

1.2.5 Mechanical Winch Project

Name of Product: Team Winch #5

Designers: Khushi Gupta, Ethan Lau, Donna Prince, and Rohin Sampeur

Duration: October 17, 2018 to October 24, 2018

Course: Principles of Engineering

Period: 7

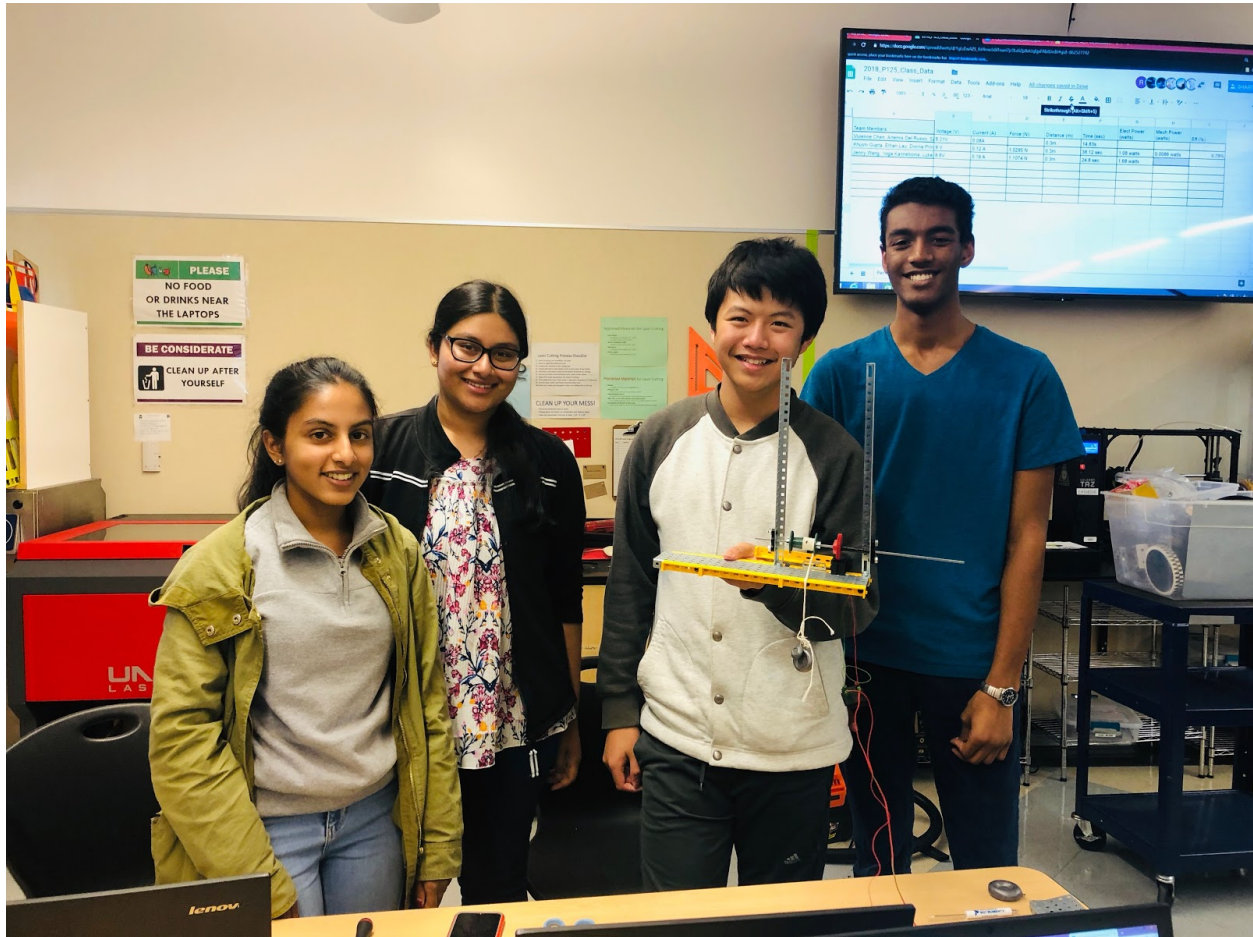


Table of Contents

Title Page	1
Table of Contents	2
Design Brief	3
NetLogo: Simulation and Code Images	4-6
Calculations	7
Product Final Image and Description	8-9
Reference List	10

Design Brief

Client/Target Consumer: Local construction company needs a lifting mechanism, to move an object 3 floors up.

Designers: Khushi Gupta, Ethan Lau, Donna Prince, Rohin Sampeur

Problem Statement: It is hard to efficiently lift materials up 3 floors by hand without a machine.

Design Statement: Design, build, and test a mechanical winch system prototype to lift up an object.

Criteria and Constraints: Prototype must lift at least 100 grams, and must be 30 centimeters over the side of the table. The winch system must be built with VEX or FT parts, powered by an FT motor and turned on an off by a switch. The prototype must be built in four days. You may use any material approved by your instructor to build the prototype.

Deliverables

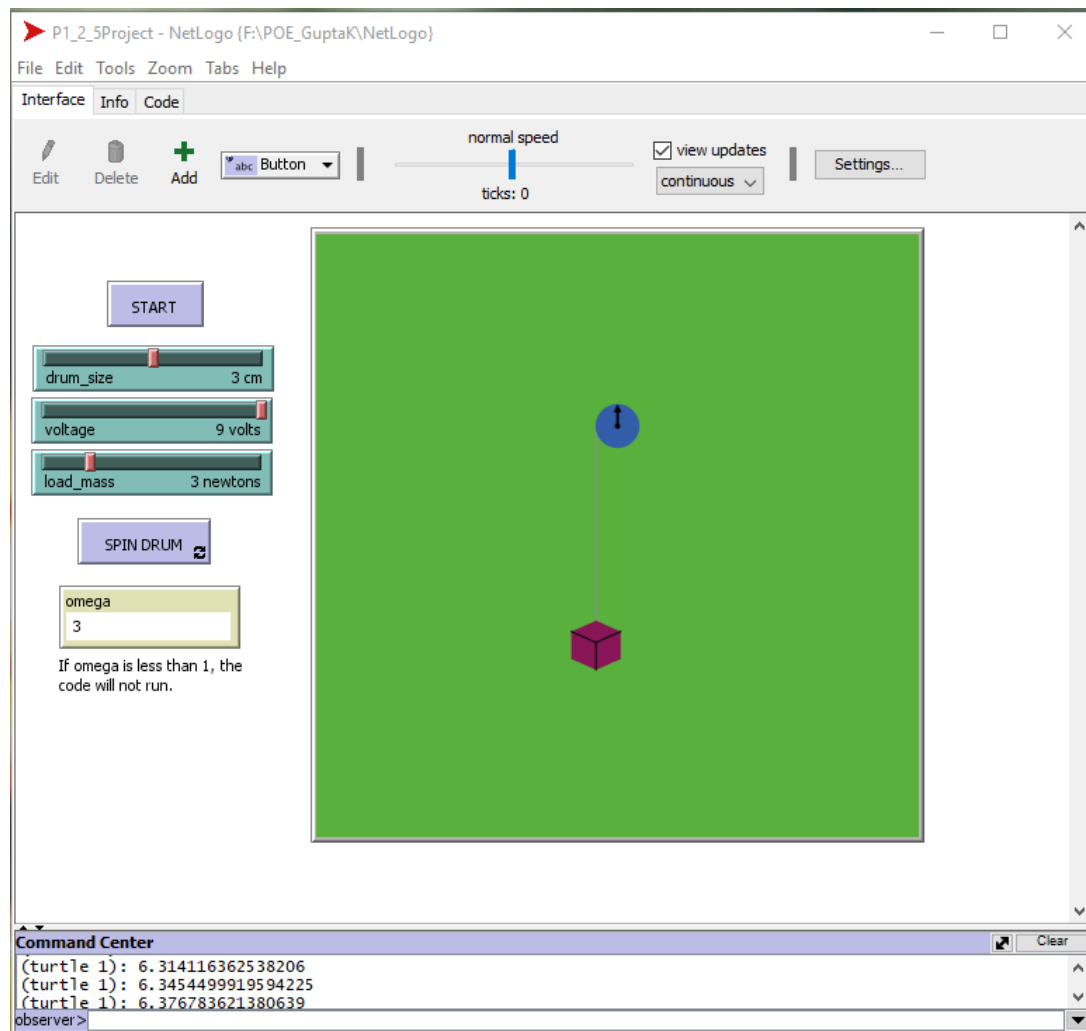
Team: Working Prototype, NetLogo Simulation and Report that includes: Title Page, Table of Contents, Design Brief, Photo of Final Product, a Paragraph description of final prototype, NetLogo Simulation Screenshots, NetLogo code/comments and a Reference list of sources.

Individual (Items in Notebook): Project Log of tasks completed each day, project setup, table of measurements, calculations of MA for final design (including clear measurements, formulas, for Work, Mechanical Output Power, Electrical Input Power, and Efficiency) and Conclusion Questions.

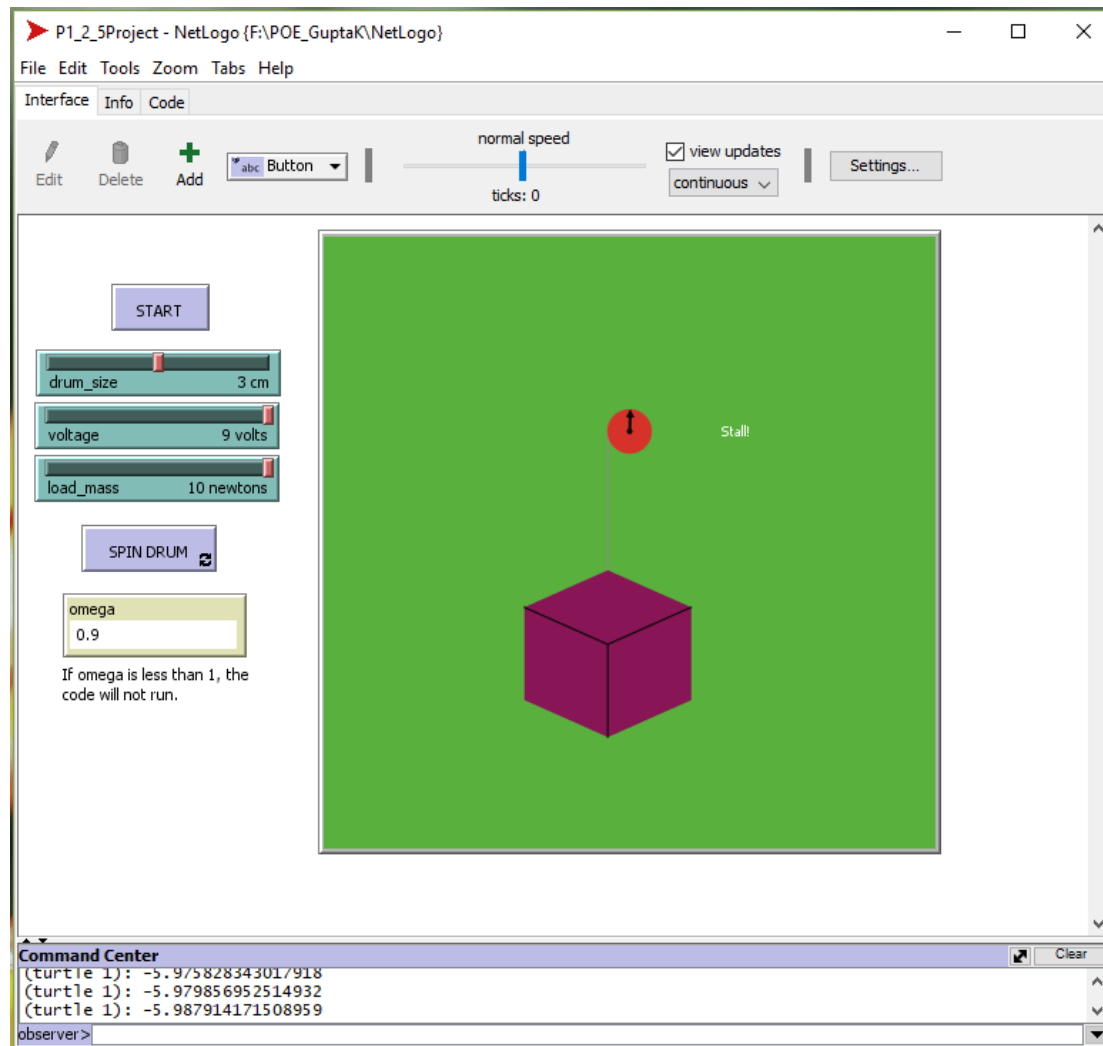
Netlogo: Simulation and Code Images

Stimulation

This screenshot depicts the GUI (General User Interface). It is an example of when there is enough power to run the stimulation. On the GUI, there is the START button to set up the world. One can then manipulate the drum_size(in centimeters), voltage(in volts), and load_mass(in newtons) sliders as they desire to view the effects of changing the variables or to match their desired real-life winch. One can then click, the SPIN DRUM button to watch the code in action and to view the effects of the sliders on the stimulation itself. The stimulation will stop once it reaches the top of the drum. There is also a monitor in the GUI that shows the omega as well as a message underneath. This is for the user to know that the code will not run if there is not enough power. For there to be enough power, the omega must be greater than 1. In the command center, the y-coordinates are shown when the code is run the view the exact distance the box is traveling.

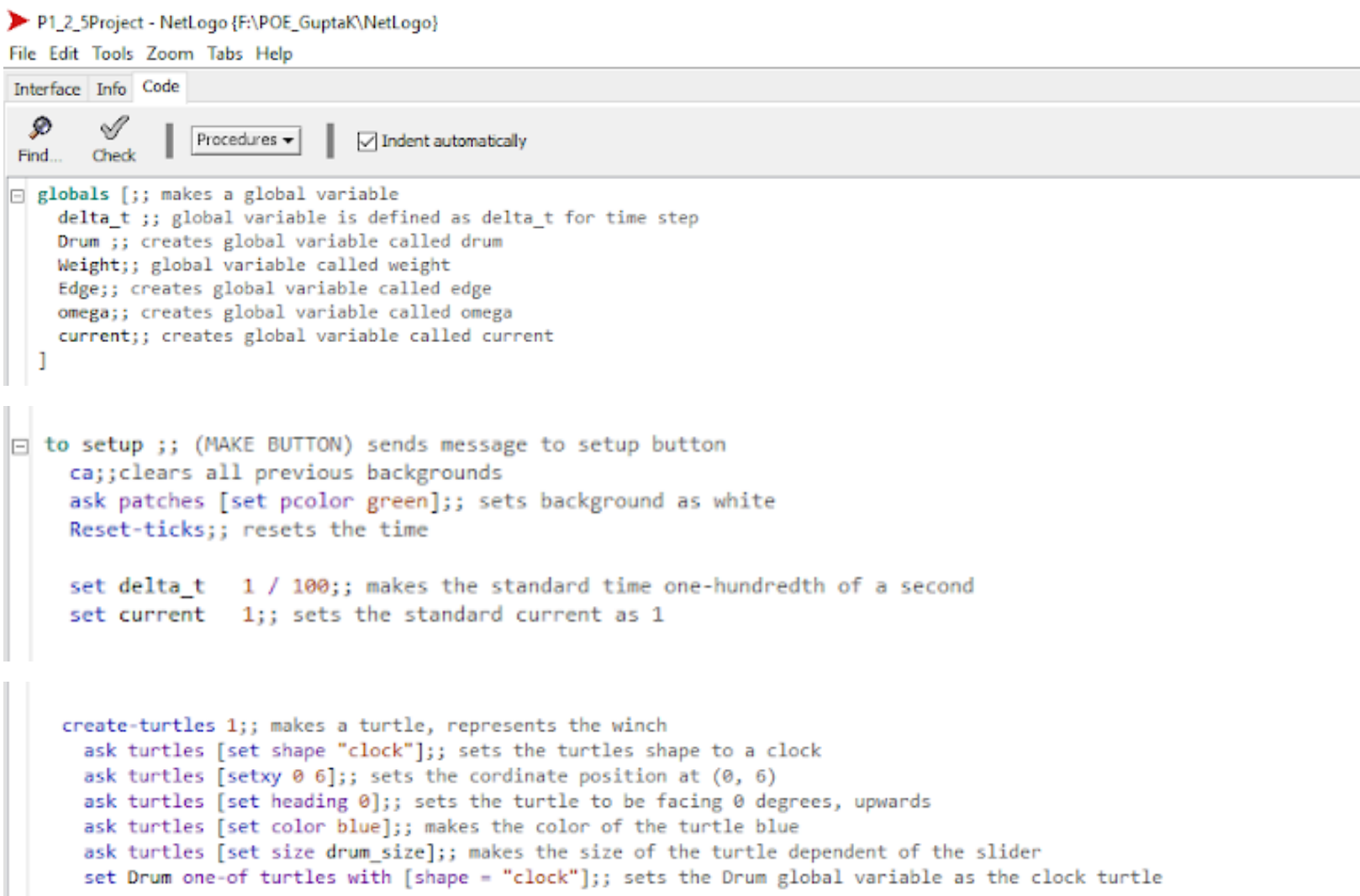


This is an example of how the GUI will look like when there is not enough power to run the stimulation. The drum will turn red and the message “Stall!” will be shown. This can easily be fixed by tampering with the sliders so that the power(ω) is once again more than one.



Code:

These are the screenshots of the code, done on NetLogo. It first starts off with creating the global variables that will be used throughout the code. Then, the first set of code is made for the START button. This includes the creation of the three turtles (box, dot, and clock), as well as their specific details such as color and location. It also includes the sliders for changing the specific variables. Lastly, we set specific values for Δt and current. The second set of instructions is to go which controls the SPIN DRUM button. This controls the speed at which the drum will spin and how fast the box will move up. The last part of this code controls whether the code will run which is dependent on the value of ω . Below are pictures of the code:



The screenshot shows the NetLogo code editor with the following code:

```

globals [;; makes a global variable
  delta_t ;; global variable is defined as delta_t for time step
  Drum ;; creates global variable called drum
  Weight;; global variable called weight
  Edge;; creates global variable called edge
  omega;; creates global variable called omega
  current;; creates global variable called current
]

to setup ;; (MAKE BUTTON) sends message to setup button
  ca;;clears all previous backgrounds
  ask patches [set pcolor green];; sets background as white
  Reset-ticks;; resets the time

  set delta_t 1 / 100;; makes the standard time one-hundredth of a second
  set current 1;; sets the standard current as 1

  create-turtles 1;; makes a turtle, represents the winch
  ask turtles [set shape "clock"];; sets the turtles shape to a clock
  ask turtles [setxy 0 6];; sets the coordinate position at (0, 6)
  ask turtles [set heading 0];; sets the turtle to be facing 0 degrees, upwards
  ask turtles [set color blue];; makes the color of the turtle blue
  ask turtles [set size drum_size];; makes the size of the turtle dependent of the slider
  set Drum one-of turtles with [shape = "clock"];; sets the Drum global variable as the clock turtle

```

```

crt 1 [set shape "box";; sets the shape to a box, represents the weight lifted
set heading 0 ;; sets the direction pointing at 0 degrees
set color 124 setxy (-3.14 / 8 * drum_size) (min-pycor + 10);; sets the color of turtle,
;;tells it the coordinates to appear, the minimum pycor is the variable plus 10
set size load_mass;; uses the load_mass slider to determine size of the box
set Weight one-of turtles with [shape = "box"];; sets the Weight global variable as the box turtle
]

```

```

crt 1 [set shape "dot";; makes a turtle with the shape box, represents the string
set size 0.1 ;; sets size of turtle
setxy (-3.14 / 8 * drum_size) (max-pycor - 10)];; creates coordinate that represents connection between weight and drum
set Edge one-of turtles with [shape = "dot"];; sets the global variable Edge to represent this turtle

ask Weight [create-link-with Edge];; connects the Weight turtle to the Edge turtle

set omega current * voltage / load_mass;; sets omega to equal current times voltage divided by load mass

end;; finishes setup function

```

```

to go;; sends message to go
ask Drum [right 180 / pi * 5 * delta_t];; makes turtle move to the amount of degrees determined by angular speed(5) slider every 0.01 seconds

wait delta_t;; tells the function to wait 0.01 seconds before performing the function repeatedly
tick-advance delta_t;; increase amount of ticks in 0.01 seconds

ask Weight [fd (omega * delta_t * 180 / pi) * drum_size / 128 show ycor];; tells the coordinates that the turtle should move up
if [ycor] of weight > 6 [stop];; tells the turtle to stop once it had reached the top

if omega < 1 [ask Drum [set color red ask patch (max-pxcor - 10) (max-pycor - 10) [set plabel "Stall!"]] stop];; if not enough
;;power to run motor, will not run code

end;; finishes go function

```

Calculations

Time (t)	$t = 38.12 \text{ sec}$
Distance (d)	$d = 30 \text{ cm} \times \frac{1 \text{ m}}{100 \text{ cm}} = 0.3 \text{ m}$
Force (F)	$F = 291.4 \text{ g} \times \frac{0.9807 \text{ N}}{100 \text{ g}} = 2.8577598 \text{ N}$
Current (A)	$I = 0.12 \text{ A}$
Voltage(v)	$V = 9 \text{ V}$

Work

$$Work = F \times d = (2.8577598 \text{ N})(0.3 \text{ m}) = 0.85732794 \text{ joules}$$

Power Output

$$P_{out} = \frac{w}{t} = \frac{0.85732794 \text{ joules}}{38.12 \text{ sec}} = 0.02249023976 \text{ watts}$$

Power Input

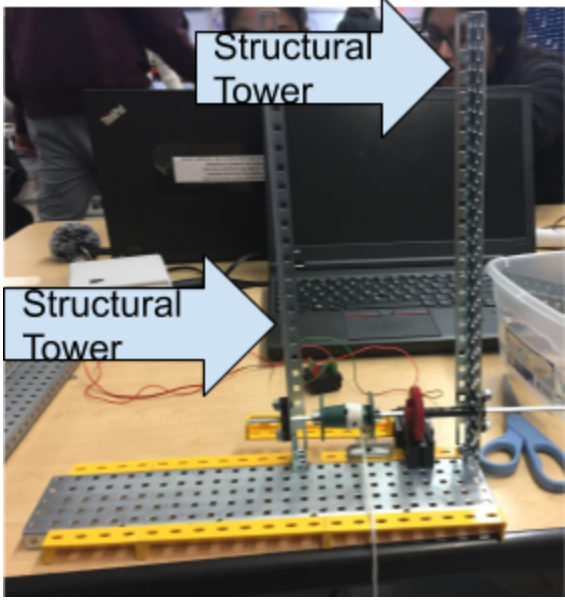
$$P_{in} = IV = (0.12 \text{ A})(9 \text{ V}) = 1.08 \text{ watt}$$

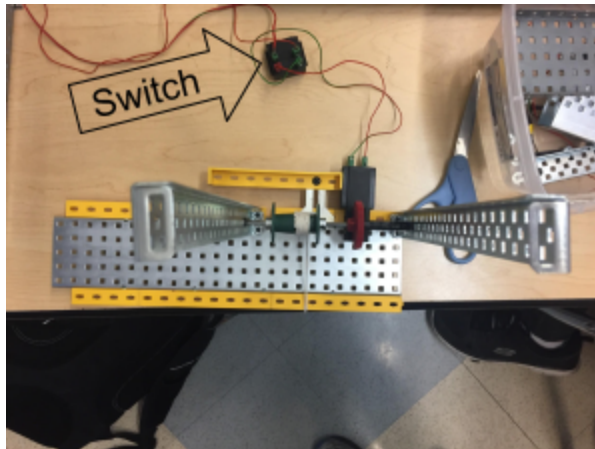
Efficiency

$$Eff = \frac{P_o}{P_i} \times 100\% = \frac{0.02249023976 \text{ watt}}{1.08 \text{ watt}} \times 100\% = 2.1\%$$

Prototype Final Image and Description

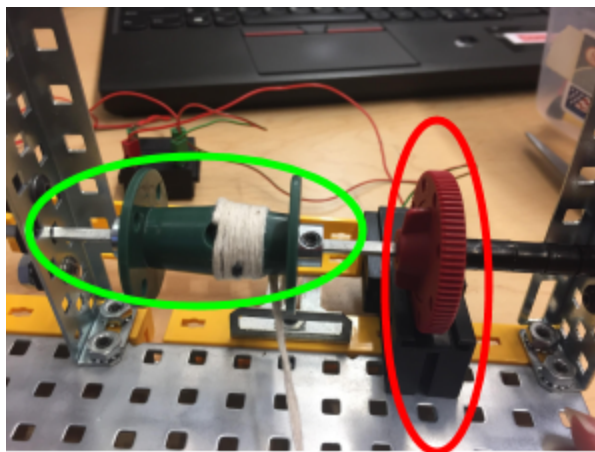
The scaffolding consists of a relatively large base along with two vertical rails. The axle is placed through the two rails secured through a barring. Around the axle contains a drum which pulls up the weight and a gear. The motor turns the red gear which turns the axle and powers the drum. As the drum turns, the string wraps around it which in turn pulls up the weight attached on the other side. This motor is secured on the rails with Fischertech pieces and angle gussets. This motor is then connected to a switch through two separate wires, which is then powered by a DC power converter through another two wires. The final design was able to pull a weight of 291.4g a distance of 30cm in a time of 38.12s. The amperage was measured to be 0.12A and the voltage was measured to be 9V. We calculated an output power of 0.0225 watts and an input power of 1.08 watts, thereby creating an efficiency of 2.1%.

Picture with Annotation	Description
	<p><u>Side View of Mechanism</u></p> <p>The structural towers(made with VEX rails) gives our mechanism's axle support. It also allows us to place the axle at the correct height to align the red FischerTech gear with the FischerTech Motor.</p>



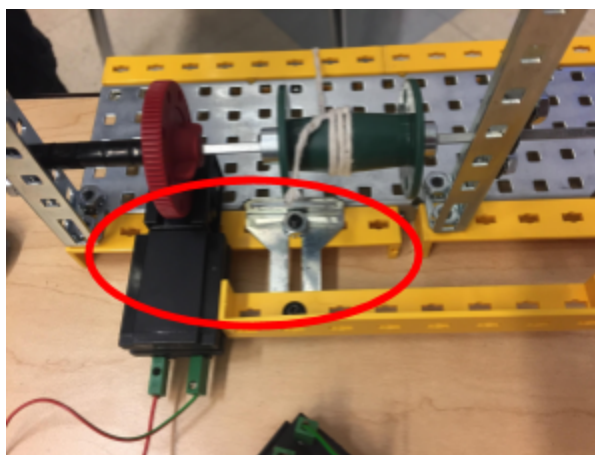
Top View of Mechanism

The Fischer Tech switch that we used allows the electricity from our power supply to activate our motor, which turns our drum. This type of switch also lets us turn our drum forwards and backwards.



Front of Gearing/Motor

As the motor spins the red gear, our green drum spins as well, wrapping the attached string around it. This pulls whatever is attached to the end of the string. We used spacing to make sure that the gears meshed correctly and collars to keep our drum from moving left or right when it was turning. The bearings also gave us less room for the axle to move which made it spin easier.



Back of Gearing/Motor

To attach our motor in a way so that it would not move when powered, we used the yellow angle channels and angle gussets. This attachment adds more support to the motor and also raises it so it can reach the red gear.

Reference List

P. (n.d.). Retrieved October 23, 2018, from PLTW Engineering Formula Sheet 2018 (v18.0)

VEX EDR. (2016). Retrieved September 5, 2018, from <https://www.robotmesh.com/>