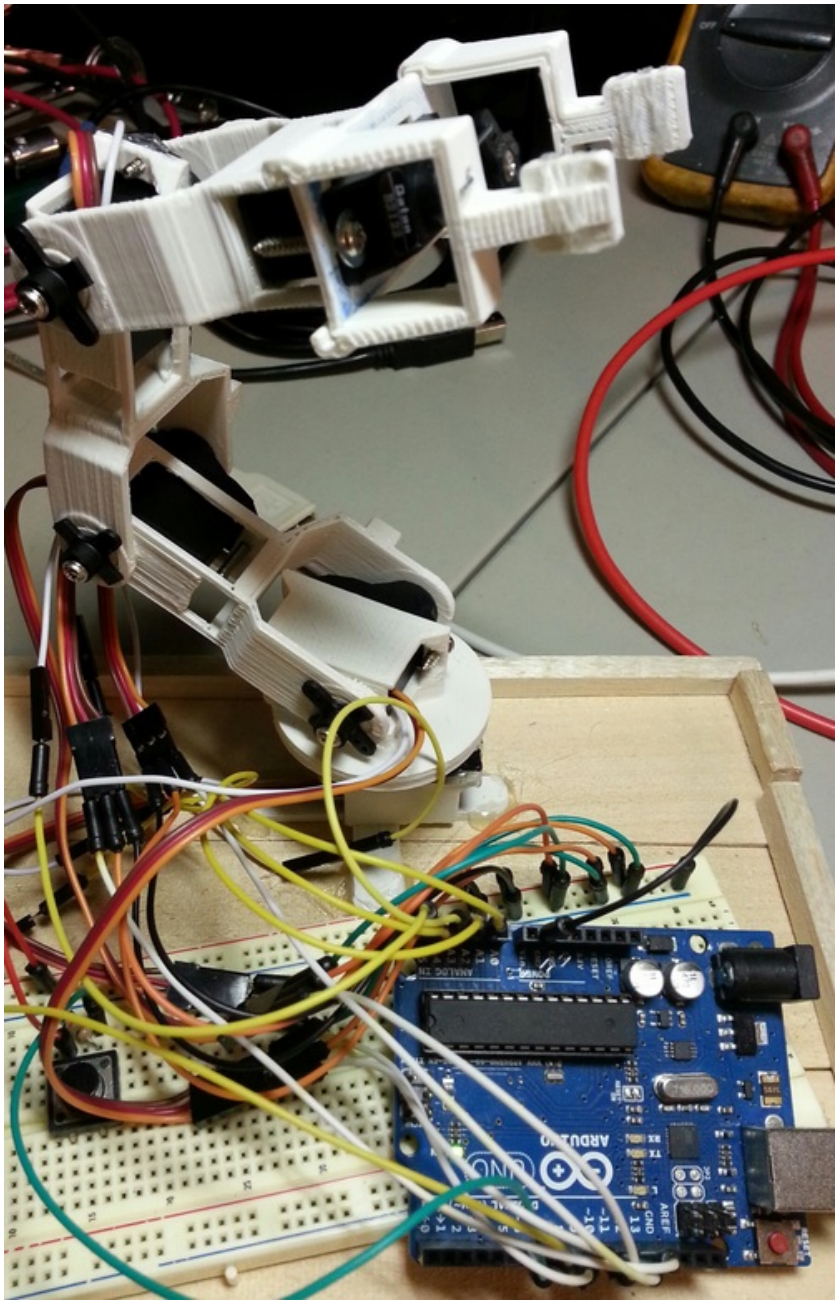
Trainable Robotic Arm

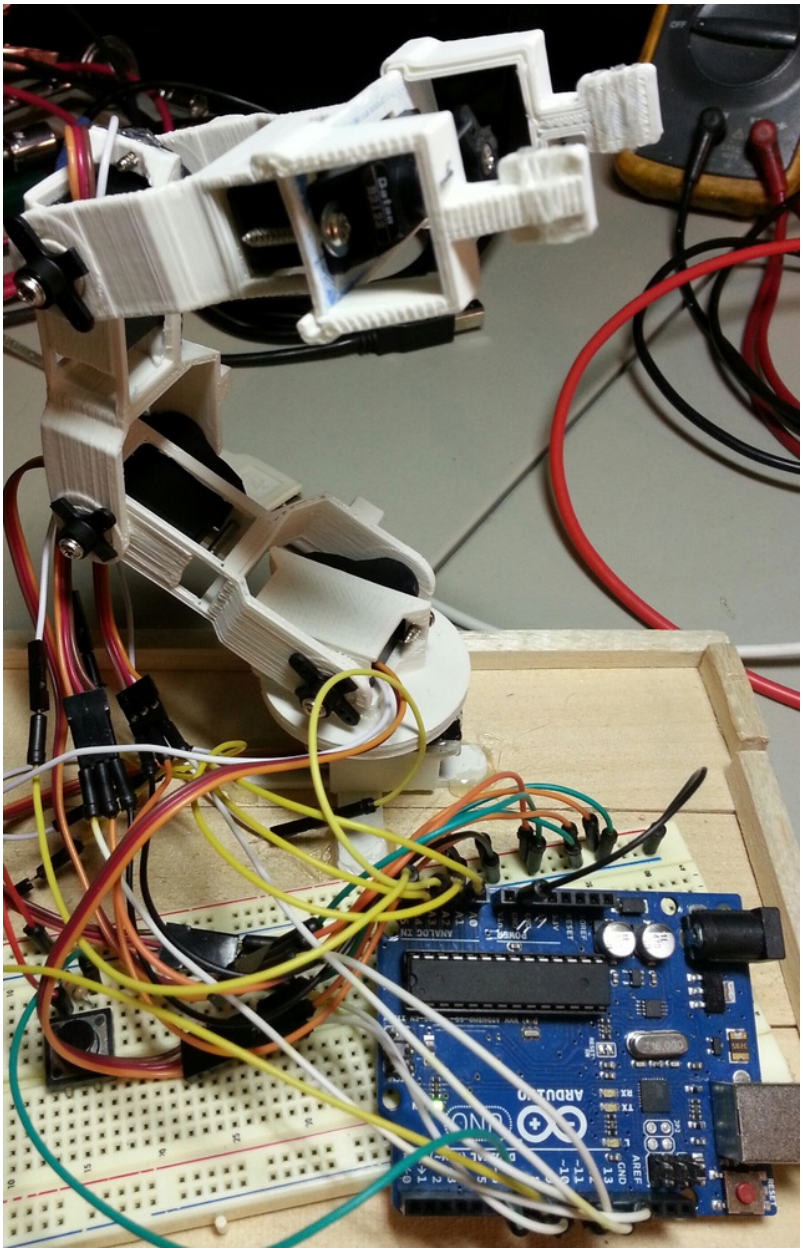
Created by Robert Svec



Last updated on 2013-11-12 06:00:29 PM EST

Guide Contents

Overview



The Baxter robot (http://adafru.it/cUz) can be easily trained to perform actions by simply moving his arms and grippers with your own hands while he records the motions. Analog feedback servos provide a way around the complicated kinematics necessary to make robotic arms operate efficiently. Interacting with a robotic arm is lots of fun and being able to actual teach it to carry out tasks is futuristic-cool.

You can build one of these trainable robotic arms because Adafruit sells the crucial analog feedback servos that make this technology possible.

3D printing allows anyone to make robotic parts. We will be printing an arm and gripper for this

project, but you could swap out the servos in an existing robotic arm also.

For more details on the servos, check out the About Analog Feedback Servos (http://adafru.it/cUA) write up.

Required Parts

You will need the following electronic hardware:

- Arduino Uno (http://adafru.it/50)
- 5 x Analog Feedback Micro Servo (http://adafru.it/1450) (metal or plastic gears up to you)
- 2 x Push buttons (http://adafru.it/367)
- 2 x 1k Resistors
- Breadboard (http://adafru.it/64)
- Jumper wires (http://adafru.it/759)

The arm parts and gripper are 3D printed so you will need a 3D printer or a 3D printing service. If you'd like to replace the servos in a retail arm you already have that should be no problem.

You will need something to attach the servos to the arm pieces. Screws will do the job, but if the plastic breaks you can use hot glue. A paperclip will be needed for the gripper.

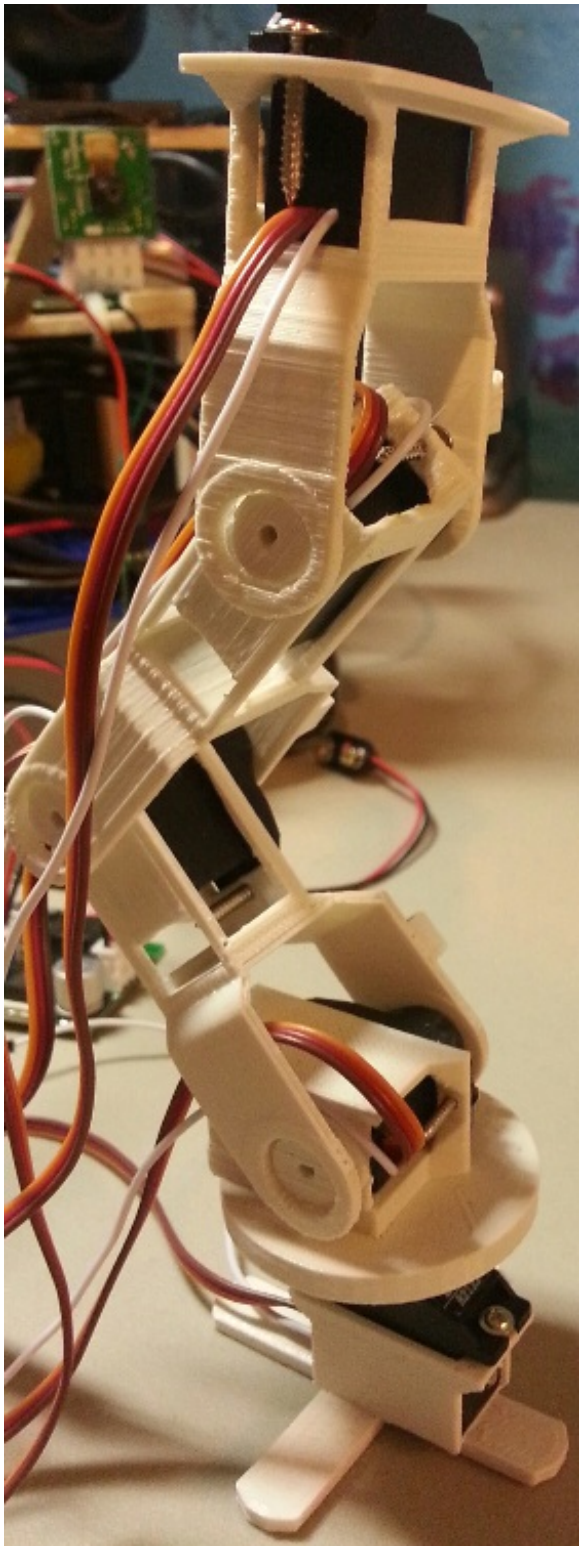
Tools:

- Small drill
- Blade
- Sandpaper
- Hot glue gun
- Needle nose pliers

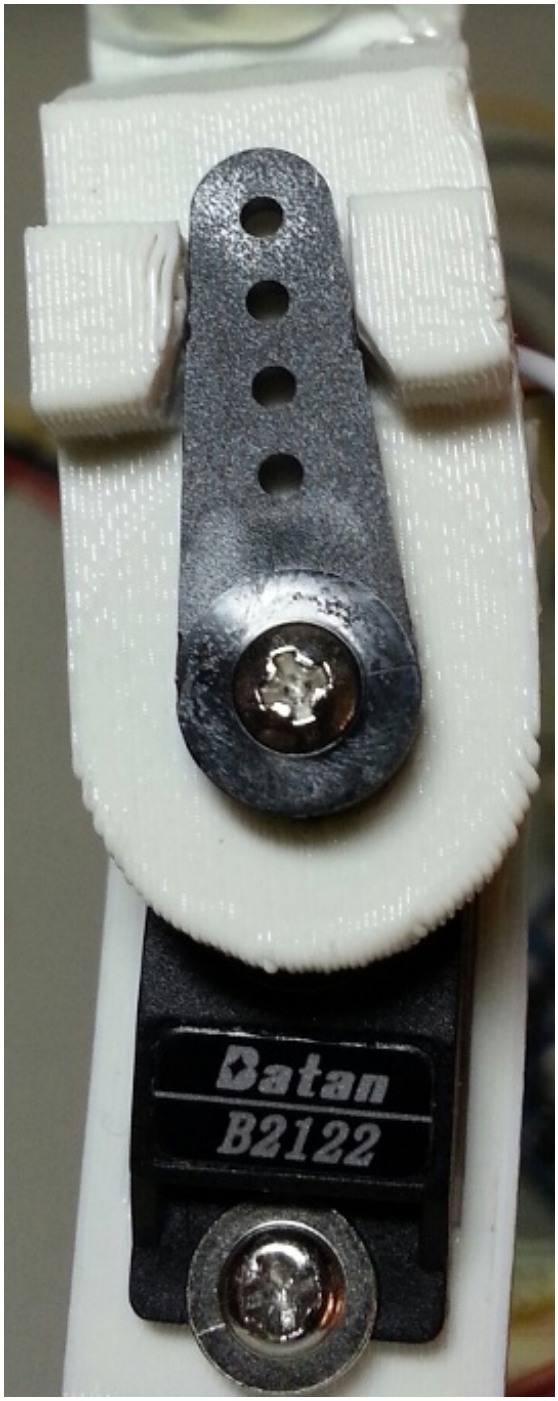
Arm Build

Thingiverse is a great place to find 3D printable robotic parts. This arm (http://adafru.it/cUB) works great for micro servos. Assembly of the parts is very simple but some modifications may be required to get everything working well.

Once the parts have been printed and the servos installed, your arm should look like this:



The square blocks that hold the servo horn still may need to be cut down to fit the horns that come with the analog feedback servos, as shown in this picture:



If the joints are slipping off the stationary ring opposite the servo horn, you can cut down an extra horn and screw it into the ring. This will keep the parts together nicely:

The legs of the base part are pretty short for the amount of weight and motion of the arm. I hot glued the base to a balsa wood platform to keep the arm stable when in use. Any form of additional base support will be great so the arm doesn't topple over.

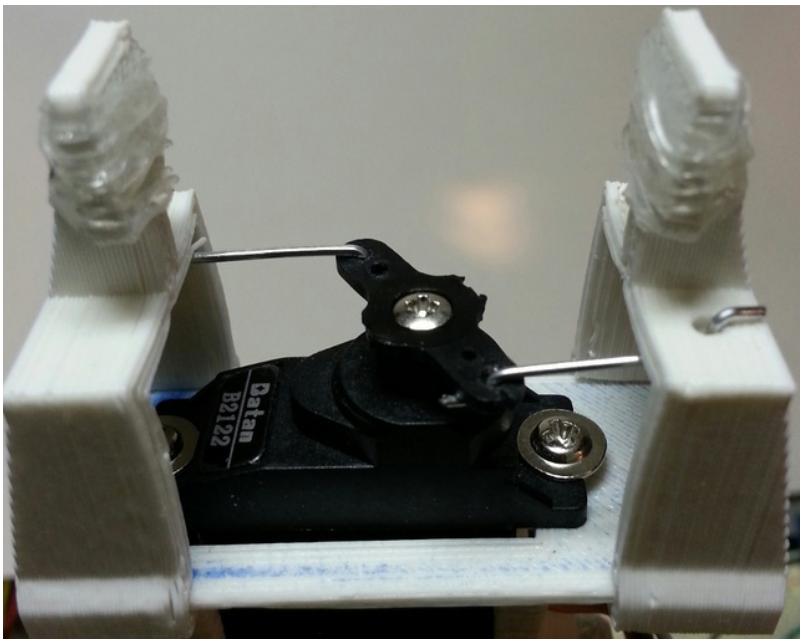
Gripper Build

This (http://adafru.it/cUC) is the gripper that fits on the end of the arm.

Drill a small hole in the gripper so a paperclip can fit through it. Once the hole is drilled and the gripper is slid onto the top of the arm, measure the length between the hole and the servo horn. This will be the length of the paperclip section needed to allow the gripper to close and open when attached to the servo horn.

Bend a section of the paperclip with needle nose pliers so it looks like this:

One end will hook into the gripper hole and the other into the servo horn. Both sides will look like this:



Note the clear rubber bands wrapped on the ends of the gripper. This is to increase the gripping ability when picking up objects. As the Thingiverse picture shows, small pieces of foam work well also.

It may take a few times to get the paperclips the right side and bent straight enough to allow a

smooth movement on the arm rails.

If there is resistance between the gripper and the arm rail when sliding, smooth out the edges of the arm rail with a blade and/or sandpaper.

Wiring

We will refer to the servos as One through Five, One being the bottom servo located in the arm base and Five being the top servo where the gripper is connected.

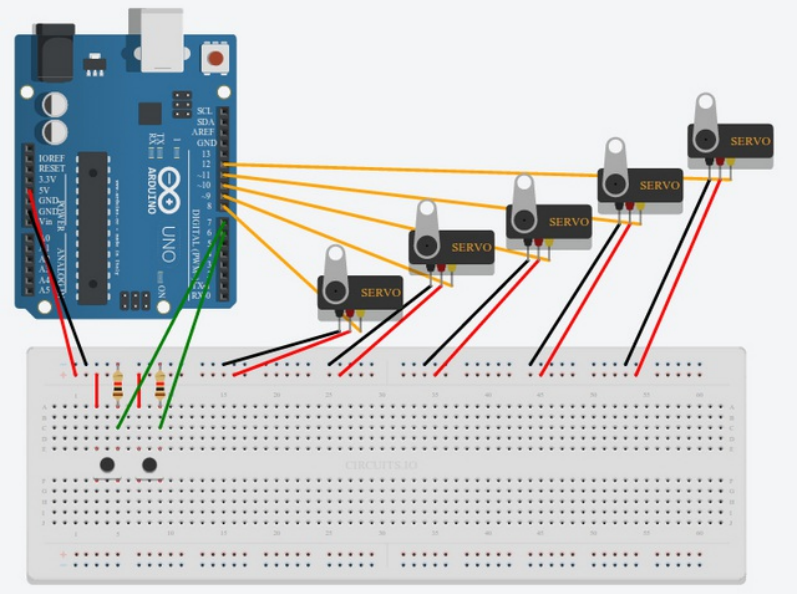
Each servo will be wired to 5V and Ground. Since there are 5 servos you may want to use a power source with more than 2A output. I used the Molex 5V power from an old PC power supply to power everything, but you can also try a 5V 4A adapter (http://adafru.it/1466) with a barrel-jack adapter (http://adafru.it/368) to power the servos separately than the Arduino

The PWM input wire (orange) and the feedback wire (white) for each servo will be attached to the Arduino pins as follows:

- Servo One - PWM -> Digital 8, Feedback -> Analog 1
- Servo Two - PWM -> Digital 9, Feedback -> Analog 2
- Servo Three - PWM -> Digital 10, Feedback -> Analog 3
- Servo Four - PWM -> Digital 11, Feedback -> Analog 4
- Servo Five - PWM -> Digital 12, Feedback -> Analog 5

We need one push button for the record function and another push button for the replay function. Wiring a push button circuit can be found at the Digital Inputs Arduino Lesson (http://adafru.it/aUw). The record button circuit will be attached to Arduino digital pin 7 and the replay button circuit will be attached to digital pin 6.

The picture below shows the wiring *with the exception of the analog feedback connections.* The circuits.io does not have analog feedback servo modules yet.



The Arduino analog pins to servo feedback wires are not shown in the image, don't forget them! See above for the wiring list

Sketch

The Arduino sketch is available here (http://adafru.it/cUD).

Complete description of the code is as follows.

We will be using the Servo and EEPROM libraries:

```
#include <Servo.h>
#include <EEPROM.h>
```

Declare the servos. One is the bottom (base) servo, counting up each joint until servo Five which is the gripper:

```
Servo one;
Servo two;
Servo three;
Servo four;
Servo five;
```

Calibrating the servos requires recording the minimum and maximum values for both the degrees and feedback. These are integers, so we will declare all these variables for each servo. We also want to create integer variables to hold the calculated replay values for each servo.

```
int minDegOne, minDegTwo, minDegThree, minDegFour, minDegFive;
int maxDegOne, maxDegTwo, maxDegThree, maxDegFour, maxDegFive;
int minFeedOne, minFeedTwo, minFeedThree, minFeedFour, minFeedFive;
int maxFeedOne, maxFeedTwo, maxFeedThree, maxFeedFour, maxFeedFive;
int posOne, posTwo, posThree, posFour, posFive;
int posOne1, posTwo1, posThree1, posFour1, posFive1;
```

Declare and integer variable to store the EEPROM address used to read/write servo values and a boolean variable to keep track of whether we have recorded arm motion or not.

```
int addr = 0;
boolean recorded = false;
```

The setup function will start serial output for debugging, attach each servo to their digital pins, set the LED on pin 13 to output and set the record and playback button pins to inputs.

The easiest method for calibrating the servos is to start them all out at 90 degrees then step through each servos motion starting from One and ending at Five to record the analog feedback values when the servos are positioned at their minimums and maximums. The micro

servos I used tended to have a good minimum of 30 and maximum of 150. The gripper servo (Five) worked will with minimum 20 and maximum 180.

When each servo is at its minimum, we read the analog value from the respective servo analog pin and store it in the respective variable. After this is complete, we place the servos back a 90 then detach them to both save power and to allow them to be moved by our own hands.

```
void setup()
{
 Serial.begin(9600);
 one.attach(8);
 two.attach(9);
 three.attach(10);
 four.attach(11);
 five.attach(12);
 pinMode(13, OUTPUT);  // LED
 pinMode(6, INPUT);    // Replay Button
 pinMode(7, INPUT);    // Train Button
 delay(100);
 // One center to left
 for (int i = 90; i > 29; i--)
 {
  one.write(i);
  delay(10);
 }
 minDegOne = 30;
 minFeedOne = analogRead(1);
 delay(500);
 // One left to right
 for (int i = 30; i < 151; i++)
 {
  one.write(i);
  delay(10);
 }
 maxDegOne = 150;
 maxFeedOne = analogRead(1);
 delay(500);
 // One right to center
 for (int i = 150; i > 89; i--)
 {
  one.write(i);
  delay(10);
 }
 delay(500);
 // Two up to forward
 for (int i = 90; i > 29; i--)
 {
  two.write(i);
  delay(10);
 }
 minDegTwo = 30;
 minFeedTwo = analogRead(2);
 delay(500);
 // Two forward to backward
 for (int i = 25; i < 151; i++)
 {
  two.write(i);
```

```
  delay(10);
 }
 maxDegTwo = 150;
 maxFeedTwo = analogRead(2);
 delay(500);
 // Two backward to up
 for (int i = 150; i > 89; i--)
 {
  two.write(i);
  delay(10);
 }
 delay(500);
 // Three up to forward
 for (int i = 90; i > 30; i--)
 {
  three.write(i);
  delay(10);
 }
 minDegThree = 30;
 minFeedThree = analogRead(3);
 delay(500);
 // Three forward to backward
 for (int i = 30; i < 151; i++)
 {
  three.write(i);
  delay(10);
 }
 maxDegThree = 150;
 maxFeedThree = analogRead(3);
 delay(500);
 // Three backward to up
 for (int i = 150; i > 89; i--)
 {
  three.write(i);
  delay(10);
 }
 delay(500);
 // Four up to forward
 for (int i = 90; i > 29; i--)
 {
  four.write(i);
  delay(10);
 }
 minDegFour = 30;
 minFeedFour = analogRead(4);
 delay(500);
 // Four forward to backward
 for (int i = 30; i < 151; i++)
 {
  four.write(i);
  delay(10);
 }
 maxDegFour = 150;
 maxFeedFour = analogRead(4);
 delay(500);
 // Four backward to up
 for (int i = 150; i > 89; i--)
 {
```

```
    four.write(i);
    delay(10);
  }
  delay(500);
  // Five up to forward
  for (int i = 90; i > 19; i--)
  {
    five.write(i);
    delay(10);
  }
  minDegFive = 20;
  minFeedFive = analogRead(5);
  delay(500);
  // Five forward to backward
  for (int i = 20; i < 181; i++)
  {
    five.write(i);
    delay(10);
  }
  maxDegFive = 180;
  maxFeedFive = analogRead(5);
  delay(500);
  // Five backward to up
  for (int i = 180; i > 89; i--)
  {
    five.write(i);
    delay(10);
  }
  delay(500);
  // Center all servos
  one.write(90);
  two.write(90);
  three.write(90);
  four.write(90);
  five.write(90);
  delay(1000);
  // Detach to save power and allow human manipulation
  one.detach();
  two.detach();
  three.detach();
  four.detach();
  five.detach();
}
```

The main code polls the state of the record and replay buttons. When the record button is pressed, the Arduino LED lights up to let you know its time to move the arm in the positions you want. The analog feedback values are read then converted into servo degrees and finally written to the EEPROM. When the replay button is pressed the stored servo positions are read from EEPROM and written to their respective servos. After recording or replaying, each servo is moved back to 90 then detached.

```
void loop()
{
  delay(100);
  if (digitalRead(7))
  {
```

```
    recorded = true;
    digitalWrite(13, HIGH);
    delay(1000);
    while (!digitalRead(7))
    {
      delay(50);
      int posOne = map(analogRead(1), minFeedOne, maxFeedOne, minDegOne, maxDegOne);
      EEPROM.write(addr, posOne);
      addr++;
      int posTwo = map(analogRead(2), minFeedTwo, maxFeedTwo, minDegTwo, maxDegTwo);
      EEPROM.write(addr, posTwo);
      addr++;
      int posThree = map(analogRead(3), minFeedThree, maxFeedThree, minDegThree, maxDegTh
      EEPROM.write(addr, posThree);
      addr++;
      int posFour = map(analogRead(4), minFeedFour, maxFeedFour, minDegFour, maxDegFour);
      EEPROM.write(addr, posFour);
      addr++;
      int posFive = map(analogRead(5), minFeedFive, maxFeedFive, minDegFive, maxDegFive);
      EEPROM.write(addr, posFive);
      addr++;
      if (addr == 512)
      {
        EEPROM.write(addr, 255);
        break;
      }
      delay(50);
    }
    EEPROM.write(addr, 255);
  }
  if (recorded || digitalRead(6))
  {
    digitalWrite(13, LOW);
    // Power up servos
    one.attach(8);
    two.attach(9);
    three.attach(10);
    four.attach(11);
    five.attach(12);
    delay(1000);
    // Center servos
    one.write(90);
    two.write(90);
    three.write(90);
    four.write(90);
    five.write(90);
    delay(1000);
    // Start playback
    addr = 0;
    while (1)
    {
      posOne = EEPROM.read(addr);
      posOne1 = EEPROM.read(addr+5);
      addr++;
      posTwo = EEPROM.read(addr);
      posTwo1 = EEPROM.read(addr+5);
      addr++;
      posThree = EEPROM.read(addr);
```

```
posThree1 = EEPROM.read(addr+5);
addr++;
posFour = EEPROM.read(addr);
posFour1 = EEPROM.read(addr+5);
addr++;
posFive = EEPROM.read(addr);
posFive1 = EEPROM.read(addr+5);
addr++;

// Check for the end of the recorded commands, if so then break out of the infinite loop
if ((posOne == 255) || (posOne1 == 255) || (posTwo == 255) || (posTwo1 == 255) || (posThre
{
 break;
}

// Step from one recording to the next for each servo
if ((posOne1 - posOne) > 0)
{
 for (int i = posOne; i < posOne1; i++)
 {
  one.write(i);
  delay(5);
 }
}
else if ((posOne1 - posOne) < 0)
{
 for (int i = posOne; i > posOne1; i--)
 {
  one.write(i);
  delay(5);
 }
}
if ((posTwo1 - posTwo) > 0)
{
 for (int i = posTwo; i < posTwo1; i++)
 {
  two.write(i);
  delay(5);
 }
}
else if ((posTwo1 - posTwo) < 0)
{
 for (int i = posTwo; i > posTwo1; i--)
 {
  two.write(i);
  delay(5);
 }
}
if ((posThree1 - posThree) > 0)
{
 for (int i = posThree; i < posThree1; i++)
 {
  three.write(i);
  delay(5);
 }
}
else if ((posThree1 - posThree) < 0)
{
```

```
      for (int i = posThree; i > posThree1; i--)
      {
        three.write(i);
        delay(5);
      }
    }
    if ((posFour1 - posFour) > 0)
    {
      for (int i = posFour; i < posFour1; i++)
      {
        four.write(i);
        delay(5);
      }
    }
    else if ((posFour1 - posFour) < 0)
    {
      for (int i = posFour; i > posFour1; i--)
      {
        four.write(i);
        delay(5);
      }
    }
    if ((posFive1 - posFive) > 0)
    {
      for (int i = posFive; i < posFive1; i++)
      {
        five.write(i);
        delay(5);
      }
    }
    else if ((posFive1 - posFive) < 0)
    {
      for (int i = posFive; i > posFive1; i--)
      {
        five.write(i);
        delay(5);
      }
    }
  }
  recorded = false;
  addr = 0;
  delay(1000);
  // Center all servos
  one.write(90);
  two.write(90);
  three.write(90);
  four.write(90);
  five.write(90);
  delay(500);
  // Detach them to save power and allow human manipulation
  one.detach();
  two.detach();
  three.detach();
  four.detach();
  five.detach();
 }
}
```

During the replay after the values have been read from EEPROM, there are a series of if statements that control the writing of the servo values. The reason for these conditionals is to create a smooth playback from one recorded position to the next. A quick calculation of the difference between the servo position being read now and the next servo position tells us how to step between those values as we write them to the servos. Instead of writing 120 then 130 to the servo, we want to write every value between 120 and 130 with a short delay in order to create the smooth motion.

Note the if statement for the replay button: **if (recorded || digitalRead(6))** Instead of just executing a replay when the replay button attached to digital pin 6 is pressed, it will execute a replay if the boolean variable recorded is true. This is why the servo motion is replayed right after you end recording because the recording code sets the variable recorded to true - before the code even gets to this if statement. The only reason for this is so the first replay happens without needing to press the replay button right after you record a motion. If you do not want the arm to automatically replay its motion after recording, remove the recorded or to make the line: **if (digitalRead(6))** instead.

Full code listing

```
#include <Servo.h>
#include <EEPROM.h>

// Servo declarations
Servo one;
Servo two;
Servo three;
Servo four;
Servo five;
// Calibration values
int minDegOne, minDegTwo, minDegThree, minDegFour, minDegFive;
int maxDegOne, maxDegTwo, maxDegThree, maxDegFour, maxDegFive;
int minFeedOne, minFeedTwo, minFeedThree, minFeedFour, minFeedFive;
int maxFeedOne, maxFeedTwo, maxFeedThree, maxFeedFour, maxFeedFive;
int posOne, posTwo, posThree, posFour, posFive;
int posOne1, posTwo1, posThree1, posFour1, posFive1;
int addr = 0;
boolean recorded = false;

void setup()
{
  Serial.begin(9600);
  one.attach(8);
  two.attach(9);
  three.attach(10);
  four.attach(11);
  five.attach(12);
  pinMode(13, OUTPUT);  // LED
  pinMode(6, INPUT);    // Replay Button
  pinMode(7, INPUT);    // Train Button
  delay(100);
  // One center to left
  for (int i = 90; i > 29; i--)
```

```
{
  one.write(i);
  delay(10);
}
minDegOne = 30;
minFeedOne = analogRead(1);
delay(500);
// One left to right
for (int i = 30; i < 151; i++)
{
  one.write(i);
  delay(10);
}
maxDegOne = 150;
maxFeedOne = analogRead(1);
delay(500);
// One right to center
for (int i = 150; i > 89; i--)
{
  one.write(i);
  delay(10);
}
delay(500);
// Two up to forward
for (int i = 90; i > 29; i--)
{
  two.write(i);
  delay(10);
}
minDegTwo = 30;
minFeedTwo = analogRead(2);
delay(500);
// Two forward to backward
for (int i = 25; i < 151; i++)
{
  two.write(i);
  delay(10);
}
maxDegTwo = 150;
maxFeedTwo = analogRead(2);
delay(500);
// Two backward to up
for (int i = 150; i > 89; i--)
{
  two.write(i);
  delay(10);
}
delay(500);
// Three up to forward
for (int i = 90; i > 30; i--)
{
  three.write(i);
  delay(10);
}
minDegThree = 30;
minFeedThree = analogRead(3);
delay(500);
// Three forward to backward
```

```
for (int i = 30; i < 151; i++)
{
  three.write(i);
  delay(10);
}
maxDegThree = 150;
maxFeedThree = analogRead(3);
delay(500);
// Three backward to up
for (int i = 150; i > 89; i--)
{
  three.write(i);
  delay(10);
}
delay(500);
// Four up to forward
for (int i = 90; i > 29; i--)
{
  four.write(i);
  delay(10);
}
minDegFour = 30;
minFeedFour = analogRead(4);
delay(500);
// Four forward to backward
for (int i = 30; i < 151; i++)
{
  four.write(i);
  delay(10);
}
maxDegFour = 150;
maxFeedFour = analogRead(4);
delay(500);
// Four backward to up
for (int i = 150; i > 89; i--)
{
  four.write(i);
  delay(10);
}
delay(500);
// Five up to forward
for (int i = 90; i > 19; i--)
{
  five.write(i);
  delay(10);
}
minDegFive = 20;
minFeedFive = analogRead(5);
delay(500);
// Five forward to backward
for (int i = 20; i < 181; i++)
{
  five.write(i);
  delay(10);
}
maxDegFive = 180;
maxFeedFive = analogRead(5);
delay(500);
```

```
  // Five backward to up
  for (int i = 180; i > 89; i--)
  {
    five.write(i);
    delay(10);
  }
  delay(500);
  // Center all servos
  one.write(90);
  two.write(90);
  three.write(90);
  four.write(90);
  five.write(90);
  delay(1000);
  // Detach to save power and allow human manipulation
  one.detach();
  two.detach();
  three.detach();
  four.detach();
  five.detach();
  /*
  // Display minimums and maximums for analog feedback
  // Uncomment for debugging
  Serial.print("One Min: ");
  Serial.println(minFeedOne);
  Serial.print("One Max: ");
  Serial.println(maxFeedOne);
  Serial.print("Two Min: ");
  Serial.println(minFeedTwo);
  Serial.print("Two Max: ");
  Serial.println(maxFeedTwo);
  Serial.print("Three Min: ");
  Serial.println(minFeedThree);
  Serial.print("Three Max: ");
  Serial.println(maxFeedThree);
  Serial.print("Four Min: ");
  Serial.println(minFeedFour);
  Serial.print("Four Max: ");
  Serial.println(maxFeedFour);
  Serial.print("Five Min: ");
  Serial.println(minFeedFive);
  Serial.print("Five Max: ");
  Serial.println(maxFeedFive);
  Serial.println();
  */
}

void loop()
{
  delay(100);
  if (digitalRead(7))
  {
    recorded = true;
    digitalWrite(13, HIGH);
    delay(1000);
    while (!digitalRead(7))
    {
      delay(50);
```

```
    int posOne = map(analogRead(1), minFeedOne, maxFeedOne, minDegOne, maxDegOne);
    EEPROM.write(addr, posOne);
    addr++;
    int posTwo = map(analogRead(2), minFeedTwo, maxFeedTwo, minDegTwo, maxDegTwo);
    EEPROM.write(addr, posTwo);
    addr++;
    int posThree = map(analogRead(3), minFeedThree, maxFeedThree, minDegThree, maxDegTh
    EEPROM.write(addr, posThree);
    addr++;
    int posFour = map(analogRead(4), minFeedFour, maxFeedFour, minDegFour, maxDegFour);
    EEPROM.write(addr, posFour);
    addr++;
    int posFive = map(analogRead(5), minFeedFive, maxFeedFive, minDegFive, maxDegFive);
    EEPROM.write(addr, posFive);
    addr++;
    if (addr == 512)
    {
      EEPROM.write(addr, 255);
      break;
    }
    delay(50);
  }
  EEPROM.write(addr, 255);
}
if (recorded || digitalRead(6))
{
  digitalWrite(13, LOW);
  // Power up servos
  one.attach(8);
  two.attach(9);
  three.attach(10);
  four.attach(11);
  five.attach(12);
  delay(1000);
  // Center servos
  one.write(90);
  two.write(90);
  three.write(90);
  four.write(90);
  five.write(90);
  delay(1000);
  // Start playback
  addr = 0;
  while (1)
  {
    posOne = EEPROM.read(addr);
    posOne1 = EEPROM.read(addr+5);
    addr++;
    posTwo = EEPROM.read(addr);
    posTwo1 = EEPROM.read(addr+5);
    addr++;
    posThree = EEPROM.read(addr);
    posThree1 = EEPROM.read(addr+5);
    addr++;
    posFour = EEPROM.read(addr);
    posFour1 = EEPROM.read(addr+5);
    addr++;
    posFive = EEPROM.read(addr);
```

```
posFive1 = EEPROM.read(addr+5);
addr++;

/*
// Display positions being written to the servos
// Uncomment for debugging
Serial.print("One: ");
Serial.print(posOne);
Serial.print("\t\t\t");
Serial.println(posOne1);
Serial.print("Two: ");
Serial.print(posTwo);
Serial.print("\t\t");
Serial.println(posTwo1);
Serial.print("Three: ");
Serial.print(posThree);
Serial.print("\t\t");
Serial.println(posThree1);
Serial.print("Four: ");
Serial.print(posFour);
Serial.print("\t\t");
Serial.println(posFour1);
Serial.print("Five: ");
Serial.print(posFive);
Serial.print("\t\t");
Serial.println(posFive1);
Serial.println();
*/

// Check for the end of the recorded commands, if so then break out of the infinite loop
if ((posOne == 255) || (posOne1 == 255) || (posTwo == 255) || (posTwo1 == 255) || (posThre
{
  break;
}

// Step from one recording to the next for each servo
if ((posOne1 - posOne) > 0)
{
  for (int i = posOne; i < posOne1; i++)
  {
    one.write(i);
    delay(5);
  }
}
else if ((posOne1 - posOne) < 0)
{
  for (int i = posOne; i > posOne1; i--)
  {
    one.write(i);
    delay(5);
  }
}
if ((posTwo1 - posTwo) > 0)
{
  for (int i = posTwo; i < posTwo1; i++)
  {
    two.write(i);
    delay(5);
```

```
  }
 }
 else if ((posTwo1 - posTwo) < 0)
 {
  for (int i = posTwo; i > posTwo1; i--)
  {
   two.write(i);
   delay(5);
  }
 }
 if ((posThree1 - posThree) > 0)
 {
  for (int i = posThree; i < posThree1; i++)
  {
   three.write(i);
   delay(5);
  }
 }
 else if ((posThree1 - posThree) < 0)
 {
  for (int i = posThree; i > posThree1; i--)
  {
   three.write(i);
   delay(5);
  }
 }
 if ((posFour1 - posFour) > 0)
 {
  for (int i = posFour; i < posFour1; i++)
  {
   four.write(i);
   delay(5);
  }
 }
 else if ((posFour1 - posFour) < 0)
 {
  for (int i = posFour; i > posFour1; i--)
  {
   four.write(i);
   delay(5);
  }
 }
 if ((posFive1 - posFive) > 0)
 {
  for (int i = posFive; i < posFive1; i++)
  {
   five.write(i);
   delay(5);
  }
 }
 else if ((posFive1 - posFive) < 0)
 {
  for (int i = posFive; i > posFive1; i--)
  {
   five.write(i);
   delay(5);
  }
 }
```

```
    }
    recorded = false;
    addr = 0;
    delay(1000);
    // Center all servos
    one.write(90);
    two.write(90);
    three.write(90);
    four.write(90);
    five.write(90);
    delay(500);
    // Detach them to save power and allow human manipulation
    one.detach();
    two.detach();
    three.detach();
    four.detach();
    five.detach();
  }
}
```

Final Steps

The final step is to upload the Arduino sketch to your board. As soon as the Arduino resets, the arm should start its calibration routine by moving each servo through its range of motion. When this completes the arm will be positioned straight up and the servos will deatch, allowing you to manipulate the arm without power being applied.

At this point you can press the record button. The onboard LED will light up which means its recording. Move the arm around any way you like then press the record button again to stop recording. The arm will straighten up again then immediately replay the motion it learned. When the replay is complete the arm will position itself straight up again.

You can press the replay button to replay the last learned routine, or teach the arm another motion sequence with the record button.

Did everything work correctly? Congrats on creating your own Baxter-like trainable robotic arm!

Troubleshooting

Some common issues and solutions for your trainable robotic arm:

- **Servos moving erratically:** Check all your ground connections. If your powering the servos with a different power source than the Arduino, they need to share the same ground or the PWM signals will corrupt between the Arduino and servos.
- **The arm performs the movements but in reverse order:** Reset the Arduino if your using one power source. If you have 2 power sources, power everything off then back on.
- **Routine starts correctly but repeats the beginning motions:** You recorded a routine that filled up the EEPROM so the commands of what would have been the end of the routine were lost in the bitbucket. Record shorter motion routines.
- **LED doesn't turn on when record button is pressed:** Double check the record button circuit, if it's good then give it a moment after calibration before pressing the record button - there is a delay and you may be pressing the button during that delay.