Parallel 3RRR Robot Classroom Guide

Advanced Robotics and Mechanism Applications Lab
Vanderbilt University

Garrison Johnston, Rashid Yasin, Long Wang, Nima Sarli, and Nabil Simaan

July 7, 2016

Contents

1 Introduction

According to the Robot Institute of America, a robot is a "reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks." The use of the word *robot* first arose from the 1921 stageplay *R.U.R. (Rossum's Universal Robots)* written by Czech playwright Karel Čapek [19]. Now, less than 100 years later, robots are an integral part of the modern economy. They are used in industrial factories, in the military, in space and ocean exploration, in medicine, and in consumer products to name only a few applications. Into the future, robots will continue to become a larger and larger part of the human experience, making human lives safer and easier.

Robot manipulators are typically subdivided into two main categories: serial and parallel. In a serial robot, the position of the end effector, the part of the robot that interacts with the environment, is determined by a series of links making an anthropomorphic (arm-like) structure [17]. Figure 1 shows a few examples of serial manipulators.

Figure 1: Serial robot examples (anthropomorphic [23], SCARA [14], gantry [22])

In parallel robots, the position of the end effector is determined by multiple arms working at the same time (in parallel) [17]. Figure 2 shows a few examples of parallel manipulators.

Figure 2: Parallel robot examples (3RRR [11], delta [15], stewart/gough [16])

This classroom activity uses a 3RRR parallel robot similar to that shown in Figure 2, which is explained in detail in Section 3.

2 Getting The Course Files

Before you begin, you should first download the course materials. The following URL links to a *Dropbox* folder called *ARMA NRI 3RRR Robot*: `https://goo.gl/ktm3Lp`. This folder contains all the files needed for this course. The course files can also be found on the main course web-page.

3 The 3RRR Parallel Robot

The name *3RRR* comes from the robot's construction. The position and orientation of the robot is controlled by three kinematic chains (the name for an assembly of mechanical linkages) hence the *3* in the name. The *RRR* comes from the three revolute joints in each kinematic chain. A revolute joint is a mechanical linkage that only allows rotation about one axis.

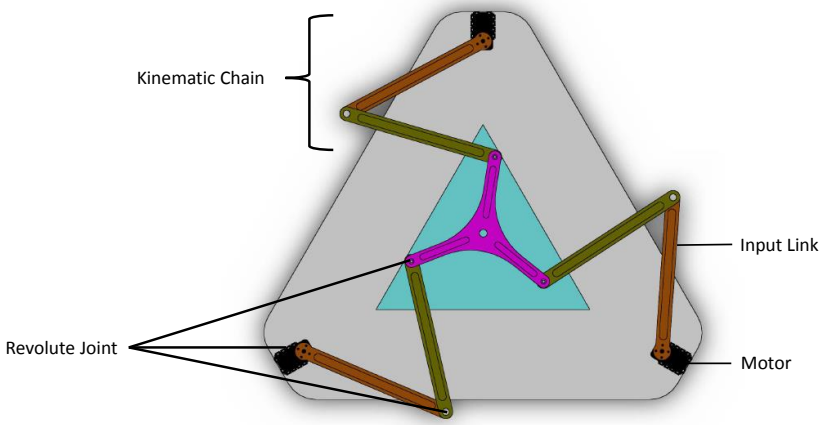

Figure 3: Kinematic chains and revolute joints (adapted from [11])

This mechanical arrangement allows controlling the position and the orientation of the moving platform in a plane parallel to the base platform. By controlling the angles of the three input links the moving platform can be controlled.

This classroom activity uses a 3RRR robot to demonstrate basic use of trigonometry and math to program and control the robot using an Arduino microcontroller.

4 Inverse Kinematics

In order for a robot to perform a task, it must be able to accurately move to a specified location in space. This may seem obvious, but it is a central problem in robotics. To solve this, engineers use the geometry of the robot to determine a mathematical expression that relates the position of the end effector to the joint angles. In other words, to what angle should the motors be set in order to make the robot move to a pre-determined location. This is called kinematic analysis [19]. There are two types of kinematic analysis: forward and inverse. Forward kinematics determines the position of the end effector as a function of its joint angles. Inverse kinematics determines the joint angles as a function of the end effector position [17]. This section will briefly explain the inverse kinematics of the 3RRR robot.

Figure 4 shows the names of the joint angles and points used in the mathematical formulation of the robot's inverse kinematics. The point P is the center of the robot which is at the orientation θ and q_1,q_2, and q_3 are the corresponding joint angles. To formulate the inverse kinematics you need to find q_1, q_2, and q_3 as a function of P and θ. Along the way you will use the other points in Figure 4 and define some new ones. The first step is to make a circle around the center of the moving platform of the robot as shown in Figure 5.

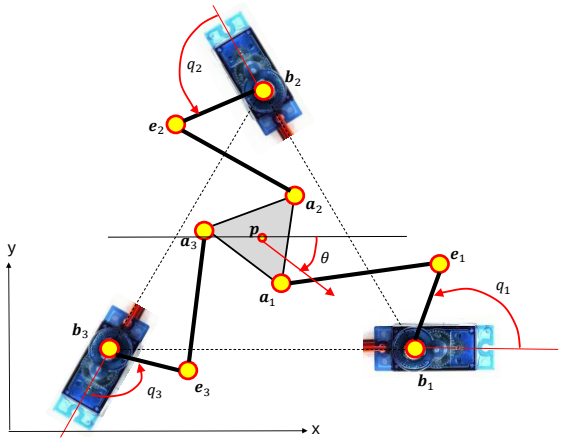

Figure 4: Schematic of the parallel robot for inverse kinematics

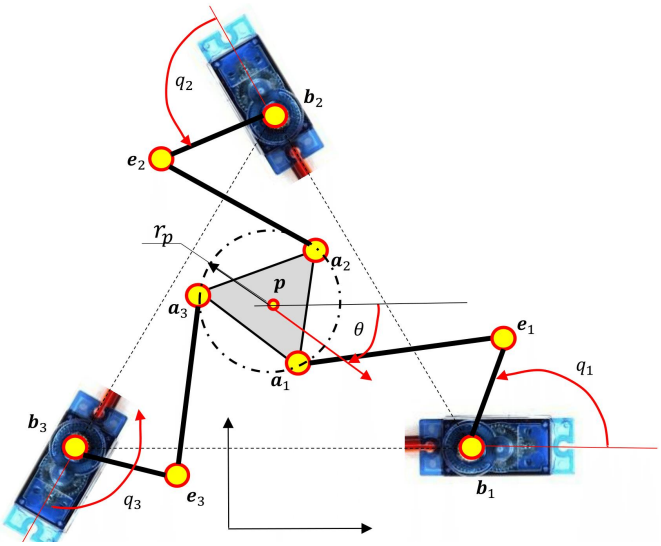

Figure 5: Inverse kinematics step one

Using this, you can derive the x and y positions of points a_1,a_2, and a_3.

$$a_{1x} = p_x + r_p \cos\left(\theta - \frac{\pi}{6}\right) \tag{1}$$

$$a_{1y} = p_y + r_p \sin\left(\theta - \frac{\pi}{6}\right) \tag{2}$$

$$a_{2x} = p_x + r_p \cos\left(\theta + \frac{\pi}{2}\right) \tag{3}$$

$$a_{2y} = p_y + r_p \sin\left(\theta + \frac{\pi}{2}\right) \tag{4}$$

$$a_{3x} = p_x + r_p \cos\left(\theta + \frac{7\pi}{6}\right) \tag{5}$$

$$a_{3y} = p_y + r_p \sin\left(\theta + \frac{7\pi}{6}\right) \tag{6}$$

Now that these positions are known, the next step is to determine the position of motor shafts, points b_1,b_2, and b_3, using a similar method to equations 1 to 6. A circle circumscribing the equilateral triangle b_1,b_2, and b_3 is drawn as shown in Figure 6.

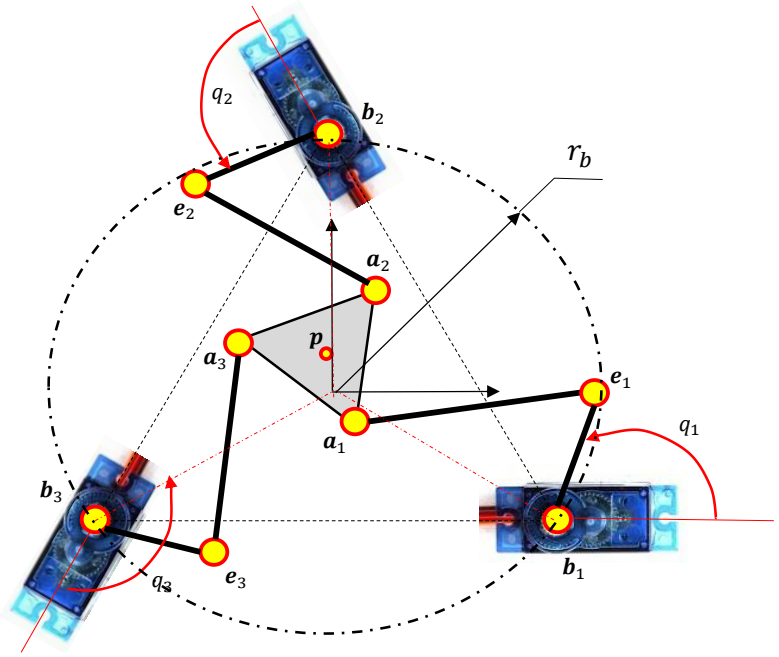

Figure 6: Inverse Kinematics step two

$$b_{1x} = r_b \cos\left(-\frac{\pi}{6}\right) = r_b \frac{\sqrt{3}}{2} \tag{7}$$

$$b_{1y} = r_b \sin\left(-\frac{\pi}{6}\right) = -\frac{r_b}{2} \tag{8}$$

$$b_{2x} = r_b \cos\left(\frac{\pi}{2}\right) = 0 \tag{9}$$

$$b_{2y} = r_b \sin\left(\frac{\pi}{2}\right) = r_b \tag{10}$$

$$b_{3x} = r_b \cos\left(\frac{7\pi}{6}\right) = r_b \frac{\sqrt{3}}{2} \tag{11}$$

$$b_{3y} = r_b \sin\left(\frac{7\pi}{6}\right) = -\frac{r_b}{2} \tag{12}$$

Because the center of the motors are fixed, these equations can be simplified further than equations 1 to 6. In order to solve for the values of q_1,q_2, and q_3, two alternate motor angles, ψ and α, must be introduced as shown in Figure 7.

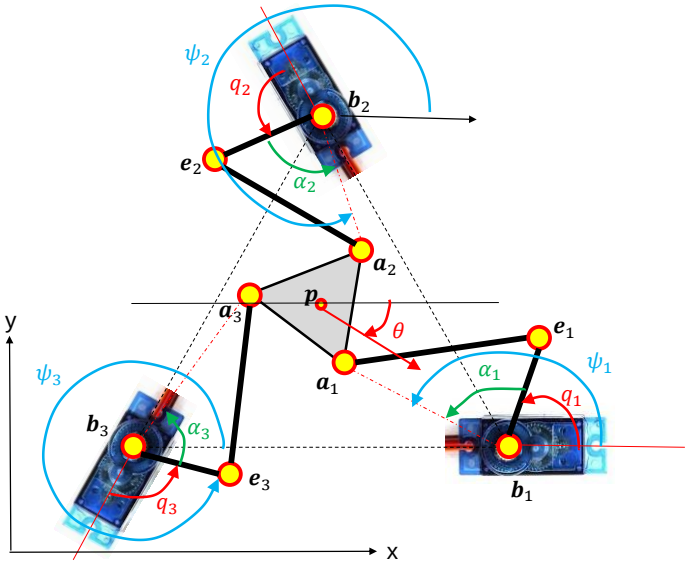

Figure 7: Inverse kinematics step three

You can now find equations for ψ_1, ψ_2, and ψ_3. These equations use the $atan2$ function. See Appendix C for an explanation of the $atan2$ function.

$$\psi_1 = atan2(a_{1y} - b_{1y}, a_{1x} - b_{1x}) \tag{13}$$

$$\psi_2 = atan2(a_{2y} - b_{2y}, a_{2x} - b_{2x}) \tag{14}$$

$$\psi_3 = atan2(a_{3y} - b_{3y}, a_{3x} - b_{3x}) \tag{15}$$

Using the law of cosines and the distance formula, you can find α_1, α_2, and α_3.

$$\alpha_1 = \arccos\left(\frac{(\overline{b_1e_1})^2 + ((b_{1x} - a_{1x})^2 + (b_{1y} - a_{1y})^2) - (\overline{a_1e_1})^2}{2\left((b_{1x} - a_{1x})^2 + (b_{1y} - a_{1y})^2\right)(\overline{b_1e_1})}\right) \tag{16}$$

$$\alpha_2 = \arccos\left(\frac{(\overline{b_2e_2})^2 + ((b_{2x} - a_{2x})^2 + (b_{2y} - a_{2y})^2) - (\overline{a_2e_2})^2}{2\left((b_{2x} - a_{2x})^2 + (b_{2y} - a_{2y})^2\right)(\overline{b_2e_2})}\right) \tag{17}$$

$$\alpha_3 = \arccos\left(\frac{(\overline{b_3e_3})^2 + ((b_{3x} - a_{3x})^2 + (b_{3y} - a_{3y})^2) - (\overline{a_3e_3})^2}{2\left((b_{3x} - a_{3x})^2 + (b_{3y} - a_{3y})^2\right)(\overline{b_3e_3})}\right) \tag{18}$$

Finally, the last step is to solve for q_1,q_2, and q_3.

$$q_1 = \psi_1 - \alpha_1 \tag{19}$$

$$q_2 = \psi_2 - \alpha_2 - \frac{2\pi}{3} \tag{20}$$

$$q_3 = \psi_3 - \alpha_3 - \frac{4\pi}{3} \tag{21}$$

These equations have been simplified, but they represent the inverse kinematic model of the 3RRR robot. Algorithm 1 summarizes the process of solving the inverse kinematics of the 3RRR robot.

Algorithm 1 Inverse kinematics summary

Inputs:
θ, $P(x, y)$
Process:
calculate: a_1, a_2, $a, 3$, (Eq. 1-6)
calculate: b_1, b_2, b_3, (Eq. 7-12)
calculate: ψ_1, ψ_2, ψ_3, (Eq. 13-15)
calculate: α_1, α_2, α_3 (Eq. 16-18)
calculate: $q_i = \psi_i - \alpha_i - (i-1)\frac{2(i-1)\pi}{3}$

The file *3RRR Kinematic Simulator.jar* is an applet that simulates the kinematics of the 3RRR robot [13]. Take some time playing with this applet to get an understanding of how the robot moves.

5 Materials Needed

In order to complete this project, you are going to need to purchase materials. Unfortunately, these materials cannot be purchased all from the same location. This section will outline all the materials needed and how to purchase them.

5.1 General Materials

To set up the robot, you will need the items in Table 1

The following link will take you to a *Digi-Key* shopping cart that has all the correct parts: `http://www.digikey.com/short/34258p`. You can buy all of the parts there if you wish or simply use it as a reference. A PDF version of the shopping cart is included in the course files in the folder *Orders*. If you purchase the materials elsewhere be careful to purchase the correct items or the project may not function correctly.

5.2 Robot Parts

The parts for the robot itself can be purchased from a 3D printing service called *Shapeways*. This link takes you to a *Shapeways* shop where you can order the parts to be printed: `https://www.shapeways.com/shops/arma-lab-3rrr-robot`. Once you follow this link, purchase both

Table 1: Bill of materials

No.	***Desciption***	***Quantity***
1	Arduino Uno	1
2	USB-A to USB-B Cable	1
3	L7805CV Voltage Regulator	1
4	0.1 μF Capacitor	1
5	0.33 μF Capacitor	1
6	Breadboard	1
7	Assorted Jumper Wires	>20
8	10kΩ Potentiometer	1
9	Tower Pro SG92R Micro Servo Motor	3
10	9V Power Source	1
11	7/16" #4-40 Screws	6
12	#4-40 Hex Nuts	6
13	1/2" #6-32 Shoulder Screws	9
14	#6-32 Hex Nuts	6
15	8mm M2 Screws	8
16	M2 Hex Nuts	8

items. Don't worry, the ARMA Lab is not making any profit off of these parts. If you have access to a 3D printer and wish to print the parts yourself, the .stl files can be found in the course folder under *Robot Parts*. These are all the parts needed to complete this project.

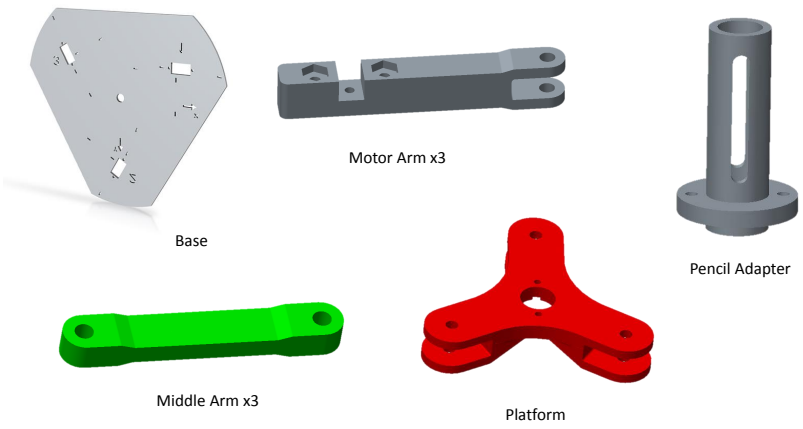

Figure 8: Robot parts

Figure 8 shows the parts needed for the robot and the names they will be referred to in this document.

5.3 Optional Materials

If you or your instructor have soldering capabilities, you can also purchase the parts for a joystick from this link: `http://sfe.io/w129046`. A PDF version is also available in the *Orders* folder. More about this is talked about in Section 12.

6 Wiring

Now that you have all the materials needed, it is time to start wiring the robot.

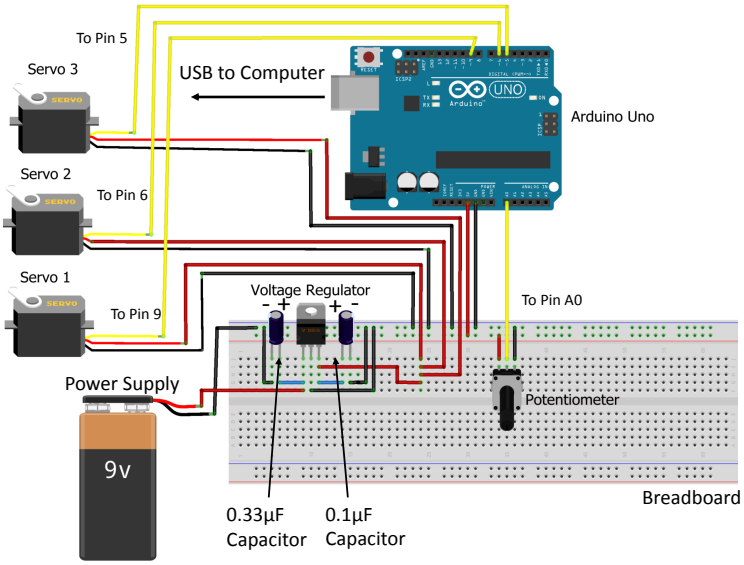

Figure 9: Wiring diagram

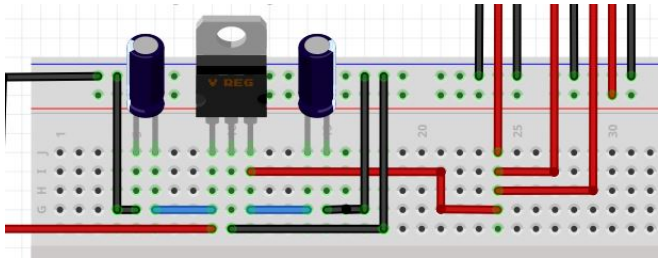

Figure 10: Wiring diagram detail view

Complete all the connections as shown in Figures 9 and 10. If you do not know how to use a breadboard please refer to Appendix A

If you used capacitors other than those recommended in the *Digi-Key* shopping cart, it is important to note that some capacitor types are *polar*. This means they have a positive and negative terminal and therefore must be oriented in the correct manner. If you used the recommended capacitors, you do not need to worry about this, but if you did not buy the recommended capacitors please refer to Appendix B.

Figure 9 shows the 9V supply as a battery, but it is better to use a power supply similar to the one recommended in items 10 and 11 in the *Digi-Key* shopping cart (this is a *wall wart* and an alligator clip adapter) because a battery would be quickly drained due to the large power draw of the circuit. However, if you cannot get access to a power supply, a 9V battery will also work.

This circuit has several components that work together to control the robot's servos. The Arduino is a microcontroller that interprets the code from the Arduino IDE and outputs an electrical signal with the correct pulse width specified in the code [3]. A potentiometer is a device that changes its impedance when the knob is turned [1]. The potentiometer in this circuit will allow you to control the position of the servos manually. The servos used in the robot operate at 5V so the voltage regulator is used to keep the voltage going to the servos at this value. The capacitors are used to increase the output stability of the regulator [21].

7 Robot Assembly

The first step to assembling your robot is attaching the servos to the base plate. The process for assembling one servo is is shown in Figure 11.

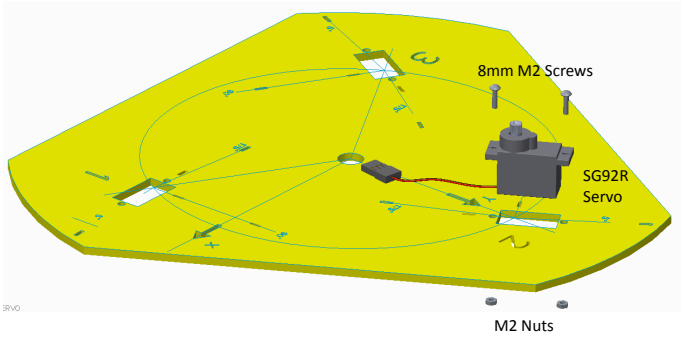

Figure 11: Servo Assembly

The next step is to attach the servo adapter that comes with the servo to the robot's motor leg. To do this, you must widen the holes shown in Figure 12 using a $\frac{1}{8}$" drill bit.

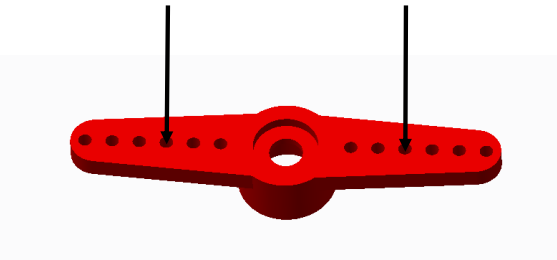

Figure 12: Holes to Widen on Motor Adapter

Next, assemble the servo adapter to the motor leg as shown in Figure 13.

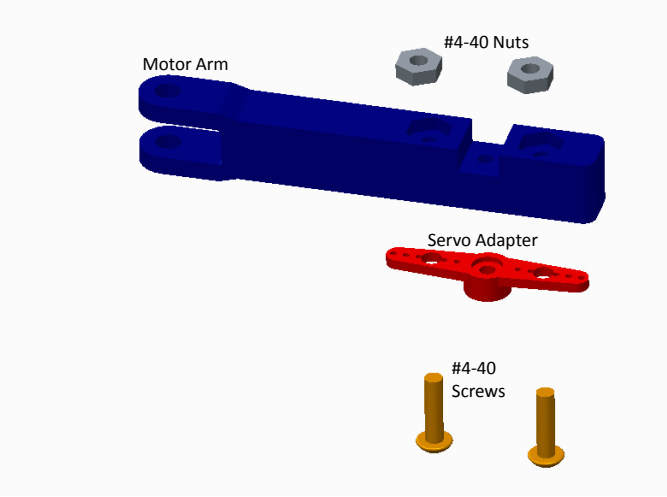

Figure 13: Assembling the Servo Adapter to the Motor Leg

After you have done this, attach the assembly from Figure 13 to the servo. To do this you will need the long screw that comes with the servo. Figure 14 shows how to do this.

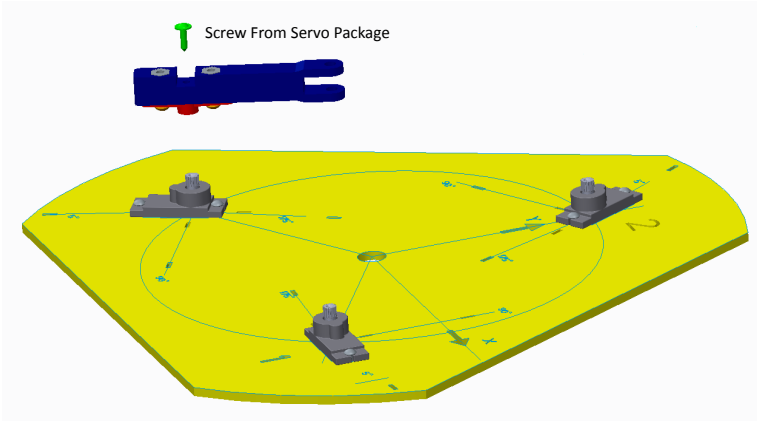

Figure 14: Assembling the motor leg to the servo

Once you have assembled this part, make sure that the motor arm can move from the 5° line to the 175° line. If it cannot, change the orientation of the assembly in 14 until it can move to both lines.

The next step is to attach the middle arm to the motor arm. Figure 15 shows this process.

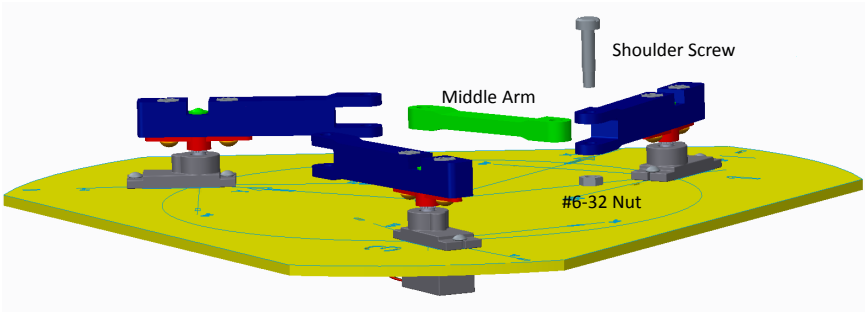

Figure 15: Assembling the middle arm

The next step is to assemble the pencil adapter to the platform as shown in figure 16. The pencil adapter allows a pencil to be added to the center of the platform to trace the location of the robot on a piece of paper.

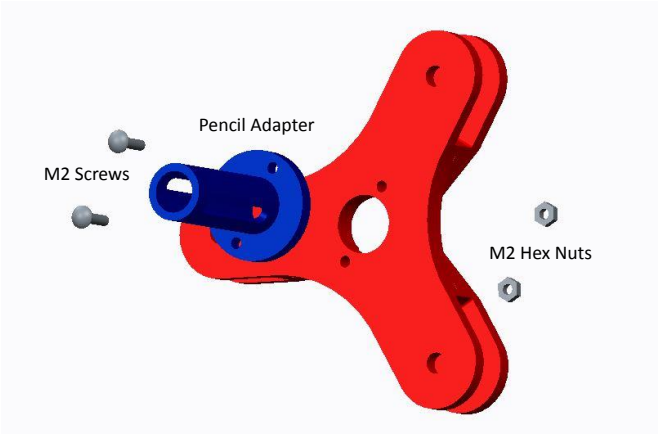

Figure 16: Assembling the pencil adapter

Finally, the platform is attached as shown in Figure 17. The other arms and pencil adapter were removed from this figure for clarity.

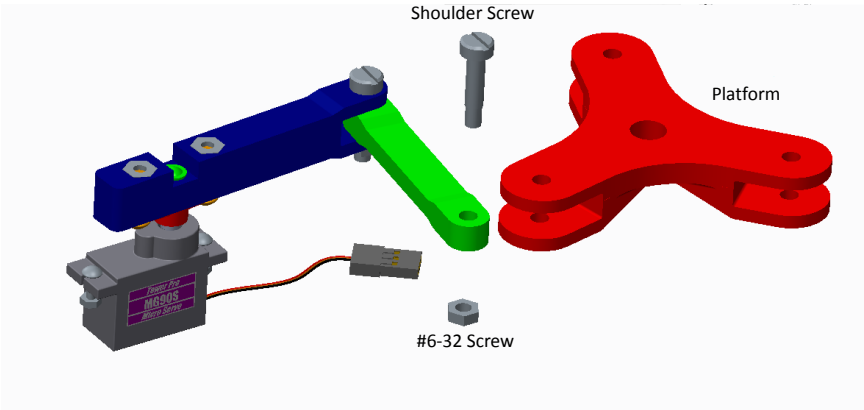

Figure 17: Assembling the platform

It is important note that there is a correct and incorrect method of assembling the center platform. The incorrect assembly is shown in Figure 18 and the correct assembly is shown in figure 19. These robots will look slightly different than yours, but the concepts are the same. The coordinate axis on the center platform was added as a visual aid.

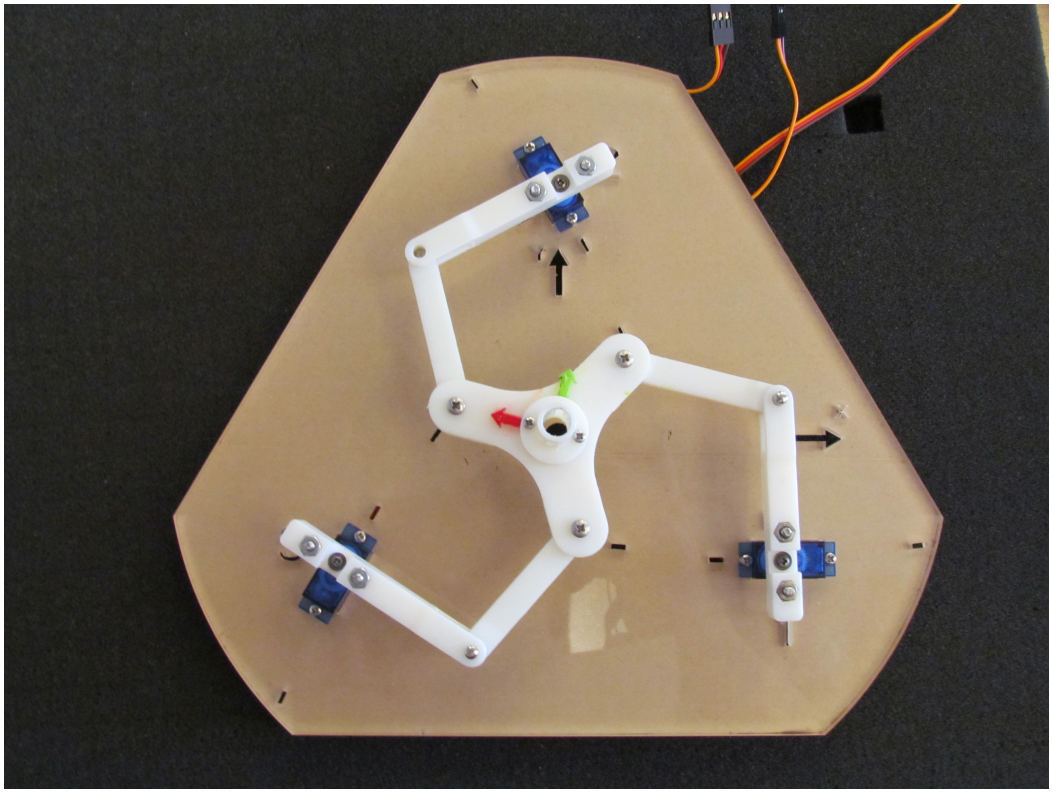

Figure 18: Incorrect assembly of center platform

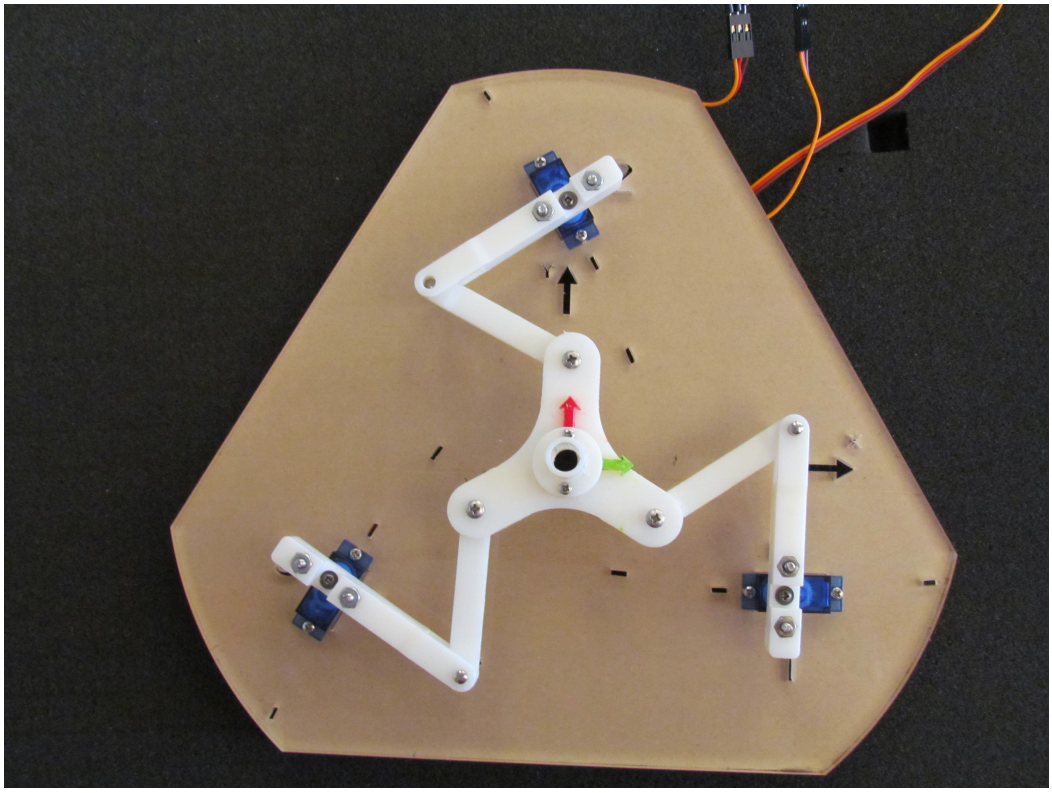

Figure 19: Correct assembly of center platfrom

If the center platform is assembled incorrectly, the robot will not be able to reach all the points in its workspace because it will hit *singularities* or configurations where the robot loses one or more degrees of freedom (ability to move in a given direction).

8 Software

8.1 Software Set Up

In order to set up the Arduino software, follow this link `https://www.arduino.cc/en/Main/Software`, download the Arduino software, and run the installation instructions. This will install the Arduino Integrated Development Environment (IDE). Programs in the Arduino IDE are written in a programming language called Arduino which is C/C++ with a slightly different organizational structure [3]. Once you have installed the Arduino IDE, open the *Arduino code* folder from the course folder you downloaded earlier. In this folder will be a subfolder called *calibration*. Copy this folder to the \documents\Arduino\libraries directory that was created on your computer during the Arduino installation. In the calibration folder there is a file called *calibration.h*. Right click on this file, go to *open with*, and select Notepad (or any other text editor). Notepad should open showing the code for a C++ library. A library is a repository of useful classes, variables, etc. that can be included in other C++ code to increase ease of development [2]. You will be editing some of the variables in this library as you calibrate the robot. Leave this file open on your computer. This is all the software set up needed.

8.2 Uploading to The Arduino

In order to run a program on an Arduino, you must *upload* the program to the board (the Arduino IDE calls a program a sketch). To do this you must open the file you wish to run in the Arduino code folder you downloaded earlier. This will open the Arduino IDE. In the IDE, press the blue button at the top of the window as indicated in Figure 20. This will upload the program to the Arduino.

Once the program is done uploading to the Arduino, the bottom of the IDE window should look similar to Figure 21.

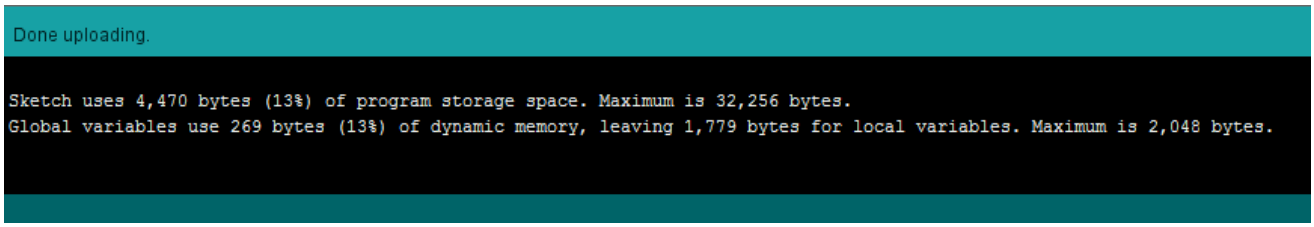

Figure 21: Arduino "done uploading" message

If everything is wired correctly, the robot should be working. If the robot doesn't work, consider these common mistakes:

- Is the 9V power source turned on?

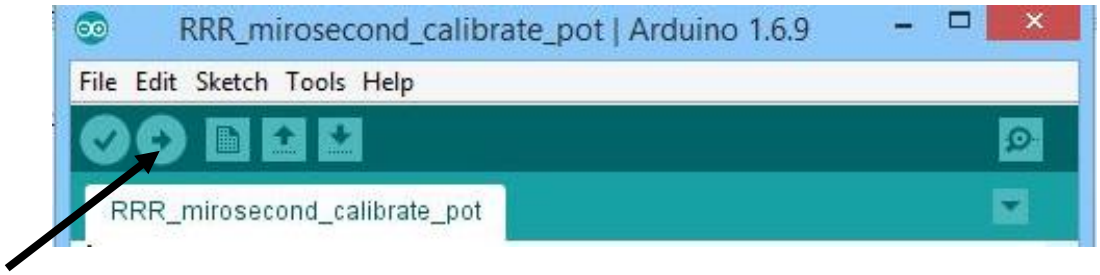

Figure 20: Arduino IDE upload button

- Have any connections been incidentally placed in the wrong row/column?
- Are all the connections making solid contact?
- Did you overlook one of the connections in Figures 9 and 10?
- Are the capacitors and voltage regulator oriented correctly?

9 Servo Motors

Figure 22: Servo motor [6]

Before you can calibrate your robot, it is important to understand servo motors and how they are controlled. Servos are a type of motor commonly used in robotics that allow engineers to specifically control the angular position of the shaft. They consist of four different parts: a DC

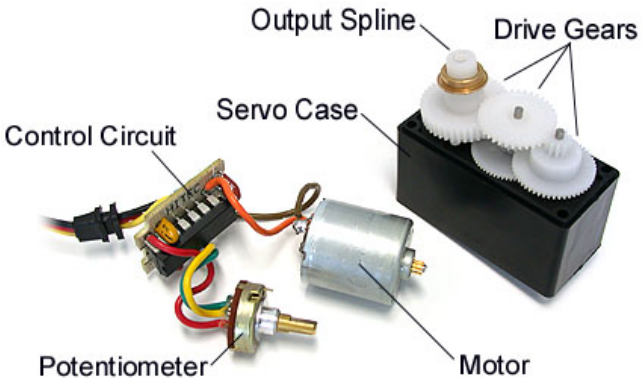

Figure 23: Servo components [18]

motor, a control circuit, a potentiometer, and usually a set of gears as shown below in Figure 23 [19].

The DC motor is used to move the mechanism, the control circuit is used to interpret the input signal and feedback loop, the potentiometer is used as a position sensor, and the gears provide a speed/torque transformation to the motion of the DC motor. If you examine the servos you will notice that there are three wires feeding into the motor. The brown wire is ground, the red wire is power, and the yellow wire is the signal used to control the servo. This signal uses pulse width modulation (PWM) to control the position of the motor [19].

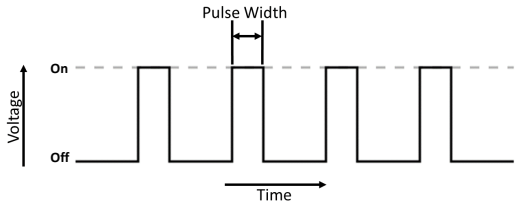

Figure 24: Pulse width modulation [7]

Pulse width modulation is a method of creating analog signals out of digital (0 or 1) signals. For this robot, an Arduino board is used to send a signal to the servo that pulses on and off. By changing the amount of time the signal is "on," shown as the "pulse width" in Figure 24, the motor can be moved to different locations. To calibrate the robot's servos you will determine the pulse widths, in microseconds (μs), that correspond to various angular positions. This will allow you to interpolate the μs value of any desired servo position. As a side note, the type of PWM that servos use differs slightly from the traditional use of the term. Usually, PWM is talked about in terms of the signal's "duty cycle" or percentage of the signal's period that it is "on," but servos simply count the amount of time the signal is "on" [4].

10 The Arduino Code Repository

In the Dropbox folder you downloaded earlier, there is a folder called *Ardiuno Code*. In this folder, you will find several Arduino sketches that command the robot to do various things. Table 2

Table 2: Arduino code descriptions

File Name	***Purpose***
RRR_Circle_Path	Commands robot to move in a circle around the origin
RRR_epicycloid_path	Commands robot to move in a epicycloid path about the origin
RRR_mirosecond_calibrate_pot	Uses potentiometer to assist in setting the microsecond delays corresponding to 5° and 175°
RRR_parallel_robot_go_to_home	Commands the robot to go to the origin
RRR_parallel_robot_roll_origin	Commands the robot to alternate its orientation back and forth from 30°to -30°while staying fixed about the origin
RRR_roll_about_origin_pot_control	Commands the robot to move in a similar fashion to RRR_parallel_robot_roll_origin except controlled by the potentiometer
RRR_Telemanipulation	Allows for the robot to be controlled using a joystick. See section 12 for more details.

After you have calibrated the robot, you can run these sketches and watch the robot move. If you want the robot to draw out its path, place a blank sheet of paper under the robot on the base and place a pen in the center hole of the platform.

11 Calibration

11.1 Introduction to Calibration

Once the kinematics of the robot is known, it is necessary to command the servos to the desired angular position. However, motors only understand electrical signals so engineers must figure out the relationship between the input signal and the angular position of the motor [19]. Also, it is impossible to manufacture a robot perfectly so there will always be some error created by the robot's construction [1]. Engineers must compensate for this error. These two things are collectively referred to as calibration.

This course's robot is calibrated by determining the pulse width that corresponds to a 5° rotation and a 175°. Using these values, any position of the servos can be interpolated. Equation 22 shows the equation used to determine the pulse width that corresponds to a desired angular position.

$$m = \frac{m_{max} - m_{min}}{\theta_{max} - \theta_{min}}(\theta_{desired} - \theta_{min}) + m_{min} \tag{22}$$

In Equation 22, m is the pulse width (in microseconds) that correspond to the desired angular position ($\theta_{desired}$). m_{max} and m_{min} are the pulse widths that correspond to a 175° (θ_{max}) and 5° (θ_{min}) position of the servos respectively.

This section will walk you through how to calibrate the course's parallel robot. There is also an online video explaining the calibration process that can be found here: `https://goo.gl/gLn7wS`. It can also be found on the course web-page.

11.2 Calibration Step One

Once you have ensured that the robot is working, remove the center and middle legs of the robot as indicated in Figure 25. For reasons that will become apparent later, this will make calibration much easier.

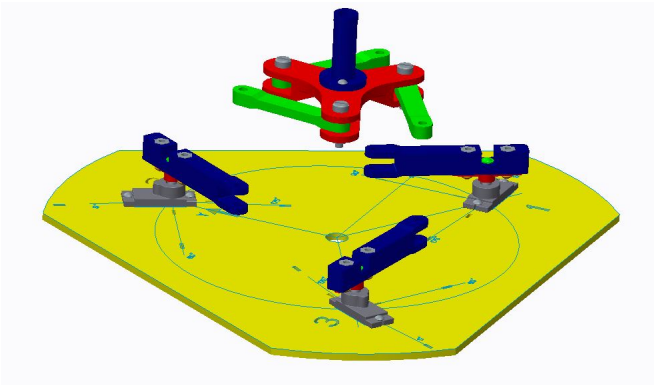

Figure 25: Removing the middle section

The first step to calibrate the robot is to find the microsecond pulse delay that corresponds to a 5° position of Servo 1. To do this, upload `RRR_microsecond_calibrate_pot.ino` to the Arduino. You will next need to open the Arduino's serial monitor. To do this click on the serial monitor button as shown in Figure 26.

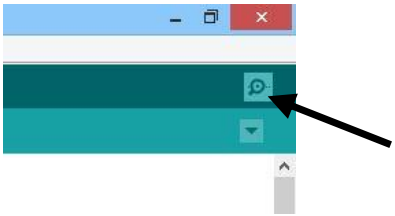

Figure 26: Button to open the serial monitor

This will open a window that displays the current pulse width in microseconds and the angle the servo thinks it is at in degrees. Next, open the calibration.h file as described earlier. In this file, you will find a line that reads **double m_of_theta_min[3]={580,713,752};**. This line tells the computer to initialize an array (a grouping of adjacent items in memory under a common name) of three double (a number rounded to 15 decimal places) data types called **m_of_theta_min** and assign it the values of 580, 713, and 752. The semicolon tells the computer the line of code is over [2]. This variable is the pulse width that the Arduino sends to Servo 1

to command it to 5°. Turn the potentiometer until you see servo 1 line up exactly with the 5° notch as shown in Figure 27. Change the first value in **m_of_theta_min** to equal the value of the pulse width on the serial monitor. If the arm does not reach the 5° notch, decrease the first value of **m_of_theta_min** until you can reach the 5° notch. Save *calibration.h* and re-upload `RRR_microsecond_calibrate_pot.ino`.

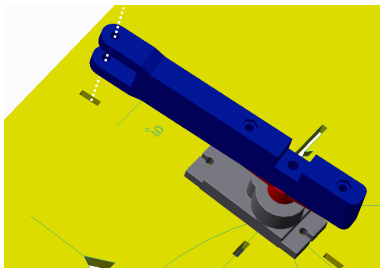

Figure 27: 5° alignment

Next, turn the potentiometer until Servo 1 lines up with the 175° notch. Then change the first value of **m_of_theta_max** to equal the pulse width value on the serial monitor.

Once this is completed, change the value of the variable **i** in the Ardino IDE to be 1. This will change the code so that Servo 2 moves when the potentiometer is turned. repeat the above process changing the second values of **m_of_theta_min** and **m_of_theta_max** making sure to save *calibration.h* each time you change a value. Again, repeat for Servo 3 by changing **i** to 2 and editing the third values of **m_of_theta_min** and **m_of_theta_max**.

You have now completed the first step to calibrating your robot. If this has been done correctly, you should be able to turn the potentiometer and make each servo move from their 5° lines to their 175° lines. You are now ready to move on to step two.

11.3 Calibration Step Two

Because of the way the robot is assembled, there will be positional errors in the robot's end effector position. In this section you will attempt to negate this. Change **i** back to zero and turn the potentiometer until servo 1 lines up with the 90° notch. Check the serial monitor to what angle the robot thinks it is at, replace the first value of the array **q_home_offset** in *calibration.h* to be $90 - \theta$ where θ is the position servo 1 thinks it is at.

Reassemble the robot, then open the file `RRR_parallel_robot_go_to_home.ino` and upload it to the Arduino. This will command the robot to the center of the board. if the robot is calibrated correctly the center of the moving platform will exactly line up with the center of the base.

11.4 Calibration Summary

Algorithm 2 summarizes the calibration process for the 3RRR robot detailed in Sections 11.1 to 11.3.

Algorithm 2 Calibration Summary

1: Open *calibration.h* (Sec. 8.1)
2: Open and upload `RRR_microsecond_calibrate_pot.ino` to the Arduino (Sec. 8.2)
3: Open serial monitor (Fig. 26)
4: Turn potentiometer until Servo arm lines up with 5° notch (Fig. 27)
5: Edit first term in array **m_of_theta_min** to equal the pulse width displayed on the serial monitor (Sec. 11.2)
6: Turn potentiometer until Servo arm lines up with 175° notch (Sec. 27)
7: Edit first term in array **m_of_theta_max** to equal the pulse width displayed on the serial monitor (Sec. 11.2)
8: Change value of **i** to 1 to change to Servo 2 (in C indexing starts at zero)(Sec. 11.2)
9: Save *calibration.h* (Sec. 8.1)
10: Repeat steps 2-7, this time editing the second values in **m_of_theta_max** and **m_of_theta_min** (Sec. 11.2)
11: Change value of **i** to equal 2 to change to Servo 3 (in C indexing starts at zero)(Sec. 11.2)
12: Save *calibration.h* (Sec. 8.1)
13: Repeat steps 2-7 this time editing the third values in **m_of_theta_max** and **m_of_theta_min** (Sec. 11.2)
14: Change **i** back to zero
15: Upload `RRR_microsecond_calibrate_pot.ino` to the Arduino (Sec. 8.2)
16: Turn the potentiometer until servo 1 lines up with the 90° notch (Sec. 11.3, Fig. 27)
17: Check the serial monitor to what angle the robot thinks it is at, replace the first value of the array **q_home_offset** in *calibration.h* to be $90 - \theta$ where θ is the position servo 1 thinks it is at. (Sec. 11.3)
18: Save *calibration.h* (Sec. 8.1)
19: Change **i** to 1 to change to servo 2
20: Repeat steps 15-18 this time changing the second value of **q_home_offset** (Sec. 11.3)
21: Save *calibration.h* (Sec. 8.1)
22: Change **i** to 2 to change to servo 3
23: Repeat steps 15-18 this time changing the third value of **q_home_offset** (Sec. 11.3)

12 Telemanipulation

Note: this section requires soldering so if you and/or your instructor do not have soldering capabilities, you will unfortunately not be able to complete this section. A soldering tutorial can be found here: `https://learn.sparkfun.com/tutorials/how-to-solder---through-hole-soldering`.

This section will show you how to move your robot by using a joystick. This is called telemanipulation. First you will need to purchase the items in this *SparkFun* wishlist: `http://sfe.io/w129046`. A PDF version of this order can be found in the course files *Orders* folder.

First you will need to solder the joystick to the breakout board and then solder five jumper cables to the holes labeled *VCC*, *VERT*, *HORZ*, *SEL* and *GND*. After this is completed, wire the Arduino using the schematic in Figure 28.

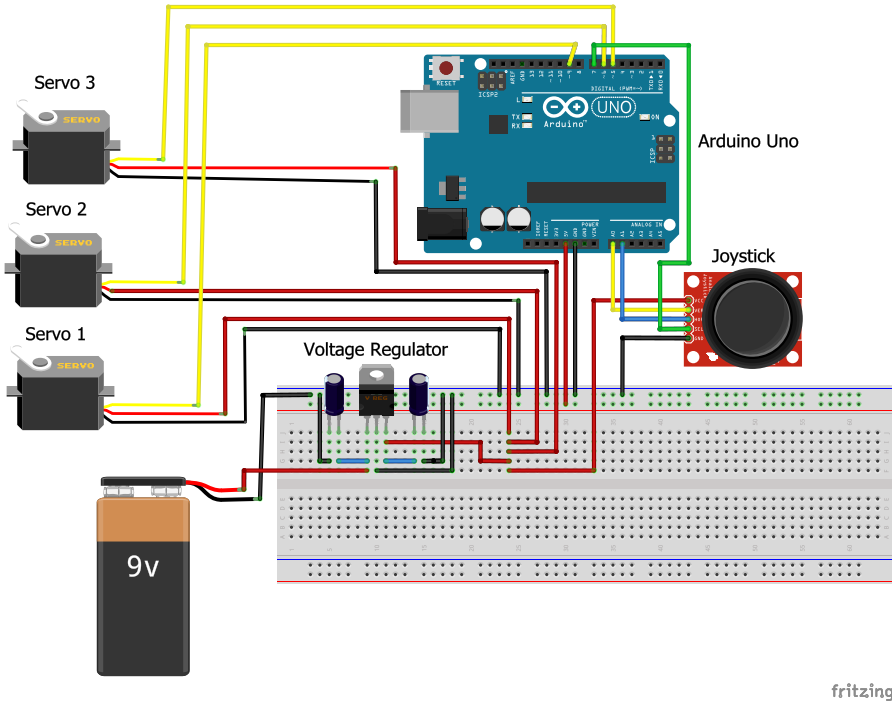

Figure 28: Telemanipulation wiring diagram

When you have finished wiring this, you will now be able to change the robot's position by moving the joystick.

References

[1] David G Alciatore. *Introduction to Mechatronics and Measurement Systems*. Tata McGraw-Hill Education, 2007.

[2] Alex Allain. Jumping into c++. *CIT*, 105, 2014.

[3] Arduino.cc. *Introduction to Arduino*, 2016.

[4] Micheal Barr. *Introduction to Pulse Width Modulation (PWM)*. Barr Group, 9 2001.

[5] Hernando Barragán. *What is a Breadboard?* Wiring.

[6] Conrad. *Modelcraft Mini servo Analogue servo Gear box material Plastic Connector system JR.*

[7] Embedded Micro. *Pulse-Width Modulation*, 2015.

[8] Every China. *CL11(PEI) MYLAR CAPACITOR*, 2016.

[9] Future Electronics. *.001uF 50V Ceramic Capacitors.*

[10] Guangdong Fenghua Advanced Technology Holding Co. Ltd. *Leaded Tantalum Electrolytic Capacitor with Large Capacitance and 4 to 50V Voltage Rating*, 2016.

[11] Human and Robot Interaction Laboratory. *Parallel Mechanisms: Design, Development and Control of 3-RRR Planar Parallel Mechanism*, 2014.

[12] MathWorks. *atan2*, 2016.

[13] Miguel Hernndez University: Automation, Robotics and Computer Vision lab. *Parallel 3RRR Robot.*

[14] Omron Adept Technologies, INC. *SCARA Robot - Adept Cobra s350 SCARA 4-Axis Robot*, 2016.

[15] Penta Robotics. *Veloce.*

[16] Physik Instrumente. *H-840 6-Axis Hexapod*, 2016.

[17] Jon M Selig. *Introductory Robotics*, volume 5. Prentice Hall Englewood Cliffs, NJ, 1992.

[18] Servo City. *How Do Servos Work?*, 2015.

[19] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control.* Springer Science & Business Media, 2010.

[20] Simple Electronics. *Capacitor Types*, 7 2013.

[21] ST Microelectronics. *Positive voltage regulator ICs*, 2 2016. Rev. 33.

[22] Jörg Tertünte and Lisa Endrijaitis. *Switching from Robot Systems to Cartesian Handling Systems.* Linear Motion Tips, 11 2014.

[23] TU: Berlin - Robotics and Biology Laboratory. *Robotics: Facilities*, 4 2016.

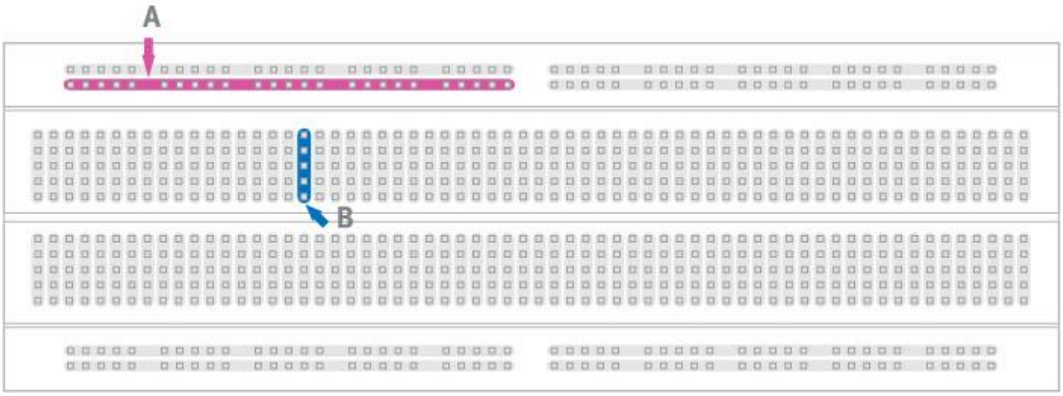

Figure 29: Breadboard connections [5]

A How to Use a Breadboard

Breadboards, which are also sometimes called protoboards, are used to easily prototype circuitry. They are a plastic board with a matrix of holes that are internally connected to expedite wiring. In Figure 29 all columns are internally connected in the manner shown by the blue column B. Any wire put into any of the holes will be connected to all the other holes in the column. On the top and bottom of the breadboard the rows are internally connected as indicated by the pink line A. These rows are called the "power rails" because they are traditionally where the power and ground are connected. It is important to note that the breaks in the power rail rows are internally connected while the breaks in the columns are not connected as indicated in Figure 29. This is to allow a DIP to be inserted into the breadboard.

B Polar vs. Non-Polar Capacitors

The positive and negative signs on the capacitors in Figure 9 indicate which orientation the capacitors should be in if they are polar. Figure 30 shows some common polar and non-polar capacitor types. Needless to say, this list is not exhaustive and you should research the type of capacitor you have to determine if it is polar. In general, if one of the capacitor's leads (the wires leading to the capacitor) is longer than the other, it is polar and the longer lead is the positive terminal [1].

C The $atan2$ Function

The $atan2$ function is a variant of the inverse tangent ($arctan$) function. It is used because tan is multivalued. This means that it has multiple valid solutions existing in different quadrants. For example, $\tan\left(\frac{3\pi}{4}\right) = \tan\left(-\frac{\pi}{4}\right) = -1$ despite the fact that $\frac{3\pi}{4}$ is in quadrant 2 and $-\frac{\pi}{4}$ is in quadrant 4. This makes the solution to $arctan(-1)$ ambiguous. The $atan2$ function solves this issue by taking the x and y coordinates as two separate arguments, $atan2(y, x)$. Using the inverse

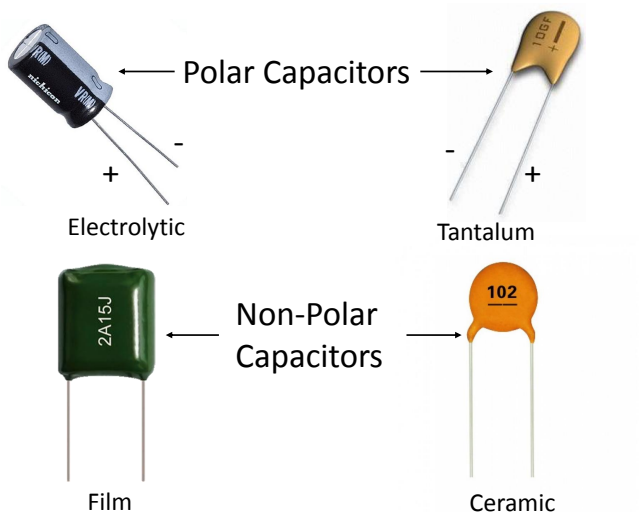

Figure 30: Common capacitor types: polar vs. non-polar (electrolytic [20], tantalum [10], film [8], ceramic [9]

of the example above, while $arctan(-1)$ could equal either $\frac{3\pi}{4}$ or $-\frac{\pi}{4}$, $atan2$ has unambiguous solutions to both cases. $atan2(-1,1) = -\frac{\pi}{4}$ and $atan2(1,-1) = -\frac{3\pi}{4}$ [12].