Rebecca Schwartz
MEM 455 Fundamentals of Robotics
Professor James Tangorra
15 March 2019

Serial Arm Redesign

The Challenge

The robot challenge is to remove rubble and rebuild in a particular orientation. The space is 8.5 by 14 inches, where half of the space is dedicated to rubble, and the other half is dedicated space for rebuild. The robot shall gather the Lego and Lego Duplo, circumvent a barrier, and place each Lego in the appropriate orientation. The taskspace shall be defined as an 8.5 by 14 by 2 inch space.

The challenge is envisioned to be completed via prismatic joints via rack gears and rotary joints. The linear motion will be completed via stepper motors, allowing for movement in the x and y directions. The rotary joints are for the arm itself: one for navigating in the z direction, one for orienting the end effector, and one for gripping. The end effector will be a 3D printed gripper, powered by a servo. As mentioned, the end effector will be able to grip and orient legos, as seen fit. The end effector will be geared so that a servo can control its opening and closing.

Motors will be controlled via an Arduino Uno. A breadboard, wires, motor drivers, and external power sources will be used to set up the hardware for success. Code will be written in the Arduino interface, which will be used to control each motor individually.

Steps to Success

1. Build robot
    a. Linears in x and y direction for base
    b. Test joints
    c. Rotary joints along x axis and to rotate end effector
    d. Test joints
    e. Attach end effector
    f. Test joints
2. 3D printing
    a. Determine what needs to be 3D printed
    b. Draw it in Creo, save as .stl
    c. Send it to Andy immediately for printing
    d. Incorporate 3D printed pieces
    e. Test
    f. Repeat steps b-e if necessary
3. Hook up motors
    a. Wire up first motor

   b. Set up external power
   c. Write Arduino code
   d. Test
   e. Repeat steps a-d, adding one motor each time until all motors incorporated
4. Attach motors to robot
   a. Test
   b. Reduce torque via gearing if necessary
   c. Test
5. Testing Phase
   a. Attempt to pick up object, orient and place (with robot)
   b. Rewrite code for easier interface if necessary
   c. Repeat steps a-b until satisfied

With a little bit of blood, sweat, time, the right tools, and support, the challenge can be faced with full force.

Critical Needs and Specifications

The following are details of critical needs and specifications, which are simplified and placed in **Table 1** below.

1.
Need: Be able to reach any space within area of play.
Specification: Prismatic joint in the x direction shall be at least 14 inches long.
Specification: Prismatic joint in the y direction shall be at least 8.5 inches long.
Specification: Workspace shall cover 100% of the task space.
Specification: Configuration should be able to reach extreme all 8 corners of the rectangular prism taskspace..
Specification: Stepper motor torque for x and y direction movement shall be less than 63 oz-in each.

2.
Need: Be able to circumvent a central barrier.
Specification: End effector must be able to reach at least 2 inches above surface of ground.
Specification: Servo motor rotating about x axis shall have torque of less than 104 oz-in.

3.
Need: Be able to control the end effector precisely.
Specification: Any movement should have an error of less than 0.5 cm and less than 10 degrees. Choose servos and stepper motors as opposed to DC motors.

4.
Need: Be able to pick up any size Lego.
Specification: Be able to pick up a Lego which is at least 1.5 inches wide.

5.
Need: Be able to support the weight of any Lego.
Specification: Be able to support up to Legos up to 0.40 kg of weight without unwanted motor or structural failure.
Specification: Servo torque for gripper should be less than 104 oz-in.

6.
Need: Be able to orient legos.
Specification: Rotary joint about y axis shall rotate the end effector via a servo motor with torque of less than 104 oz-in.
Specification: End effector should have at least 2 dof; one for gripping motion and one for orienting the gripper.

7.
Need: Be able to power all motors sufficiently.
Specification: Supply 12 V, 2 A to steppers and 6V 1.5 A to the 3 servos.

Table 1: Needs and Specifications

#	**Category**	**Need**	**Priority**	**Metric**	**Target Value**	**3R Robot Compliance**
1	Spatial	Reach any space along x direction.	1	x direction prismatic joint length	14 inches	*Does not comply; only reaches up to 7.5 inches in the x direction.*
2	Spatial	Reach any space along y direction.	1	y direction prismatic joint length	8.5 inches	*Does not comply; only reaches up to 7.5 inches in the y direction.*
3	Spatial	Be able to circumvent a central barrier and place legos above surface level (for	1	z direction movement	>=2 inches	*Does comply; can reach from 0 to 4.5 inches in the z direction when outstretched.*

		stacking).				
4	Spatial	Workspace shall embody the task space.	1	Workspace coverage of taskspace	100%	*Does not comply; workspace only covers 47% of the taskspace.*
5	Config.	Reach the following points, in inches: (0,0,0), (14,0,0), (0,8.5,0), (14,8.5,0), (0,0,2), (14,0,2), (0,8.5,2), (14,8.5,2)	2	ratio of configs reached	8/8	*Does not comply; only reaches 2/8 extreme configurations.*
6	Orientation	End effector should be able to rotate, pick up, and place Legos.	2	dof	>= 2	*Does not comply; end effector only has 1 dof.*
7	Motor	Be able to move along the x direction.	1	Torque	< 63 oz-in	*Does comply; requires only 3.3 oz-in to move along the x direction.*
8	Motor	Be able to move along the y direction.	1	Torque	< 63 oz-in	*Does comply; requires only 3.3 oz-in to move along the y direction.*
9	Motor	Be able to move end effector up and down, via a rotary joint.	2	Torque	< 104 oz-in	*Does not comply; J1 requires 114 oz-in of torque to move the end effector up and down.*
10	Motor	Be able to orient objects.	2	Torque	< 104 oz-in	*Does not comply; no rotary joint present to rotate gripper.*
11	Motor	Be able to open and close gripper.	1	Torque	< 104 oz-in	*Does comply; gripper only needs 6.7 oz-in of torque.*

12	Control	Be able to control the end effector precisely.	2	Linear/ angular	+- 0.5 cm / +- 10 deg	*Does comply; could likely control with fairly good precision.*
13	Gripper	Be able to pick up any size Lego.	3	Gripper opening	0 <= opening width <= 1.5in or more	*Does comply; gripper can open up to about 3 inches.*
14	Power	Be able to supply power to all motors sufficiently.	1	Power	12V 2A for steppers, 6V 1.5A for each servo	*Does not comply; Need 6V 1.5A per servo, not for 3 servos.*

Note that all torque values take into account the expected maximum weight of a lego, at 0.3 kg. See Appendix A for MATLAB code on calculated torques.

Shortcomings of Original 3R Serial Arm

The baseline robot was much too small to cover the workspace. Set in one spot, the arm cannot reach the entire 8.5 by 14 inch space, let alone the fact that there would be no control over orienting the pieces. Specifically, the robot could only reach as far as 7.5 inches (L2+L3), but only when the end effector is about 5 inches above the surface of the ground, well above where legos might be. With the end effector on the surface, the robot can only reach up to 6 inches (sqrt[$(L2+L3)^2$ - $(L1)^2$]), which is well below the area needed. The only strong advantage of this robot is that it has a lot of z direction motion. It can reach as high as 12 inches, or 4.5 inches when outstretched. See Figure 2 below to see the links L1, L2, and L3. Note that for the purposes of discussion, axis x in figure 2 is axis y in discussion. Axis y in figure 2 is axis -x in discussion.

The workspace of the original robot is a donut with the outer radius being 6 inches and the inner radius being 1.5 inches (apx width of arm). This workspace is not up to par with the 8.5 inch by 14 inch by 2 inch space needed, as defined by the task space. With a 6 inch radius reach on the surface, the workspace only covers 47% of the taskspace. The configuration extremes needed of the robot are not satisfied by the 3R robot. (0,0,0) and (0,0,2,) can be reached, which is only 2/8 of the extreme configurations which can be reached. The end effector only has 1 dof characterized by the opening and closing of the end effector. The end effector cannot be oriented, falling short of the 2 dof required for sufficient Lego orienting.

- *Specification 1: Does not comply; only reaches up to 7.5 inches in the x direction.*
- *Specification 2: Does not comply; only reaches up to 7.5 inches in the y direction.*

- *Specification 3: Does comply; can reach from 0 to 4.5 inches in the z direction when outstretched.*
- *Specification 4: Does not comply; workspace only covers 47% of the taskspace.*
- *Specification 5: Does not comply; only reaches 2/8 extreme configurations.*
- *Specification 6: Does not comply; end effector only has 1 dof.*

As for moving the x and y directions, the original arm does that, but only in the rotary sense. The torque required to move along the x and y direction under challenge conditions (servo at each rotary joint, holding 0.3kg Lego in end effector) is 3.3 oz-in (rotation at J1). The robot may move up and down via rotary joints. At J1, the torque required to move along the z direction is 114 oz-in. At J2, the torque required to move along the z direction is 47 oz-in. The original robot cannot orient Legos as it does not have the capability to rotate the gripper. With a gripper, only 6.7 oz-in of torque is needed. See Appendix B for MATLAb code on calculated torques.

- *Specification 7: Does comply; requires only 3.3 oz-in to move along the x direction.*
- *Specification 8: Does comply; requires only 3.3 oz-in to move along the y direction.*
- *Specification 9: Does not comply; J1 requires 114 oz-in of torque to move end effector up and down.*
- *Specification 10: Does not comply; no rotary joint present to rotate gripper.*
- *Specification 11: Does comply; gripper only needs 6.7 oz-in of torque.*

If hooked up to an Arduino, it can be assumed that control would be fairly precise. The gripper would be able to grip any Lego theoretically, although it might have to grip the long side of the Lego. This is due to lack of orienting abilities. With the intense need from the servo at J1, 6V 1.5A is likely not enough power to power all servos.

- *Specification 12: Does comply; could likely control with fairly good precision.*
- *Specification 13: Does comply; gripper can open up to about 3 inches.*
- *Specification 14: Does not comply; Need 6V 1.5A per servo, not for 3 servos.*

The 3R robot has 3 degrees of freedom, but all are revolute joints, making the robot fairly limited in the x and y directions, where the most amount of mobility is necessary. Below shows details constraints of the original design.

A side view of the 3R serial arm is shown below in **Figure 1**.

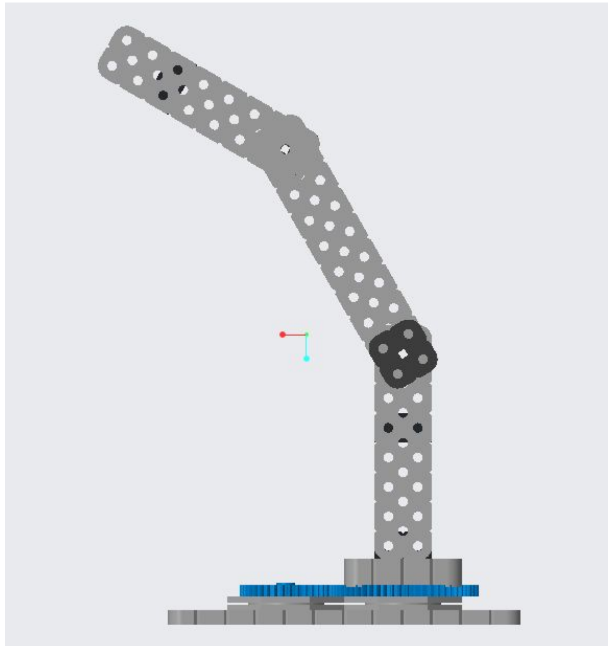

Figure 1: Side View of 3R Arm

Constraint/ Configuration Equations of End Effector

Note that (X1,Y1,Z1) are at J1 and are (0,0,0).
Note that (X4,Y4,Z4) is where the end effector would be.

L() is to denote the length of something. Length shall be in meters.
L(L1) = 0.1143 m
L(L2) = 0.1016 m
L(L3) = 0.0889 m

Constraints are to be defined so that G(c) = 0.

Assume that in the current state of the 3R arm (as shown in **Figure 2** below), that Ang1 = 0.

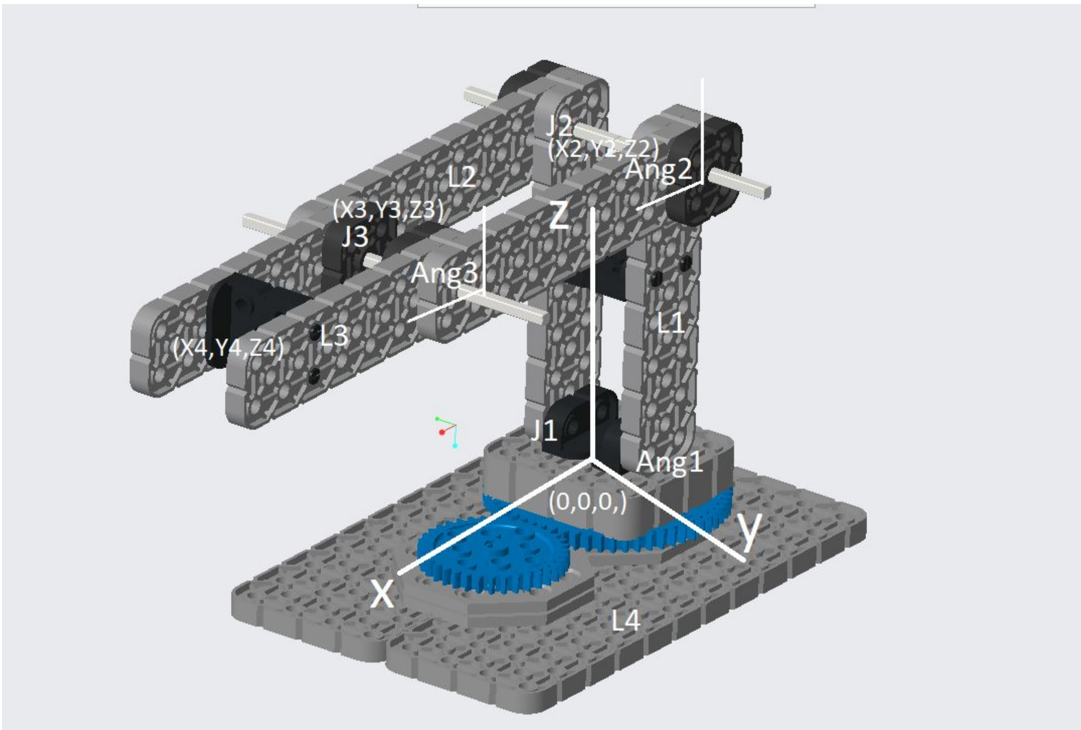

Figure 2: 3R Arm Configuration for Constraint Equations

Ang2 is measured against the z axis.
Ang3 is measured against the z axis.
So when the robot arm is extended to its maximum height (in z direction), then Ang 2 and Ang3 are both 0 degrees.

Working

With this simplified definition of Ang2 and Ang3, the configuration equations for the end effector become much simpler (as opposed to my previous configuration equations).

X4 = (L(L3)*sin(Ang3) + L(L2)*sin(Ang2)) * cos(Ang1)
Constraint 1: g1(c) = X4 - [(L(L3)*sin(Ang3) + L(L2)*sin(Ang2))] * cos(Ang1)]= 0

Y4 = (L(L3)*sin(Ang3) + L(L2)*sin(Ang2)) * sin(Ang1)
Constraint 2: g2(c) = Y4 - [(L(L3)*sin(Ang3) + L(L2)*sin(Ang2))] * sin(Ang1)] = 0

Z4 = L(L3)*cos(Ang3) + L(L2)*cos(Ang2)
Constraint 3: g3(c) = Z4 - [L(L3)*cos(Ang3) + L(L2)*cos(Ang2)] = 0

The main pitfall of this design is that the workspace is not up to par. If the robot cannot reach the necessary points, then it is totally useless.

Space Jacobian

This describes the Jacobian as referenced by the space frame, which is at (X4,Y4,Z4). Note that the first column corresponds to J1, second column to J2, third to J3.
The Jacobian given angles of 0 deg for all joints gives the following:

```
Jb0 =

     0     0     0
     0     1     1
     1     0     0
     0     0     0
     0     0     0
     0     0     0
```

The Jacobian given angles of 90 deg for all joints gives the following:

```
Js90 =

         0   -0.8940   -0.8940
         0   -0.4481   -0.4481
    1.0000         0         0
         0         0         0
         0         0         0
         0         0         0
```

The Jacobian for a typical pick-up or place might require J1 to be 45 deg, J2 to be 90 deg, and J3 to be 45 deg. The Jacobian turns out to be:

```
Jswork =

         0   -0.8509   -0.8509
         0    0.5253    0.5253
    1.0000         0         0
         0         0         0
         0         0         0
         0         0         0
```

See Appendix C for the MATLAB code for the space Jacobian for the 3R robot.

Wrench

Wrench of end effector related to other joints, values are in Oz-in and Oz when loaded with 0.2kg at the end effector, and J2 is 90 degrees.

$F4 = [Adj\ Ta4]^T\ Fa$

Related to Joint 1:

```
F41 =

         0
         0
    5.3950
         0
         0
         0
```

Related to Joint 2:

```
F42 =

         0
    5.3950
         0
         0
         0
         0
```

Related to Joint 3:

```
F43 =

         0
    2.5176
         0
         0
         0
         0
```

See Appendix C.1 for the MATLAB code.

Twist

Twist of of other joints related to end effector in ang/s

V = Stheta_dot

Joint 1:

```
V1 =

         0
         0
         0
   -0.1905
         0
   -0.1143
```

Joint 2:

```
V2 =

         0
         0
         0
   -0.1905
         0
         0
```

Joint 3:

```
V3 =

         0
         0
         0
   -0.0889
         0
         0
```

See MATLAB code in Appendix C.2.

Inverse Kinematics

To achieve the end effector being 0.227 meters along the x direction and zero in y and z, the angles suggested to give the robot are

J1 = 0

J2 = 0.875 pi

J3 = 0.625 pi

See Appendix D for the MATLAB code.

Robot Redesign

The biggest issue with the original robot is that its workspace does not cover the task space (specifications 1, 2, 4, 5) and the required torque is too large for one of the joints (specification 9). Building a serial arm robot with more links could solve the workspace issue, but the torque required would also increase. This has been carefully considered through the redesign process.

To overcome the issue, it has been decided to rely on prismatic joints to cover the task space. This way, an arm's outstretched length can be minimized, therefore minimizing torque necessary to complete the challenge.

The end effector for the redesign shall be 3D printed and actuated via a servo. Unlike the 3R robot, this end effector will be able to rotate, and thus orient itself to pick up Legos with ease.

The primary navigation will be completed by the stepper motors, allowing movement along the x and y directions. The main arm rotary joint will be used to lift and lower Legos. The orienting rotary joint will be used to orient the end effector, thus orienting Legos. The gripper is defined by a rotary joint which will work to open and close the end effector.

The redesigned robot meets almost all specifications.

The robot is able to reach every point on the 14 by 8.5 inch space (specifications 1 and 2) and is capable of moving more than 2 inches above the surface of the space (specification 3, 4, 5). The end effector can grip and orient legos; it has 2 dof (specification 6).

Joint torque requirements are computed assuming that the lego it is holding is 0.3 kg. See Appendix A for MATLAB code for finding torques required. The joint which allow movement in the x direction requires 1.97 oz-in of torque, which is well below the 63 oz-in of torque provided by the stepper (specification 7). The joint which allow movement in the y direction requires 1.24 oz-in of torque, which is well below the 63 oz-in of torque provided by the stepper (specification 8). The main arm joint requires 97.4 oz-in of torque, which is below the 104 oz-in of torque provided by the servo (specification 9). The orienting joint requires 10.7 oz-in of torque, which is below the 104 oz-in of torque provided by the servo (specification 10). The gripper joint requires 6.67 oz-in of torque, which is well below the 104 oz-in of torque provided by the servo (specification 11).

Control is at least as fine as +- 0.5cm and +- 10 deg (specification 12). The gripper can be actuated into the close position and open position, which exceeds 1.5 inches of grip width (specification 13).

The redesigned robot is not supplied with power sufficiently (specification 14). The stepper motors are sufficiently powered with 12V 2A total, but the servos need about two times as much power as is supplied to support the torque required for the revolute joints. Each servo maxes out (gives a torque of 104 oz-in) when supplied with 6V and 1.5A, as much as is supplied by 4 AA batteries in series. However, twice the amperage is required to achieve the torques required. In the robot, this is characterized by slow response time and struggling servomotors. This goes to show how much torque is necessary, even with a very small serial arm.

Engineering Analysis of Redesign

The redesigned robot meets thirteen out of fourteen specifications. To achieve the power specification, a wall socket is likely needed. It might also be worth trying to power the steppers with the battery pack and the servos with the cord. This might work considering that the torque required of the steppers is so low compared to the maximum capability of each stepper. The cord allows 12V and 2A of power, which could possibly be enough for the servos. However, too much voltage might deem the servos useless too.

The new robot has 5 degrees of freedom; one degree of freedom per independent joint. Below shows the constraints/ configuration of the new design.

An approximate side view of the redesigned robot, shown in **Figure 3**.

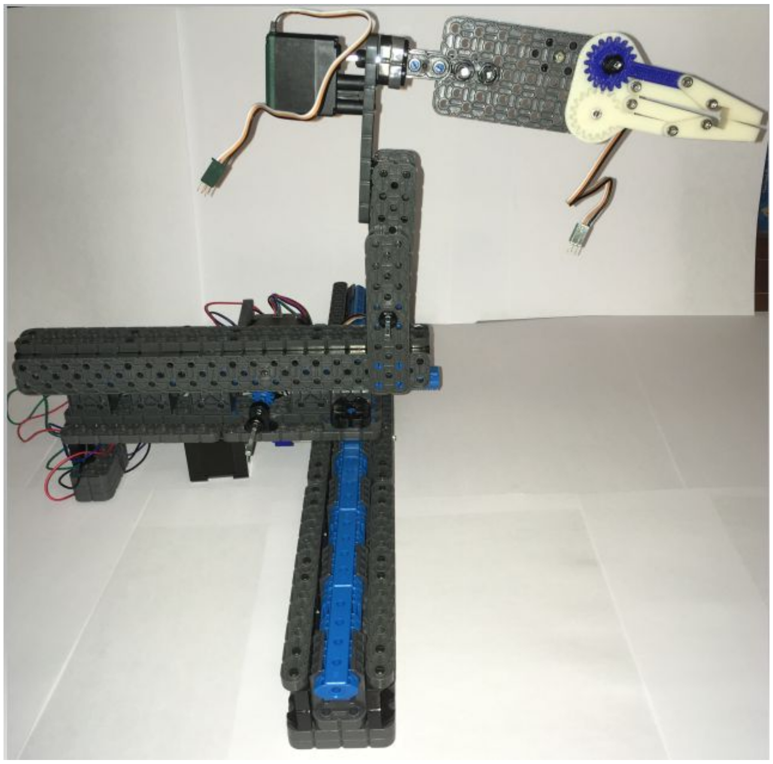

Figure 3: Side View of Redesigned Robot

An approximate side view of the redesigned robot, shown in **Figure 4**.

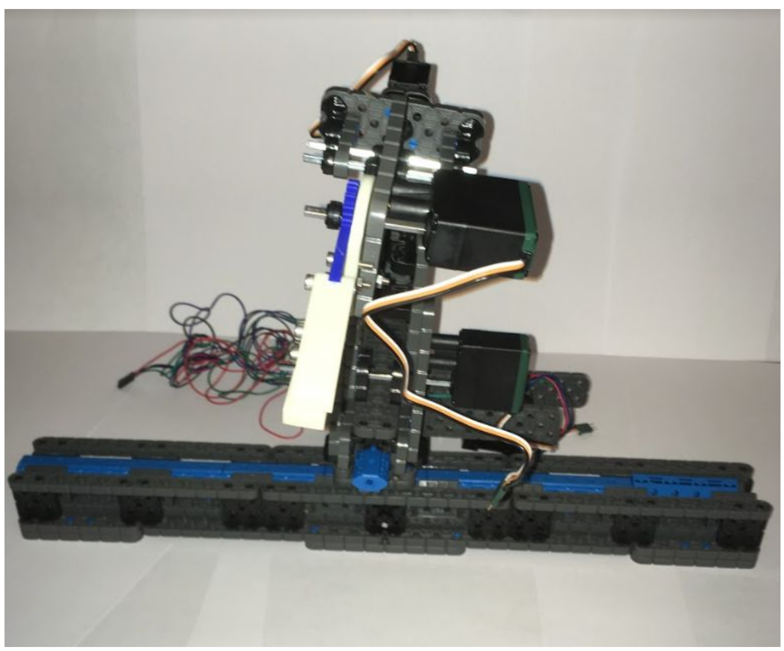

Figure 4: Side View of Redesigned Robot

Constraint/ Configuration Equations of End Effector

Note that (X1,Y1,Z1) are at J1 and are (0,0,0).
Note that (X6,Y6,Z6) is the tip of the end effector.

L() is to denote the length of something. Length shall be in meters.
L(J1) = -0.18 min; +0.18 max
W(L1) = 0.04
L(J2) = -0.08 min; +0.22 max
W(L2) = 0.065
L(L3) = 0.02
L(L4) = 0.135
L(L5) = 0.12
L(L6) = 0.08

Ang1, 2, 3, 4, 5 are at joints 1, 2, 3, 4, 5 respectively.

Constraints are to be defined so that G(c) = 0.

Assume that in the current state of the 3R arm (as shown in **Figure 5** below), that L(J1) and L(J2) are at the zero position. Ang 4 and 5 are also at the zero position.

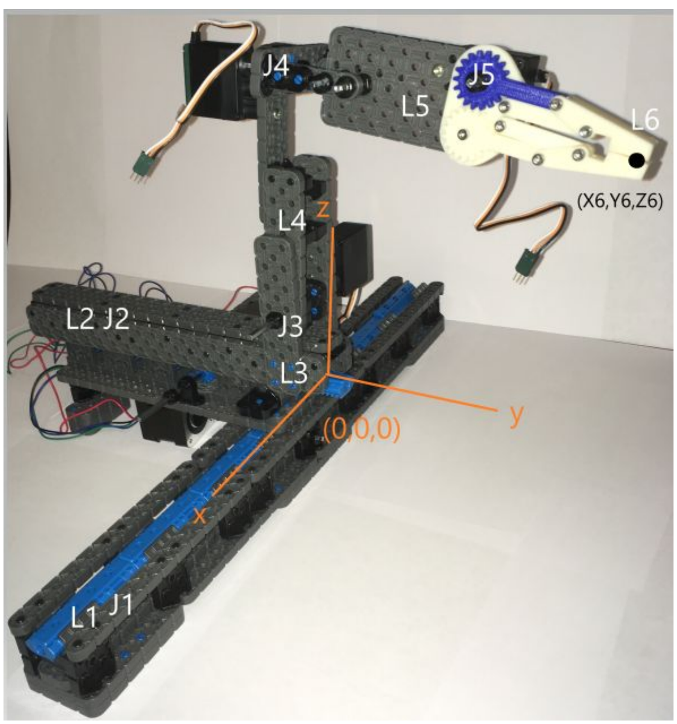

Figure 5: Robot Configuration for Constraint Equations

Working

X6 = L(J1)
Constraint 1: g1(c) = X6 - L(J1) = 0

Y6 = L(J2) + L(L4)*-sin(Ang3) + (L(L5) + L(L6))*-cos(Ang3) - 0.08 (for gap)
Constraint 2: g2(c) = Y6 - [-0.02 + L(J2) + L(L4)*-sin(Ang3) + (L(L5) + L(L6))*cos(Ang3)] = 0

Z6 = L(L1) + L(L2) + L(L3) + L(L4)*-cos(Ang3) + (L(L5) + L(L6))*sin(Ang3)
Constraint 3: g3(c) = Z6 - [L(L1) + L(L2) + L(L3) + L(L4)*-cos(Ang3) + (L(L5) + L(L6))*sin(Ang3)] = 0

Every corner of the taskspace can be reached. For example, the left far corner and the right near corner can be reached given the following angles. Assume a 8 cm gap between taskspace and edge of robot along L1.

Left far corner: (X6,Y6,Z6) = (-0.18, 0.30, 0)
L(J1) = -0.18 m
L(J2) = +0.22
Ang3 = -80 deg
Ang4 = anything
Ang5 = anything

Test with Constraints:
X6 = -0.18
Y6 = 0.22
Z6 = -0.04 = 0
IT WORKS

Right near corner: (X6,Y6,Z6) = (+0.18, 0.08, 0)
L(J1) = +0.18 m
L(J2) = -0.08
Ang3 = -80 deg
Ang4 = anything
Ang5 = anything

Test with Constraints:
X6 = +0.18

Y6 = 0.08
Z6 = -0.04 = 0
IT WORKS

Workspace
The robot's workspace is defined by a 0.36 (x) by 0.30 (y) by 0.26 m (z) general space. **Figure 6** below shows a cross section of the workspace. The black line represents the border of the space, not the whole space.

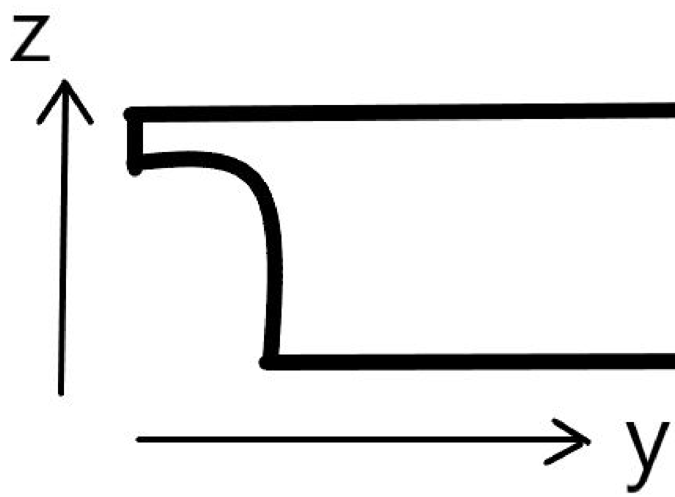

Figure 6: Cross Section of Robot Workspace

Note that when looking at the workspace in the xz plane, the workspace would look like a rectangle.

Space Jacobian
This describes the Jacobian as referenced by the space frame, which is at (X6,Y6,Z6).
The Jacobian given angles of 0 deg for all joints gives the following:

```
Js0 =

     0     0     1     0     1
     0     0     0     1     0
     0     0     0     0     0
     1     0     0     0     0
     0     1     0     0     0
     0     0     0     0     0
```

The Jacobian given angles of 90 deg for all revolute joints and 1 for all prismatic joints:

```
Js90 =

         0         0    1.0000         0   -0.4481
         0         0         0   -0.4481    0.7992
         0         0         0    0.8940    0.4006
    1.0000         0         0    0.8940    0.4006
         0    1.0000         0   -0.8940   -0.4006
         0         0   -1.0000   -0.4481    1.2473
```

The Jacobian for a typical pick-up or place might require 0 for all angles except -80 for Ang3.

```
Jswork =

         0         0    1.0000         0    1.0000
         0         0         0   -0.1104         0
         0         0         0    0.9939         0
    1.0000         0         0         0         0
         0    1.0000         0         0         0
         0         0         0         0         0
```

See Appendix E for the MATLAB code for the body Jacobian for the redesigned robot.

Wrench

Wrench of end effector related to other joints, values are in Oz-in and Oz.

F6 = $[\text{Adj Ta6}]^T$ Fa

Related to Joint 1:

```
F61 =

         0
   -0.0036
    0.0028
    0.0139
         0
         0
```

Related to Joint 2:

```
F62 =

    0.0019
         0
         0
         0
    0.0088
         0
```

Related to Joint 3:

```
F63 =

    0.6878
         0
         0
         0
         0
         0
```

Related to Joint 4:

```
F64 =

         0
    0.0756
         0
         0
         0
         0
```

Related to Joint 5:

```
F65 =

         0
    0.0471
         0
         0
         0
         0
```

See Appendix E.1 for the MATLAB code.

Inverse Kinematics

To achieve the end effector being at (0.05,0.12,0), the following angles are suggested.

```
thetalist =

    0.0500
   -0.0800
   90.0000
   -0.0000
         0
```

meters for 1,2; degrees for 3-5

See Appendix F for the MATLAB code.

Arduino Code

See Appendix G for Arduino Code.

Recognition

Thank you Professor Tangorra for teaching me. This will surely help me along in my academic and professional careers.

Thank you Andy Drago for being a fantastic TA. From moral support to 3D printing to troubleshooting, Andy has helped me make my robot what it is today.

Thank you Liam Walsh, for the wonderful Arduino code with pointers for tokenizing code.

And of course, thank you to my family and friends for the moral support.

Appendix A: MATLAB code for calculating needed torques for Redesign Robot

```
clear all
clc

g = 9.81;
rpinion = 0.00635;

%weight in kg
mgripper = 0.02;
mservo = 0.045;
mstepper = 0.310;
mlego = 0.3; %expected max mass of lego in kg

%x direc
mvexx = 0.2; %apx weight of vex load
mx = mgripper + 3*mservo + mstepper + mlego + mvexx;
u = 0.2; %friction coef
a = 0.3; %acc of motor
Ffric = 0.01;
Fx_N = mx*g*u + mx*a + Ffric;
Tx_Nm = Fx_N*rpinion;
Tx_Ozin = Tx_Nm*141.612

%y direc
mvexy = 0.15; %apx weight of vex load
my = mgripper + 3*mservo + mlego + mvexy;
u = 0.2; %friction coef
a = 0.3; %acc of motor
Ffric = 0.01;
Fy_N = my*g*u + my*a + Ffric;
Ty_Nm = Fy_N*rpinion;
Ty_Ozin = Ty_Nm*141.612


%arm rot about x
mvexrotx = 0.05; %apx weight of vex load
mrotx = mgripper + 2*mservo + mlego + mvexrotx;
rrotx = 0.1524; %rad to center of mass
Frotx_N = mrotx*g;
```

```
Trotx_Nm = Frotx_N*rrotx;
Trotx_Ozin = Trotx_Nm*141.612

%arm rot about y
mvexroty = 0.02; %apx weight of vex load
mroty = mgripper + 1*mservo + mlego + mvexroty;
rroty = 0.02; %rad to center of mass
Froty_N = mroty*g;
Troty_Nm = Froty_N*rrotx;
Troty_Ozin = Troty_Nm*141.612

%gripper
mgrip = mgripper + mlego;
rrotg = 0.015; %rad of gear
Fgrip_N = mgrip*g;
Tgrip_Nm = Fgrip_N*rrotg;
Tgrip_Ozin = Tgrip_Nm*141.612
```

MATLAB's Output, Torques in oz-in

```
Tx_Ozin = 1.9719
Ty_Ozin = 1.2396
Trotx_Ozin = 97.3894
Troty_Ozin = 10.700
Tgrip_Ozin = 6.6682
```

Appendix B: MATLAB code for calculating needed torques for Original Robot

```
clear all
clc

g = 9.81;
rgear = 0.020;

%weight in kg
mgripper = 0.02;
mservo = 0.045;
mlego = 0.3;
```

```
%orig arm x and y direc
mvexx = 0.05; %apx weight of vex load
mx = mgripper + 3*mservo + mlego + mvexx;
u = 0.2; %friction coef
a = 0.3; %acc of motor
Ffric = 0.01;
Fx_N = mx*g*u + mx*a + Ffric;
Tx_Nm = Fx_N*rgear;
Txy_Ozin = Tx_Nm*141.612

%orig arm rot about x1
mvexrotx1 = 0.02; %apx weight of vex load
mrotx1 = mgripper + 2*mservo + mlego + mvexrotx1;
rrotx1 = 0.1905; %rad to center of mass
Frotx1_N = mrotx1*g;
Trotx1_Nm = Frotx1_N*rrotx1;
Trotx1_Ozin = Trotx1_Nm*141.612

%arm rot about x2
mvexrotx2 = 0.015; %apx weight of vex load
mrotx2 = mgripper + 1*mservo + mlego + mvexrotx2;
rrotx2 = 0.0889; %rad to center of mass
Frotx2_N = mrotx2*g;
Trotx2_Nm = Frotx2_N*rrotx2;
Trotx2_Ozin = Trotx2_Nm*141.612

%gripper
mgrip = mgripper + mlego;
rrotg = 0.015; %rad of gear
Fgrip_N = mgrip*g;
Tgrip_Nm = Fgrip_N*rrotg;
Tgrip_Ozin = Tgrip_Nm*141.612
```

MATLAB's Output, Torques in oz-in

```
Txy_Ozin = 3.2636
Trotx1_Ozin = 113.7974
Trotx2_Ozin = 46.9304
Tgrip_Ozin = 6.6682
```

Appendix C: MATLAB code for calculating Jacobian of 3R Robot

```
clear all
clc

%space jacobian for 3R robot

Slist = [[0;0;1;0;0;0],[0;1;0;0;0;0],[0;1;0;0;0;0]];
thetalist0 = [0;0;0];
thetalist90 = [90;90;90];
thetalistwork = [45;90;45];

Js0 = JacobianSpace(Slist,thetalist0)
Js90 = JacobianSpace(Slist,thetalist90)
Jswork = JacobianSpace(Slist,thetalistwork)
```

Appendix C.1: MATLAB code for calculating Wrench of 3R Robot

```
clear all
clc

%wrench of end effector related to other joints
L1 = 0.1143;
L2 = 0.1016;
L3 = 0.0889;

M1 = 0.2*(L3+L2)*141.6;
M2 = 0.2*(L3+L2)*141.6;
M3 = 0.2*L3*141.6;

%joint 1
T14 = [1,0,0,L2+L3;0,1,0,0;0,0,1,L1;0,0,0,1];
F1 = [0;0;M1; 0; 0;0];
F41 = (Adjoint(T14))'*F1

%joint 2
T24 = [1,0,0,L2+L3;0,1,0,0;0,0,1,0;0,0,0,1];
F2 = [0;M2;0; 0; 0;0];
```

```
F42 = (Adjoint(T24))'*F2

%joint 3
T34 = [1,0,0,L3;0,1,0,0;0,0,1,0;0,0,0,1];
F3 = [0;M3;0; 0; 0;0];
F43 = (Adjoint(T34))'*F3
```

Appendix C.2: MATLAB code for calculating Twist of 3R Robot

```
clear all
clc

%twist related to end effector, J4
L1 = 0.1143;
L2 = 0.1016;
L3 = 0.0889;

%joint 1
T41 = TransInv([1,0,0,L2+L3;0,1,0,0;0,0,1,L1;0,0,0,1]);
V1 = se3ToVec(MatrixLog6(T41))

%joint 2
T42 = TransInv([1,0,0,L2+L3;0,1,0,0;0,0,1,0;0,0,0,1]);
V2 = se3ToVec(MatrixLog6(T42))

%joint 3
T43 = TransInv([1,0,0,L3;0,1,0,0;0,0,1,0;0,0,0,1]);
V3 = se3ToVec(MatrixLog6(T43))
```

Appendix D: MATLAB code for using Inverse Kinematics on the 3R Robot

```
%use IKinSpace to find joint angles

clear all
close all
clc
```

```
Slist = [[0;0;1;0;0;0],[0;1;0;0;0;0],[0;1;0;0;0;0]];
M = [[1, 0, 0, 0.1905]; [0, 1, 0, 0]; [0, 0, 1, 0.1143]; [0, 0, 0, 1]];
T = [[0, 0, -1, -0.227]; [0, 1, 0, 0]; [1, 0, 0, 0]; [0, 0, 0, 1]];
thetalist0 = [0; pi/2; pi/4]; %rad
eomg = 0.001; %rad
ev = 0.001; %0.1mm
[thetalist, success] = IKinBody(Slist,M,T,thetalist0,eomg,ev)

thetalist_pi_rad = thetalist/pi
```

Appendix E: MATLAB code for calculating Jacobian of Redesigned Robot

```
clear all
clc

%space jacobian for redesigned robot

Slist = [[0;0;0;1;0;0],[0;0;0;0;1;0],[1;0;0;0;0;0],[0;1;0;0;0;0],[1;0;0;0;0;0]];
thetalist0 = [0;0;0;0;0];
thetalist90 = [1;1;90;90;90];
thetalistwork = [0;0;-80;0;0];

Js0 = JacobianSpace(Slist,thetalist0)
Js90 = JacobianSpace(Slist,thetalist90)
Jswork = JacobianSpace(Slist,thetalistwork)
```

Appendix E.1: MATLAB code for calculating wrench of Redesigned robot

```
clc
%wrench of end effector related to other joints
W1 = 0.04;
W2 = 0.065;
L3 = 0.02;
L4 = 0.135;
L5 = 0.12;
L6 = 0.08;
M1 = 1.97;
M2 = 1.24;
M3 = 97.4;
```

```
M4 = 10.7;
M5 = 6.67;
%convert to Nm
M1 = M1/141.6119;
M2 = M2/141.6119;
M3 = M3/141.6119;
M4 = M4/141.6119;
M5 = M5/141.6119;
%joint 1
T16 = [1,0,0,0;0,1,0,L5+L6;0,0,1,W1+W2+L3+L4;0,0,0,1];
F1 = [0;0;0; M1; 0;0];
F61 = (Adjoint(T16))'*F1
%joint 2
T26 = [1,0,0,0;0,1,0,L5+L6;0,0,1,W2+L3+L4;0,0,0,1];
F2 = [0;0;0; 0; M2;0];
F62 = (Adjoint(T26))'*F2
%joint 3
T36 = [1,0,0,0;0,1,0,L5+L6;0,0,1,L4;0,0,0,1];
F3 = [M3;0;0; 0; 0;0];
F63 = (Adjoint(T36))'*F3

%joint 4
T46 = [1,0,0,0;0,1,0,L5+L6;0,0,1,0;0,0,0,1];
F4 = [0;M4;0; 0; 0;0];
F64 = (Adjoint(T46))'*F4
%joint 5
T56 = [1,0,0,0;0,1,0,L6;0,0,1,0;0,0,0,1];
F5 = [0;M5;0; 0; 0;0];
F65 = (Adjoint(T56))'*F5
```

Appendix F: MATLAB code for using Inverse Kinematics on the Redesigned Robot

```
%use IKinSpace to find joint variables

clear all
close all
clc

Slist = [[0;0;0;1;0;0],[0;0;0;0;1;0],[1;0;0;0;0;0],[0;1;0;0;0;0],[1;0;0;0;0;0]];
M = [[1, 0, 0, 0]; [0, 1, 0, 0.2]; [0, 0, 1, 0.26]; [0, 0, 0, 1]];
```

```
T = [[1, 0, 0, 0.05]; [0, 0, -1, 0.12]; [0, 1, 0, 0]; [0, 0, 0, 1]];
thetalist0 = [0; 0; pi/2; 0 ;0]; %rad
eomg = 0.001; %rad
ev = 0.001; %0.1mm
[thetalist, success] = IKinBody(Slist,M,T,thetalist0,eomg,ev)
[thetalist, success] = IKinBody(Slist,M,T,thetalist0,eomg,ev)
thetalist(3:5) = thetalist(3:5)/pi
thetalist(3:5)= thetalist_pi_rad(3:5)*180 % in degrees for revolute
```

Appendix G: Arduino Code

```
#include <Stepper.h> //import stepper library
#include <Servo.h>

const float Pitch_Radius = 10; //Pitch radius of pinion gear in mm
const float pi = 3.14159265358979323846264338327950; //Value of pi
const float y = 360/(2* pi* Pitch_Radius); //combine values to reduce calcs in loop

//Arc_length = 2*pi*R*(C/360)

//Define pins on Arduino
  //Motor1
int in1Pin = 10;
int in2Pin = 11;
int in3Pin = 12;
int in4Pin = 13;

  //Motor2
int in5Pin = 6;
int in6Pin = 7;
int in7Pin = 8;
int in8Pin = 9;

int servoPin1 = 5;
Servo servo1;
int angle1 = 90;

int servoPin2 = 4;
```

```
Servo servo2;
int angle2 = 0;

int servoPin3 = 3;
Servo servo3;
int angle3= 0;


//Change this to the number of steps on your motor anything from 200-1600
  #define STEPS1 200
  #define STEPS2 200
  #define STEPS3 200

//Motor setup to communicate with H-bridge
  Stepper motor1(STEPS1, in1Pin, in2Pin, in3Pin, in4Pin);
  Stepper motor2(STEPS2, in5Pin, in6Pin, in7Pin, in8Pin);
  //Stepper motor3(STEPS3, in9Pin, in10Pin, in11Pin, in12Pin);

void setup() //begin loop
{
  //H-Bridge to motor communication
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);
  pinMode(in5Pin, OUTPUT);
  pinMode(in6Pin, OUTPUT);
  pinMode(in7Pin, OUTPUT);
  pinMode(in8Pin, OUTPUT);

  while (!Serial);

  Serial.begin(9600); //communication speed between computer and arduino
  motor1.setSpeed(100); //set speed of motor usually between 20-100
  motor2.setSpeed(100);

  servo1.attach(servoPin1);
  servo2.attach(servoPin2);
  servo3.attach(servoPin3);
```

```
  delay(500);
  servo1.write(angle1);
  servo2.write(angle2);
  servo3.write(angle3);
  delay(500);
}

void loop()
{
  if (Serial.available()) //waits for an input on the serial monitor
  {
    int bufferSize = 20;
    char abc[bufferSize];
    Serial.readBytes(abc, bufferSize);


   float Arc_length1 = 0.0;
   float Arc_length2 = 0.0;

   char *strtokIndx;
   strtokIndx = strtok(abc,",");
   Arc_length1 = atof(strtokIndx);
   strtokIndx = strtok(NULL,",");
   Arc_length2 = atof(strtokIndx);
   strtokIndx = strtok(NULL,",");
   angle1 = atof(strtokIndx);
   strtokIndx = strtok(NULL,",");
   angle2 = atof(strtokIndx);
   strtokIndx = strtok(NULL,",");
   angle3 = atof(strtokIndx);

    //the divisor is the step angle based on the STEPS in one full rotation
    const float StepsRequired1 = (Arc_length1*y)/1.8;
    const float StepsRequired2 = (Arc_length2*y)/1.8;

    //tells the motors to step based on the previous equation
    motor1.step(StepsRequired1);
    motor2.step(StepsRequired2);
```

```
    servo1.write(angle1);
    servo2.write(angle2);
    servo3.write(angle3);
    delay(100);
  }
}
```

Appendix G: Needs and Specs related to Robot Compliance

Table 1: Needs and Specifications						
#	**Category**	**Need**	**Priority**	**Metric**	**Target Value**	**Pick and Place Robot Compliance**
1	Spatial	Reach any space along x direction.	1	x direction prismatic joint length	14 inches	***Does comply; reaches at least 14 inches in the x direction***
2	Spatial	Reach any space along y direction.	1	y direction prismatic joint length	8.5 inches	***Does comply; reaches at least 8.5 inches in the y direction***
3	Spatial	Be able to circumvent a central barrier and place legos above surface level (for stacking).	1	z direction movement	>=2 inches	*Does comply; can reach from 0 to 4.5 inches in the z direction when outstretched.*
4	Spatial	Workspace shall embody the task space.	1	Workspace coverage of taskspace	100%	***Does comply; reaches 100% of the task space***
5	Config.	Reach the following points, in inches: (0,0,0), (14,0,0), (0,8.5,0), (14,8.5,0), (0,0,2), (14,0,2), (0,8.5,2), (14,8.5,2)	2	ratio of configs reached	8/8	***Does comply; reaches all 8 corners of the task space***

6	Orientation	End effector should be able to rotate, pick up, and place Legos.	2	dof	>= 2	***Does comply; end effector able to pick, place, and rotate objects***
7	Motor	Be able to move along the x direction.	1	Torque	< 63 oz-in	*Does comply; requires only **1.97 oz-in (down from 3.3 oz-in)** to move along the x direction.*
8	Motor	Be able to move along the y direction.	1	Torque	< 63 oz-in	*Does comply; requires only **1.24 oz-in (down from 3.3 oz-in)** to move along the y direction.*
9	Motor	Be able to move end effector up and down, via a rotary joint.	2	Torque	< 104 oz-in	***Does comply; requires only 97.4 oz in to move the end effector up and down via a rotary joint***
10	Motor	Be able to orient legos.	2	Torque	< 104 oz-in	***Does comply; only requires 10.7 oz in of rotary torque to rotate objects***
11	Motor	Be able to open and close gripper.	1	Torque	< 104 oz-in	*Does comply; gripper only needs 6.7 oz-in of torque.*
12	Control	Be able to control the end effector precisely.	2	Linear/ angular	+- 0.5 cm / +- 10 deg	*Does comply; could likely control with fairly good precision.*
13	Gripper	Be able to pick up any size Lego.	3	Gripper opening	0 <= opening width <= 1.5in or more	*Does comply; gripper can open up to about 3 inches.*

14	Power	Be able to supply power to all motors sufficiently.	1	Power	12V 2A for steppers, 6V 1.5A for each servo	***Does not comply; has 12V 2A for steppers, but only has 6V 1.5A for all 3 servos***