**Operations Manual**

**Remote-Controlled Fuse-Switching Device**

**Sponsored by Florida Power & Light**

FAMU-FSU College of Engineering

2023 Team 510

Ebony Bland, Austin Bruen, Davis Goolsby, Emilio Manzo, and Garfield Murphy

# Project Overview

The robotic telescoping fuse switching device (RTFSD) was developed to help utility workers remove and switch damaged fuses from utility poles. Utility workers currently must manually control the height of the telescoping stick and balance the stick with their hands as they maneuver the stick around the fuse cutout. This process causes a significant amount of strain on the operators’ upper body as well as fatigue from repeating the process.

The robotic telescoping fuse-switching device has two major components. One of them being the robotic arm. The robotic arm is used to pull out the broken or damaged fuses. The other major component is the base. The base is used to help the arm extend up to a height of 10 feet.

# Component/Module Description

## Telescoping Pole

Each layer of the telescoping pole will have a corresponding custom-made L-shaped segment that will be press fit and reinforced with PVC cement. These segments will be flush with the top of each layer and in line with the vertical cutout of each telescoping layer. Each custom block will have a protrusion at the top with a cutout intended to tie the steel wire around. This cutout will ideally be reinforced with a non-conductive composite such as carbon fiber to prevent tear-out during operation. The bottom side of the blocks has an ovular recession to allow for a pulley block to be mounted flush, however, do not mount the pulley blocks to the telescoping pole with hardware that punctures the telescoping pole as this would interfere with the internal pulley mechanism. Rather, the pulley block will be contained by an included support plate that can be mounted to the custom-made segments with a non-conductive fastener, such as nylon screws, or screws with an appropriate non-conductive coating. A shroud will be included to protect the pulley blocks from environmental impurities during operation and transport. The internal pulley mechanism will use non-conductive rope, either synthetic or properly insulated steel cable. Regardless, the rope must be a low stretch to ensure the telescoping pole reaches the desired height under load. The pulley mechanism consists of spans of rope elevating each layer with one end attaches to the cutout on the L-shaped segment, wrapping around the pulley, feeding through the vertical cutout, and then attaching to the circular cutout at the bottom of the driven telescoping layer. With this pulley chain, each layer moves relative to the previous, making the diagnosis in case of failure simple to identify. The telescoping pole will be driven by a DC motor paired to a hand winch. While code will be implemented as a fail-safe, it is the operator’s responsibility to ensure slack isn’t completely pulled out of the system which would cause excessive strain on the PVC. The ratchet will be two-way, meaning directionality is controlled by a safety pawl. This also implies that the user can change the directionality of the telescoping pole (i.e., extending or retracting) as desired or in case of an emergency. Note that if the safety pawl is set in the opposite direction as the DC motor, the motor will back-drive the ratchet resulting in no motion of the device. The base layer of the telescoping pole will be rigidly attached to a baseplate with a set screw. This base plate can then be attached to the included tripod to provide additional stability.

### Assembly instructions

1. Download G-Code files for the 3 pulley mounts: 0.75-inch, 1-inch, 1.25-inch.
2. Print pulley mounts with 40% infill, and wall thickness of 5mm.
3. Once printed, Check if pulley with base fits into the flat edge of the pulley mount.
4. Sometimes a 3D printer may print more material than needed, if the pulleys do not fit, used rotary tool to carve out more space
5. Once the pulleys fit onto the flat edge affix them with zip ties or an adhesive. There are holes for screws, but they would interfere with the PVC pipe meshing.
6. Press fit the pulley mounts onto their respective PVC pipes, hammering them in may be necessary.
7. Cut the PVC pipes as directed.
8. Thread the wire connected to hand winch through the Pulley system and the PVC pipes as directed.
9. Affix the largest PVC pipe to the base using a 0.35-inch diameter set screw

Chart

Description automatically generated with medium confidence

Figure 1: Telescoping Pole

## Robotic Arm Prototype

The robotic arm component of the fuse-switching device has 3 degrees of freedom to remove a blown fuse link and replace it with a new one. A prototype was built to model the full-scale capabilities of the robotic arm without being attached to the rest of the fuse switching device. The figure below shows a Computer Aided Design (CAD) of the prototype.

A picture containing LEGO, toy

Description automatically generated

Figure 2: CAD Model of Robotic Arm Prototype in default starting position.

Figure 3 displays each major component of the prototype. The arm has 2 joints: the shoulder and the elbow. The shoulder joint has two degrees of freedom. It can rotate 360 degrees about the central axis or z-axis of the telescoping pole or base link. This is called the Shoulder Z Linkage.

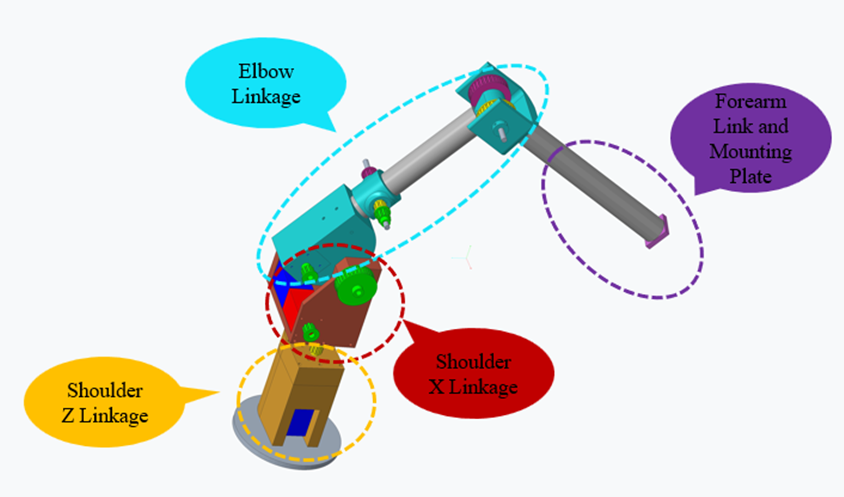


Figure 3: Major Components of Robotic Arm Prototype

The shoulder joint also can rotate 150 degrees from its starting position about the axis perpendicular to the central axis of the telescoping base, or x-axis. This is referred to as the Shoulder X Linkage. Attached to the shoulder joint is a cylindrical bicep link. Another cylindrical link is connected to the bicep through the elbow joint. The elbow joint can rotate the forearm link 270 degrees about the x-axis. At the end of the forearm link is a mounting plate that is used to mount a gripper and a camera.

### Shoulder Z Linkage

The image below shows an exploded model of the Shoulder Z Linkage. This set of components allows the entire robotic arm to rotate 360 degrees about the z-axis. The motion is driven by a pulley and timing belt system connected to a stepper motor.

A picture containing toy

Description automatically generated

Figure 4: Shoulder Z Linkage Exploded View

A NEMA 24 stepper motor is connected to an EG Series Planetary Gearbox with a gear ratio of 20:1. The motor is shown has a blue box in the image and the gear box attachment is shown in red. The motor and gearbox are in the orange 3D printed motor housing shown in Figure 4.

The exposed shaft of the gearbox is attached to a yellow 3D-printed L-Series timing belt pulley. This pulley has an outer diameter of approximately 1 inch. Another yellow pulley is attached to a grey shaft that will be supported by bearings located in the motor housing part. The output pulley is a 3D-printed L-Series with a 3-inch diameter. The driver pulley is attached to the output pulley by an L-Series timing belt. These pulleys will be connected using an L series timing belt.

To assemble this linkage, the two orange parts of the Shoulder Z housing and pulleys need to be 3D printed. They were printed for this prototype using a DREMEL 3D45 printer with 30 percent infill and a wall thickness of 5mm. All other settings were left on default.

Once printed, ¼-20-inch heat inserts need to be inserted into the two portions of the shoulder z housing. First, insert the heat inserts into the holes around the perimeter of thinner orange housing plate. Then, put the heat insets into the top and bottom holes of the large housing component. An up-close image of these holes is shown in the images below. The heat inserts can be put into the holes using a soldering iron.

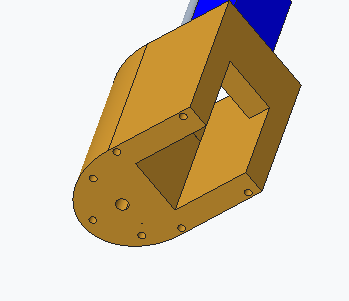
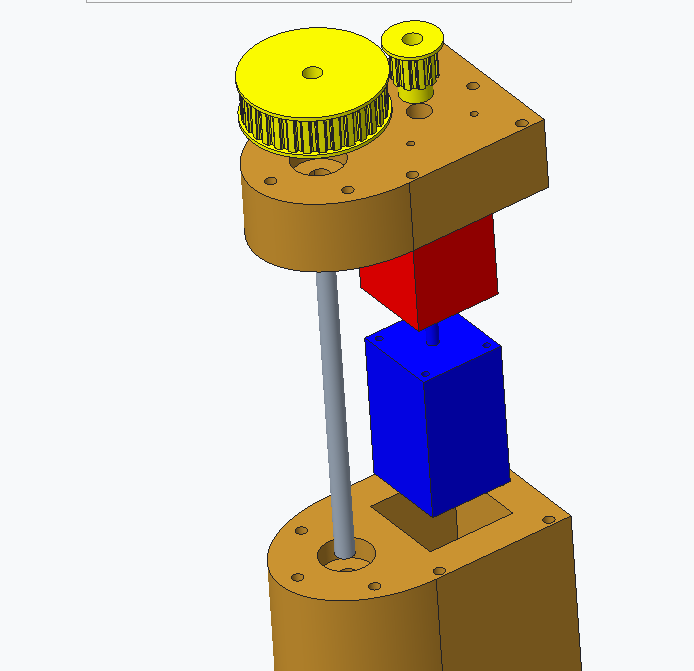


Figure 5: Close up view of Shoulder Z Linkage Housing

The next step is to mount the motor to the gearbox using the screws that were included in the motor’s original packaging. Then, the gearbox and motor can be mounted to the top portion of the Shoulder Z housing using screws provided with the gearbox. A bearing is placed in the large circular hole of the bottom housing piece. Then, a shaft is put through this bearing and hole. Then, the top portion of the shoulder Z housing with the motor configuration can be placed on top of the shaft and screwed down with ¼-20 ich screws and washers in holes around the perimeter. Then, the yellow pulleys need to have #10 heat insert put into the set screw holes with a soldering iron. Then use #10 set screw to screw the pulleys on to the shaft and gear box shaft.

### Shoulder X Linkage

The shoulder X linkage allows the entire robotic arm to rotate about the x-axis 270 degrees from its default starting position. This rotation of the arm is facilitated by a NEMA 24 stepper motor connected to an EG Series Planetary Gearbox with a gear ratio of 10:1. The motor and gearbox setup is held in a 3D printed structure that mounts it and holds the shaft that goes through the bicep link. The exposed shaft of the gearbox is attached to a 3D-printed 1-inch diameter L-Series timing belt pulley. The driver pulley is attached to another 3D-printed 3-inch pulley by an L-Series timing belt. The larger pulley is attached to a shaft that rotates the entire arm.

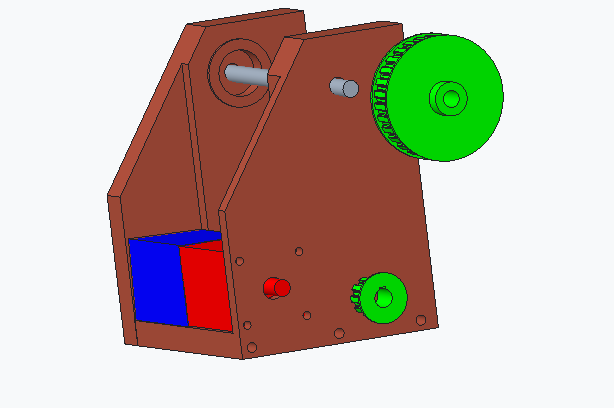
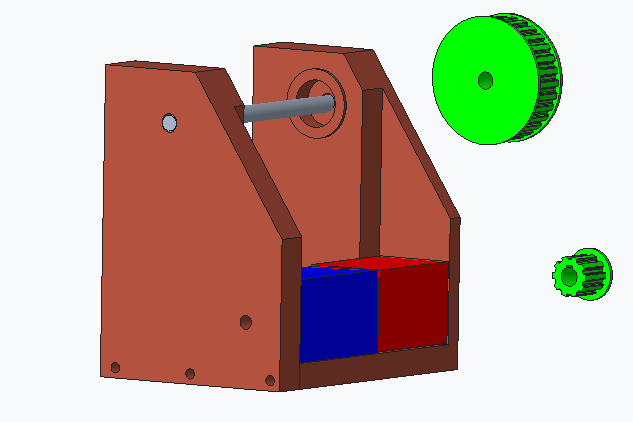


Figure 6: Exploded View of Shoulder X Linkage

To assemble this linkage, the brown shoulder X plate housing parts need to be 3d printed using the same settings and printer as mentioned in the previous linkage section. The green pulleys also need to be 3D printed. Then, ¼-20 heat inserts need to be plated into the arches of the shoulder plate and middle plate. #10 heat inserts need to be placed in set screw holes of the pulleys. #10 screws are used to secure the pulleys to the shafts. The motor will be mounted to the gearbox and shoulder x housing in the same fashion as the motor setup in the should z housing.

### Elbow linkage

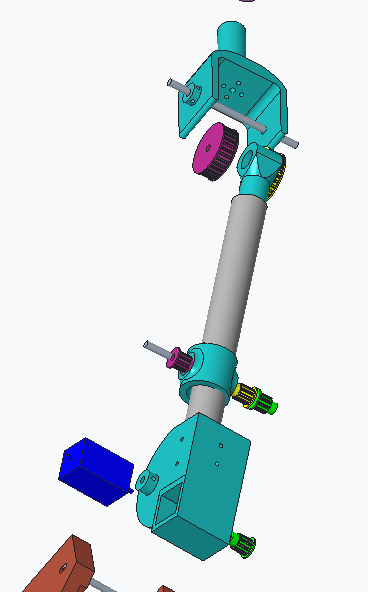
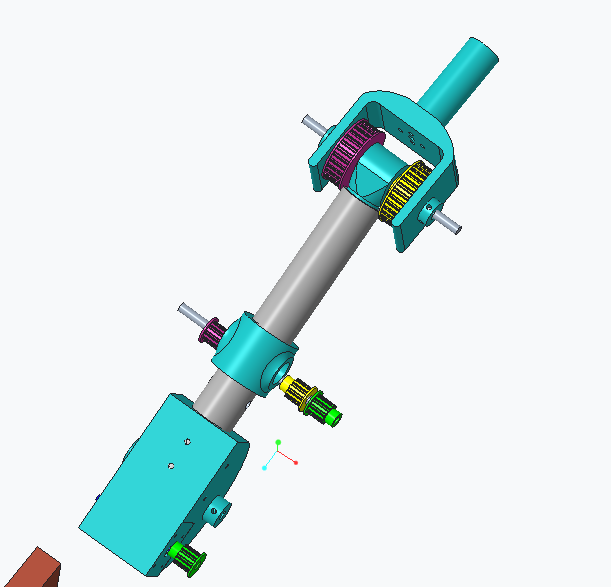
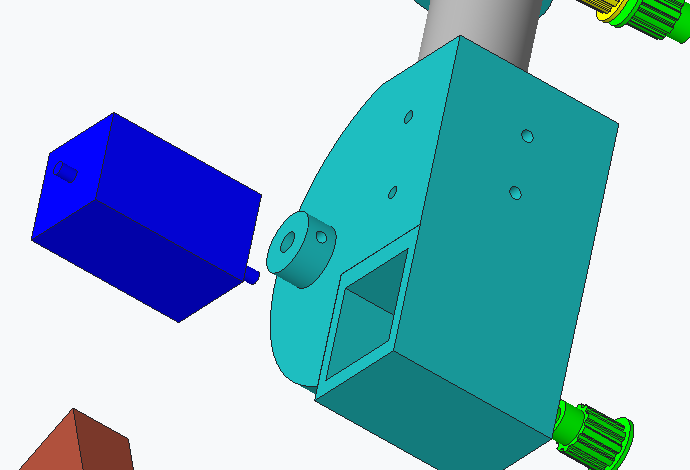
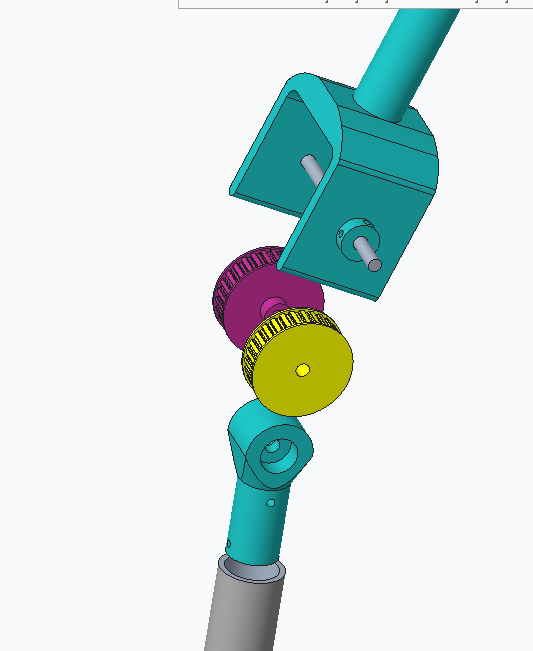


Figure 7: Elbow Linkage General and Exploded View

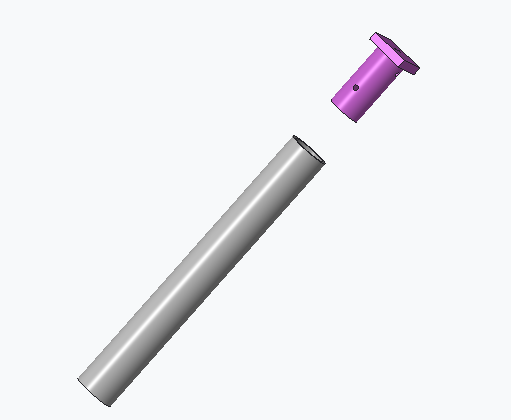
Drill 1/2-inch holes through PVC pipe arm linkages at the point where the bicep and forearm will meet (indicated in appendix drawings). where the arm would meet. At the joint between the two-forearm linkage and the elbow linkage insert the grey shaft between the joint and the arm link. Insert onto the shaft the purple pulley piece, yellow shaft ring and green pulley piece respectively as shown in figure 7. Once the joint is connected mount the motor in the light blue bottom joint, connecting it to the green pulley piece.

The elbow linkage is made of three sets of pulleys connected with timing belts. Each set is in a different color. The motor that drives this train of timing belts and pulleys is a NEMA 24 motor. It is mounted to a 3D-printed motor housing that is rigidly attached to the PVC bicep link. The large output pulleys are rigidly attached to the shaft at the elbow joint. An elbow connector piece is also rigidly attached to this shaft. Therefore, when the motor rotates the driver pulley, the output pulleys rotate the shaft. This rotates the elbow connector. The elbow connector is rigidly attached to the PVC forearm link with screws.

Figure 8: Close up of Bicep End Piece Components and Bicep Elbow Components and Elbow Linkage



### Forearm and end effector mount



*Figure 9: Close up view of Foreman (Gray piece) and end effector moun*t *(Purple Piece)*

The PVC forearm connects the elbow to the gripper and camera with the use of an end effector mount. The PVC forearm link is rigidly attached to the elbow connector piece with 1/4-20 screws. The end effector mount will have a camera attached to the top part of it and the gripper is attached to the flat side of the effector mount in the figure. The PVC forearm is attached to the end effector mount using ¼ - 20 screws. This will allow the end effector mount to be screwed together with the foreman as well as the addition of washers to make sure the connection is tight. The end effector mount will have ¼ inch holes on the flat side of it. This will be done using a drill bit of ¼ inches. This will allow for the camera and the gripper to be mounted on the robotic arm. The camera will be mounted using a butterfly mount to the top of the end effector mount. This will allow the operator to view the broken or damaged fuse and see the movements of the robotic arm.

The gripper comes with a servomotor. The first step is to use a screwdriver that fits M3 flathead screws. Connect the M3 screws to the 10 mm couplings on the bottom of the servomotor in the correct slot. Connect the M3 flathead screws on the gripper with the 10 mm coupling. This should establish the connection between the gripper and the servo motor. Next, unscrew the screws using the same flat head screwdriver on the bottom servo horn. Do a similar process with the M3 screws to the 10 mm couplings on the top of the servo motor. This time screw into two M3 flat head screws on the sides of the servo horn. This should connect the top of the servomotor with the servo horn. In the center of the servo horn screw in the last M3 flat head screw. This should allow the servomotor to be completely hooked up to the gripper. For more information visit the Gitnova website. The gripper will be attached to the effector mount using a combination of screws as well as zip ties to correctly hold it in place. As the gripper comes with a base with a single screw holder using ¼-20 screws will allow it to be mounted to the effector mount. Another screw hold can be created using a drill bit.

The camera can be attached to the mounting plate using a butterfly mount. This mount allows the user to be able to screw it into the top of the effector mount. Using a screwdriver for ¼ inch screws. This keeps the camera in place. There are no pre-made holes for this installation. The holes can be drilled into the mounting plate using a drill bit associated with the screws needed to install butterfly mount.

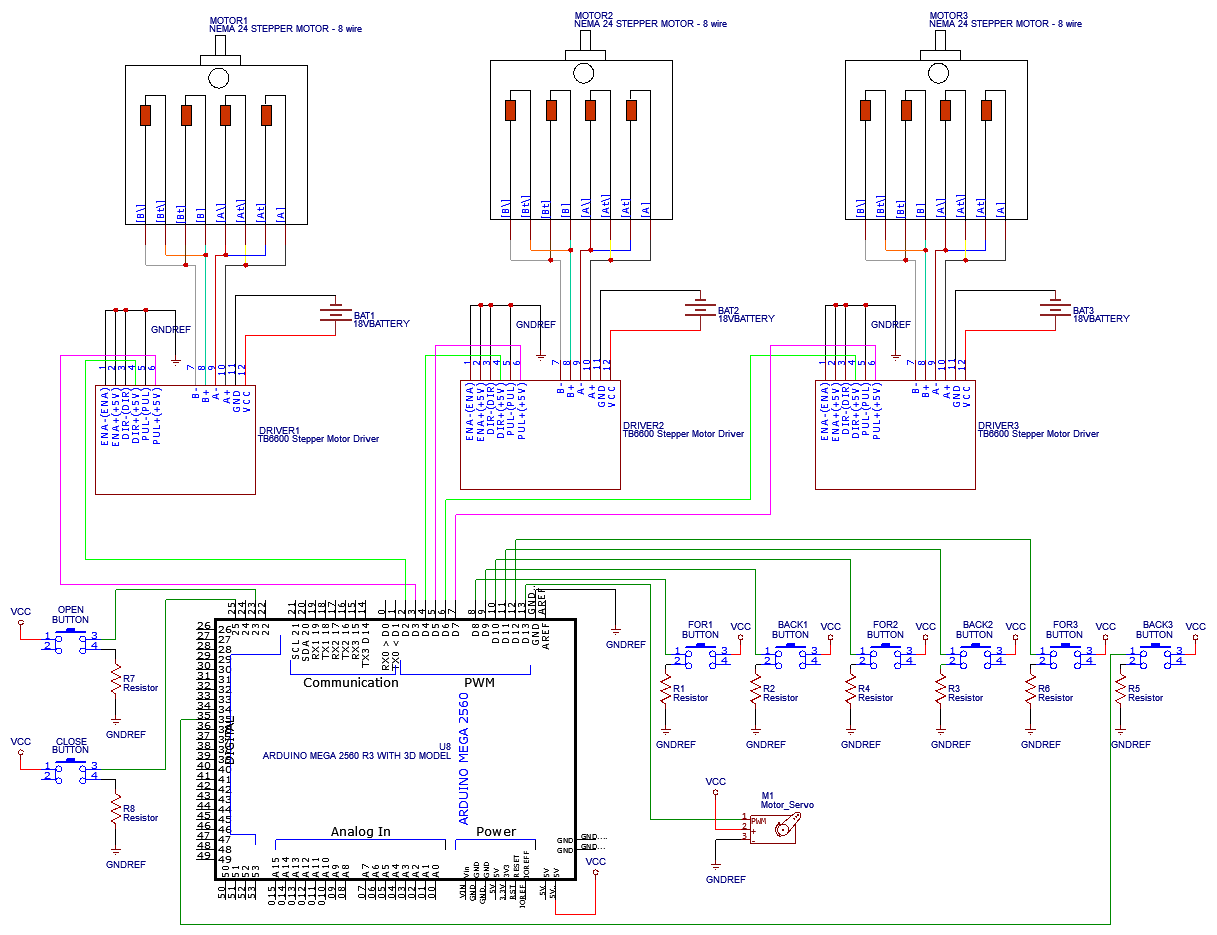
# Electronics

## Wiring Diagram

The wiring diagram for the robotic arm is shown in Figure 8. It includes a detailed view of how the stepper motors, motor drivers, batteries, Arduino microcontroller, gripper, and buttons connect to each other electrically.

Each stepper motor is connected to one 18 V battery and one TB6600 motor driver. The motors interact with the batteries and the Arduino through the motor drivers. Commands from the Arduino board cause the stepper motors to move. Buttons connected to the Arduino determine which commands will be sent to the stepper motors. In this way, the user can control the stepper motors by pressing the buttons.

The gripper’s stepper motor is also controlled by buttons connected to the Arduino. It does not require an external battery or a motor driver. As such, the gripper’s stepper motor is connected directly to the Arduino board.



*Figure 10: Wiring Diagram*

### Components

* + - 3 NEMA 24 8-wire stepper motors
    - 3 TB6600 Stepper Motor Drivers
    - 3 18 V batteries
    - 1 Gripper with a built-in servo motor
    - 1 Arduino Mega board
    - 8 pushbuttons (labeled FOR1, BACK1, FOR2, BACK2, FOR3, BACK3, OPEN, and CLOSE)
    - 8 Resistors (all 100 kΩ)

## Code

The code used in the Arduino Mega turns button presses to commands that move the stepper motors and the gripper. Buttons FOR1 and BACK1 move Motor 1 forwards and backwards. Buttons FOR2 and BACK2 move Motor 2 forwards and backwards. Buttons FOR3 and BACK3 move Motor 3 forwards and backwards. Buttons OPEN and CLOSE open and close the gripper by moving its servo motor.

The full code can be found in Appendix C. The following is a detailed explanation of the code. Please refer to Appendix C to follow along with the explanation.

### Code Explanation

During the setup stage of the code, pins are declared as input or output pins. Pins 2, 3, 4, 5, 6, and 7 are declared as output pins and will be used to control the pulse and direction pins of the stepper motor drivers. Pin 13 is attached to the servo motor. Pins 8, 9, 10, 11, 12, 35, 23, and 25 are declared as input pins and will receive input from the pushbuttons. The stepper motors are then initialized in the forward (positive) direction, and the gripper is initialized in its fully open position (pos = 0).

In the loop stage of the code, the code is separated into sections for motor 1, motor 2, motor 3, and the gripper. During each loop iteration, each of these sections is run once. The loop is run continuously for the remainder of the code’s operation.

The section for motor 1 starts by reading in the inputs for pushbuttons FOR1 and BACK1. If the button FOR1 is pressed, the direction of motor 1 is set to the forwards direction (direction pin set to HIGH) and the stepper motor is moved 1 step (pulse pin turned on and off once). If the button FOR1 is not pressed and the button BACK1 is pressed, the direction of motor 1 is set to the backwards direction (direction pin set to LOW) and the stepper motor is moved 1 step (pulse pin turned on and off once). If neither button is pressed, the section for motor 1 is skipped. Please note that the way this code is set up makes it so that if both buttons are pressed, the stepper motor will move forwards. This gives precedence to the button FOR1 over the button BACK1.

The sections for motors 2 and 3 work in the same way as the section for Motor 1 except that they move motors 2 and 3 with inputs from buttons FOR2 and BACK2, and FOR3 and BACK3, respectively.

The section for the gripper starts by checking the inputs for buttons OPEN and CLOSE. An important consideration for this section is that 0 degrees indicates the position of the gripper when fully open, and 180 degrees indicates the position of the gripper when fully closed. If the CLOSE button is pressed and the gripper’s position is less than 180 degrees, the gripper is closed by 1 degree (position increased by one degree, pos += 1). If the CLOSE button is not pressed, the OPEN button is pressed, and the gripper’s position is more than 0 degrees, then the gripper is opened by 1 degree (position decreased by one degree, pos -= 1). If neither button is pressed, the gripper stays in the same position. Please note that the code is set up so that if both buttons are pressed, the gripper will close. This prevents the gripper from releasing its grip of the fuse if both buttons are pressed by accident. This means that the CLOSE button has precedence over the OPEN button.

# Integration

## MATLAB Script

Appendix D contains the MATLAB script used to calculate the inverse kinematics of the robotic arm. The equations in the next section were used to create this MATLAB script.

## Equations of motion

1. **theta1 = atand(y/x);**
2. r1 = sqrt(x^2 + y^2);
3. r2 = z – a1;
4. phi2 = atand(r2/r1);
5. r3 = sqrt(r1^2 + r2^2)
6. phi1 = acosd((r3^2 + a2^2 – a3^2)/(2\*r3\*a2));
7. **theta2 = phi1 + phi2;**
8. phi3 = acosd((a3^2 + a2^2 – r3^2)/(2\*a3\*a2));
9. **theta3 = phi3 – 180;**

Chart

Description automatically generated with low confidence

r3

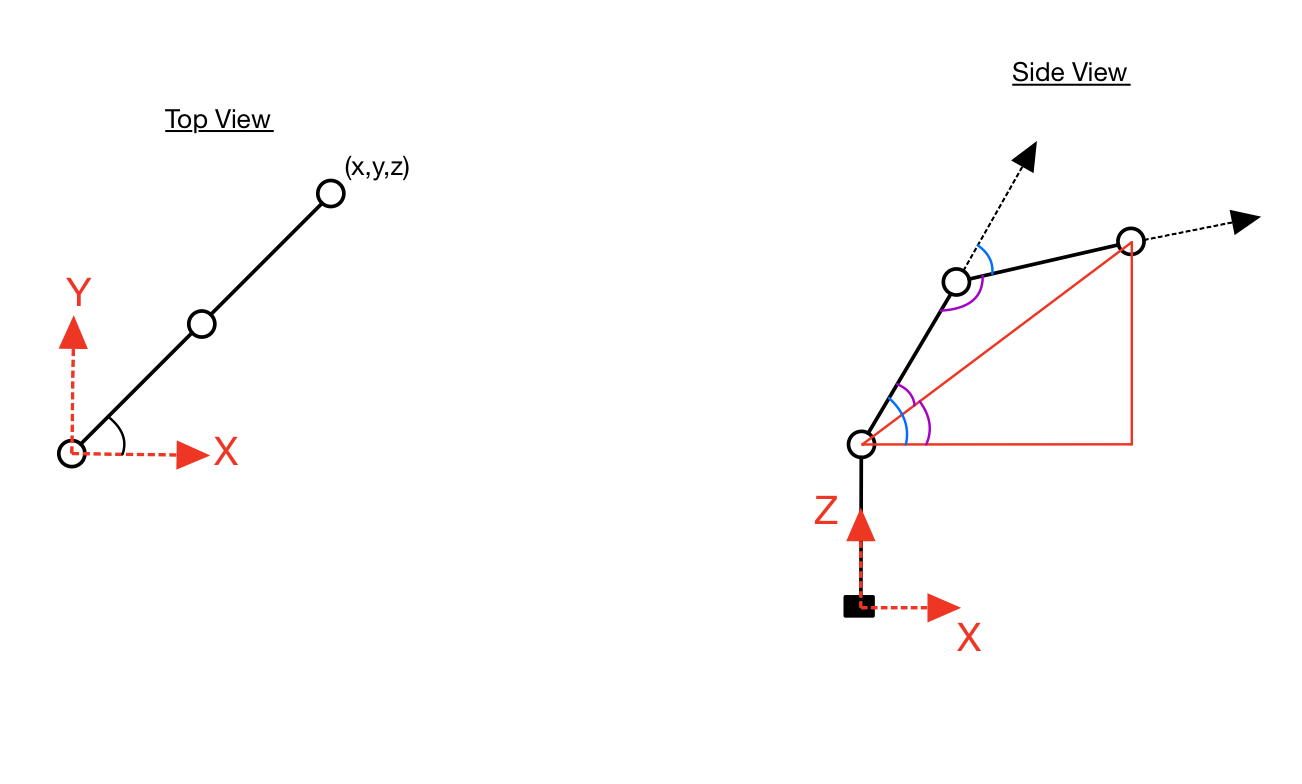
****

Figure 9: Reference Images for the Equations of Motion

θ1

φ3

φ2

φ1

θ2

r1

r2

a2

θ3

a1

# Operation

## Stepper Motor Instructions

There are eight buttons on the robot arm controller: FOR1, BACK1, FOR2, BACK2, FOR3, BACK3, CLOSE, and OPEN.

* FOR1 is used for moving motor 1 (joint 1) forwards
* BACK1 is used for moving motor 1 (joint 1) backwards
* FOR2 is used for moving motor 2 (joint 2) forwards
* BACK2 is used for moving motor 2 (joint 2) backwards
* FOR3 is used for moving motor 3 (joint 3) forwards
* BACK3 is used for moving motor 3 (joint 3) backwards
* CLOSE is used for moving the gripper to the closed position
* OPEN is used for moving the gripper to the open position

When no buttons are pressed, none of the motors will move.

When FOR1 is pressed, motor 1 will move forwards. When BACK1 is pressed, motor 1 will move backwards. If both FOR1 and BACK1 are pressed, FOR1 will take precedence and the motor will move forwards.

FOR2 and BACK2 control motor 2 in the same way that FOR1 and BACK1 control motor 1. FOR3 and BACK3 will control motor 3 in the same way as well.

When CLOSE is pressed, the gripper will begin closing. The gripper will not move if this button is pressed while the gripper is fully closed. When OPEN is pressed, the gripper will begin opening. The gripper will not move if this button is pressed while the gripper is fully open. If both CLOSE and OPEN are pressed, CLOSE will take precedence and the gripper will begin closing.

Motors 1, 2, 3, and the gripper can move at the same time. This means that if FOR1 and BACK1 are pressed at the same time, both motors will move forwards. The operation of buttons FOR1 and BACK1 is independent from the operation of buttons FOR2 and BACK2, of buttons FOR3 and BACK3, and of the gripper.

# Troubleshooting

## If the telescoping base does not move

1. Check the pulley connections between each layer are attached

2. Check the steel cable for any damages

3. Check the DC motor

4. Check the hand winch

## If the robotic arm does not move

1. Check to see if all the connections are wired incorrectly

2. Check to see if the batteries are dead

3. Check the connection to the controller

4. Check the stepper motors and gearboxes for the correct wiring connection

5. Check the Motor drivers for the correct wiring connections

## If the gripper does not move

1. Check the wiring of the gripper

2. Check the servo motor for any damages

3. Check the gripper for any damages

## Emergency Procedure

In the event of an emergency, power down the robotic arm on the controller and return the base to its original height. If necessary, manually unplugging the batteries will turn the device off. Report the incident to a supervisor. If any injuries occur call 911.

# Appendices

## Robotic Arm CAD Drawings

**Diagram, engineering drawing

Description automatically generated**

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

## Telescoping Pole CAD Drawings

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram

Description automatically generated

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated

## Arduino Mega Code

#include <Servo.h>

int dir1 = 2; // Motor 1 direction pin

int pul1 = 3; // Motor 1 pulse pin

int dir2 = 4; // Motor 2 direction pin

int pul2 = 5; // Motor 2 pulse pin

int dir3 = 6; // Motor 3 direction pin

int pul3 = 7; // Motor 3 pulse pin

// Declaring button inputs

int for1 = 8; // Moves motor 1 forwards

int back1 = 9; // Moves motor 1 backwards

int for2 = 10; // Moves motor 2 forwards

int back2 = 11; // Moves motor 2 backwards

int for3 = 12; // Moves motor 3 forwards

int back3 = 35; // Moves motor 3 backwards

int servoOpen = 23; // Opens servo motor

int servoClose = 25; // Closes servo motor

// Servo motor pins

Servo gripper; // Create gripper servo object

int servoPin = 13; // Servo motor pin

int pos = 0; // Stores servo position

void setup() {

// Declare direction and pulse pins as output pins

pinMode(dir1, OUTPUT); // pin 2 - direction 1

pinMode(pul1, OUTPUT); // pin 3 - pulse 1

pinMode(dir2, OUTPUT); // pin 4 - direction 2

pinMode(pul2, OUTPUT); // pin 5 - pulse 2

pinMode(dir3, OUTPUT); // pin 6 - direction 3

pinMode(pul3, OUTPUT); // pin 7 - pulse 3

gripper.attach(servoPin); // pin 13 - gripper servo

// Declare button pins as input pins

pinMode(for1, INPUT); // pin 8 - FOR1

pinMode(back1, INPUT); // pin 9 - BACK1

pinMode(for2, INPUT); // pin 10 - FOR2

pinMode(back2, INPUT); // pin 11 - BACK2

pinMode(for3, INPUT); // pin 12 - FOR3

pinMode(back3, INPUT); // pin 35 - BACK3

pinMode(servoOpen, INPUT); // pin 23 - OPEN

pinMode(servoClose, INPUT); // pin 25 - CLOSE

// Start with forwards direction

digitalWrite(dir1, HIGH);

digitalWrite(dir2, HIGH);

digitalWrite(dir3, HIGH);

// Start with gripper fully open

gripper.write(pos); // pos = 0 - fully open

}

void loop() {

// ----- MOTOR 1 ----- //

// Move forwards if FOR1 button is pressed

if(digitalRead(for1) == HIGH) {

// Set direction to forwrds

digitalWrite(dir1, HIGH);

// Move the motor

digitalWrite(pul1, HIGH);

digitalWrite(pul1, LOW);

}

// Move backwards if BACK1 button is pressed

else if(digitalRead(back1) == HIGH) {

// Set the direction to backwards

digitalWrite(dir1, LOW);

// Move the motor

digitalWrite(pul1, HIGH);

digitalWrite(pul1, LOW);

}

// ----- END OF MOTOR 1 ----- //

// ----- MOTOR 2 ----- //

// Move forwards if button FOR2 is pressed

if(digitalRead(for2) == HIGH) {

// Set direction to forwards

digitalWrite(dir2, HIGH);

// Move the motor

digitalWrite(pul2, HIGH);

digitalWrite(pul2, LOW);

}

// Move backwards if button BACK2 is pressed

else if(digitalRead(back2) == HIGH) {

// Set direction to backwards

digitalWrite(dir2, LOW);

// Move the motor

digitalWrite(pul2, HIGH);

digitalWrite(pul2, LOW);

}

// ----- END OF MOTOR 2 ----- //

// ----- MOTOR 3 ----- //

// Move forwards if FOR3 button is pressed

if(digitalRead(for3) == HIGH) {

// Set direction to forwrds

digitalWrite(dir3, HIGH);

// Move the motor

digitalWrite(pul3, HIGH);

digitalWrite(pul3, LOW);

}

// Move backwards if BACK3 button is pressed

else if(digitalRead(back3) == HIGH) {

// Set the direction to backwards

digitalWrite(dir3, LOW);

// Move the motor

digitalWrite(pul3, HIGH);

digitalWrite(pul3, LOW);

}

// ----- END OF MOTOR 3 ----- //

// ----- GRIPPER ----- //

// Move servo

if(pos < 180 && digitalRead(servoClose) == HIGH) {

// close servo by 1 deg

pos += 1;

gripper.write(pos);

delay(5);

}

else if(pos > 0 && digitalRead(servoOpen) == HIGH) {

// open servo by 1 deg

pos -= 1;

gripper.write(pos);

delay(5);

}

// ----- END OF GRIPPER -----//

// Vary this delay to change motor speed

delayMicroseconds(400);

}

## MATLAB Script for Inverse Kinematics

%% INTRODUCTION

% Name: Davis Goolsby

% Assignment: Equations of Motion for Robotic Arm

clc

clear all

close all

format compact

%% 3D

% Theta 1 - Rotation of vertical axis

% Theta 2 - Rotation of the bicep

% Theta 3 - Rotation of the forearm

% Arm lengths:

base\_height = 18;

bicep\_length = 22;

forearm\_length = 18;

% Prompt user for desired position:

prompt = "Input your desired x value in inches: ";

x = input(prompt);

prompt = "Input your desired y value in inches: ";

y = input(prompt);

prompt = "Input your desired z value in inches: ";

z = input(prompt);

if (x>(bicep\_length+forearm\_length)||...

(y>bicep\_length+forearm\_length)||...

(z>bicep\_length+forearm\_length+base\_height))

fprintf('Invalid coordinate input.\n');

else

% Inverse kinematics:

theta1 = atand(y/x);

r1 = sqrt(x^2 + y^2);

r2 = z - base\_height;

phi2 = atand(r2/r1);

r3 = sqrt(r1^2 + r2^2);

phi1 = acosd((r3^2 + bicep\_length^2 - (forearm\_length^2))/...

(2 \* bicep\_length \* r3));

theta2 = phi2+phi1;

phi3 = acosd((forearm\_length^2 + bicep\_length^2 - (r3^2))/...

(2 \* bicep\_length \* forearm\_length));

theta3 = phi3-180;

% Test whether the desired point is reachable:

test1 = isreal(theta1);

test2 = isreal(theta2);

test3 = isreal(theta3);

% Return angles:

if (test1 == 1) && (test2 == 1) && (test3 == 1)

fprintf('Angle of link 1 is: %f\n', theta1);

fprintf('Angle of link 2 is: %f\n', theta2);

fprintf('Angle of link 3 relative to link 2 is: %f\n', theta3);

else

fprintf('Invalid coordinate input.\n');

end

end

%Plot points of interest:

zero = 0;

shoulderx = 0;

shouldery = 0;

shoulderz = base\_height;

elbowx = bicep\_length\*cosd(theta2)\*cosd(theta1);

elbowy = bicep\_length\*cosd(theta2)\*sind(theta1);

elbowz = base\_height+(bicep\_length\*sind(theta2));

end\_effectorx = elbowx + (forearm\_length\*cosd(theta2+theta3)\*cosd(theta1));

end\_effectory = elbowy + forearm\_length\*cosd(theta2+theta3)\*sind(theta1);

end\_effectorz = elbowz + forearm\_length\*sind(theta2+theta3);

zero = [0 0 0];

shoulder = [shoulderx shouldery shoulderz];

elbow = [elbowx elbowy elbowz];

end\_effector = [end\_effectorx end\_effectory end\_effectorz];

plot = [end\_effector;elbow;shoulder];

figure(1)

plot3(plot(:,1),plot(:,2),plot(:,3),'r');

grid on

xlabel("x"); ylabel("y"); zlabel("z");

xlim([-50 50]); ylim([-50 50]); zlim([-50 50]);