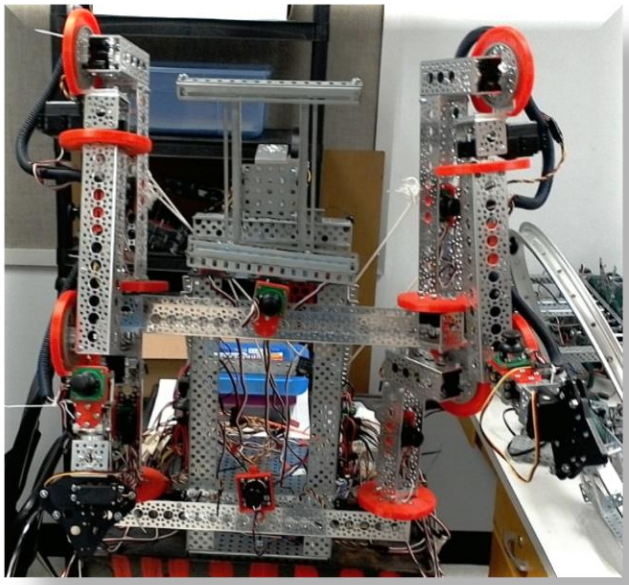
BAXTER PORTFOLIO

Robotics Capstone Project

Nicholas Fox

Metric 1

Drawings, Photos, Scans



1 Baxter, the platform used for this project, at the beginning of the year

2 The actual Baxter, and industrial robot

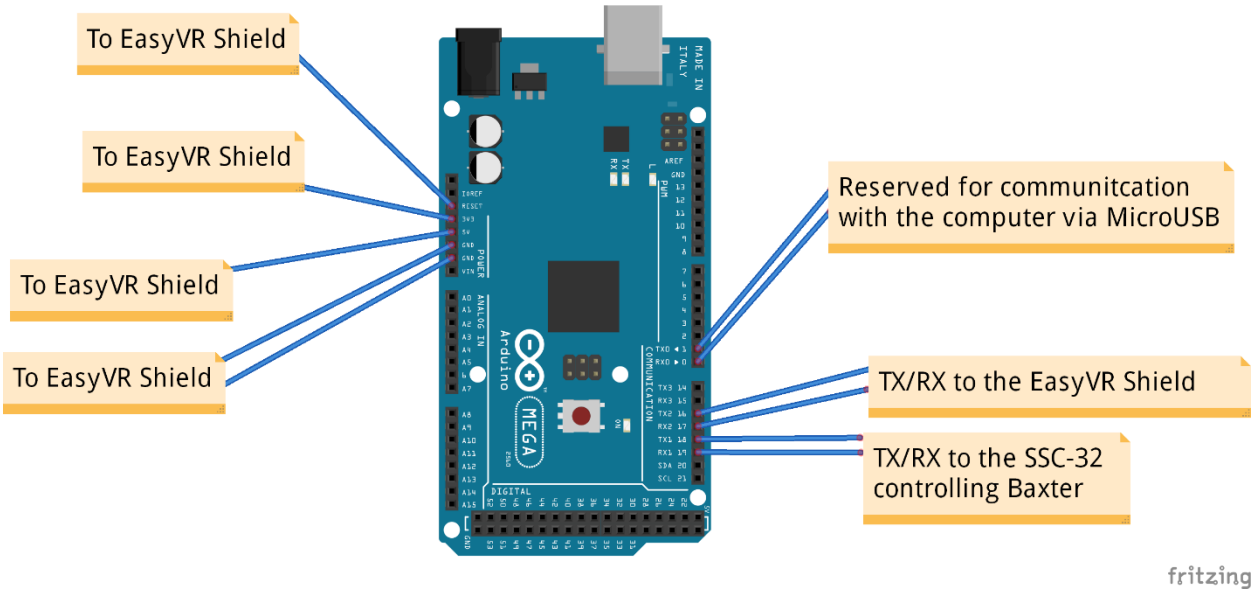
Description of Project

The purpose of this project is to make the robot I've been working on, titled Baxter, to better interact with humans. In order to achieve this goal, I will be adding the ability to shake hands with a person without crushing the hand and the ability to carry on a conversation with the person, albeit a simple conversation.

The "why" of the Project

I am attempting to bring Baxter to a greater degree of human interaction because human-robot interaction is a developing field. Additionally, I am trying to continue bringing our half-scale model of Baxter closer to the real Baxter (see the second photo), a fully capable industrial robot. Furthermore, I am interested in working with cutting edge technology such as speech-to-text in my senior year that way I can have an understanding as I move into college at, preferably, a technology institution. Finally, there is something very interesting about human-computer interaction and as I see Baxter as "my" project, I want to bring it to "life".

Sensor and Actuator Mapping



3 Initial diagram of the sensors and actuators

General Parts List

- Baxter, a robot with two 8 degree-of-freedom arms controlled by serial servos
- Arduino Mega 2560
- Possibly an Arduino Due and/or .NETduino
- 2 proMOTION CDS5516 serial servos
- VeeaR EasyVR Shield for Arduino

Metric 2

Primary Goals

1. Assemble Baxter with a hand that can judge its applied force when doing a task
2. Baxter shakes the hand of a person and gently picks up a tennis ball
3. Baxter converses with a user, using a speaker to talk and replies to user input in a console window
4. Baxter accepts user commands with audio feedback, using a speaker to talk and takes commands via a voice synthesizer

Secondary Goals

1. Baxter can shake someone's hand when called for in a conversation
2. Baxter can hold an egg without breaking it

Roadblocks to Goals

Primary Goals

1. A roadblock to building a hand that can judge its applied force is the method of judging the applied force. There are two options to judge the force: smart servos and pressure sensors. The issue with the smart servo is we need to integrate it into the Arduino board. The issue with the pressure sensors is we would need to buy them and then mount them in a safe place on the hand.
2. A roadblock to Baxter shaking the hand of a person is the pressure that Baxter would apply to the hand. If too much force is applied, the person's hand could be injured. If too little force is applied, then the hand shake won't happen. Likewise, with the picking up of a tennis ball, if too much force is applied it will not be gentle; if too little force is applied it will not pick up the ball.
3. A roadblock to Baxter's conversation with a user is creating or implementing a good text-to-speech program with Arduino. While there are text-to-speech boards, these would have to be integrated with the Arduino to create a realistic voice and response time. Getting the user input from a console window requires that a computer, likely the tablet head, to be connected to the Arduino throughout the duration of the program. As the tablet is mounted on a rotating mount, this would be difficult to type on.
4. A roadblock to Baxter's replying to spoken commands or statements is the turning of voice input into serial input for the Arduino board. If a computer were used, it would have to run simultaneously to the Arduino program and would have to have the speech-to-text program send the data to the Arduino.

Secondary Goals

1. There are a couple roadblocks to the goal of Baxter shaking someone's hand in the right part of a conversation. The aforementioned roadblocks of Baxter shaking someone's hand and of holding a conversation aside, these are knowing where the person's hand is and when the right time to go for a handshake is.
2. A roadblock to this goal is finding out the right pressure limits or force limits for holding an egg without breaking it. If Baxter is used, it would need to be protected from splatter upon the breaking of an egg.

Steps to Overcome Roadblocks

Primary Goals

1. To overcome the roadblock of how to judge the force applied, we could use smart servos controlled from a board or shield that is designed to integrate with the Arduino. Additionally, a library could be made or used which would read the servo's values and take action based on them.
2. To overcome the roadblock of applying too much or too little force when either shaking a hand or holding a tennis ball, tests would be a necessity. These would start with the tennis ball and testing through trial and error how much force is required to hold the ball. Then, starting with that data, testing on a human hand would begin. First, I would start with the force used to hold the tennis ball and see how that felt. Any change in the amount of force applied would be done using very small increments to prevent injury and a kill switch would be placed within easy reach of the test subject should the grip be too tight.
3. To overcome the roadblock of the text-to-speech board, I will work with the Arduino Mega and trying to integrate a text-to-speech board with it. If that doesn't work out, then I will try to integrate either an Arduino Duo or a Netduino into Baxter to handle the speech.
4. To overcome the roadblock of speech-to-text, I will try to integrate the tablet's built in programs, and perhaps Cortana, with the board that I am using via serial input/output, making use of the USB port on the tablet. If that doesn't work out, I'll try integrating a speech-to-text board with the board I'm using.

Secondary Goals

1. To overcome the roadblock of knowing where the other person's hand is I will have Baxter extend its hand to a predefined position and let the other person meet its hand. When the person's hand touches the button at the back of the hand, Baxter will then perform the shaking motion. If there is time this year, I would like to use either a 3D scanner or a Kinect to judge the precise location of the other person's hand and move to meet it. To overcome the roadblock of determining when in the conversation the hand should be extended, I will set up a very basic conversation sequence for introductions, where Baxter would extend its hand during the introductions.
2. To overcome the roadblock of finding the force or pressure limits on an egg I would use trial and error. Most of this would be done with a force sensor pushing down on the egg, pressing it into a table top, and recording how much pressure does what, through breaking the egg. This would be repeated multiple times until a basic range for holding was determined. Additionally, Baxter would get some kind of cover, either a 3D printed "shell" or a plastic sheet draped over it to protect from the splatter of the egg if it were dropped or crushed.

Timeline for Project

Addition of smart servos to Baxter's hands..... October 22th, 2015

Integration of smart servos with the Arduino control board..... October 30th, 2015

Programming of the smart servos to hold a tennis ball .....November 13th, 2015

Programming of the smart servos to shake a person's hand.....November 19th, 2015

Basic integration of a text-to-speech board with control board ..... December 4th, 2015

Programming of basic conversation between Baxter and console input ..... January 8th, 2016

Decision of an appropriate speech-to-text input .....January 22nd, 2016

Basic speech-to-text demonstration ..... January 28th, 2016

Demonstration of all primary goals ..... First week of April, 2016

Programming of default hand position .....Second week of April, 2016

Programming of basic introduction sequence ..... First week of May, 2016

Demonstration of first secondary goal ..... Second week of May, 2016

Testing of egg force limits completion .....Third week of May, 2016

Programming of holding an egg and tests.....Last week of May, 2016

Demonstration of second secondary goal.....First week of June, 2016

Metric 3

Detailed Parts List

- Dell Venue 11 Tablet
- Robot with at least one arm ending in a claw and a head
- 4x Flexifoce Pressure Sensor
- Arduino Due
- Wires (Male-Male, Male-Female)
- 4x Flexiforce Pressure Sensor interpreters
- Speech Synthesis Shield for Arduino
- SFE VoiceBox Arduino Shield
- EasyVR Speech Recognition Module (microphone included)
- USB-to-MicroUSB cable
- Speaker
- Ear buds

Cost of Parts, Itemized

Item	Price	Quantity	Sub-Total
Dell Venue 11 Tablet	$390.00	1	$390.00
4x Flexifoce Pressure Sensor	$6.90	4	$27.60
Arduino Due	$56.00	1	$56.00
4x Flexiforce Pressure Sensor interpreters	$11.00	4	$44.00
Speech Synthesis Shield for Arduino	$42.05	1	$42.05
SFE VoiceBox Arduino Shield	$39.95	1	$39.95
EasyVR Speech Recognition Module	$38.25	1	$38.25
Total	-	-	$687.85

Money Source Planning

In order to purchase the Flexiforce sensors and their interpreters, the instructor employed class funds. The remainder of the parts were available in class. The tablet was provided by the school and the robot was a previous project that was available to be used as a base for future projects.

Budget Timeline

Arduino Due, Venue 11 Tablet, USB cable, wires ..... November 2nd, 2015

Flexiforce sensors and interpreters ..... December 4th, 2015

Text-to-speech board and speaker/ear buds ..... December 10th, 2015

Speech-to-text Board and microphone ..... January 8th, 2016

Expense Tracking

Since none of the parts were ordered by me or on a grant of my proposal, I do not have the specifics of the orders. An order did go in for the Flexiforce sensors and signal interpreters in the week of November 9th and the sensors and interpreters were available in class on November 30th.

Metric 4 – Daily Journal

January 6th, 2016

Goals

- Take off the burnt out servo from Baxter and give to Mr. Ness to be replaced
- Tighten all screws on the arms, head, and torso

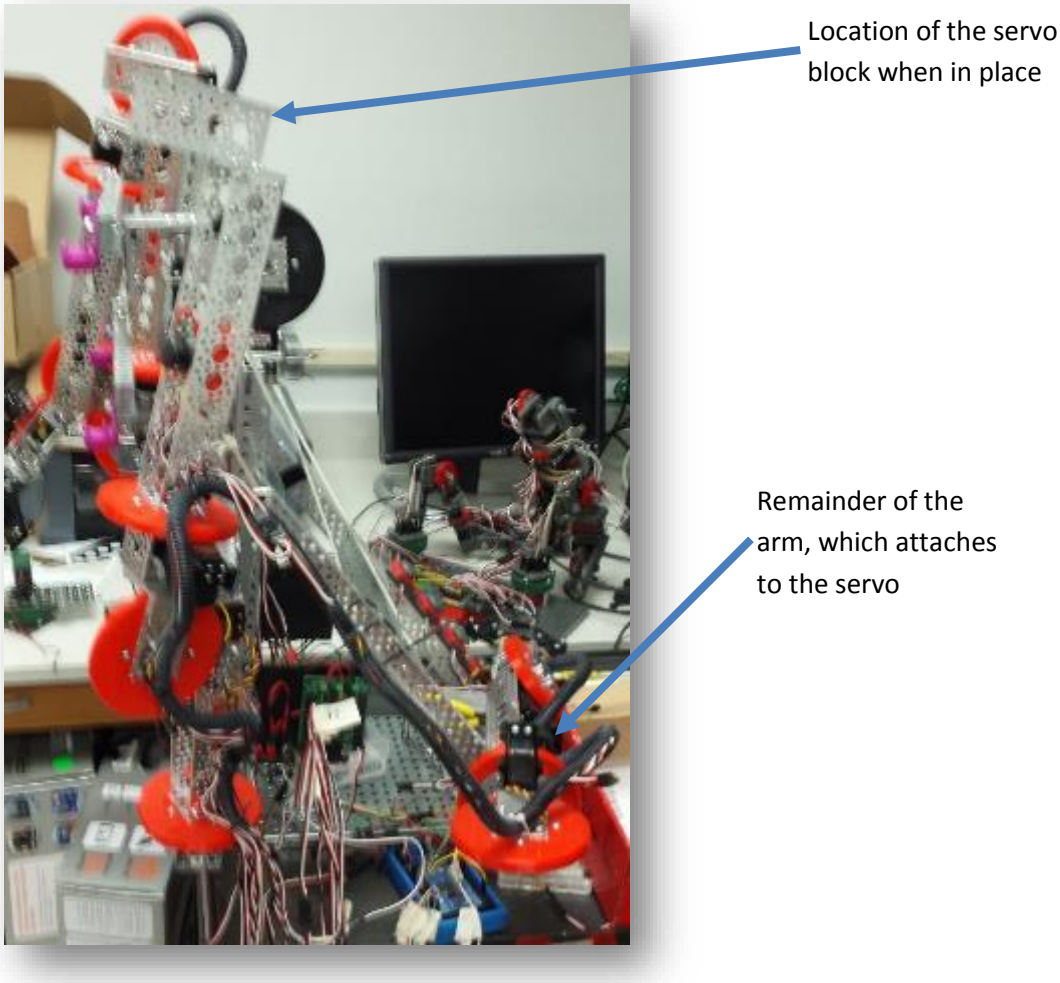
Accomplished Today

Today, I removed a burnt out servo from Baxter's left arm. The servo was controlling the elbow joint and had become locked after burning out, making any motion impossible. After removing the burnt servo, I double checked all the other servos by using an Arduino Mega linked to a SSC-32, sending a command for each servo to move a little bit and then seeing if it did, indeed, move. All the other servos worked. Next, I tightened all the screws and double checked all the joins – none of which needed repair.

Where are you in the project?

At the end of the day, the next action is to replace the servo block which was removed (with the new servo) and to come up with a plan for mounting the force sensors.

Picture or Diagram



February 4th, 2016

Goals

- Connect touch sensor using pre-packaged parts
- Solder a sensor-system together
- Begin testing for values

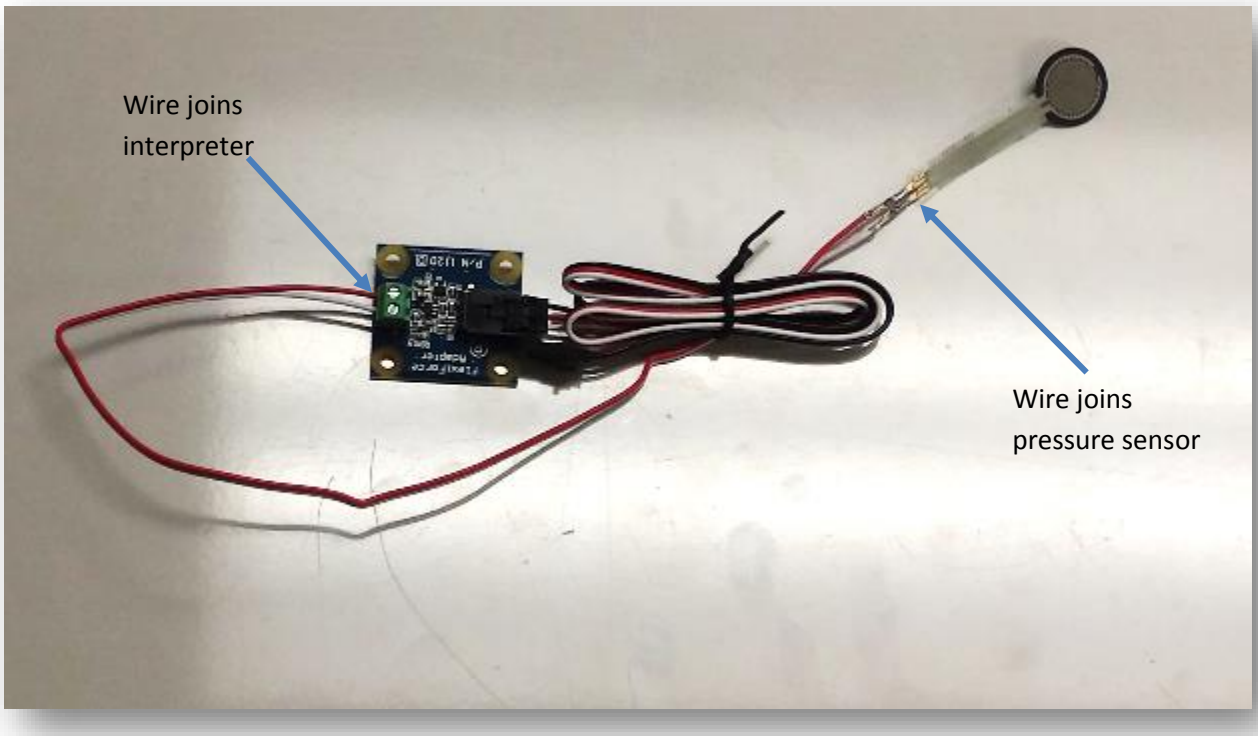
Accomplished Today

Today I fully assembled a touch sensor (see diagram), using the user manual to determine which lead is the ground and which is the signal. The screws to hold the wire from the sensor to the interpreter work well, though a very small flathead screwdriver is needed. The connection between the touch sensor and the wire is similar to that which we use for our batteries, but needs solder to hold it in place. The lead (male pin) on the sensor slides into a female pin on the wire, with the solder holding it in place. The wire from the interpreter to the Arduino is a standard 3-wire cable, with a unique join to the interpreter. After constructing the sensor assembly, I began testing for maximum and minimum values for the touch sensor by applying pressure to it and seeing what values the Arduino received.

Where are you in the project?

The next step in this project is to develop a system for attaching the touch sensor to Baxter and to finish testing the touch sensor in action.

Picture or Diagram



February 8th, 2016

Goals

- Decide on a design for the touch sensor mount
- Final test run of touch sensor in action

Accomplished Today

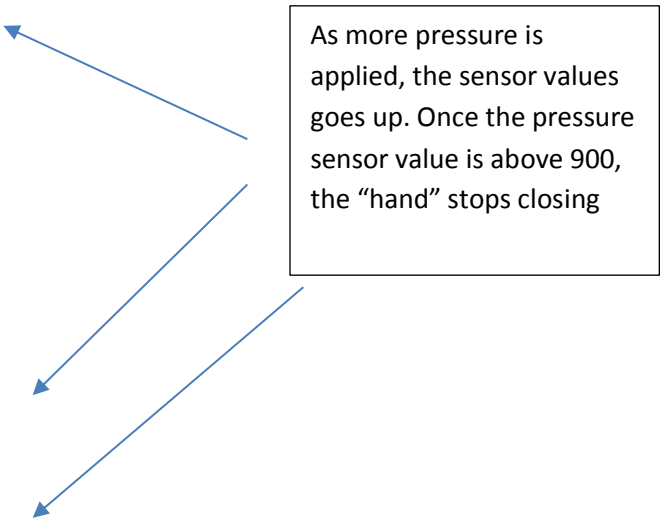
After considering the shape of the interpreter, I decided that it would be best to create a mount for it instead of mounting it directly to the claw of Baxter. The mount would be a long, flat plate underneath the pressure sensor, attached by a piece of double sided foam tape, and connect to the claw through the holes in the claw and screw holes in the mount. Additionally, I determined the maximum value for demonstrations of the hand, such as shaking someone's hand. For these purposes, I will use a value of 900 counts or greater, even though that is 124 less than the max value of the sensor. This buffer is to account for the increase in pressure not exactly hitting 900 counts, and to have an increased reaction to the pressure, instead of waiting until the object could be crushed. I intend to have this limit be set by the user, or through the console, in future iterations.

Where are you in the project?

The next step of the project is to print the touch sensor mount, then attach the sensor to the mount and the mount to the claw.

Picture or Diagram

```
Beginning to close hand
#0 P1260 #1 P1230 #2 P1480 #3 P2010 #4 P1350 #5 P1030 #6 P1870 #7 P1390 T12500
Hand closing
#7 P1365 T2500PressureVal: 0
Hand closing
#7 P1340 T2500PressureVal: 4
Hand closing
#7 P1315 T2500PressureVal: 3
Hand closing
#7 P1290 T2500PressureVal: 1
Hand closing
#7 P1265 T2500PressureVal: 4
Hand closing
#7 P1240 T2500PressureVal: 3
Hand closing
#7 P1215 T2500PressureVal: 3
Hand closing
#7 P1190 T2500PressureVal: 11
Hand closing
#7 P1165 T2500PressureVal: 752
Hand closing
#7 P1140 T2500PressureVal: 695
Hand stopped
#7 P1140 #T2500PressureVal: 1023
```



February 10th, 2016

Goals

- Print touch sensor mount
- Attach sensor to mount
- Attach mount to claw
- Run semi-live test

Accomplished Today

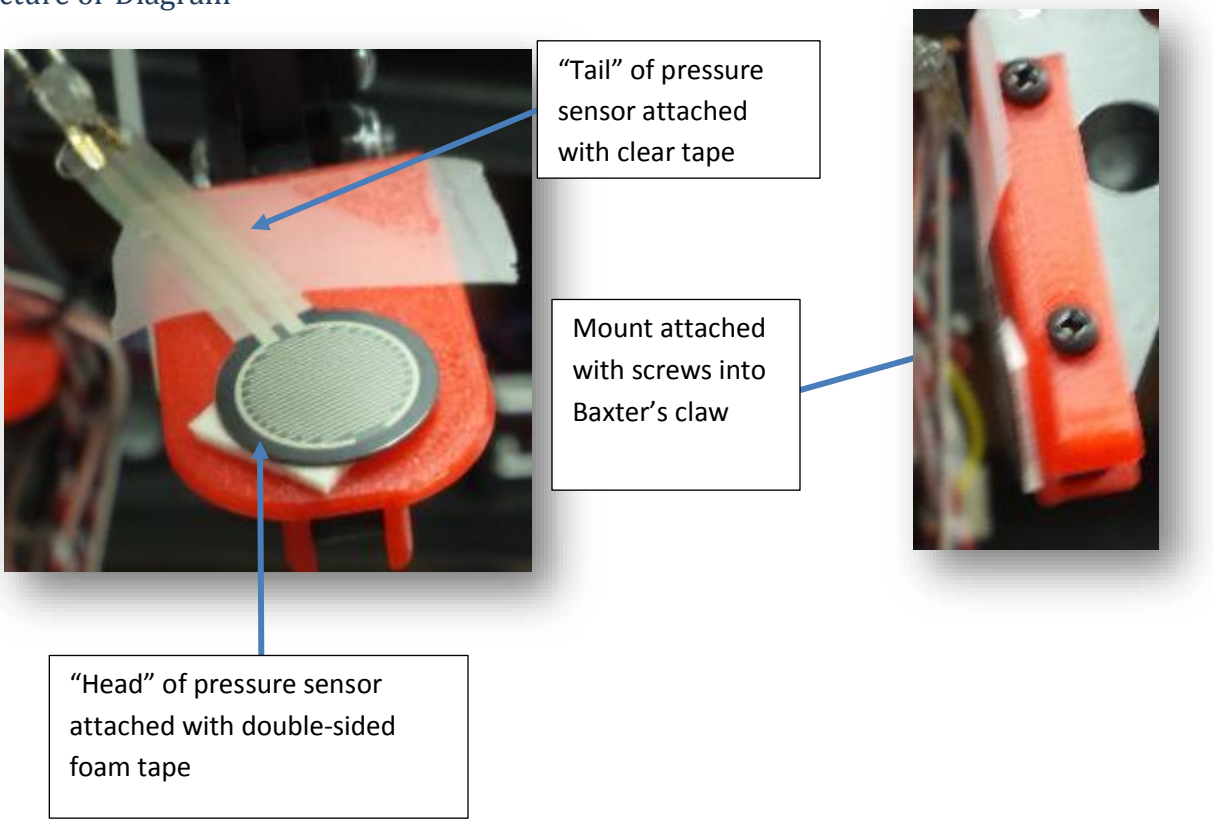
After the touch sensor mount was designed and printed, I attached the sensor to the mount using clear tape and the double sided foam tape. The foam tape connected the “head” of the sensor – the actual sensor pad – to the mount, whereas the clear tape was used to attach the “tail” of the sensor – the part of the sensor leading from the sensor pad to the leads – to the mount. Next, I attached the mount to the claw using the screws and nuts through pre-set holes in the mount and claw. Once the pressure sensor had been mounted to the claw, I preceded to run several additional tests of the stop command being sent when the threshold pressure was reached (900+ counts). Upon successful results from these tests, I

proceeded to attach the interpreter of the sensor to the upper wrist section of Baxter and clean up the wiring of the sensor.

Where are you in the project?

The next step is to calibrate the servos on one of Baxter's arms and create the code for the closing action.

Picture or Diagram



February 12th, 2016

Goals

- Calibrate the servos in one of Baxter's arms
- Input the servo home position values into the pressure sensor code

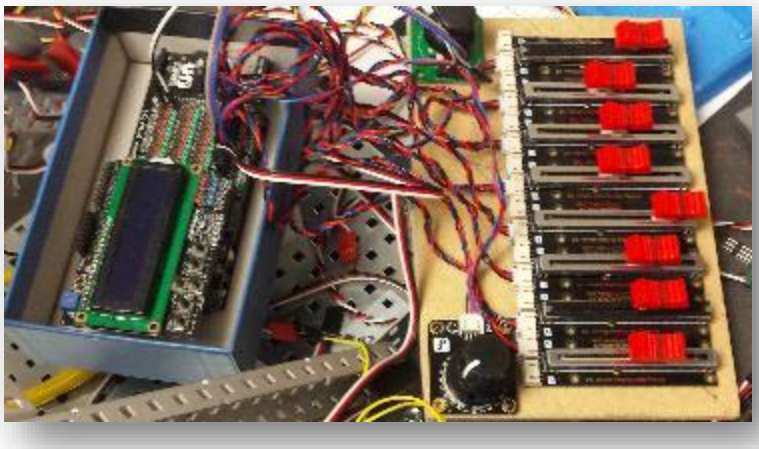
Accomplished Today

Today I was able to find the home position, maximum position, and minimum position for each servo in the left arm (see the table below) and recorded them. The maximum and minimum positions are a few counts shy of actually being a hazard, giving a safety distance in the code. For the pressure sensor code, I added the home positions and the maximum and minimum positions of all the servos into arrays to the code to prevent it from burning out the servo.

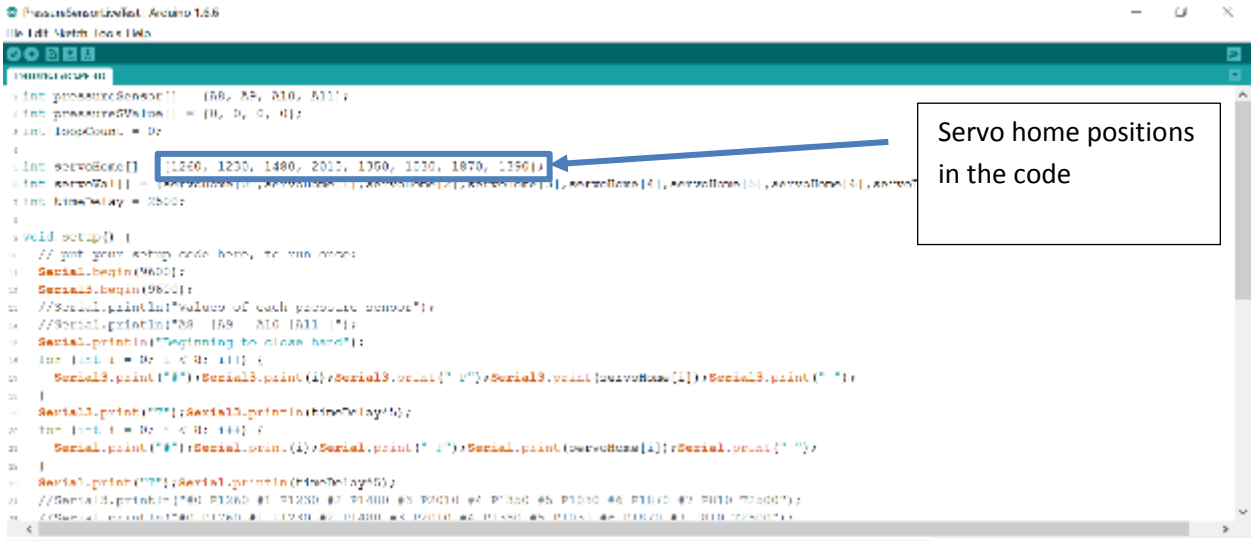
Where are you in the project?

The next step is to finish the programming of the pressure sensor and conduct test runs for the career education breakfast on the 19th.

Picture or Diagram



Servo value device: Each slider sends values to the Arduino, which displays them on the LCD and sends them to the SSC-32, allowing for easy, visual servo position testing



Servo	Home	Descrip	Max	Descrip	Min	Descrip
0	1260	Arm out front	1710	Arm in back	1202	Arm in chest
1	1220	Uarm up	1622	Hand down low	1220	Hand up high
2	1488	Hand out front	1694	Hand out wide	1298	Hand in chest
3	2010	Larm down	2274	Larm up	1530	Larm low
4	1320	Hand out front	1074	Hand out wide	1740	Hand in chest
5	960	Slightly down	926	Out	1240	Back
6	1458	Top, up	2374	Top, out	538	Top, in
7	600	Slightly open	1390	Open	574	Close

Servo positions

Each row has the servo number, home position and description, max position and description, and min position and description

February 17th, 2016

Goals

- Finish programming pressure sensor based on results of live tests
- Present pressure sensor to career education breakfast attendees when they come through (early!)

Accomplished Today

Today I discovered that the deadline for a demo of my project had been moved up to today. I was still able to finish the pressure sensor code in time by using live testing to fine-tune the parameters and speed. Additionally, the demo went well. I demonstrated the pressure sensor by having Baxter's hand close on and hold a small paper cup without crushing it. Additionally, Baxter would close gently on a person's hand and stop with a firm but not crushing grip.

Where are you in the project?

The next step is to achieve communication between Arduino Mega and Arduino Due over serial TX/RX.

Picture or Diagram

Pressure sensor value is read.

If the value is greater than 900, the hand stops for the rest of the program.

Otherwise, the hand closes by 25 counts and repeats the process

February 23rd, 2016

Goals

- To make visible progress in serial communication between the Arduino Mega and Due

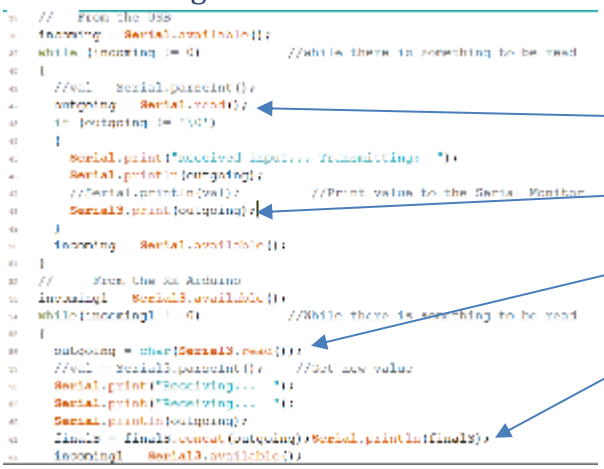
Accomplished Today

I created a basic serial communication program. The Due sends a number to the Mega, which adds 5 to it, then sends it back to the Due, which prints the new value. The number sent from the Due is input in the Serial Monitor and must be a single-digit number.

Where are you in the project?

The next step is to get the communication to work with characters (letters or numbers, one at a time).

Picture or Diagram



Due Code:

Receives value from computer (USB/Serial) and sends to Mega (Serial3)

Receives value from Mega (Serial3) and sends it to computer (Serial)

February 25th, 2016

Goals

- Establish Serial to Serial communication using characters

Accomplished Today

Today I modified the program which adds 5 to the input number to take a string of characters and shift them all 1 ASCII symbol over, then return it. This currently only works with the first string of characters before the communication breaks down, which is less than optimal.

Where are you in the project?

The next step is to send more than just one series of characters before communication fails.

Picture or Diagram

```
void loop()
{
  incoming = Serial3.available();
  while (incoming != 0)                  //While there is something to be read
  {
    newWord = Serial3.read();
    content.concat(newWord);
   // val = Serial3.parseInt(); //Reads integers as integer rather than ASCII. Anything else returns 0

    //Serial3.print(val);                  //Send the new val back to the Tx
    Serial.println(content);
    incoming = Serial3.available();
    for (int i = 0; i < content.length(); i++) {
    char origWord = content.charAt(i);
    wordVal = int(origWord) + 1;
    reply.concat(char(wordVal)); }
    //newWord = newWord + yes;
    //val = val + 5;
    Serial3.println(reply);
 Serial.println(reply);
Serial3.parseInt(); }

}
```

Mega Code:

Upon receipt of a string, it translates each character up one in ASCII value, then sends it back.

March 2nd, 2016

Goals

- To continue exploring options for making a series of long strings be interchanged between the two Arduinos

Accomplished Today

I modified the program which translates characters to translate strings. A user inputs their string into the Serial Monitor on the Due, ending in a '0' (for example: "Hello World0"). The Due sends it to the Mega, which shifts each non-zero character 1 ASCII character. It then sends the new string back to the Due (for example: "").

Where are you in the project?

The next step is to make both Arduinos receive a communication correctly after a successful exchange.

Picture or Diagram

```
void loop()
{
  incoming = Serial3.available();
  while (incoming != 0)                  //While there is something to be read
  {
    newWord = Serial3.read();
    content.concat(newWord);
   // val = Serial3.parseInt(); //Reads integers as integer rather than ASCI. Anything else returns 0

    //Serial3.print(val);                  //Send the new val back to the Tx
    Serial.println(content);
    incoming = Serial3.available();
    if (content.indexOf('0') > 0) {
    for (int i = 0; i < content.length(); i++) {
    char origWord = content.charAt(i);
    wordVal = int(origWord) + 1;
    reply.concat(char(wordVal)); }
    //newWord = newWord + yes;
    //val = val + 5;
    Serial3.println(reply);
 Serial.println(reply);
Serial3.parseInt(); }
  }
}
```

Mega Code:

Upon receipt of each character, it first checks if the character is "0"

If so, it runs the translation operation and returns the translated string

Otherwise, it waits

March 28th, 2016

Goals

- To wire the EasyVR3 board to the Arduino Mega
- To begin a Fritzing diagram of the EasyVR3 to Arduino Mega 2560

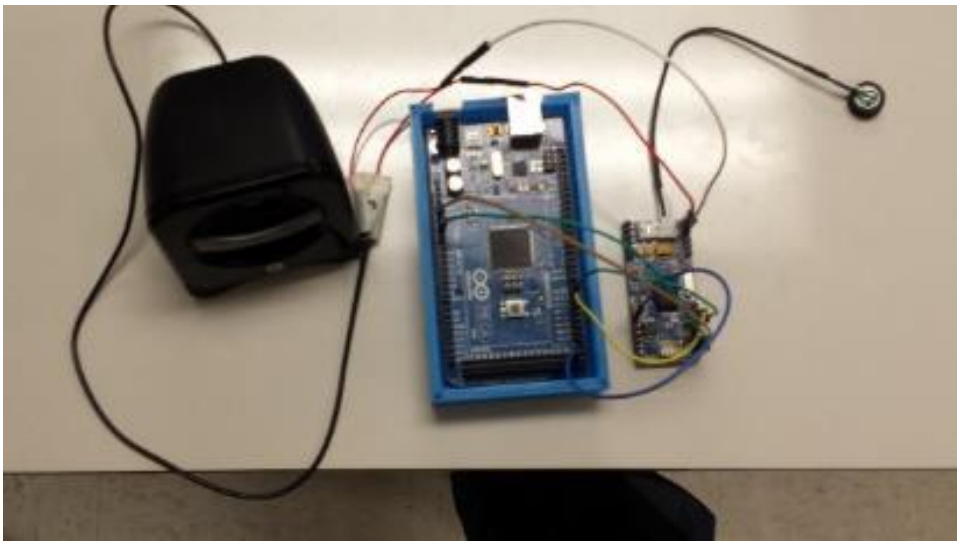
What I did today

With the guidance of the EasyVR3 user manual, I connected an external speaker to the board. As the manual did not specify what wire went were, outside of saying that pin 1 is SP+ and pin 2 is SP-, I made the judgement call that SP+ is signal and SP- is ground. If this doesn't work, hopefully the speaker won't be blown out and I can switch the two. In addition to wiring the speaker into the EasyVR3 board, I also connected the EasyVR3 board to the Arduino Mega. By the end of the week, I hope to have a Fritzing wiring diagram of the connection, but this will require creating the EasyVR3 board in Fritzing, which I had not anticipated. Therefore, here are the ports: 5V on Arduino is connected to 5V on EasyVR3 by a long brown wire; ground on Arduino is connected to ground on EasyVR3 by a long green wire; TX3 on Arduino is connected to RX on EasyVR3 by a short green wire; RX3 on Arduino is connected to TX on EasyVR3 by a short yellow wire; digital pin 22 on Arduino is connected to the reset pin on EasyVR3 (a pull-up reset). It should be noted that the SP- on the EasyVR3 is a white wire to the speaker's ground and the SP+ is a red wire to the speaker's signal. All of the wires from the EasyVR3 to the Arduino are male-female and the wires to the speaker are female-female.

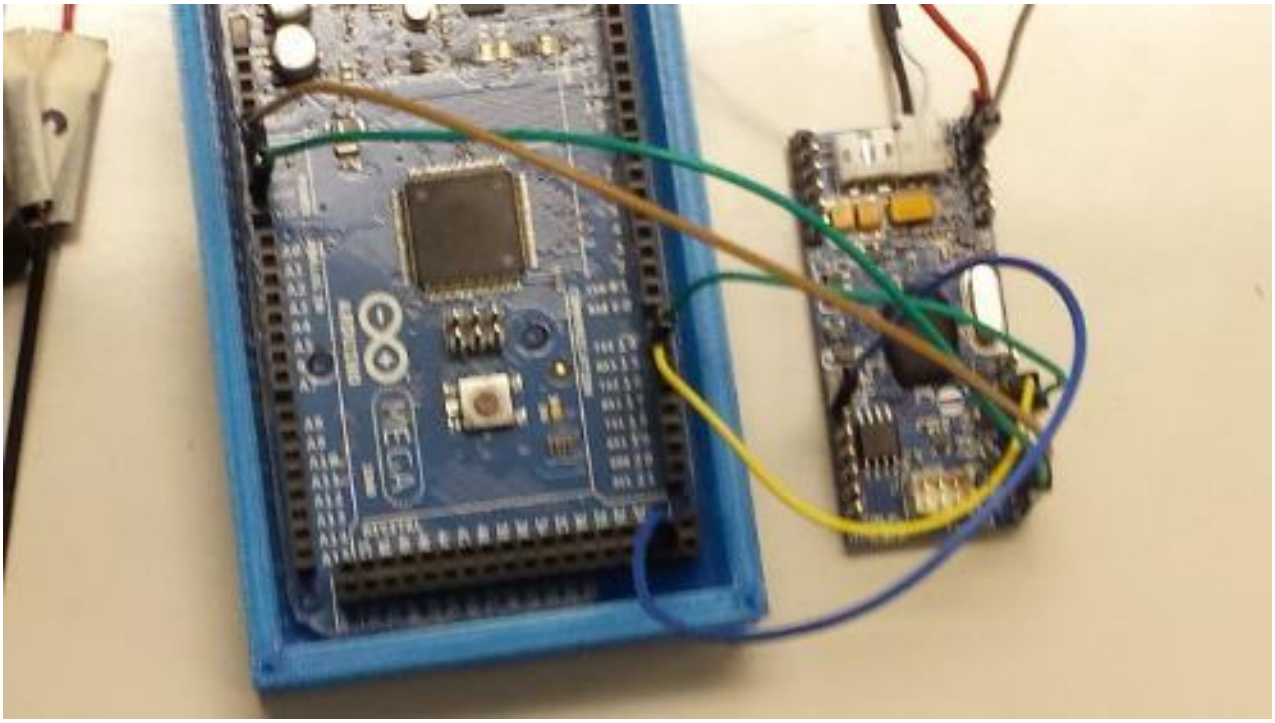
Next Step

The next step is to continue the Fritzing diagram and to begin basic communication between the Arduino and the EasyVR3 boards.

Picture/Diagram



4 Full set-up, including speaker and microphone



5 Close-up on the connections between the speech-to-text board and the Arduino

April 5th, 2016

Goals

- Set up basic serial communication between Arduino and EasyVR
- Get the EasyVR to play a sound as output

What I did today

After reading more of the user manual for the EasyVR3, I determined that there could be a work around to using the shield for communications. Instead of using the library already created for Arduino, but specifically for use with the shield, I began converting the pseudocode in the manual into Arduino code. The first procedure – waking up the EasyVR – was very easy to accomplish. However, further procedures which required waiting for a signal could not just be done with a single send/receive sequence. Instead, what is required is an initial send command, followed by a while loop that waits until the confirmation in received and prints the current Serial read to the console as well as re-sending the command. With this modification, I was able to complete the basic setup and make the EasyVR play a dial tone as output.

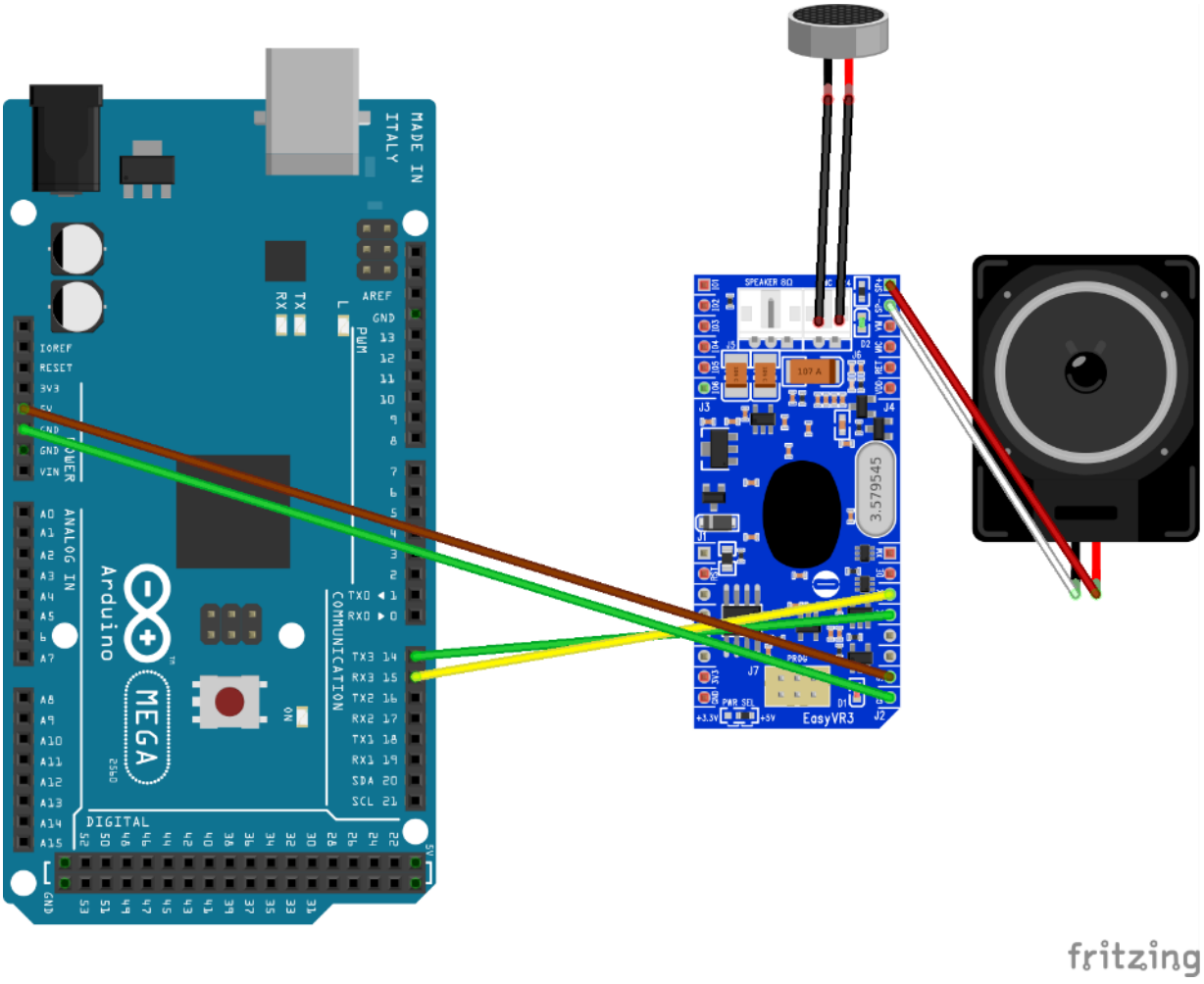
Additionally, I have competed the Fritzing diagram begun earlier and will use it when I return to speech-to-text to rewire the EasyVR3 into the Arduino.

Next Time

After further consideration and reading of the manual, I realized that the EasyVR only works for voice recognition and not also text-to-speech, as I had hoped. Thus, the next step is to put the speech-to-text off until text-to-speech is completed. For this end, the next task is to establish a basic output from the Speech Synthesis module by modifying various sample codes and studying the libraries involved, namely SpeechSynthesis and Delay.

Picture/Diagram/Code

Wire Diagram



6 Wiring diagram

Wake up, Set-up, and Tone Code

```
int timeOutRead = 10; // delay time in ms for any reading of the Serial3
void setup() {
  // put your setup code here, to run once:
  Serial3.begin(9600);
  Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:

  // Wake up Procedure
  Serial.println("Beginning Wake up Procedure");
  Serial3.print('b');
  while (Serial3.read() != 'o') {
      Serial3.print('b');
    }
  Serial.println("Connected to EasyVR3");

  // Set up procedure
  Serial.println("Beginning Set up Procedure");
  Serial.print("Firmware ID Check...");
  Serial3.print('x');
  while (Serial3.read() != 'x') {
      Serial.print(Serial3.read());Serial.print(" ");
      Serial3.print('x');
```

```
    }
  Serial.println("Good");
  Serial.print("Read Status Check...");
  Serial3.print(' ');
  delay(timeOutRead);
  char id = Serial3.read();
  if (id == 'B') {
      Serial.println("EasyVR");
    }
    else {
        Serial.println("Next Version?");
      }
  Serial.print("Language Set (English)...");
  Serial3.print('l');
  Serial3.print('A');
  while (Serial3.read() != 'o') {
      Serial.print(Serial3.read());Serial.print(" ");
      Serial3.print('l'); Serial3.print('A');
    }
  Serial.println("Good");
  Serial.print("Time out Set (5 seconds)...");
  Serial3.print('o');
  Serial3.print('F');
  while (Serial3.read() != 'o') {
      Serial.print(Serial3.read()); Serial.print(" ");
      Serial3.print('o');Serial3.print('F');
    }
  Serial.println("Good");
  Serial.println("Set up Procedure Complete");
  // Test Sound (DTMF key tone)
  Serial.println("Beginning Test Tone (-1,0,0)");
  Serial3.print('w'); Serial3.print('@');Serial3.print('A');Serial3.print('C');
  while (Serial3.read() != 'o') {
      Serial.print(Serial3.read()); Serial.print(" ");
      Serial3.print('w'); Serial3.print('@');Serial3.print('A');Serial3.print('C');
    }
  Serial.println("Good");
  Serial.println("Tone Sounded");
  while(true) {}
}
```

April 13th, 2016

Goals

- Get the Speech Synthesis module to convey a basic message
- Get the Speech Synthesis module to convey a message in response to a query through the serial console

What I did today

Today I was able, somehow, to get the Speech Synthesis module to convey the basic message of the sample code. It worked twice, though I am still trying to figure out the specific steps I took for it to work. Even though it should loop the code, it only repeated a few times before turning off. It also was talking too fast, so I tried having it go half as fast, with the result of it going half as fast and being more intelligible. Until I can get the module to produce viable output on command, converting a query response to vocal output must wait.

As of now, it will only work if you first upload the code with the switch set to Prog. Once the upload is finished, switch it to Run. Then hit reset, unplug the speaker from power, hit reset, plug the speaker in, and hit reset again. Wait for a bit, and it will start talking. It will then wait about 30 seconds from when it started talking to start again, or about 25 seconds after it finishes talking. I would strongly recommend using a speed of 2 or lower.

Next Time

Continue working on getting the speech module to produce viable output consistently on demand. If that is accomplished, try having it convert inputted values to text.

Picture/Diagram/Code

```
#include <SpeechSynthesis.h>
void setup() {
  Serial.begin(9600);
}
 byte ssr[500];//define a character string
void loop() {
SpeechSynthesis.buf_init(ssr);//Clear the buffer
SpeechSynthesis.English(ssr,2,"2");//speed of speaking: grade 5
SpeechSynthesis.English(ssr,4,"5");//volume in grade 5
SpeechSynthesis.English(ssr,6,"Hello World");//"6" means synthesis in English; "cooki"is the
content
SpeechSynthesis.English(ssr,4,"5");
SpeechSynthesis.English(ssr,6,"Hello World");
SpeechSynthesis.English(ssr,2,"10");
SpeechSynthesis.English(ssr,6,"Hello World");
//SpeechSynthesis.English(ssr,5," ");//speed,volume,intomation are all set into default
//SpeechSynthesis.English(ssr,6,"Hello World");
SpeechSynthesis.Espeaking(0,19,4,ssr);//Executive commands above, "0" is synthesis command; "19"
select speaker; "4" speech function
delay(30000);
}
```

April 19th, 2016

Goals

- Change voice
- First version of verbal debugging

What I did today

Today I changed the voice from the original sample code voice which was hard to understand to a new voice (called John, voice number 6) which is easier to understand. I also began doing verbal debugging of the pressure sensor. I connected the pressure sensor to the ground (black wire), 5V power supply (red wire), and to the A8 analog input (white wire). In the code, I then took the value of the pressure sensor and had to convert it first from an integer to a string, then from a string to a constant character pointer array. This is because the function SpeechSynthesis.English() takes an argument of a constant character pointer array.

Next Time

Begin working on a more personalized output of the status, possibly through pre-loaded audio files for specific events.

Picture/Diagram/Code

```
const int pressurePort = A8;
int pressureVal = 0;
bool ran = false;
#include <SpeechSynthesis.h>
void setup()
{
  Serial.begin(9600);

}
 byte ssr[500];//define a character string
void loop()
{
```

```
SpeechSynthesis.buf_init(ssr);//Clear the buffer
SpeechSynthesis.English(ssr,2,"3");//speed of speaking: grade 5
SpeechSynthesis.English(ssr,4,"5");//volume in grade 5
pressureVal = analogRead(pressurePort);
char pressureText[10];
String(pressureVal).toCharArray(pressureText, 10);
const char* pChar = pressureText;
if (ran == false) {SpeechSynthesis.English(ssr,6,"Pressure Sensor Values"); ran = true;}
//SpeechSynthesis.English(ssr,6,"P-Value:");//"6" means synthesis in English; "cooki"is the
content
SpeechSynthesis.English(ssr,6,pressureText);
SpeechSynthesis.Espeaking(0,6,4,ssr);//Executive commands above, "0" is synthesis command; "19"
select speaker; "4" speech function
 delay(2750);
/* while(Serial.read()!=0x41)//waiting synthesis complete
  {}
 while(Serial.read()!=0x4F)//waiting play complete
  {}
SpeechSynthesis.buf_init(ssr);
SpeechSynthesis.English(ssr,6,"hello cooki 123");
SpeechSynthesis.Espeaking(0,19,4,ssr);

   while(Serial.read()!=0x41)
  {}
 while(Serial.read()!=0x4F)
  {}
SpeechSynthesis.buf_init(ssr);
SpeechSynthesis.English(ssr,6,"hello cooki 123");
SpeechSynthesis.Espeaking(0,19,7,ssr);
   */
}
```

May 3rd, 2016

Goals

- Close the hand until a certain pressure value is reached, then stop closing the hand
- State that the hand will close until a given value is reached
- Give live debug while hand closes
- Announce when the hand has closed as far as it was supposed to
- Begin setting up either learning or stylus control for the right arm

What I accomplished

Today I was able to adapt the pressure sensor code from earlier to close the left hand of Baxter. Currently, the code runs thus. First, the robot initializes itself to the “home” position, then opens the claw to the widest point. Then, it enters a cycle of checking the pressure sensor and incrementally closing the claw, 50 counts at a time. If the pressure value exceeds 900, then the claw ceases to close. Throughout this process, there is a visual debugging being run through the Serial Monitor. Unfortunately, when I came in today, the Speech Synthesis board had ceased to work, so I was unable to have audio debugging as well. The initial output states the hand will close until the value of 900 is reached on the pressure sensor. In the loop, it states the pressure sensor value and then when 900 is reached it states “Stopped”.

In addition, I decided to re-implement the learning code for Baxter, adapting it to the right arm, since the pressure sensor is only on the left arm. While it took very little time to wire the servos to the Due I am using to control the right arm, I realized that I still needed to set up the wiring for the potentiometers.

Next Time

Next class, I need to wire the potentiometers for the right arm. I also need to find the various positions for the servos of the right arm and the potentiometer ranges for the right arm. Then I need to test those positions and ranges in the code itself.

Pictures/Diagram/Code

```
#include <SpeechSynthesis.h>
// Global Vaiables, Constants, and Arrays
const int pressurePort = A8; // input port of pressure sensor
int pressureVal = 0; // value of pressure sensor
byte ssr[500]; // string array for speech
int maxPVal = 900; // max pressure value; stopping value
int baseTime = 250; // time to say one character
int stepSize = 25; // number of counts each step takes
int servoNum = 7; // which servo port on SSC-32
int newVal = 1390; // counts of the servo
String greetingStr = "Hello distinguished guests. Please observe as my hand closes until a
pressure value is reached. The value is " + maxPVal;
//greetingStr.concat(maxPVal);
//greetingStr.concat(" is reached.");
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // Communication to the Speech Synthesis board
  Serial3.begin(9600); // Communication to the SSC-32
  outputStr(greetingStr, greetingStr.length());
  //delay(greetingStr.length() * baseTime);
  goHome();
  Serial3.println("#7 P1390 T1000");delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  // Read pressure value
  pressureVal = analogRead(pressurePort);
  // Check if max is reached
  if (pressureVal > maxPVal || pressureVal == maxPVal) {
      String closingStr = pressureVal + ". Thank you for your time.";
      outputStr(closingStr, closingStr.length());
      //delay(closingStr.length() * baseTime);
      Serial.println("Stopped");
      while(true) {}
    }
  // if max isn't reached, state pressure value and close 1 step
  outputStr(String(pressureVal), 10);
  //delay(1000 * baseTime);
  moveStep();
}

void output(String textStr, int arrayLen) {
  byte textOut[500];
  char text[arrayLen];
  textStr.toCharArray(text, arrayLen);
  SpeechSynthesis.buf_init(textOut);
  SpeechSynthesis.English(textOut,2,"3");
  SpeechSynthesis.English(textOut,4,"5");
  SpeechSynthesis.English(textOut,6,text);
  SpeechSynthesis.Espeaking(0,6,4,textOut);
}

void outputStr(String textStr, int arrayLen) {
  Serial.println(textStr);
}

void moveStep() {
  newVal = newVal - stepSize;
  //String text = "#";
  //text = text + servoNum + " P" + newVal + " T500";
  Serial3.print("#");Serial3.print(servoNum);Serial3.print("
P");Serial3.print(newVal);Serial3.println(" T2000");
  delay(500);
}

void goHome() {
  Serial3.println("#0 P1260 #1 P1220 #2 P1488 #3 P2010 #4 P1320 #5 P960 #6 P1458 #7 P600 T5000");
  delay(5000);
```

}

May 5th and 6th, 2016

Goals

- Get the servo values for the right arm
- Begin code for right arm
- Wire potentiometers for right arm

What I accomplished

Over these two days, I got servo values for the right arm. These consist of the home position (starting position), the maximum position (max servo value without injury to robot), and the minimum position (min servo value without injury to robot). In the chart, I also include a basic description of what the position looks like. Additionally, I soldered joins for the potentiometer connections. I need to figure out how the potentiometers were wired, before plugging it into the Arduino, so that I do not fry or otherwise compromise the potentiometers. However, I was able to begin coding. The code is a modified version of learning code for the left arm, with the servo home positions changed. Later, I will modify the code to be more array- and function-based instead of its current procedural programming style. This is the first version of the code, awaiting testing when I get the potentiometers sorted out. Finally, I updated the pressure sensor code, reducing the speed at which it closes and decreasing the pressure cutoff value from 900 counts to 700.

Next time

The next goal is to get the wiring down for the potentiometers. If possible, a beginning for the second version of the learning code should be worked on.

Picture/Diagram/Code

Servo	Home	Descrip	Max	Descrip	Min	Descrip
0	1738	Centered, pointing forward	1840	Closed as far in as possible	1386	Arm making a 45 deg angle from the back centerline - SE
1	1794	Upper arm pointing up	1796	As far back as possible	1290	As far down as possible (necessitates moving the other joints)
2	1498	Upper arm U with U pointing back	1814	Closed as far in as possible	1278	Gear-guard just in front of servo
3	1096	Lower arm pointing down	1864	Arm pointing straight up	1096	Arm pointing down
4	1506	Lower arm U with U pointing towards the center	1800	Lower arm U with U pointing up	1028	Lower arm U with U pointing out
5	1674	Hand parallel to the floor	1796	Hand pointing up	1406	Hand is pointing down
6	530	Servo on hand pointing down	2316	Servo on hand pointing up	530	Servo on hand pointing down

7	728	Claw a finger's width apart	1464	Claw fully open	558	Claw fully closed

May 9th, 2016

Goals

- Set up the wiring for the potentiometers using a breadboard
- Test the current version of the learning code
- Begin working on the next version of the learning code ONLY IF the current version works

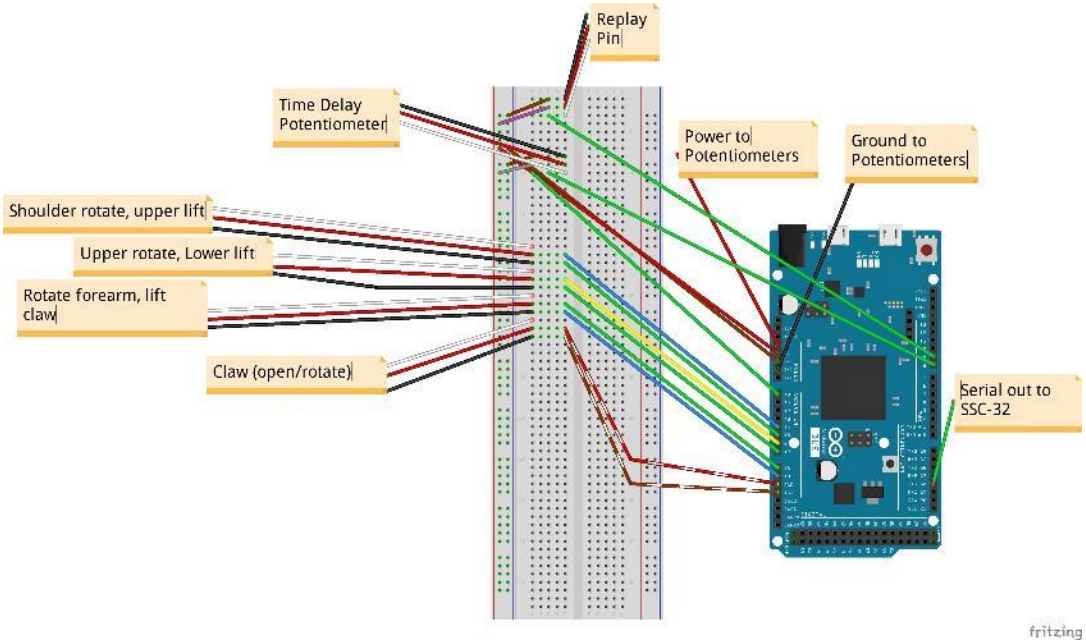
What I accomplished

Today I set up the wiring for all the potentiometers using a breadboard. These include the four 2-axis joysticks and the time delay potentiometer. I also added the replay button to the breadboard. For the two-axis joysticks, the white wire carries no signal, the black wire carries the Y-axis signal, and the red wire carries the X-axis signal. The buttons of the joysticks are all wired together and feed into one signal wire, so that if one button is pressed, the positions are learned. An unexpected roadblock prevented me from testing the current version of the learning code: the code was not uploading to the Arduino Due. There were two errors that came up – that the COM7 port was not found and that the COM7 port was busy. I tried re-uploading while pressing reset, re-uploading after erasing the board, and re-uploading after unplugging all non-programming wires then plugging them in after the attempt. None of these worked.

Next Time

The next step is to try with a different upload cable. If that fails, then to try with a different board. If that fails, then I need to do additional research. After the program uploads, I need to test the code and fix it for Wednesday.

Picture/Diagram/Code



7 Wiring diagram. The banded wires represent the joins of wires. Red banded is red to green; brown banded is brown to white.

May 10th, 2016

Goals

- To upload the new learning code
- To test the code, modifying it for the servos and sensors of this arm
- To have a verbal debugging of at least one of the arm's motions
- To be presentable for tomorrow

What I accomplished

Today started off with a success as I was able to upload the new learning code by using a different micro USB cable. For the most part the code worked exactly as planned, however, servo 4 was closing even when the potentiometer was at rest position. To fix this, I sent the potentiometer 4 value to the serial monitor while it was running, and discovered the problem was this potentiometer's dead band was in a different location that the other potentiometers. Thus, once I created a specific dead band in the code, I was able to prevent it from moving when it was supposed to be at rest. After asking for another person to come and try out the arm to get an idea of what a user would want, I decided to change which servos the potentiometers control so that each potentiometer controls the section of arm it is on. This resulted in a new wiring layout, shown below. To account for the shift of the corrected dead band, I moved it to its new servo. I also had to modify the code to match the direction the potentiometer. For example, if the potentiometer on the forearm is moved toward the chest of the robot, the arm twists in.

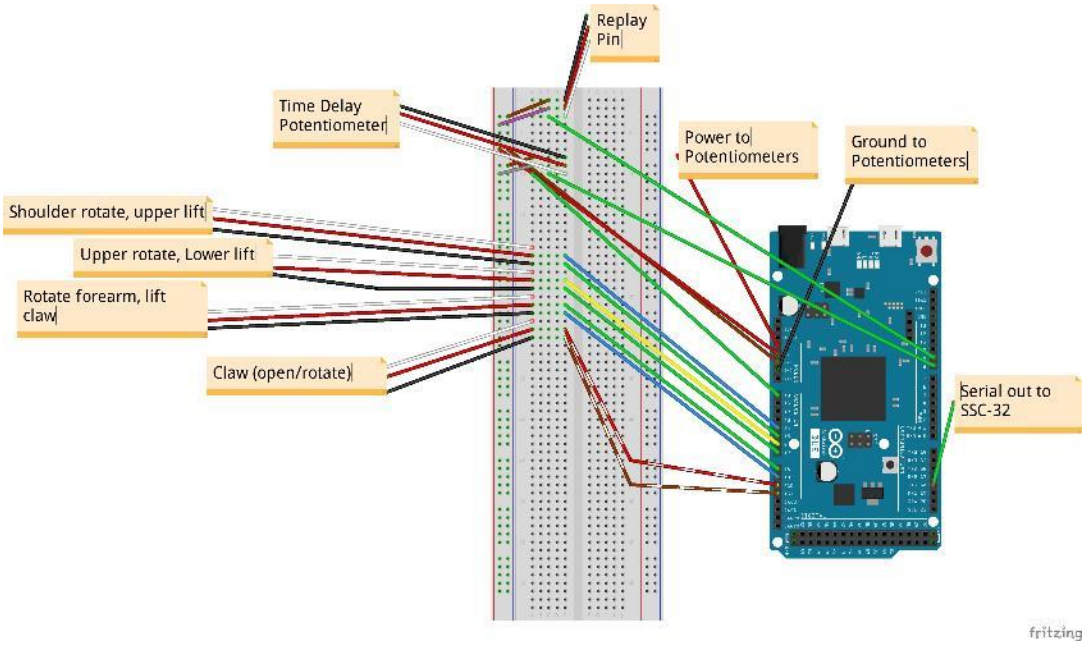
Additionally, I worked on adding verbal debugging to Baxter. First, I attempted to add it to the Due and the learning code, but this failed due to the SpeechSynthesis library not supporting the Arduino Due microcontroller. I then put the SpeechSynthesis board on the Arduino Mega. Now, when the pressure sensor code is run, it states the pressure sensor value before the claw closes its next step. It also greets

the guests using it and states what value – currently set at 800 – it will stop closing the claw at. Once all the way closed, it will thank the user for their time.

Next time

The next step is to create a visual debugging for the Due, such as a LCD screen, that can be shown without using a computer for the serial monitor.

Picture/Diagram/Code



8 New wiring diagram, accounting for shifted potentiometer control

May 12th, 2016

Goals

- To wire the LCD screen into the Due
- To complete a tutorial on how to use the LCD Screen with the due
- To begin a design of how to include visual debugging in the code (plan only)

What I accomplished

I wired the LCD screen into the Due by means of the breadboard. The power for the screen is drawn at 5V, requiring no change for the breadboard. The signal must go into a software serial port (I will use PWM port 2) and the common ground is used. When I tried to compile the sample code, it gave me an error that software serial did not exist. When I tried using it just on the Serial3 port, it did not compile either. I commented all the stuff out and tried to upload the old file, and it gave me a bossac.exe error (below).

Next Time

Figure out how to integrate this LCD screen. If that doesn't work, switch to a different board.

Pictures/Diagram/Code

```
bossac.exe: extra arguments found
Try 'bossac.exe -h' or 'bossac.exe --help' for more information
```

May 13th, 2016

Goals

- Fix bossac.exe error
- Test the sample LCD screen code
- Modify the sample LCD screen code
- Plan how to implement the LCD screen

What I accomplished

I was able to fix the bossac.exe error. It appears that this error is with the 1.6.8 update to Arduino, blocking it from talking to the Due. To fix it, I had to go into the platform.txt file in the "C:\Users\User\AppData\Local\Arduino15\packages\arduino\hardware\sam\1.6.8\" directory and change the tools.bossac.upload.pattern line to remove a particular argument (see code below). After this, it worked fine.

When I tested the LCD sample code, it worked perfectly, even though it was an adapted version, using the Serial3 port instead of SoftwareSerial on port 2. I modified the code to print "The quick brown fox jumped over the lazy dog." and it printed "Error" as it should when the string is longer than 32 characters. When I tested the string "0123456789 0123456789 0123456789" it printed "0123456789 012345" then an unknown characters, seemingly a cross between a 9 and a 5, followed by "6789 012345678". To fix it, I modified the code for the screen display method. Instead of telling it when to start the next line, I just let it figure out when the next line should start. This worked perfectly. When I then tried to put a list of 8 strings, each 4 characters long, it outputted "Error".

Next Time

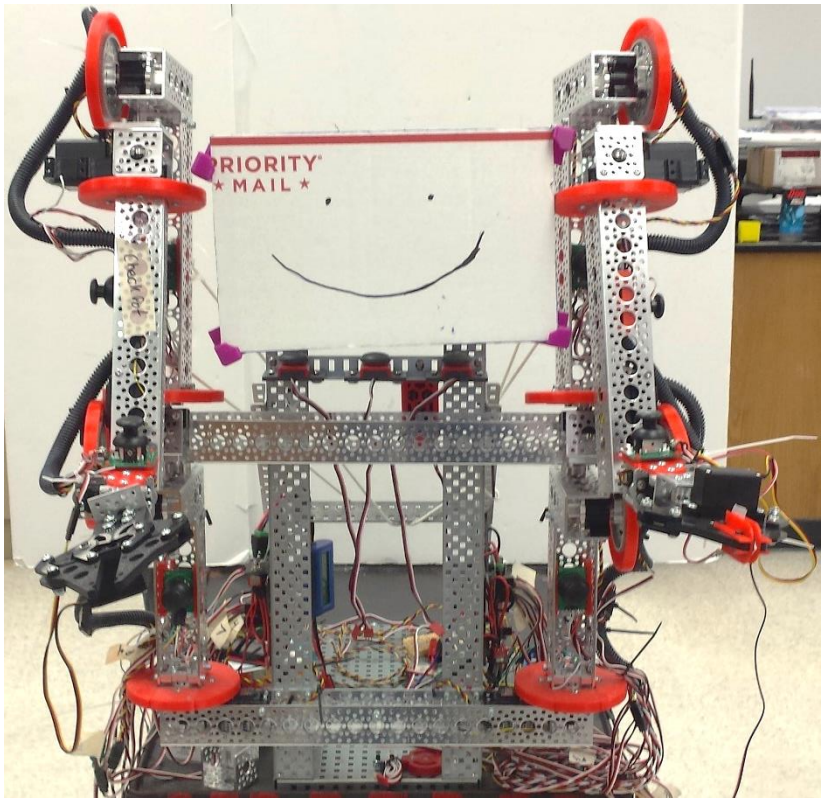
Figure out why it displays an output of "Error" and correct it.

Picture/Diagram/Code

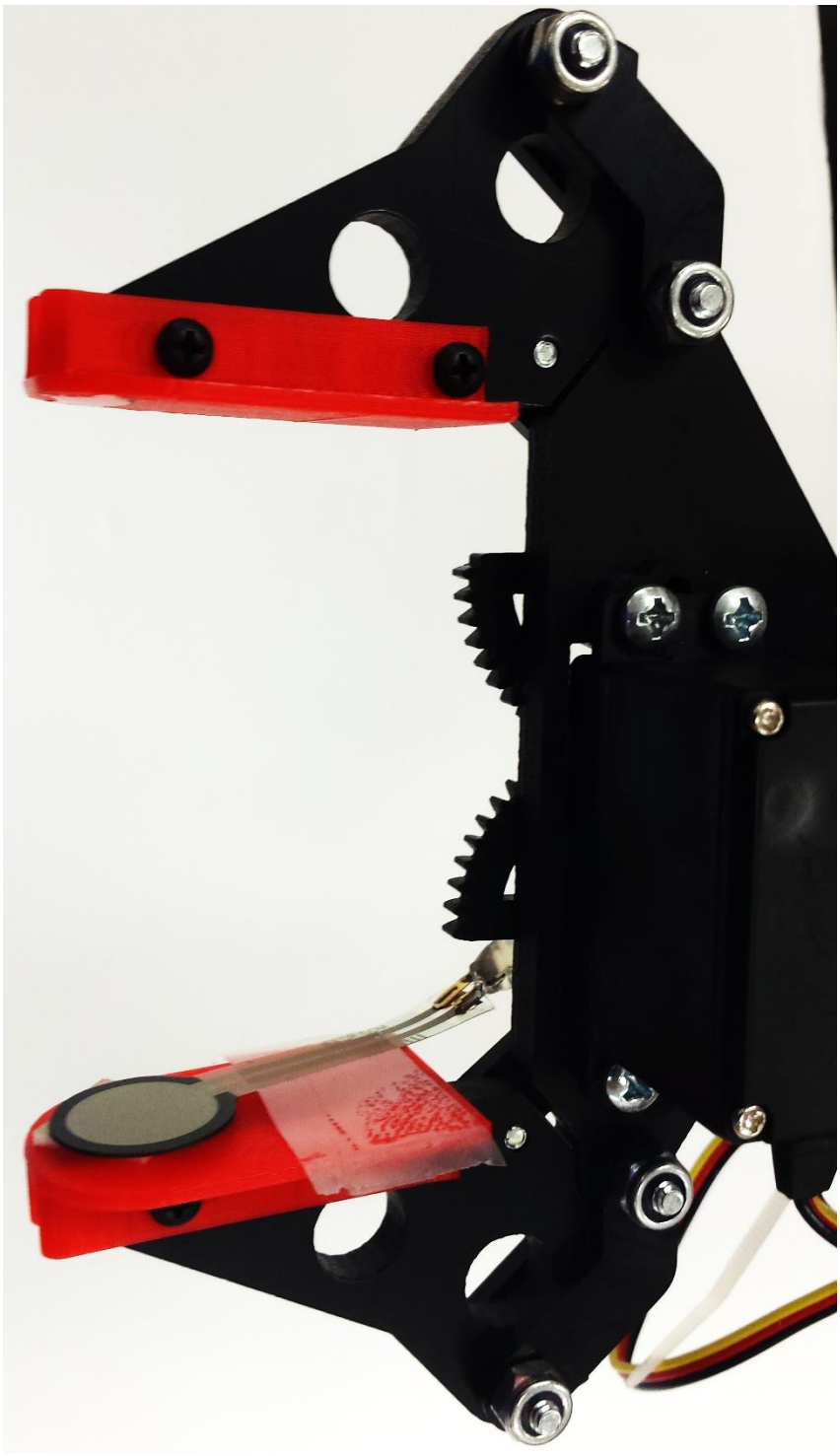
9 LCD Screen displaying "Hello, world!"

Metric 5

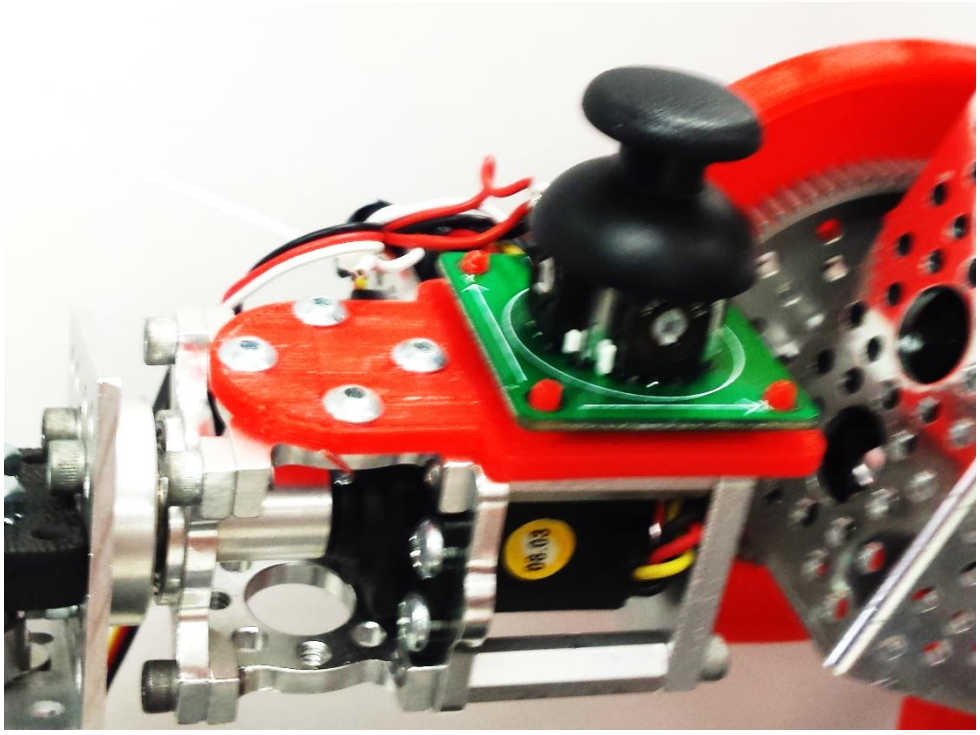
Finished Pictures and Diagrams



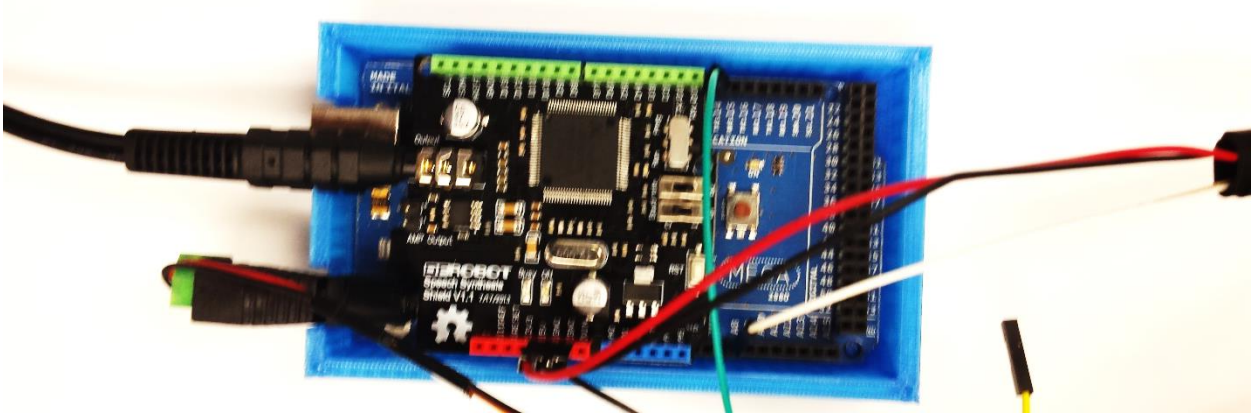
10 Front View of Baxter



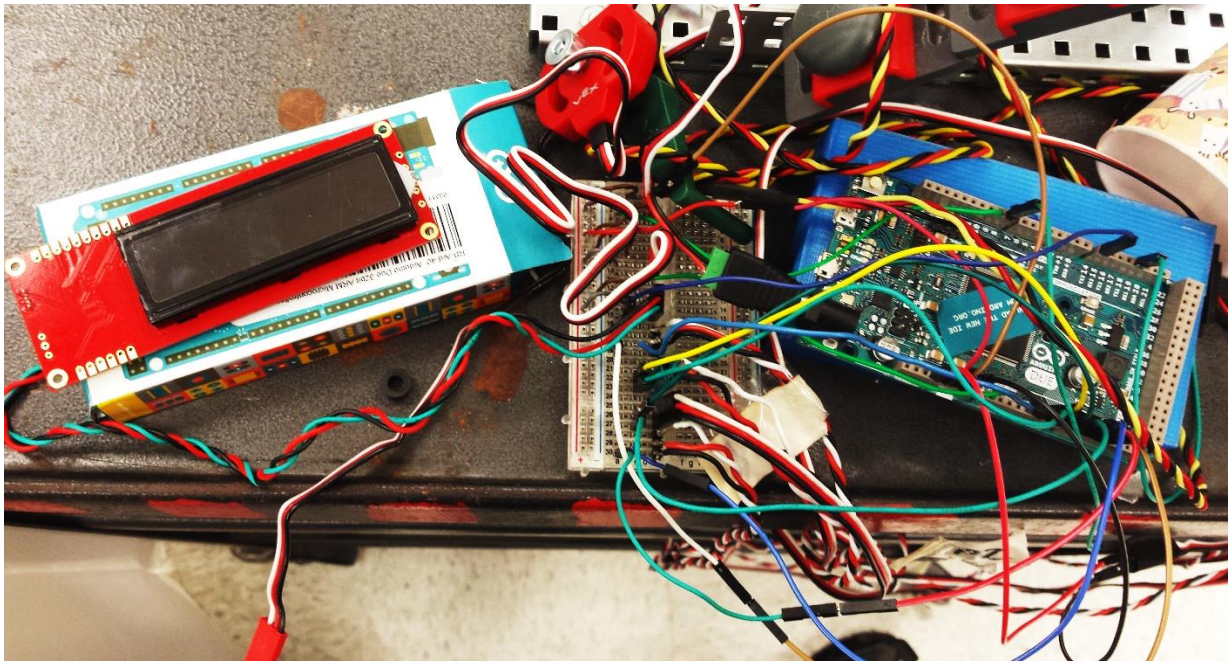
11 Close up of left claw, showing attachment of pressure sensor



12 Close up of right hand's joystick, showing the two axes

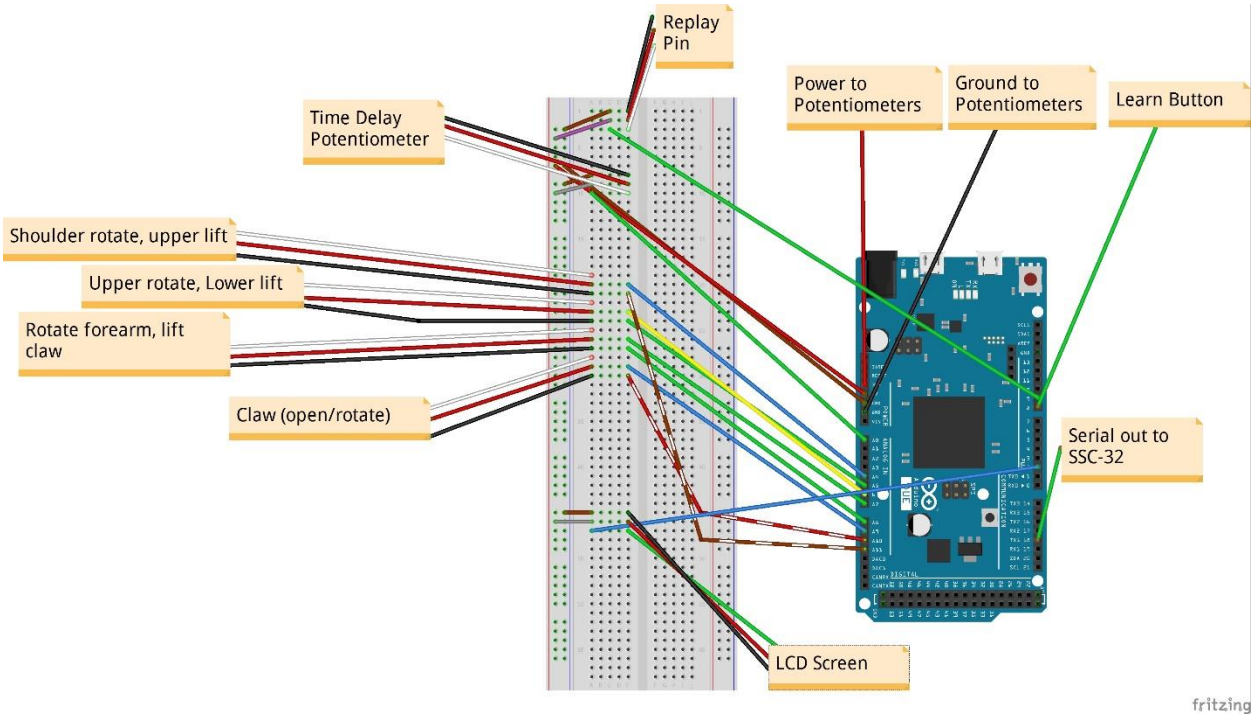


13 Close up of the SpeechSynthesis board, showing how the speaker and pressure sensor connect

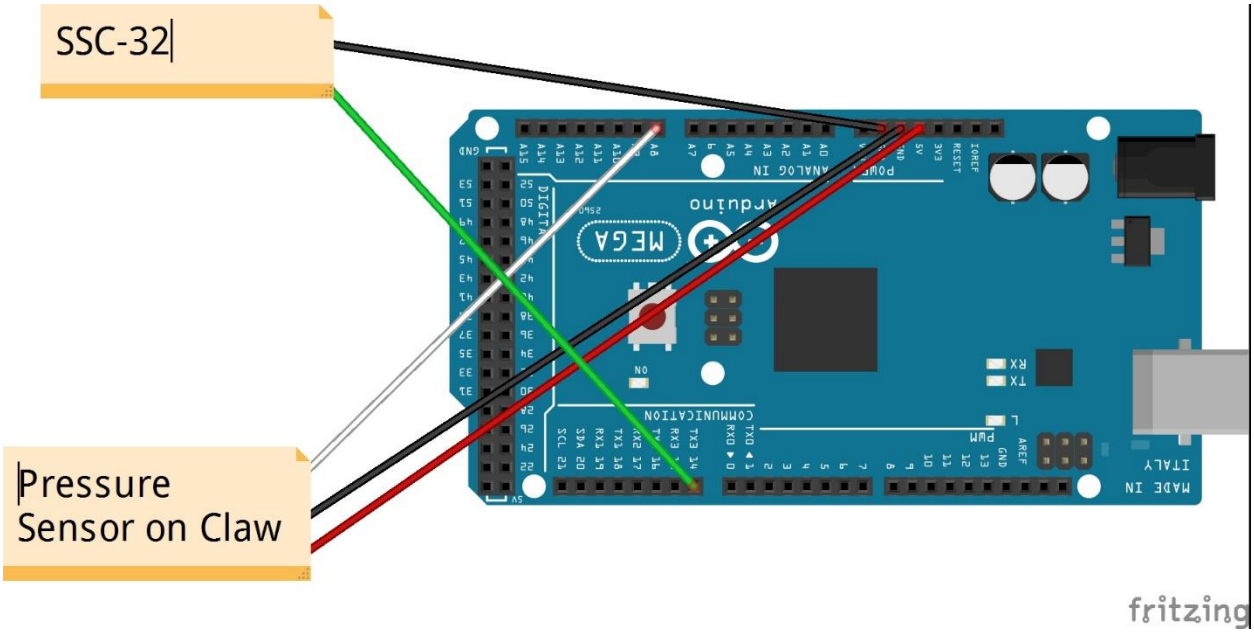


14 Due control board, with breadboarding and LCD screen

15 Close up of the output screen with servo position readouts



16 Right arm control board wiring diagram



17 Left arm control board wiring

Reflections

Summary

Baxter is able to complete two tasks simultaneously – learning code with visual debugging output on a LCD screen and closing until a value is reached with verbal debugging via the Speech Synthesis board. The verbal debugging states the pressure sensor value each time it is read, with one last statement at

the end when the maximum value is exceeded. The screen displays the value of all eight servos on one of the arms, with the values changing in real time as the command is sent to the SSC-32 from the Arduino.

Primary Goals

In retrospect, the primary goals were overly optimistic. I was able to partially complete the first one. Baxter does have a hand that can judge its applied force when doing a task, though this force is reported in counts not in pounds or Newton's. For the second goal, Baxter is able to shake the hand of a person gently. I overestimated the size of the claw that I'm working with – it is too small to hold a tennis ball. As for having Baxter carry on a conversation via the serial monitor and speaker (third goal), I developed a program for it, but was unable to test it. If I get the chance over the summer, I will attempt to get this working. For the fourth goal, I was unable to get a speech-to-text interface to work with Arduino, so I was unable to complete this goal.

Secondary Goals

In hindsight, my secondary goals were overly optimistic. While Baxter can shake someone's hand, expecting it to recognize the correct point in the conversation was too much for one year. If this project were to be continued, this could be finished in the next year or so. Additionally, with Baxter holding an egg without breaking it, I did not consider how confident I would have to be to test this concept.

Extension/Application

Possible applications of pressure sensors used in conjunction with hand servos include any robots which will interact with humans, such as nurse robots or elderly care robots, in addition to robots which will need a careful grip, such as a robot surgeon or a robot which will diffuse bombs “by hand”. An extension of the pressure sensor/servo pairing would be a simulated mouth. The sensor would tell the jaw when to stop closing when a firm grip of the food was attained, so that the jaw doesn’t damage itself.

Possible applications of the voice simulation include any human-robot interaction, as in the aforementioned nurse, surgeon, or elderly care robots. An extension of the voice simulation would be a personalized voice, one like the one of Lt. Commander Data in Star Trek, where the robot’s voice is the same as a person’s without any tinge of mechanical noise in it.

Primary Learning

Programmatically, I learned how to use the Speech Synthesis and Scheduler libraries for Arduino. I also learned how to troubleshoot various issues with the Arduino Mega and the Arduino Due connecting to the computer and to each other.

Code

Learning and Screen Display

```
//#include <SpeechSynthesis.h>
//#include <NewSoftSerial.h>
// learning button on D9
// replay pin on D8
// manual control pot on
// time control pot on A0
// ssc-32 on serial1
```

```
int screenPositions[2][16] =
{{128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143},{192,193,194,195,196,197,198,1
99,200,201,202,203,204,205,206,207}};
int clearScreenPosition = 254;

const int pot0 = A4;  // Analog input pin that the potentiometer is attached to
const int pot1 = A5;  // Analog input pin that the potentiometer is attached to
const int pot2 = A1;  // Analog input pin that the potentiometer is attached to
const int pot3 = A7;  // Analog input pin that the potentiometer is attached to
const int pot4 = A8;  // Analog input pin that the potentiometer is attached to
const int pot5 = A9;  // Analog input pin that the potentiometer is attached to
const int pot6 = A10;  // Analog input pin that the potentiometer is attached to
const int pot7 = A11;  // Analog input pin that the potentiometer is attached to
//const int pot8 = A13;  // Analog input pin that the potentiometer is attached to
//const int pot9 = A6;  // Analog input pin that the potentiometer is attached to

const int timedelay = A0;  // Analog input pin that the potentiometer is attached to
const int learnpin = 9;     // the number of the learning pin
const int replaypin = 8;     // the number of the replay pin

int s0 = 1738; int vpot0 = 0;  //homeposition for #0 servo
int s1 = 1794; int vpot1 = 0;  //homeposition for #0 servo
int s2 = 1498; int vpot2 = 0;  //homeposition for #0 servo
int s3 = 1096; int vpot3 = 0;  //homeposition for #0 servo
int s4 = 1506; int vpot4 = 0;  //homeposition for #0 servo
int s5 = 1674; int vpot5 = 0;  //homeposition for #0 servo
int s6 = 530; int vpot6 = 0;  //homeposition for #0 servo
int s7 = 728; int vpot7 = 0;  //homeposition for #0 servo
//int s8 = 1470; int vpot8 = 0;  //homeposition for #0 servo
//int s9 = 1625; int vpot9 = 0;  //homeposition for #0 servo


int vtimedelay = 0;
int vlearnpin = 0;
int vreplaypin = 0;

int timed = 0;        // value read from the time delay pot
int reply = 0;              //replay bool 0 when not replaying
int divd = 50;          // divider for manual motion
int wait = 1;          // milisecond wait on manual motion update
int learningcounter = 1;  // learning counter
int learningdelay = 2000;  // pause between each learning step
int potdeadbanda = 500;    // lower deadband for manual control pots
int potdeadbandb = 524;    // upper deadband for manual control pots
int potdeadbanda5 = 480;
int potdeadbandb5 = 575;
int ta =2000;           // ta is time between step 1 and 2
int tb =2000;           // ta is time between step 1 and 2
int tc =2000;           // ta is time between step 1 and 2
int td =2000;           // ta is time between step 1 and 2
int te =2000;           // ta is time between step 1 and 2
int tf =2000;           // ta is time between step 1 and 2
int tg =2000;           // ta is time between step 1 and 2
int th =2000;           // ta is time between step 1 and 2

int s0a =1500; int s0b =1500; int s0c =1500; int s0d =1500; int s0e =1500; int s0f =1500; int s0g
=1500; int s0h =1500;
int s1a =1500; int s1b =1500; int s1c =1500; int s1d =1500; int s1e =1500; int s1f =1500; int s1g
=1500; int s1h =1500;
int s2a =1500; int s2b =1500; int s2c =1500; int s2d =1500; int s2e =1500; int s2f =1500; int s2g
=1500; int s2h =1500;
int s3a =1500; int s3b =1500; int s3c =1500; int s3d =1500; int s3e =1500; int s3f =1500; int s3g
=1500; int s3h =1500;
int s4a =1500; int s4b =1500; int s4c =1500; int s4d =1500; int s4e =1500; int s4f =1500; int s4g
=1500; int s4h =1500;
int s5a =1500; int s5b =1500; int s5c =1500; int s5d =1500; int s5e =1500; int s5f =1500; int s5g
=1500; int s5h =1500;
int s6a =1500; int s6b =1500; int s6c =1500; int s6d =1500; int s6e =1500; int s6f =1500; int s6g
=1500; int s6h =1500;
int s7a =1500; int s7b =1500; int s7c =1500; int s7d =1500; int s7e =1500; int s7f =1500; int s7g
=1500; int s7h =1500;
int s8a =1500; int s8b =1500; int s8c =1500; int s8d =1500; int s8e =1500; int s8f =1500; int s8g
=1500; int s8h =1500;
int s9a =1500; int s9b =1500; int s9c =1500; int s9d =1500; int s9e =1500; int s9f =1500; int s9g
=1500; int s9h =1500;
int hptime = 1000;

//SoftwareSerial mySerial(3,2); // 2 = TX

void setup() //initializing serial communication baud rate
{
```

```
    Serial.begin(115200); //sets the baud rate at 115200 for the computers serial monitor
    Serial1.begin(9600); //sets the baud rate at 9600 to match the one set on the SSC-32 board
    Serial3.begin(9600);
    pinMode(learnpin, INPUT_PULLUP);
    pinMode(replaypin, INPUT_PULLUP);
    delay(10000);    //safety wait
      // delay time for home move
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7);
     //Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9);
        Serial1.print(" T"); Serial1.println(hptime);

     //output("Going to home position", String("Going to home position").length());
     Serial.print(" #0 "); Serial.print(s0);
     Serial.print(" #1 "); Serial.print(s1);
     Serial.print(" #2 "); Serial.print(s2);
     Serial.print(" #3 "); Serial.print(s3);
     Serial.print(" #4 "); Serial.print(s4);
     Serial.print(" #5 "); Serial.print(s5);
     Serial.print(" #6 "); Serial.print(s6);
     Serial.print(" #7 "); Serial.print(s7);
     //Serial.print(" #8 "); Serial.print(s8);
     //Serial.print(" #9 "); Serial.print(s9);
      Serial.print(" T"); Serial.println(hptime);

    delay(hptime);
}



void loop()
{
  /*while (true) {   Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7);
     //Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9);
        Serial1.print(" T"); Serial1.println(hptime);

     Serial.print(" #0 "); Serial.print(s0);
     Serial.print(" #1 "); Serial.print(s1);
     Serial.print(" #2 "); Serial.print(s2);
     Serial.print(" #3 "); Serial.print(s3);
     Serial.print(" #4 "); Serial.print(s4);
     Serial.print(" #5 "); Serial.print(s5);
     Serial.print(" #6 "); Serial.print(s6);
     Serial.print(" #7 "); Serial.print(s7);
     //Serial.print(" #8 "); Serial.print(s8);
     //Serial.print(" #9 "); Serial.print(s9);
      Serial.print(" T"); Serial.println(hptime);

    delay(hptime);}*/
  //clearScreen();
  //disScreen("Welcome");
  //Serial.println("Outputted to LCD");

  if ((digitalRead(learnpin) == HIGH) && (digitalRead(replaypin) == HIGH))   //learning button is
not pressed and replay =0
  {
    int vpot0 = analogRead(pot0);
    int vpot1 = analogRead(pot1);
    int vpot2 = analogRead(pot2);
    int vpot3 = analogRead(pot3);
    int vpot4 = analogRead(pot4);

    int vpot5 = analogRead(pot5);//Serial.print("Pot 5 ");Serial.println(vpot5);
    int vpot6 = analogRead(pot6);
    int vpot7 = analogRead(pot7);
   // int vpot8 = analogRead(pot8);
```

```
    //int vpot9 = analogRead(pot9);

    int vtimedelay = analogRead(timedelay);
    int vlearnpin = digitalRead(learnpin);int vreplaypin = digitalRead(replaypin);
    timed = (vtimedelay*3);

    if (vpot0 > potdeadbandb && s0 > 1386)      {s0 = s0 - ((vpot0 - potdeadbandb)/divd);}
    if (vpot0 < potdeadbanda && s0 < 1840)       {s0 = s0 + ((potdeadbanda - vpot0)/divd);}

    if (vpot1 > potdeadbandb && s1 < 1796)      {s1 = s1 + ((vpot1 - potdeadbandb)/divd);}
    if (vpot1 < potdeadbanda && s1 > 1290)       {s1 = s1 - ((potdeadbanda - vpot1)/divd);}

    if (vpot2 > potdeadbandb && s2 > 1278)      {s2 = s2 - ((vpot2 - potdeadbandb)/divd);}
    if (vpot2 < potdeadbanda && s2 < 1814)       {s2 = s2 + ((potdeadbanda - vpot2)/divd);}

    if (vpot3 > potdeadbandb && s3 < 1864)      {s3 = s3 - ((potdeadbanda - vpot3)/divd);} // {s3
= s3 + ((vpot3 - potdeadbandb)/divd);}
    if (vpot3 < potdeadbanda && s3 > 1096)       {s3 = s3 + ((vpot3 - potdeadbandb)/divd);} //{s3
= s3 - ((potdeadbanda - vpot3)/divd);}

    if (vpot4 > potdeadbandb && s4 > 1028)      {s4 = s4 - ((vpot4 - potdeadbandb)/divd);} //
swapped the <> signs
    if (vpot4 < potdeadbanda && s4 < 1800)       {s4 = s4 + ((potdeadbanda - vpot4)/divd);}

    if (vpot5 > potdeadbandb5 && s5 < 1796)      {s5 = s5 - ((potdeadbanda5 - vpot5)/divd);}//{s5
= s5 + ((vpot5 - potdeadbandb5)/divd);}
    if (vpot5 < potdeadbanda5 && s5 > 1406)       {s5 = s5 + ((vpot5 -
potdeadbandb5)/divd);}//{s5 = s5 - ((potdeadbanda5 - vpot5)/divd);}

    if (vpot6 > potdeadbandb && s6 < 2316)      {s6 = s6 + ((vpot6 - potdeadbandb)/divd);}
    if (vpot6 < potdeadbanda && s6 > 530)       {s6 = s6 - ((potdeadbanda - vpot6)/divd);}

    if (vpot7 > potdeadbandb && s7 < 1464)      /*{s7 = s7 - ((potdeadbanda - vpot7)/divd);} //
*/{s7 = s7 + ((vpot7 - potdeadbandb)/divd);}
    if (vpot7 < potdeadbanda && s7 > 558)       /*{s7 = s7 + ((vpot7 - potdeadbandb)/divd);} //
*/{s7 = s7 - ((potdeadbanda - vpot7)/divd);}
   /*
    if (vpot8 > potdeadbandb && s8 < 2400)      {s8 = s8 + ((vpot8 - potdeadbandb)/divd);}
    if (vpot8 < potdeadbanda && s8 > 600)       {s8 = s8 - ((potdeadbanda - vpot8)/divd);}

    if (vpot9 > potdeadbandb && s9 > 600)      {s9 = s9 - ((vpot9 - potdeadbandb)/divd);}
    if (vpot9 < potdeadbanda && s9 < 2400)       {s9 = s9 + ((potdeadbanda - vpot9)/divd);}
*/
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7);
     /*Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8);
     Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9); */
        Serial1.print(" T"); Serial1.println(wait);

     Serial.print(" #0 "); Serial.print(s0);
     Serial.print(" #1 "); Serial.print(s1);
     Serial.print(" #2 "); Serial.print(s2);
     Serial.print(" #3 "); Serial.print(s3);
     Serial.print(" #4 "); Serial.print(s4);
     Serial.print(" #5 "); Serial.print(s5);
     Serial.print(" #6 "); Serial.print(s6);
     Serial.print(" #7 "); Serial.print(s7);
     /*Serial.print(" #8 "); Serial.print(s8);
     Serial.print(" #9 "); Serial.print(s9);     */
      Serial.print(" T"); Serial.println(wait);

    Serial.print(" TD "); Serial.print(timed);  Serial.print(" TDpot ");
Serial.print(vtimedelay);
    Serial.print(" Learn ");Serial.print(vlearnpin);Serial.print(" Replay
");Serial.println(vreplaypin);
    // sprintf(rpmstring,"%4d",temp);
    char str0[4],str1[4],str2[4],str3[4],str4[4],str5[4],str6[4],str7[4];

sprintf(str0,"%4d",s0);sprintf(str1,"%4d",s1);sprintf(str2,"%4d",s2);sprintf(str3,"%4d",s3);sprin
tf(str4,"%4d",s4);sprintf(str5,"%4d",s5);sprintf(str6,"%4d",s6);sprintf(str7,"%4d",s7);
    String valForScreen = "";

valForScreen.concat(str0);valForScreen.concat(str1);valForScreen.concat(str2);valForScreen.concat
(str3);valForScreen.concat(str4);valForScreen.concat(str5);valForScreen.concat(str6);valForScreen
.concat(str7);
```

```
    clearScreen();
    disScreen(valForScreen);

   delay(wait);

   //Serial.print(" Cpot ");Serial.print(vpot0);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 1))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7;// s8a=s8; s9a=s9;
     s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; //s8b=s8; s9b=s9;
     s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; //s8c=s8; s9c=s9;
     s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; //s8d=s8; s9d=s9;
     s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; //s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 2))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; //s8b=s8; s9b=s9;
     s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; //s8c=s8; s9c=s9;
     s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; //s8d=s8; s9d=s9;
     s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; //s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 3))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; //s8c=s8; s9c=s9;
     s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; //s8d=s8; s9d=s9;
     s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; //s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 4))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     //s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; s8c=s8; s9c=s9;
     s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; //s8d=s8; s9d=s9;
     s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; //s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }
```

```
  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 5))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     //s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; s8c=s8; s9c=s9;
     //s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; s8d=s8; s9d=s9;
     s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; //s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 6))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     //s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; s8c=s8; s9c=s9;
     //s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; s8d=s8; s9d=s9;
     //s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; s8e=s8; s9e=s9;
     s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; //s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 7))  //if learning button is pressed
and replay =0
  {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     //s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; s8c=s8; s9c=s9;
     //s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; s8d=s8; s9d=s9;
     //s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; s8e=s8; s9e=s9;
     //s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; s8f=s8; s9f=s9;
     s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; //s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

  if ((digitalRead(learnpin) == LOW)&&(learningcounter == 8))  //if learning button is pressed
and replay =0
   {
    timed = (analogRead(timedelay)*5);
    Serial.print("step "); Serial.print(learningcounter); Serial.print("time Delay ");
Serial.println(timed);
     //s0a=s0; s1a=s1; s2a=s2; s3a=s3; s4a=s4; s5a=s5; s6a=s6; s7a=s7; s8a=s8; s9a=s9;
     //s0b=s0; s1b=s1; s2b=s2; s3b=s3; s4b=s4; s5b=s5; s6b=s6; s7b=s7; s8b=s8; s9b=s9;
     //s0c=s0; s1c=s1; s2c=s2; s3c=s3; s4c=s4; s5c=s5; s6c=s6; s7c=s7; s8c=s8; s9c=s9;
     //s0d=s0; s1d=s1; s2d=s2; s3d=s3; s4d=s4; s5d=s5; s6d=s6; s7d=s7; s8d=s8; s9d=s9;
     //s0e=s0; s1e=s1; s2e=s2; s3e=s3; s4e=s4; s5e=s5; s6e=s6; s7e=s7; s8e=s8; s9e=s9;
     //s0f=s0; s1f=s1; s2f=s2; s3f=s3; s4f=s4; s5f=s5; s6f=s6; s7f=s7; s8f=s8; s9f=s9;
     //s0g=s0; s1g=s1; s2g=s2; s3g=s3; s4g=s4; s5g=s5; s6g=s6; s7g=s7; s8g=s8; s9g=s9;
     s0h=s0; s1h=s1; s2h=s2; s3h=s3; s4h=s4; s5h=s5; s6h=s6; s7h=s7; //s8h=s8; s9h=s9;

    learningcounter ++; ta = timed; delay(learningdelay);
  }

   if (digitalRead(replaypin) == LOW) //if replay button is pressed and replay =0
  {
    delay(2000);    //safety wait
        //step 1
     Serial.println("Step 1");
     //Serial.print("#0"); Serial.print(" P"); Serial.print(s0a); Serial.print(" T");
Serial.println(ta);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0a);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1a);
```

```
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2a);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3a);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4a);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5a);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6a);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7a);
    /* Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8a);
     Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9a); */
       Serial1.print(" T"); Serial1.println(ta); delay(ta);

        //step 2
     Serial.println("Step 2");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0b);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1b);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2b);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3b);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4b);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5b);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6b);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7b);
    // Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8b);
     //Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9b);
        Serial1.print(" T"); Serial1.println(tb);  delay(tb);

        //step 3
    Serial.println("Step 3");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0c);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1c);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2c);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3c);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4c);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5c);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6c);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7c);
 //    Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8c);
   //  Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9c);
        Serial1.print(" T"); Serial1.println(tc);  delay(tc);

        //step 4
    Serial.println("Step 4");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0d);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1d);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2d);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3d);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4d);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5d);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6d);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7d);
  //   Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8d);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9d);
        Serial1.print(" T"); Serial1.println(td);  delay(td);

       //step 5
    Serial.println("Step 5");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0e);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1e);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2e);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3e);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4e);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5e);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6e);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7e);
  //   Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8e);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9e);
        Serial1.print(" T"); Serial1.println(te);  delay(te);

       //step 6
    Serial.println("Step 6");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0f);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1f);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2f);
```

```
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3f);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4f);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5f);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6f);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7f);
  //   Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8f);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9f);
        Serial1.print(" T"); Serial1.println(tf);  delay(tf);

       //step 7
    Serial.println("Step 7");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0g);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1g);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2g);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3g);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4g);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5g);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6g);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7g);
  //   Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8g);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9g);
        Serial1.print(" T"); Serial1.println(tg);  delay(tg);

        //step 8
    Serial.println("Step 8");
     // Serial.print("#0"); Serial.print(" P"); Serial.print(s0b); Serial.print(" T");
Serial.println(tb);
     Serial1.print("#0"); Serial1.print(" P"); Serial1.print(s0h);
     Serial1.print(" #1"); Serial1.print(" P"); Serial1.print(s1h);
     Serial1.print(" #2"); Serial1.print(" P"); Serial1.print(s2h);
     Serial1.print(" #3"); Serial1.print(" P"); Serial1.print(s3h);
     Serial1.print(" #4"); Serial1.print(" P"); Serial1.print(s4h);
     Serial1.print(" #5"); Serial1.print(" P"); Serial1.print(s5h);
     Serial1.print(" #6"); Serial1.print(" P"); Serial1.print(s6h);
     Serial1.print(" #7"); Serial1.print(" P"); Serial1.print(s7h);
  //   Serial1.print(" #8"); Serial1.print(" P"); Serial1.print(s8h);
    // Serial1.print(" #9"); Serial1.print(" P"); Serial1.print(s9h);
        Serial1.print(" T"); Serial1.println(th);  delay(th);
  }
}
/**/
void clearScreen() {
    Serial3.write(254);
    Serial3.write(128);
    Serial3.print("                ");
    Serial3.print("                ");
  }

void disScreen(String text) {
    Serial3.write(clearScreenPosition);
    Serial3.write(screenPositions[0][0]);
    if (text.length() < 17) {
        Serial3.print(text);
      }
      else if (text.length() < 34) {
        String text1 = text.substring(0,16);
        String text2 = text.substring(16);
        Serial3.print(text1);
        //Serial3.write(screenPositions[1][0]);
        Serial3.print(text2);
        }
        else {
            Serial3.print("Error");
          }
  }/**/

/*
void output(String textStr, int arrayLen) {
    byte textOut[500];
    char text[arrayLen];
    textStr.toCharArray(text, arrayLen);
    SpeechSynthesis.buf_init(textOut);
    SpeechSynthesis.English(textOut,2,"3");
    SpeechSynthesis.English(textOut,4,"5");
    SpeechSynthesis.English(textOut,6,text);
    SpeechSynthesis.Espeaking(0,6,4,textOut);

  }
```

```
*/
```

Pressure Sensor and Voice Simulation

```
#include <SpeechSynthesis.h>
// Global Vaiables, Constants, and Arrays
const int pressurePort = A8; // input port of pressure sensor
int pressureVal = 0; // value of pressure sensor
byte ssr[500]; // string array for speech
int maxPVal = 800; // max pressure value; stopping value
int baseTime = 250; // time to say one character
int stepSize = 10; // number of counts each step takes
int servoNum = 7; // which servo port on SSC-32
int newVal = 1390; // counts of the servo
String greetingStr = "Hello distinguished guests. Please observe as my hand closes until a
pressure value is reached. The value is " + maxPVal;
//greetingStr.concat(maxPVal);
//greetingStr.concat(" is reached.");
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // Communication to the Speech Synthesis board
  Serial3.begin(9600); // Communication to the SSC-32
  output("Hello distinguished guests. Please observe as my hand closes until a pressure value of
at least 800 is reached.", 200);
  delay(15000);
  //output(greetingStr, greetingStr.length());
  //delay(greetingStr.length() * baseTime);
  goHome();
  Serial3.println("#7 P1390 T1000");delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  // Read pressure value
  pressureVal = analogRead(pressurePort);
  // Check if max is reached
  if (pressureVal > maxPVal || pressureVal == maxPVal) {
      //String closingStr = pressureVal + ". Thank you for your time.";
      output(String(pressureVal), 10);
      delay(3000);
      output("Thank you for your time", 30);
      //delay(closingStr.length() * baseTime);
      //Serial.println("Stopped");
      while(true) {}
    }
  // if max isn't reached, state pressure value and close 1 step
  output(String(pressureVal), 10);
  //delay(1000 * baseTime);
  moveStep();
}

void output(String textStr, int arrayLen) {
  byte textOut[750];
  char text[arrayLen];
  textStr.toCharArray(text, arrayLen);
  SpeechSynthesis.buf_init(textOut);
  SpeechSynthesis.English(textOut,2,"3");
  SpeechSynthesis.English(textOut,4,"5");
  SpeechSynthesis.English(textOut,6,text);
  SpeechSynthesis.Espeaking(0,6,4,textOut);
}

void outputStr(String textStr, int arrayLen) {
  Serial.println(textStr);
}

void moveStep() {
  //if ((pressureVal > (maxPVal - 200))) {stepSize = stepSize / 2;}
  newVal = newVal - stepSize;
  //String text = "#";
  //text = text + servoNum + " P" + newVal + " T500";
  Serial3.print("#");Serial3.print(servoNum);Serial3.print("
P");Serial3.print(newVal);Serial3.println(" T2000");
  delay(750);
}

void goHome() {
  Serial3.println("#0 P1260 #1 P1220 #2 P1488 #3 P2010 #4 P1320 #5 P960 #6 P1458 #7 P600 T5000");
  delay(5000);
}
```