A Three-Dimensional Reconfigurable Surface: Investigation of Solenoid Driven Linear Actuation and Implementation of a Graphical User Interface

by

Jack Snodgrass

Submitted in partial fulfillment of the

requirements for Departmental Honors in

the Department of Engineering

Texas Christian University

Fort Worth, Texas

May 4, 2020

A Three-Dimensional Reconfigurable Surface: Investigation of Solenoid Driven Linear Actuation and Implementation of a Graphical User Interface

Project Approved:

Supervising Professor: Stephen Weis, Ph.D.

Department of Engineering

Robert Bittle, Ph.D.

Department of Engineering

Eric Hanson, Ph.D

Department of Mathematics

ABSTRACT

Contributions and further research suggestions for a project focused on creating a three-dimensional reconfigurable surface for the creation of custom fit orthotics are outlined. The functionality and feasibility of solenoids as a method to miniaturize components and perform smooth linear motion is analyzed. The analytical approach provides a direct answer on their viability to be used as a driver for this given project. In conjunction with testing of solenoids, preliminary research is reported on Linear Tubular Motors as a possible alternative electromagnetic linear motion driver. The creation of an open-source GUI, which can be downloaded onto a personal computer and controls the motion of each element is also reported. Also, two possible avenues to create three-dimensional models of any created surface are theorized. Unfortunately with the onset of the COVID-19 outbreak, personal research and experimental data gathering ceased and this research became purely theoretical. As such, the theoretical steps and calculations will be left for the sake of posterity and continued research.

A Three-Dimensional Reconfigurable Surface: Investigation of Solenoid Linear Actuation and Implementation of a Graphic User Interface

Jack E Snodgrass[1], *Student Member, IEEE*

[1]Department of Engineering, Texas Christian University, Fort Worth, Texas 76129, USA

Abstract – **Contributions and further research suggestions for a project focused on creating a three-dimensional reconfigurable surface for the creation of custom fit orthotics are outlined. The functionality and feasibility of solenoids as a method to miniaturize components and perform smooth linear motion is analyzed. The analytical approach provides a direct answer on their viability to be used as a driver for this given project. In conjunction with testing of solenoids, preliminary research is reported on Linear Tubular Motors as a possible alternative electromagnetic linear motion driver. The creation of an open-source GUI, which can be downloaded onto a personal computer and controls the motion of each element is also reported. Also, two possible avenues to create three-dimensional models of any created surface are theorized. Unfortunately with the onset of the COVID-19 outbreak, personal research and experimental data gathering ceased and this research became purely theoretical. As such, the theoretical steps and calculations will be left for the sake of posterity and continued research.**

I. Introduction

The application of electromagnetic machinery to provide linear motion has increased steadily throughout recent years, especially in areas where unique and innovative implementations can mean the success or failure of a design. The need for such implementations can be found in the ever-expansive field of healthcare. For the past seven years, a project has been slowly progressing at Texas Christian University (TCU), which encompasses the unique healthcare field of orthotics. Faculty at TCU proposed an idea to a local orthotist that a research project be put in motion to replicate the process in which he uses to align an individual's posture. For this project, student engineers at TCU are tasked with finding an innovative way to create a three-dimensional reconfigurable surface. Said surface must hold a patient being fitted for an implant and hold the surface mold of the patient that aligns his/her posture. After the mold of the patient's orthotic is taken down, the surface must then be able to return to its equilibrium position.

The reconfigurable surface has been created by connecting the tops of individual rods to a pliable silicone-fabric composite material, which while remaining ergonomic and comfortable to the user provides enough rigidity to sustain a surface to retrieve an orthotic impression. A 3x3 array of these rods supports the individual standing on array and allows for individual movement of each rod, thus altering the surface. However, the rods only act as a structure to form the surface needed to correct the patients posture and as anchors for the surface themselves. The current method of loading is by attaching springs to hold and store the displaced energy from the weight of the patient and allow a linear driver to make up only the differential work needed to align the patients posture. Since the rods pinned into the surface material must be elevated to different heights, it stands to reason that the material chosen must have a resiliency to plastic deformation effects. As such, it was decided that a layered composite material be created with the outside layers being comprised of ductile materials while the inner material be comprised of a stiffer material. This composite material was created

in house with on-hand materials by casting a silicone surface around a silk sheet thus offering both a flexible yet rigid anchor for the driven shafts.

Formerly, a host of ideas, such as pneumatics, hydraulics and screw drives have been assessed for viability as the driver for the motion of each individual element in the reconfigurable surface's array. Unfortunately, the pneumatic and hydraulic variants presented numerous and rapid modes of failure. While the screw drive works, the motors controlling the driving rods take up a great deal of space and limit the amount of elements in the array. As such, miniaturization of the driving elements was the prime reason for analyzing solenoids for their potential as drive replacements.

Ideally, the machine should be controllable via a tablet or handheld device that allows for the orthotist to move around the patient and accurately assess the individual's unique needs. To allow for this autonomy in movement, an interactive graphic user interface was created using an open source coding platform and processor in tandem with java script. This GUI is capable of running the current 3x3 grid of motors installed in the prototype reconfigurable surface with the aid of an Arduino Uno and two stackable Adafruit motor control boards.

II. Defining the Mechanical Properties of a Silicone Rubber-Silk Composite

Given the design parameters and available materials, a prototype surface was created to bind the ends of the rods together. The material was made by layering stock silk fabric between a cast of a polydimethylsiloxane elastomer, sourced from a company called Smooth-On [3], [6]. The product Dragon Skin™ 10 FAST had already been purchased by the department and, as such, was used as the silicone-based medium. Manufacturing the surface was rather simple. Dragon Skin™ was poured into an 8"x13" mold that had a similarly cut sheet of silk cloth placed inside. The silicone reagents were mixed thoroughly and the silk sheet was smoothed out by hand. Then the composite was left to cure for 75min at 73ºF (23ºC) and was removed from the mold.

Unfortunately, the finished surface was never able to be directly tested due to the lockdown of campus. However, the process by which the material should be tested was formulated, and therefore, can be lain out for posterity. In order to test the composite material reliably, all procedures must adhere to the ASTM code for tensile testing. As such, multiple test samples should be cut from the composite material that coincide with the measurements of the ASTM D-3039 standard, shown below [1].

A tensile test performed on a sample size of these style of specimens should produce results for important mechanical property values such as: Elastic Modulus (E), Yield Stress (σ_{yield}), Ultimate Stress ($\sigma_{Ultimate}$) and the percent elongation nearing failure. This experimental data does not exist because the project was terminated before such testing could be conducted, but affixing the specimens to the SATEC 20000 Universal Tensile Testing machine should yield easily

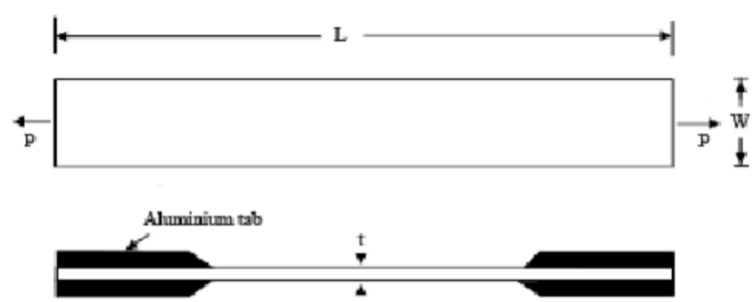

Fig. 1. The cross sectional area of an ASTM D-3039 standard test sample. L = 250mm; W = 25mm; t = 3mm [1]

interpretable stress-strain curves.

While testing a sample of the created material will give a more exact evaluation of the mechanical properties for the composite material, the data does not exist. However, the found mechanical properties for both Dragon Skin™ and stock silk can be used to give a preliminary assessment of the composites properties [3], [4].

	Elastic Modulus (E) (GPa)	($\sigma_{Ultimate}$) (MPa)	% Elongation @ rupture
Dragon Skin 10	9x10-5	3.278	1000% (10x)
Silk Brin Fibers	17.33 ±2.62	284.67± .058	13.77% ± 2.76 (1.3x)

Fig. 2. Note: the values obtained for the silk fibers are a mean value over three independent trials

From the data, the conclusions can be drawn that, while remaining capable of supporting the weight of the average person, the composite material will most likely have enough ability to deform itself and return to its original shape. This can be found definitively through analytical means by determining the point at which the material begins to deform past the boundary of elastic recovery. This can be found by utilizing the relationship between the stress and strain (1) [8].

$$\varepsilon = E\sigma \tag{1}$$

When the elastic recovery of a material results in a deformation of $\varepsilon = .02$, the corresponding stress becomes the yield stress (σ_{yield}). Once a stress-strain curve is obtained for the composite material, the value for σ_{yield} can be found by rearranging the values in (1) such that the following equations become true.

$$\varepsilon_{elastic\ recovery} = E\sigma_{yield} \tag{2}$$

$$\varepsilon_{yield} - \varepsilon_{elastic\ recovery} = 0.02 \tag{3}$$

This σ_{yield} value can then be used to directly calculate the value for the maximum force the surface can support without entering into plastic deformation. Once the σ_{yield} value has been exceeded, the composite material will no longer be able to fully recover to its original dimensions. Therefore, this still unknown yield stress value will be the defining factor in evaluating the viability of the surface for this project.

III. Evaluation of Electromagnetic Sources as Linear Drivers

A. Evaluation of Push/Pull Solenoids

For a solenoid driver, linear motion is produced by the interaction of the ferromagnetic properties of the shaft material and the magnetic field produced by the imposed current running through the helical wire. Fundamentally, solenoids are governed by two major laws from electromagnetics, Ampère-Maxwell's Law. Ampere-Maxwell's Law states that the magnitude of the magnetic field density (**B**) directly relates to the magnitude of the current and the distance from the line carrying the current [11].

$$\oint B \cdot dL = \mu_0 \left(I_{enc} + \varepsilon_0 \frac{d}{dt} \oint E \cdot \hat{n}\, ds \right) \tag{4}$$

However, in a solenoid a time-varying electric field will be a rather negligible factor, if at all. Thus the equation can further be reduced to the following.

$$\oint B \cdot dL = \mu_0 I_{enc} \tag{5}$$

Now the integral can be easily solved by integrating along a square path enclosing the cross-sectional area (shown below) of the solenoid.

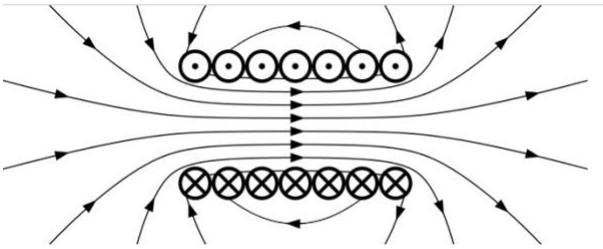

Fig. 3. The cross-sectional are of a solenoid exposing the turns and showing the direction of current flow (Top to Bottom)

The left-hand side of (5) can be broken into its parts. Each new integral is the line integral of a specific line segment on the loop.

$$\int_A^B B \cdot dL + \int_B^C B \cdot dL + \int_C^D B \cdot dL + \int_D^A B \cdot dL$$

Another simple observation can be made to reduce this integral further. Because angle between the incremental length vector and the magnetic field density vector is 90º, the dot product of the second and fourth integrals are zero. Likewise, the third integral lies outside of the solenoid, meaning that there exists such a small amount of magnetic field flux that the density (**B**) may be considered negligible. The final, reduced integral can be expressed as the following. With the substitution of the findings of (6) back into (5), an equation can be formed to find the number of turns it takes to support a given magnetic field density (B) (8).

$$\oint B \cdot dL = \int_A^B B \cdot dL \qquad (6)$$

$$if\ A \rightarrow B\ is\ the\ full\ length\ L\ then,$$

$$\oint B \cdot dL = BL\ \therefore\ BL = \mu_0 I_{enc} \qquad (7)$$

Where,

$$N = number\ of\ turns;\ \ I_{enc} = IN$$

$$BL = \mu_0 IN$$

$$N = \frac{BL}{\mu_0 I} \qquad (8)$$

By applying the design constraints of the reconfigurable surface system, an actual figure can be reached for how many turns the solenoid must be in order to meet requirements. Firstly, the 3x3 array of pins must be able to move the weight of an adult man, 870.15 [N], the necessary maximum of 7.65 [cm] (3”). Secondly, the current output cannot exceed 40 [mA] for an Arduino Uno. Finally, the assumption is made that there is no airgap between the 9.53x10-4[m] (3/8”) diameter iron rod and the solenoid. In order to evaluate the solenoids capabilities, a comparison of the maximum energy output, at ideal conditions and no factor of safety, will be assessed. The result of which can be seen in the following mathematical calculations.

$$U = (871\ [N])(.076[m]) = 66.3\ [J]$$

$$U_{solenoid} = L\pi R^2 \frac{B^2}{2\mu_0}\frac{\mu_m}{\mu_0};\ \ \mu_0 = 4\pi \cdot 10^{-7} \qquad (9)$$

$$L = .0765\ [m]; R = 9.53 \cdot 10^{-4}[\mathrm{m}];$$
$$\mu_{m,Fe} = 5000$$

After substituting the necessary energy required and all other needed variables into (9), the value for magnetic field density is as follows.

$$B = .0761\ [\frac{Wb}{m^2}] \qquad (10)$$

This found value can now be substituted back into the derived equation (8). Doing so will render a true value for the number of turns (N) the solenoid must have in order to support the necessary magnetic field (B).

$$N = \frac{(0.0761)(0.0765)}{(4\pi \cdot 10^{-7})(0.04)}$$

$$N = 96{,}083\ [turns]$$

Unfortunately, the number of turns required to sustain the necessary magnetic field density is too large to be implemented into the design. Using wiring of the correct gauge for the maximum current, the solenoid would need to be seven meters long in order to satisfy the projects requirements. It remains important to remember that the following calculations were made assuming the absolute best case for the given parameters. Although the solenoids proved to be incapable of fitting the needs of the project, this does not preclude all forms of electromagnetic drivers.

B. Theory of Linear Tubular Motors

Linear tubular motors (LTMs) create linear motion through the interaction of a stationary magnetic field and one that is produced by a movable ferrous conductor. Motion is created much the same as a synchronous motor machine seen in Fig. 4.

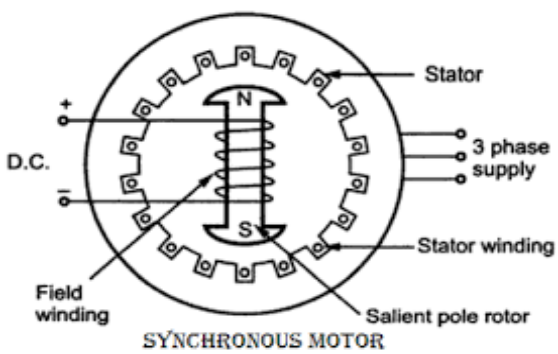

Fig. 4. Diagram of a synchronous motor and its subsequent pole structure

In a synchronous motor a rotational magnetic field is induced electrically in the stator coils via alternating currents of varying phase displacements passing through stator windings. This allows for the rotor of the machine to latch to the synchronous speed of the stator magnetic field, producing smooth rotational motion [2]. An LTM works in much the same way as its rotational counterpart except the poles of the stator are laid out linearly and the permanent magnet rotor sits in the airgap between the set of stator teeth, as shown below [10].

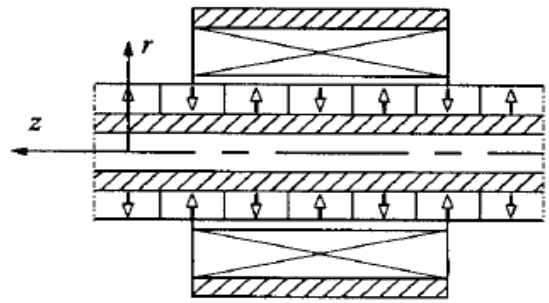

Fig. 5. The adopted setup for the LTM driver for the project. (A permanent magnet with set magnetization domains between a single tooth of a current-induced electromagnet) [7]

Much like the stator's phase windings in a synchronous motor, each of the slots of the stator combs contains a dense winding that, when energized, excites a magnetic field. Depending on the structure of the given tubular motor, we can impose the forcing on either the stator or the rotor, making their designations rather arbitrary in nature. However, in order to not change the design of the reconfigurable surface entirely, the setup should be similar to Fig. 5. This entails that the stator contain concentrated windings that allow the user to control the strength of the induced magnetic field and that the permanent magnet rotor be assembled in a quasi-Halbach array, such that only the outward facing portion of the permanent magnet be extending any magnetic flux [9]. Therefore, when the coils in the stator are then energized, the magnetic domain of the permanent magnet rotor are realigned and forced in the needed direction of motion. Significant design and experimentation will need to be resolved for the viability of an LTM drive, but the initial theory seems promising as a replacement for

the current screw drive for the individual elements in the reconfigurable surface array.

IV. Creating a System Controller Via a Graphical User Interface

A. Creating a GUI Via Open Source platform, Processing3

For the control of the reconfigurable surface system, a simple graphic user interface was created to easily articulate human input to actionable electrical control via the Arduino Uno. Allowing for independent control over each of the nine motors in the array, the GUI has been thoroughly tested and is useable both through a wired USB connection and remotely.

Created on an open-source, java-based coding platform called Processing (P3), the GUI wrapper is a rudimentary display that allows you to toggle the screw drives to either advance or retract as the user pleases (A2). It was created using a Java-based GUI library called Control_P5. Control is an easily altered script that allows the user a greater control in interface design. Its use allows for posterity to alter the aesthetic portion of the interface without altering the command code. The simple interface means that the orthotist can easily manipulate the reconfigurable surface while examining the patient, so as to best align their posture.

It should be noted that the created interface is a graphical wrapper. This means that the actual command code must be loaded to the Arduino microcontroller before launching the GUI program. Otherwise, the commands received will not be transferred to an output from the board. All code will be appended, but the general idea for the controlling code is as follows. Each individual motor has two possible actuation directions, up or down. Each of those directions were assigned a serial identifier. When the command is given to move a motor on the interface, the command is interpreted as a string of bits that triggers the current output at the desired motor control pins on the Arduino.

B. Communication between the GUI and Arduino

Like most digital logic commands, the master system (tablet or PC) communicates with the slave system (Arduino Uno board) via a serial bus connection. This means that for a given command sent from the master the slave will receive and interpret a string of bits in a given communication protocol.

For the Arduino, the protocol used is the Universal Asynchronous Receiver/Transmitter (UART) protocol. In this communication scheme, the master systems transmission signal is tied to the slaves input.

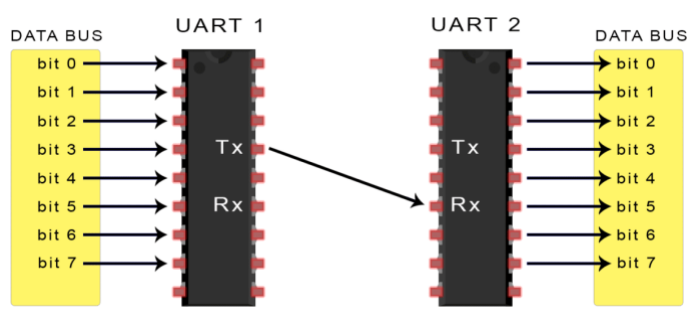

Fig. 6. Diagram of a Master-Slave UART connection

Unlike synchronous serial transmission, which requires a clock to notify the slave when to sample data, the asynchronous nature of the UART system means that, as soon as the input from the GUI is entered, the command is immediately read by and transferred to the output of the Arduino [5].

The type of UART control that was established to confirm the performance of the GUI's ability was a USB-3 connection between the master and slave system. However, a USB-3 requires a wired connection and was not desirable for the final product, so after the testing on the GUI's capability, a simple change was made to create a wireless system. The purchase of a USB-

Bluetooth dongle allowed for the host laptop to connect to the Arduino system wirelessly; thus satisfying the requirement for the project to allow the user full autonomy of movement.

V. Conclusions

Contributions both tangible and in theoretical research to the creation of a three-dimensional reconfigurable surface have been analyzed and developed. Analytical proofs for the viability of types of electromagnetic drivers, as well as the functionality of a finished Graphical User Interface, are discussed. Due to the onset of the COVID-19 outbreak, the partially completed research for the strength and viability of the composite surface material meant to be the mold for the three-dimensional reconfigurable surface is included as well. The basis for the analysis of all of these separate systems was drawn from a deep list of verified scholarly sources and educational texts. This research should serve as a basis for future student researchers to continue this work.

Acknowledgments

I would like to thank Dr. Stephen Weis for his invaluable guidance, both academic and otherwise, over the last four years and making this research experience possible. I would also like to thank the rest of my research council Dr. Robert Bittle and Dr. Eric Hanson, the faculty and staff of the College of Science of Engineering and the John V. Roach Honors College. A special thank you to Mrs. Teresa Berry for stopping me from quitting this project and for being a constant light in the lives of the Engineering students. A final and very special thank you must be shared to my family, blood and otherwise, who have pushed me to be the absolute best I can be. Mom, Dad, Scott, Erin, Andrea, Jill, Brady, Neal and Lexton, thank you for everything.

References

[1] ASTM D3039 / D3039M-17, Standard Test Method for Tensile Properties of Polymer Matrix Composite Materials, ASTM International, West Conshohocken, PA, 2017, www.astm.org

[2] S. J. Chapman, Electric machinery fundamentals. McGraw-Hill, 2012.

[3]S. Chen, M. Liu, H. Huang, L. Cheng, and H.-P. Zhao, "Mechanical properties of Bombyx mori silkworm silk fibre and its corresponding silk fibroin filament: A comparative study," Materials & Design, vol. 181, p. 108077, 2019.

[4] "Dragon Skin™ 10 FAST Product Information," Smooth, 2020. [Online]. Available: https://www.smooth-on.com/products/dragon-skin-10-fast/. [Accessed: 03-May-2020].

[5] B. Mealy and J. Mealy, Digital McLogic Design. 2012

[6]H.-A. Oh, D. Park, K.-S. Han, and T. S. Oh, "Elastic Modulus of Locally Stiffness-variant Polydimethylsiloxane Substrates for Stretchable Electronic Packaging Applications," Journal of the Microelectronics and Packaging Society, vol. 22, no. 4, pp. 91–98, 2015.

[7] B.-T. Ooi, "A generalized machine theory of the linear induction motor," IEEE Transactions on Power Apparatus and Systems, vol. PAS-92, no. 4, pp. 1252–1259, 1973.

[8] J. F. Shackelford, Introduction to materials science for engineers. Boston: Pearson, 2016.

[9] D. Trumper, W.-J. Kim, and M. Williams, "Design and analysis framework for linear permanent magnet machines," Proceedings of 1994 IEEE Industry Applications Society Annual Meeting, vol. 32, no. 2, Apr. 1996.

[10] J. Wang, W. Wang, K. Atallah, and D. Howe, "Comparative studies of linear permanent magnet motor topologies for active vehicle suspension," 2008 IEEE Vehicle Power and Propulsion Conference, 2008.

[11] W. H. Hayt and J. A. Buck, Engineering electromagnetics. New York, NY: McGraw-Hill Education, 2019.

Appendix A

GUI Associated JavaScript

```
import controlP5.*; //import controlP5 library
import processing.serial.*;

Serial port;

ControlP5 cp5; //create Control P5 object
PFont  font;


void setup(){ //same as arduino program

  size(1000,800);    //window size, (width, height)

  printArray(Serial.list());    //prints all available serial ports

  //port = new Serial(this, "/dev/cu.usbmodem1411", 9600); //NEED TO FIND OUT WHAT
SERIAL PORT TO USE

  //add button to empty window

  cp5 = new ControlP5(this);
  //font = createFont("calibri light bold", 20);

  cp5.addButton("Release")
    .setPosition (415,635)
    .setSize (200,120)
    ;
  cp5.addButton("Up_M1")  //"Up" is the name of button
    .setPosition(75,135) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
  cp5.addButton("Down_M1")  //"Up" is the name of button
    .setPosition(200,135) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
  cp5.addButton("Up M2")  //"Forward" is the name of button
    .setPosition(400,135) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
  cp5.addButton("Down M2")  //"Forward" is the name of button
    .setPosition(525,135) //x and y coordinates of upper left corner of button
```

```
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M3")  //"Forward" is the name of button
  .setPosition(725,135) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Down M3")  //"Forward" is the name of button
  .setPosition(850,135) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M4")  //"Up" is the name of button
  .setPosition(75,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Down M4")  //"Forward" is the name of button
  .setPosition(200,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M5")  //"Up" is the name of button
  .setPosition(400,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Down M5")  //"Forward" is the name of button
  .setPosition(525,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M6")  //"Up" is the name of button
  .setPosition(725,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Down M6")  //"Forward" is the name of button
  .setPosition(850,335) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M7")  //"Up" is the name of button
  .setPosition(75,535) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Down M7")  //"Forward" is the name of button
  .setPosition(200,535) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
cp5.addButton("Up M8")  //"Up" is the name of button
  .setPosition(400,535) //x and y coordinates of upper left corner of button
  .setSize(100, 80)  // (width, height)
  ;
```

```
  cp5.addButton("Down M8")  //"Forward" is the name of button
    .setPosition(525,535) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
  cp5.addButton("Up M9")  //"Up" is the name of button
    .setPosition(725,535) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
  cp5.addButton("Down M9")  //"Forward" is the name of button
    .setPosition(850,535) //x and y coordinates of upper left corner of button
    .setSize(100, 80)  // (width, height)
    ;
}

void draw(){  //same as loop in arduino

  background(77,25,121); // background color of window (r,g,b) or (0 to 255);

  //lets give title to our window
  fill(255, 255, 255);
  //textFont(font);
  text("Motor Controller for Reconfigurable Surface", 400, 20);  //(text, coordinate,coordinate)
  text("Motor 1x1", 150, 115);
  text("Motor 1x2", 475, 115);
  text("Motor 1x3", 800, 115);
  text("Motor 2x1", 150, 315);
  text("Motor 2x2", 475, 315);
  text("Motor 2x3", 800, 315);
  text("Motor 3x1", 150, 515);
  text("Motor 3x2", 475, 515);
  text("Motor 3x3", 800, 515);
  fill(0,0,0);
  text("L", 15, 375);
  text("R", 985,375);
  text("Back", 500, 650);
  text("Up", 500, 80);
}

void Up_M1(){
  port.write('u');
}
void Release(){
  port.write('r');
}
void Down_M1(){
  port.write('d');}
```

Arduino/Screw Drive Associated Control Code

```
#include <Adafruit_MotorShield.h>
#include <Wire.h>
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *Motor1x1 = AFMS.getMotor(1);

void setup() {

  Serial.begin(9600);   //start serial communication 09600 bps
  AFMS.begin(1000);
  Motor1x1->run(RELEASE);
  }

void loop() {
  if(Serial.available());{  //id data is available to read

    char val = Serial.read();

    if(val=='u'){
        Motor1x1->run(FORWARD);
        Motor1x1->setSpeed(200);
        delay(10);
    }
    if(val=='d'){
        Motor1x1->run(BACKWARD);
        Motor1x1->setSpeed(200);
        delay(10);
    }
    if(val=='r'){
      Motor1x1->run(RELEASE);
    }
   }
  }
```

GUI Image


Motor Controller for Reconfigurable Surface
Up
Motor 1x1
UP_M1
DOWN_M1
Motor 1x2
UP M2
DOWN M2
Motor 1x3
UP M3
DOWN M3
Motor 2x1
UP M4
DOWN M4
Motor 2x2
UP M5
DOWN M5
Motor 2x3
UP M6
DOWN M6
L
R
Motor 3x1
UP M7
DOWN M7
Motor 3x2
UP M8
DOWN M8
Motor 3x3
UP M9
DOWN M9
RELEASE