
846
DESIGN
HIGHLIGHTS
2019

Intentionally Blank

TEAM 846
THE FUNKY MONKEYS

present

SPACEREX

Intentionally Blank

Climber a.k.a. “The Cobra”

Garrett Peake (Senior)

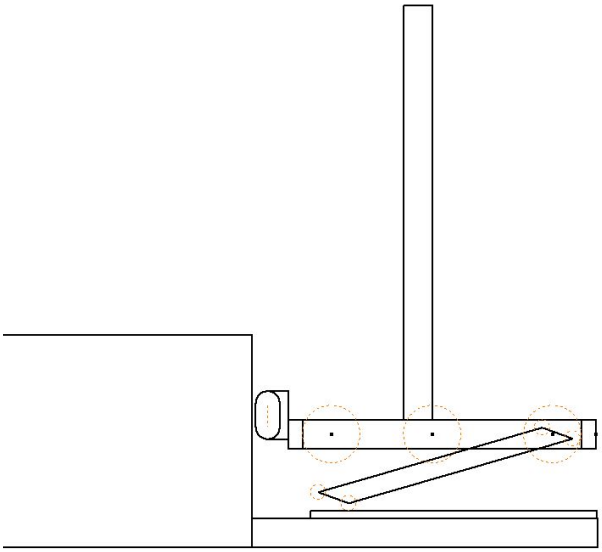
The climber lifts the robot to the level 3 HAB platform. The four bar linkage design allows for a steady rotational climb while keeping the robot parallel to the ground. A variable reduction gearbox allows for a climb which increases in speed as the robot rises while keeping motor load constant.

Design Requirements:

1. Align for climb within 10 seconds (Actual: <1s)
2. Complete the climb within 10 seconds (Actual: 3.5s)
3. Climb straight from HAB level 1 to HAB level 3

Conceptual Design

Garrett Peake with Hari Parthasarathy (Freshman)



I created sketches to ensure that *The Cobra* would work in many different situations. These sketches helped to find the different lengths and dimensions that were finally transferred to the official design. I only analyzed a level 3 climb because we found that the cycles lost spending time on a level 2 climb were more than the points gained.

I recognized that due to the nature of *The Cobra* sliding on both the ground and the side of the HAB, we didn't need precise alignment to complete the climb.

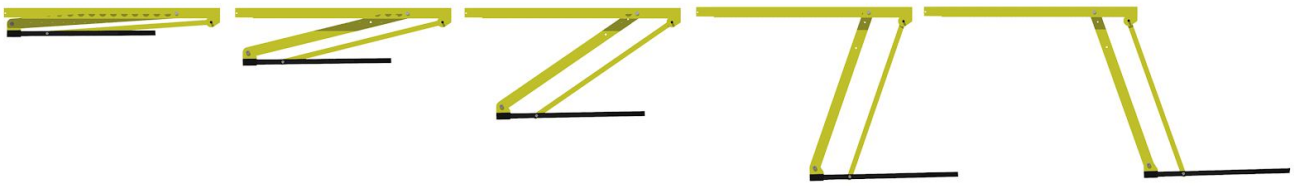
Verifying the concept would work using CAD geometry

Four Bar Linkage

The four bar linkage used by The Cobra to rise while not tipping

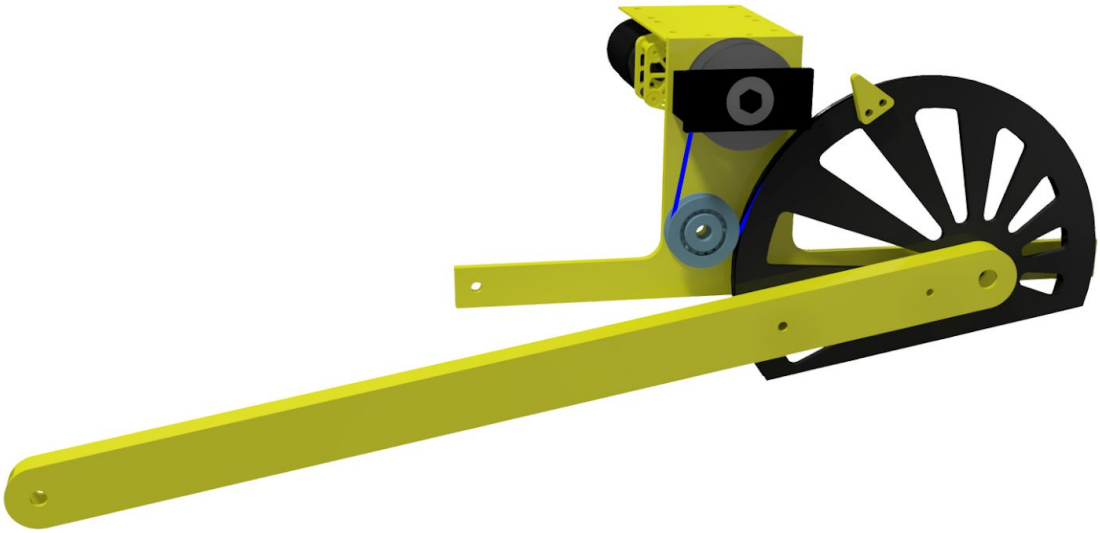
I found that the two non-driven beams were always in tension while the robot's CG was above the foot and the driven beam was under a large torque. Because of this, I used a

smaller material for the non-driven beams. The top linkage consists of beams that span across our drivetrain and the bottom linkage makes up a foot to keep our robot stable during a climb.



A four bar linkage maintains parallel beams during the entire actuation

Gearbox



The gearbox used to lift the robot to level 3

The gearbox uses an eccentric pulley in combination with a planetary gearbox that creates a variable gear ratio increasing *The Cobra's* torque by a factor of 2 at the beginning of the climb and conversely our speed by a factor of two at the end of the climb. The eccentric pulley allows for a constant force on the motors while the torque on the climber changes non-linearly as a sin function.

Required Torque					775 Pro			
	Weight	150	lb			Stall Torque	0.71	N-m
	Max CG distance from Drivetrain Pivot	20	in			Free Speed	18730	rpm
	Torque on Pivot at Beginning	3000	in-lb			Max Power	348.15	W
	Torque on Pivot at Beginning	250	ft-lb					
					Rev NEO			
Required RPS						Stall Torque	2.6	N-m
	Target Climb Time	3	sec			Free Speed	5676	rpm
	Required Climb Angle	135	deg			Max Power	386.35	W
	Output RPS	0.125	RPS					
	Output RPM	7.5	RPM					
Selected Motor								
		775 Pro	ul					
	Stall Torque	0.71	N-m					
	Free Speed	18730	rpm					
	Operating Point	80%	ul					
	Speed @ Operating Point	14984	rpm					
	Torque @ Operating Point	0.142	N-m					
	Power output @ Operating Point	222.82	W					
Required Reduction								
	Overall reduction necessary for speed	1997.9	ul	Uses Motor Data	$Torque\ Ratio = \frac{Input\ Rev}{Output\ Rev}$			
Output Torque								
	Torque Out	283.7	N-m	Uses Motor Data				
	Torque Out	209.2	ft-lb					
Initial Torque on Reduction Before CAM								
	Intial cam radius	6	in					
	Final CAM Radius	4.5						
	Radius of Spool	0.5	in					
	Torque on 2nd to last stage	250	in-lb					
	Torque on 2nd to last stage	20.83	ft-lb	Versaplanetary load rating is 116 ft-lb				
	Intial CAM Reduction	12.00	ul					
	Final CAM Reduction	9.00	ul					
	Additional Reduction Required	166.5	ul					
Safety Factor for Torque								
	Safety Factor	10.0			$Safety\ Factor = \frac{Output\ Torque}{Required\ Torque}$			

Failure Mode	N-m	ft-lbs
1/2" Hex Shaft Yield Stress	157	116
1/2" Round Shaft Yield Stress	130	96
3/8" Hex Shaft Yield Stress	57	42
CIM Motor Shaft Yield Stress	29	21
7:1, 9:1, 10:1 Carrier Plate Yield Stress	100	74

Table 9 – VersaPlanetary Output Torque Limits (Pure Torsion)

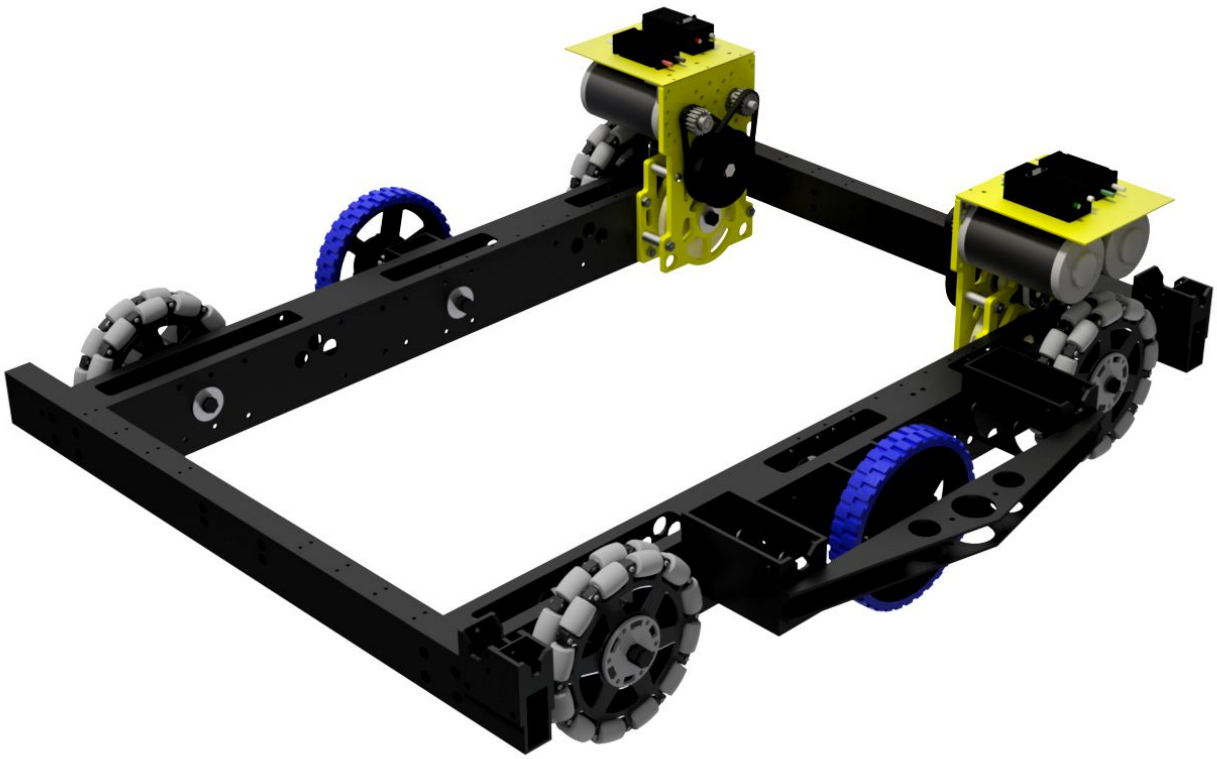
Using a target time, we found the required reduction (Worked on by Atul Nair (Senior))

We knew we wanted a fast climb and by knowing the power limits of our motors, we found the best time we could reasonably achieve. Using this time figure, we found the necessary reduction to achieve it. We found that a climb time of 3 seconds was more than feasible and made that our target (We are currently at 3.5s using only 75% power!)

Drivetrain (Girls' Subsystem)

Eesha Deepak (Senior), Anna Shaposhnik (Junior), Catherine Zheng (Sophomore), Claire Chen (Sophomore)

The drivetrains consist of the gearbox, chain runs, frame, and bumpers.



Design Requirements:

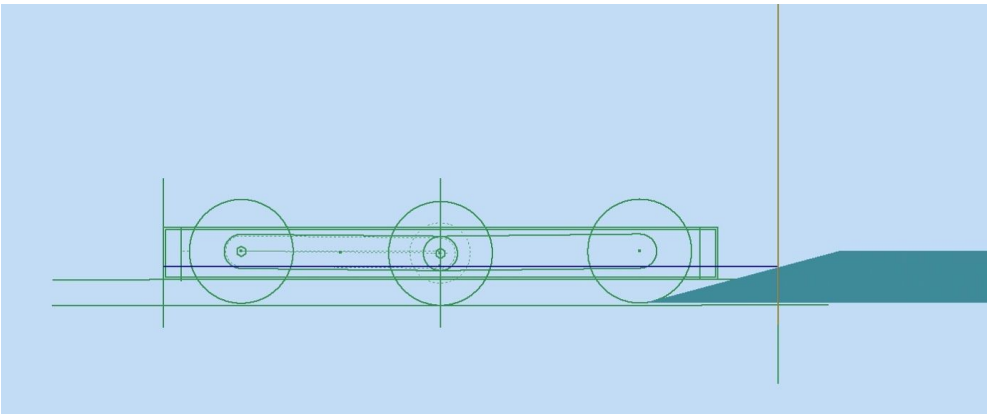
1. Drive quickly around the field
2. Turn quickly
3. Give clearance for The Cobra
4. Clear the cable protectors and platform ramp
5. Minimize rocking

Drivetrain Frame

Eesha, Catherine

This year we chose to have a smaller frame to give benefits to other subsystems. With a smaller frame, we were able to allocate space for our hexagonal bumpers and give the rest of the subsystems extending space. The hexagonal bumpers play an important role in defense with only an increase of 1.6 inches to the perimeter, highlighted further.

We also did some tests to see if we would be able to clear the platform and cable protectors. With these simulations, we came to the conclusion that 6 inch wheels were the best option to balance chain slop and clearing these barriers.



Using CAD to check that our drivetrain can climb the platform.

Additionally, we wanted our robot to be able to turn quickly, but not tip over. Therefore, we used a 6-wheel layout, with omni-wheels on the front and back, and traction wheels in the center, with a slight center drop. This arrangement allows us to make our robot agile and quick to turn while keeping the pushing power of traction wheels. The front and back tubes of the robot are 1 inch higher to allow the Cobra to swing back.

This year we also created an offset block, a block of metal that extends from the side of the frame on the middle wheel shaft. The offset block pushes the traction wheel out 1 inch, increasing the distance from the wheel to the center of the robot (the pivoting point). Similar to pushing a door, the further the wheels are from the pivoting point, the more leverage there is, leading to around 8% more torque when turning the robot.

Sprocket and Chain System		
Wheels Diameter	6	Choose
Wheels Drop	0.075	Final
Chain Height	0.23	Given
Chain Pitch	0.25	Given
Chain Width	0.343	Given
Sprocket Pitch	0.25	Given
Sprocket Hex Diameter	0.5	Given
Sprocket Teeth	18	Choose
Center to Center	11.25	Chain Pitch*# of pitch
Sprocket Pitch Diameter	1.439693	Sprocket Pitch * Sprocket:Teeth / PI
Outer Diameter	1.675693	Given number + Sprocket Pitch Diameter

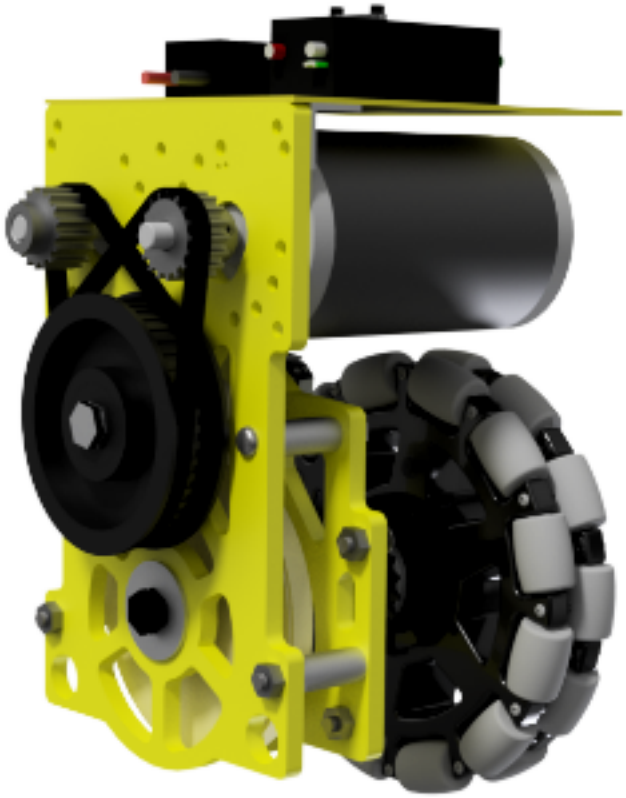
Calculations to find chain length and the distance between bearings

Frame Perimeter Calculations		
Side of Frame	16	
Bumper Tip to Tip	30	Robot Width + 4 in
Bumper Side Length	16.125	Pythagorean Theorem: 0.5 ul * ((((Bumper Tip to Tip- Robot Width) ^ 2 ul + (Robot Length) ^ 2 ul)) ^ 0.5 ul)
Frame Perimeter	116.5	2 ul * Robot Width + 4 ul * Bumper Side Length

Frame perimeter calculations with hexagonal bumpers

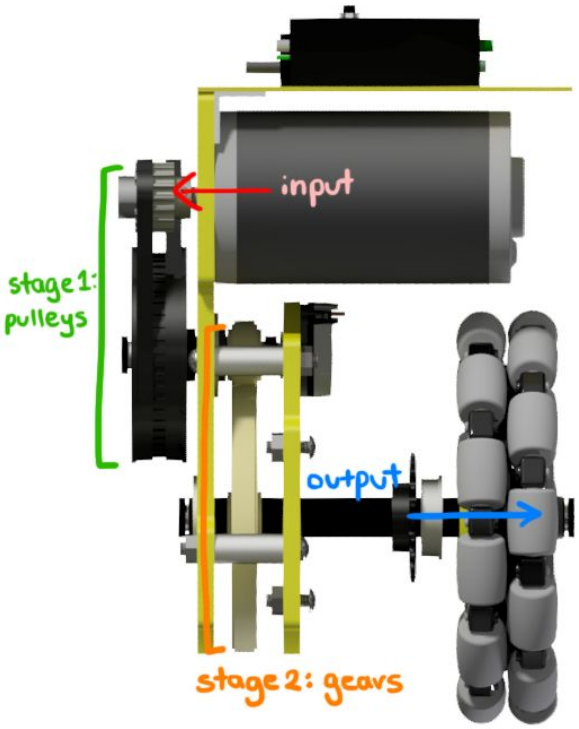
Gearbox

Anna



Our gearbox is specially designed with a systems approach. Features like over-the-top motors and a speed controller mounting plate allow the gearbox to be effective and out of the way of other subsystems. We have two stages, the first uses belts to easily extend across the distance needed to clear the wheels and the second uses gears.

The gearbox is tasked with balancing speed, acceleration, and consequently pushing power (torque) with the gear reduction. Keeping the center to center distance the same between the gear second stage, we are able to vary the reduction by adding multiples of two teeth to one gear and taking it away from the other. We came up with 4 possible free speeds that we will be able to test and change as needed. Because the robot in this year's game is traveling shorter distances, a quick acceleration was more important to us than a fast top speed to get from point A to point B the quickest.



Target RPM Calculations					
	Wheel Diameter	we want →	6	in	0.5 ft
	Target Wheel Speed		15	ft/sec	10800 in/min
	Circumference of Circle		1.57	ft	
	Number of Rotations per Second		9.55	number of circumfereces in 15 ft	
	Number of Rotations per Minute		572.96	RPM	
CIM Motor Specs					
	Free Speed		5330	RPM	
	Free Current		2.7	amps	
	Stall Torque		2.41	N-m	
	Stall Current		131	amps	
Determine target gear ratio					
	Free Speed Wheel Velocity		8372.34	ft/min	
			139.54	ft/sec	
Target Gear Ratio		we need →	9.303	ul	
	DP		20	T/in	

STAGE 1				
	CIM pinion teeth	17	T	min is 17 timing pulley
	CIM pinion pitch diameter	27.056	mm	25.9 mm
	CIM pinion pitch	5	mm	1.01968504 in
	CIM Output Gear Teeth	48	T	timing pulley
	CIM Output Gear pitch Diameter	76.394	mm	
	CIM Output Gear Pitch	5	mm	
	First Stage Reduction	2.823529412	ul	
	Free Speed with First Stage	49.420	ft/sec	
	Second Stage Expected	3.295	ul	
SECONG STAGE CALCULATIONS				
	Sum Teeth	94	Teeth	allows enough space for encoder on top of 3 inch frame
	Center to Center Distance	2.35	in	
	Wheel Input Gear Teeth	22	T	smallest possible is 18
	Wheel Input Pitch Diameter	1.1	in	
	Wheel Input Diameter	1.2	in	
	Wheel Output Gear Teeth	**72**	T	
	Wheel Output Pitch Diameter	3.6	in	
	Wheel Output Diameter	3.7	in	must be less or equal to 3.5
	Second Stage Reduction	3.272727273	ul	
	Actual Gear Reduction	9.2406		→ compare to target, use goal seek
	Actual Wheel Velocity	15.101	ft/sec	

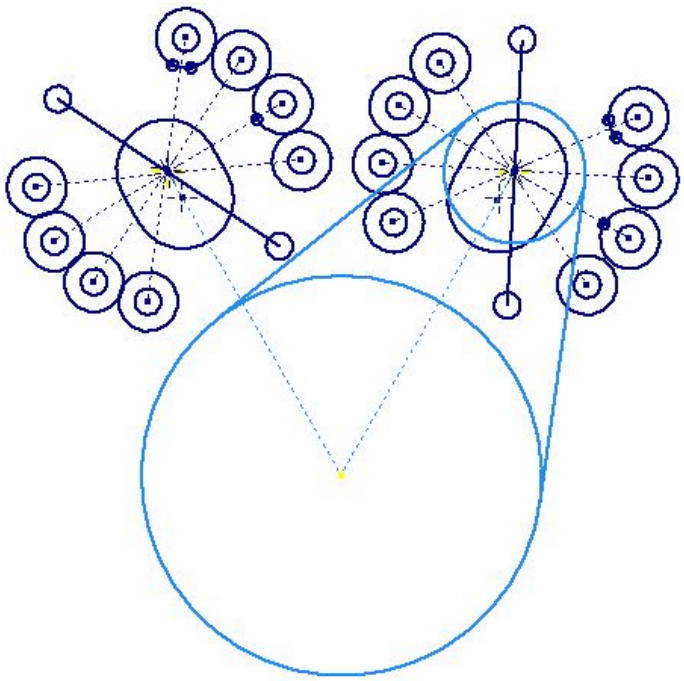
We first figure out our target total gear ratio of stage 1 + stage 2, then select teeth based on design and availability constraints, and get to our target using goal seek

Possible Free Speeds					
Selected	18	20	22	24	T
Larger Gear	76	74	72	70	T
2nd Ratio	4.222	3.7	3.27272727	2.917	ul
Free Speeds	**11.705**	13.3567807	15.101	**16.944**	**ft/sec**

Selections of second stage gear ratios and corresponding free speeds

On the first stage, a unique feature is the belt tensioning system, made through an array of motor mounting holes that move the motor closer and further away from the big output pulley. that allows for 2 settings tighter than nominal and 1 looser to account for variation in the belts.

Belt Var (Belt Length)	CC	Belt Length		Percent Change	Belt Var (CC)
0.00	2.623	305.00	mm	0.00%	
0.56	2.635	305.56	mm	0.18%	0.012
1.13	2.647	306.12	mm	0.37%	0.024
-0.53	2.612	304.47	mm	-0.17%	-0.011



Geometry of the motor mounting holes for belt tensioning

To figure out the center to center distance between timing pulley for the belts to fit we started by laying out all the relationship between center to center distance, the circumference of the

belt, and the amount of teeth. Then we used goal seek to an integer number of teeth to make sure a belt would fit mathematically perfectly.

$$(R_2 - R_1)^2 + L^2 = CC^2$$

$$L^2 = CC^2 - (R_2 - R_1)^2$$

we need the angle of tangency, α

$$\sin\alpha = \frac{R_2 - R_1}{CC}$$

now we need the arc lengths of the angles:

$$A_1 = \alpha\, \mathrm{R}_1$$
$$A_2 = \alpha\, \mathrm{R}_2$$

we need to account for the arcs made by alpha

$$\Sigma \mathrm{A} = 2\alpha_2 - 2\alpha_1$$
$$\Sigma A = 2\,\alpha(R_2 - R_1)$$

we need the half circles
full circle is 2π so half circle is just π!

$$\Sigma\, C = \pi R_1 + \pi R_2$$
$$\Sigma\, C = \pi(R_1 + R_2)$$

total length is the sum of the half circles + sum of the arcs + 2L

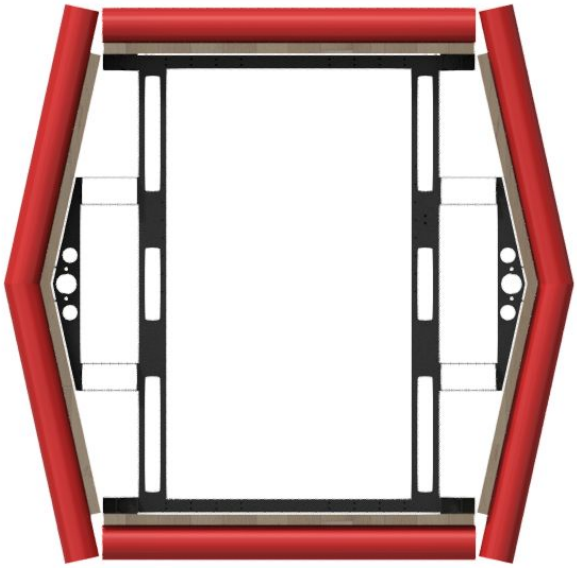
$$L_T = \pi(R_1 + R_2) + 2\,\alpha(R_2 - R_1) + 2\mathrm{L}$$

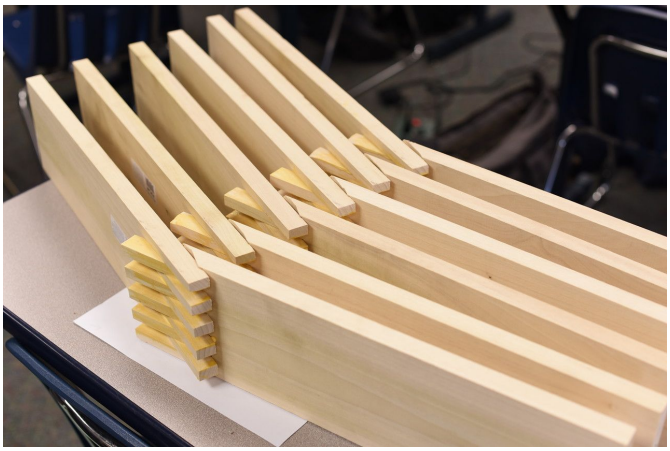
Formulas we applied to calculate the first stage belt size

Bumpers

Eesha, Catherine, Claire

This year, we are using hexagonal bumpers. The shape of the bumpers gives us a defensive advantage on the field. If a robot were to attempt to pin us from the sides, the angle of the bumpers will allow us to escape. Since only a small angle is created, the difference between the side of the bumpers and the frame only increased slightly, which makes it easy to accommodate.



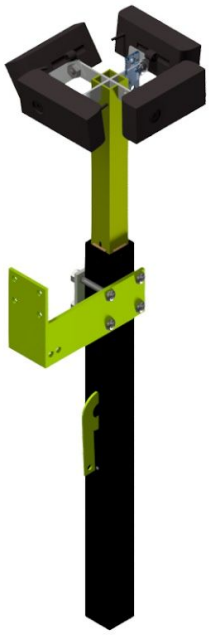


Finger joints form the angle of the side bumpers

Birds Eye Camera

Shaunak Bhandarkar (Senior)

For times during the match when the drivers' view is obstructed, we determined we wanted an aerial view. Using 4 cameras each facing a side of the robot, we get a view from all directions. We needed to mount the Birds Eye cameras as high on the robot as possible to avoid view obstruction from the top of the elevator.



Design Requirements:

1. Achieve an aerial view during the whole match
2. Stay within height limits during both beginning and end of match
3. Mount as high as possible on the robot

Our idea was to have the camera below limits at the beginning of the match, but for the initial movement of the elevator to trigger it to extend it to maximum height. Our current idea involves a telescoping arm composed of two tubes. When released by a rope pulled by the elevator, the inner tube holding the cameras shoots upward, powered by a constant force spring. The inner tube is expected to remain extended throughout matches.

Carriage Intake

Andrew Ng (Senior), Kunal Patel (Junior), Conrad Ho (Freshman)

The carriage intake picks up and shoots out cargo from the ground and uses an actuated hook to pick up and place the hatch panel. A crown, or a convex raised section along the roller, on the cargo rollers allow us to center the ball relative to the robot.

Design Requirements:

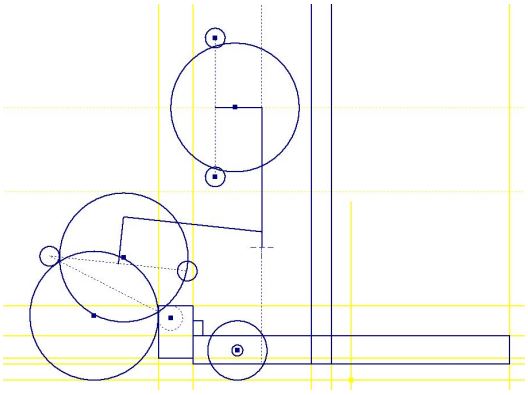
Hatch Panels

1. Center hatch panel to the robot
2. +/- 1.5” of tolerance
3. Securely hold hatch panels when traveling

Cargo

1. “You touch it you own it” (no bouncing!)
2. Center cargo to the robot
3. +/- 5” of tolerance

Cargo Collection

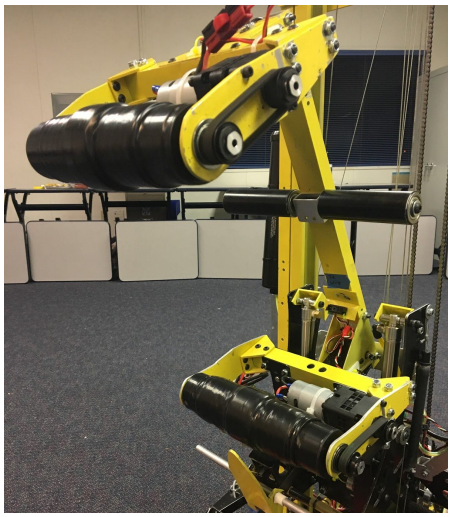


The cargo collection pulls the ball up and over the bumpers

To pick the cargo ball up off the ground, we push the arm down and spin the two rollers inward. The front roller pulls the ball up against the bumper and in between the two rollers.

Centering Cargo

A few years ago, we realized that a ball that is spinning on a crowned roller will tend to ride up to the high parts. To center the cargo relative to our robot, we spin the ball in place by spinning the bottom roller out and the top roller in. We crowned the top and bottom collector rollers using foam weather stripping and covered them with black tennis grip. We also added a pair of idle rollers at the back to lower the friction between the ball and the back of the collector.



The cargo tends to ride up the crowned rollers as we spin it in place, centering the cargo

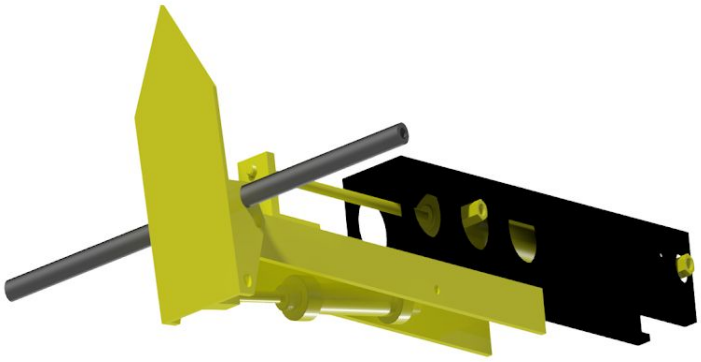
Mecanum wheel

		
Old FingerTech Mecanum Purchased Weight: 55 grams Problem: Rollers fall off during matches.	Rubber squeegee Mecanum Custom designed Weight: 27 grams Problem: Does not direct balls toward the center efficiently.	New bearing-based Mecanum. Custom Designed Weight: 52 grams Current Mecanum

Evolution of Mecanums (right to left)

To direct balls on the ends of the roller into the center, we designed and printed this custom Mecanum. These 3D printed mecanums have six bearings positioned on a 30° tilt. Pairs of mirrored Mecanums ensures collection anywhere near the rollers.

Hatch Panel Collection



The hatch panel slider and hook

To collect and place hatch panels, we used a linear slider and a hook. When collecting, the pointed shape of the beak also allows the driver to push the slider forward and move the slider back and forth until the beak slips into the hatch panel hole. The hook takes advantage of gravity to center the hatch panel relative to the robot. When deploying, the linear slider allows the driver to drive all the way against a scoring target whether or not the bumpers can get under the target and do the same actuation to push the plate against the wall.

Ground Collector

James Jiao (Senior), Kunal Patel (Junior), Jonah Bond (Freshman)

The Ground Collector collects and centers cargo over a wide range, picks up hatch panels from the ground, and hands off the game pieces to the Carriage Collector.

Design Requirements:

Hatch Panels

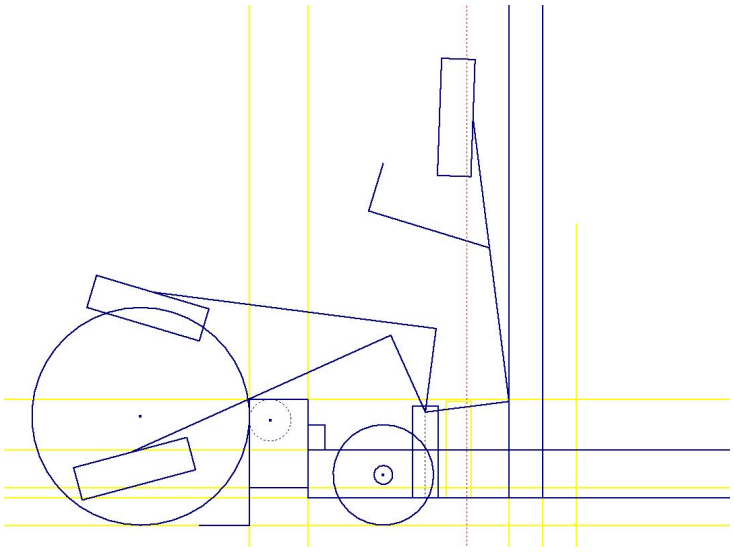
4. Ground intake, able to pick off from floor
5. Center hatch panel to the robot
6. +/- 1.5” of tolerance

Cargo

4. “You touch it you own it” (no bouncing!)
5. Center cargo to the robot
6. +/- 5” of tolerance

Cargo Collection & Centering

An important design requirement was centering the cargo after owning the game piece. To accomplish this task, we used four rollers wrapped in grippy tennis grip to increase the range of collection and effectively center the cargo.

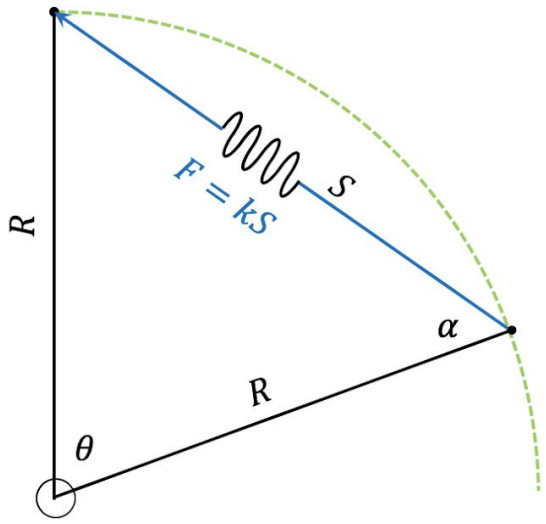


Sketch of centering rollers at different arm positions

Counterbalancing

To reduce strain on the pivot motor and speed up the movement of the arm, we counterbalanced the Ground Collector. This meant that using a surgical tubing spring that followed Hooke's Law, we were able to perfectly counterbalance the collector arms at all angles. This means the arm will always hold its position, even without the help of a motor.

The torque on the arm due to gravity is a sinusoidal curve. A torsion spring placed at the pivot would only balance the arm at two positions (because the torque it provides is linear). However, our team discovered that a simple linear spring was able to perfectly counterbalance the arm because the torque it provided fit the sinusoidal curve exactly.



Using Trigonometry:

$$\vec{T} = \vec{R}\times\vec{F} = \vec{R}\times k\vec{S} = k(\vec{R}\times\vec{S}) \quad (1)$$

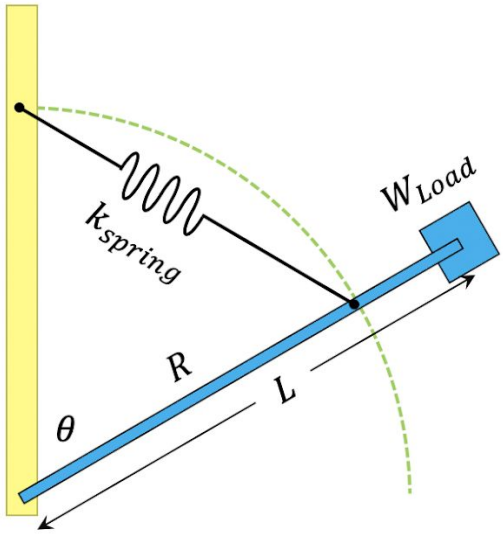
$$\mathrm{T} = \|\vec{T}\| = kRS\sin\alpha \quad (2)$$

From law of sines:

$$\frac{S}{\sin\theta} = \frac{R}{\sin\alpha} \quad \rightarrow \quad S\sin\alpha = R\sin\theta$$

$$\mathrm{T} = kR(R\sin\theta) \quad (3)$$

$$T = kR^2 sin\theta \quad (4)$$



For the Spring,

$$T_{spring} = kR^2\sin\theta$$

And for the Arm:

$$T_{Load} = L\,W_{Load}\sin\theta$$

Or more generally,

$$T_{Load} = T_{90°}\sin\theta$$

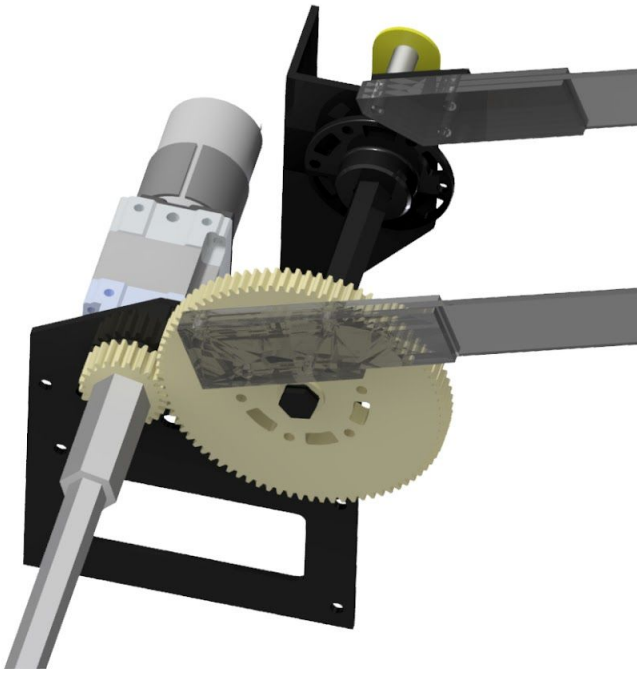
Where, in this case for simplicity, $T_{90°} = L\,W_{load}$ but $T_{90°}$ should also include the arm's mass and distribution.

The spring balances the arm at all angular positions, θ, when:

$$T_{90°} = kR^2 \quad !!!$$

Pivot & Arm

To protect the Versaplanetary gearbox on the pivot from high forces that may be transferred from impacts on the arm, we added a 3.5:1 reduction with gears after the gearbox. This means the pivot of the arm will be able to take 3.5 times the torque of the gearbox by itself.

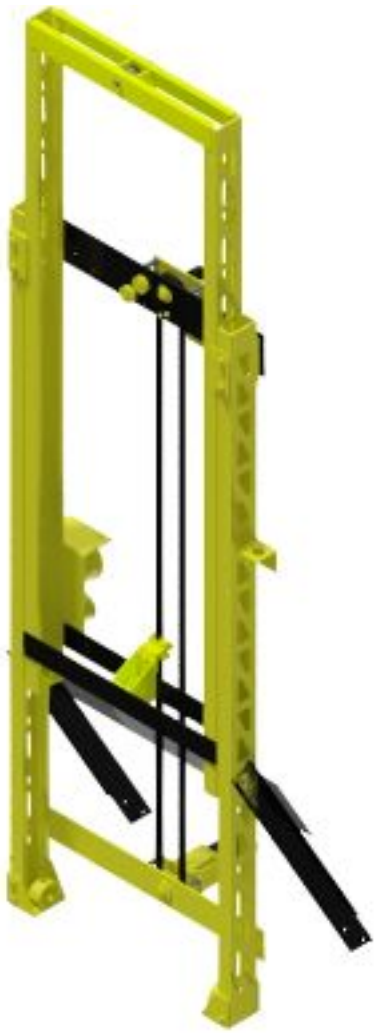


Final gear reduction + polycarbonate fingers

In addition to the gear reduction, we built the middle section of the arms out of polycarbonate. Combined with polycarbonate fingers that spread out the bending force over a longer distance, the polycarbonate arms provided flexure that would dampen forces on the arm. The polycarbonate pieces could also be easily manufactured and swapped out.

Elevator

Jonah Soong (Junior), Sam Pickholtz (Sophomore)



Design Requirements:

1. Reach up to the top port on rocket and down to lowest port
2. Lift to max height in a second

Cascading Lift Design

This years' lift is a cascading design, where the second and third stages move up relative to each other, and the third stage traverses twice the distance of the second in the same amount of time.

Neg'ator Spring Counterbalancing

A Neg'ator spring motor uses constant torque springs wound around a spool to wind up a rope. As the rope is pulled out of the Neg'ator spring motor, the springs unwind and provide a constant amount of force that counterbalances the lift. In other words, it is counteracting the force of gravity, allowing us to use only a single motor to power the lift.

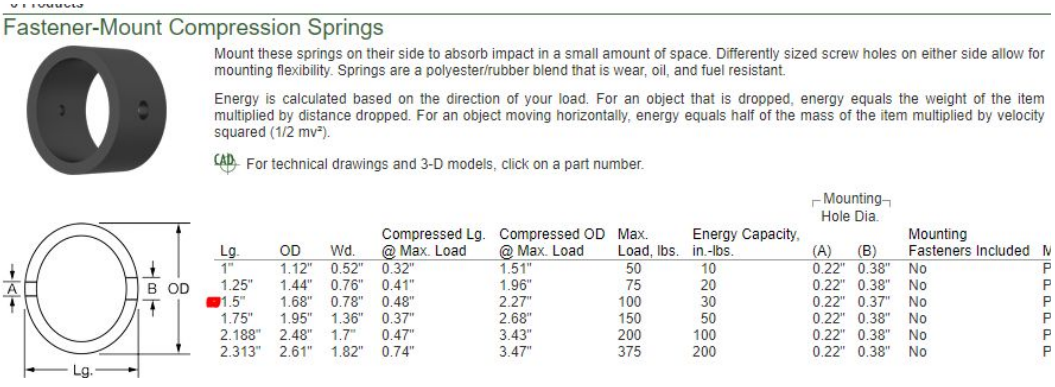
Spring Hardstop

To prevent the lift from damaging any components when traveling down. We did calculations for a rubber spring to absorb any potential impact when the lift is moving to its lowest position.

Second Stage Deceleration			
	Weight	**6.2**	lb
	Weight	2.8	kg
	Avg Velocity	0.9	m/s
Third Stage Deceleration			
	Weight	**13.3**	lb
	Weight	6.0	kg
	Avg Velocity	1.7	m/sec
KE Second Stage		1.0	J
KE Third Stage		8.7	J
KE Lift(Joules)		9.8	J
KE Lift(lb ft)		7.2	lb ft
KE Lift(lb in)		86.4	lb in

Calculations for the amount of energy the lift stages will slam down with if powered into the bottom.

From these calculations we chose a spring that would absorb the energy of a falling lift.



Fastener-Mount Compression Springs

Mount these springs on their side to absorb impact in a small amount of space. Differently sized screw holes on either side allow for mounting flexibility. Springs are a polyester/rubber blend that is wear, oil, and fuel resistant.

Energy is calculated based on the direction of your load. For an object that is dropped, energy equals the weight of the item multiplied by distance dropped. For an object moving horizontally, energy equals half of the mass of the item multiplied by velocity squared (1/2 mv²).

CAD For technical drawings and 3-D models, click on a part number.

Lg.	OD	Wd.	Compressed Lg. @ Max. Load	Compressed OD @ Max. Load	Max. Load, lbs.	Energy Capacity, in.-lbs.	Mounting Hole Dia. (A)	Mounting Hole Dia. (B)	Mounting Fasteners Included	M
1"	1.12"	0.52"	0.32"	1.51"	50	10	0.22"	0.38"	No	P
1.25"	1.44"	0.76"	0.41"	1.96"	75	20	0.22"	0.38"	No	P
1.5"	1.68"	0.78"	0.48"	2.27"	100	30	0.22"	0.37"	No	P
1.75"	1.95"	1.36"	0.37"	2.68"	150	50	0.22"	0.38"	No	P
2.188"	2.48"	1.7"	0.47"	3.43"	200	100	0.22"	0.38"	No	P
2.313"	2.61"	1.82"	0.74"	3.47"	375	200	0.22"	0.38"	No	P

With two spring the one marked in red is pretty close to what we need.

The use of these springs protects the lift from destroying itself even in the worst of situations.

Horizontal Slider and Pivot

Sam Pickholtz (Sophomore)

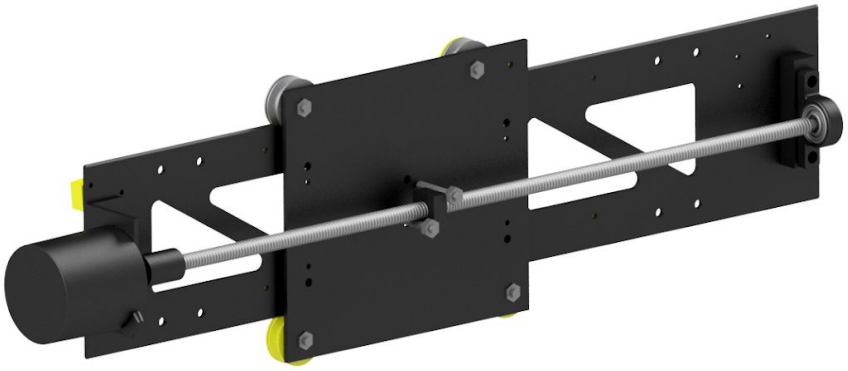
Our team predicted that precise positioning of game pieces is necessary for this year's game. The driver is required to position hatch panels and cargo within 1.5 inches in either direction while standing 20 feet away and additionally has to align the robot at oblique angles to place game pieces. To add to the challenge, defending robots can push our robot during the alignment process. To combat both these problems we implemented a horizontal slider. The slider moves our collector from side to side, left to right, 6 inches in each direction.

Design Requirements:

1. Adjust Horizontally by at least 6 inches in either direction in less than 1.5 seconds
2. Have horizontal positioning accuracy of at least ¼ inch
3. Pivot Carriage Intake in less than 0.5 seconds
4. Stow Carriage Intake within the frame of the robot

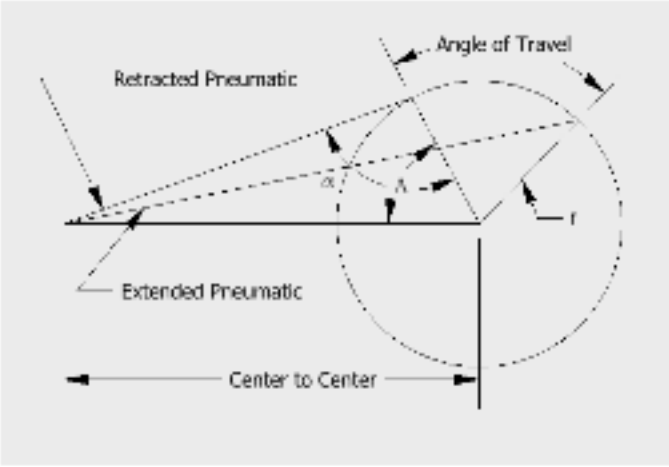
Lead Screw Powered Slider

Lead screws (similar to threaded rods) push a lead screw nut back and forth along the length of the screw. This design allows for a large reduction between the input rotational speed to the output linear motion. The 12 turn per inch lead screw used on this design translates the 4000 rpm of the REV Neo brushless motor directly to 6 in/sec of horizontal travel without gearboxes. As a bonus, the Rev Neo provides an integrated encoder to measure the position of the slider.



Pneumatic Cylinder Pivot

The pivot is powered by pneumatic cylinders which, when attached to an arm, transfer their linear force into a torque that moves the arm up and down. The advantages of pneumatics over a gearbox are twofold. The first advantage is in simplicity. With pneumatics, no sensors are required, and accurate positioning of the arm is simple. The second advantage is that when a large force is applied to the arm in the direction of travel, the pneumatics will compress. If a gearbox was used in this case, the gearbox would break under such load.



Equation For Pneumatic Torque Relative to:

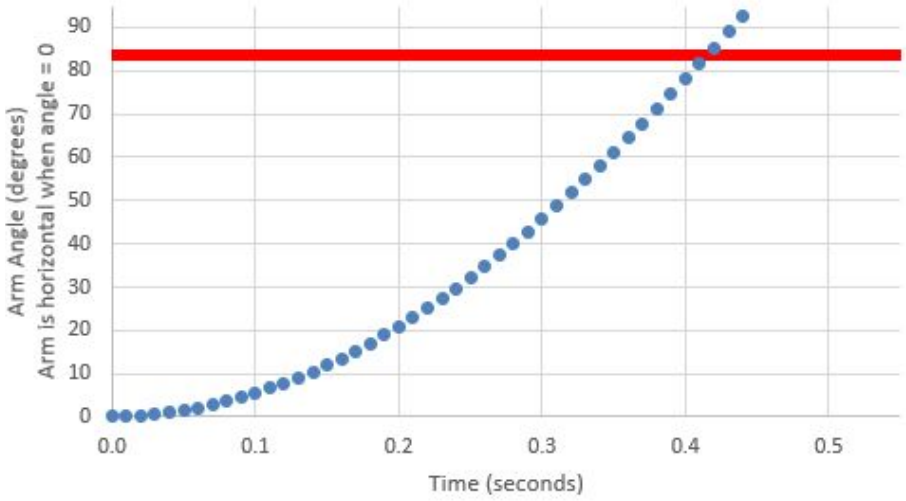
C_C-Center to Center Distance
F_P-Force From the Pneumatic
r-The Radius of Pneumatic attachment
A-The angle from x-axis to pneumatic
α-Angle, pneumatic end to circle center

This is derived by first doing law of cosines to find L, then using the law of sines to find α, and then τ is found with a cross product.

$$\tau = F_P * r * \frac{C_c * \sin(A)}{\sqrt{r^2 + {C_c}^2 - 2rC_C * \cos(A)}}$$

This formula provides the torque of the pneumatic over its range of motion.

When counterbalancing is added (counterbalancing explained in next section) even quite small pneumatics can rotate the entire collector assembly in less than half a second.



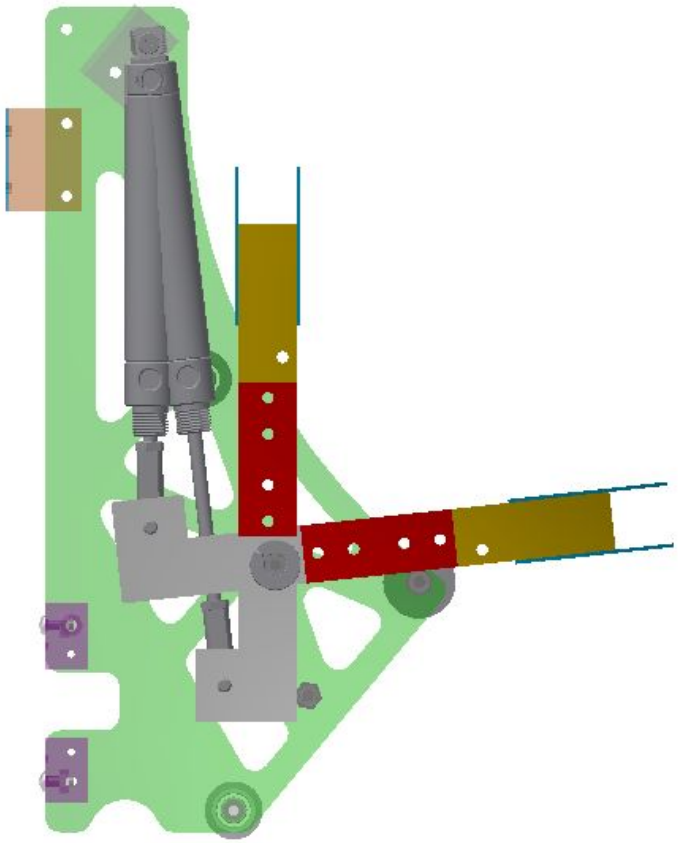
Arm angle (blue) over time. Red line represents arm vertical position

This graph was calculated through numerical analysis of the torque provided by the pneumatic. The pneumatic torque changes based on the angle of the arm (follow the function above) rather than time. In order to relate the arm position and time I imputed time and the starting arm angle, and then solved for the angular acceleration. From the angular acceleration I calculated the angular velocity at the end of a given time interval, then integrated final angular velocity to calculate radians traveled over that amount of time. I then added this output distance traveled to the previous angle to calculate the next values.

$\tau = I\alpha$ $\omega_F = t * \alpha$ $\theta = \frac{\omega_1 - \omega_0}{2} * t$

		lbf-in					
Time	Pneumatic Angl	Torque	Angular Acc.	Fin. Ang. Vel.	Rad. Traveled	Arm Angle (rad)	Arm Arm (deg)
0	1.23	131.68	34.73	0.35	0.00	0.00	0.00
0.01	1.24	131.69	34.73	0.69	0.00	0.00	0.10
0.02	1.24	131.72	34.74	1.04	0.01	0.01	0.30
0.03	1.24	131.76	34.75	1.39	0.01	0.01	0.60
0.04	1.25	131.81	34.76	1.74	0.01	0.02	1.00
0.05	1.26	131.86	34.78	2.08	0.01	0.03	1.49
0.06	1.27	131.90	34.79	2.43	0.01	0.04	2.09
0.07	1.28	131.94	34.80	2.78	0.01	0.05	2.79
0.08	1.30	131.95	34.80	3.13	0.02	0.06	3.58
0.09	1.31	131.94	34.80	3.48	0.02	0.08	4.48
0.1	1.33	131.89	34.78	3.82	0.02	0.10	5.48
0.11	1.35	131.79	34.76	4.17	0.02	0.11	6.57
0.12	1.37	131.62	34.71	4.52	0.02	0.14	7.77
0.13	1.39	131.38	34.65	4.87	0.02	0.16	9.06
0.14	1.42	131.06	34.57	5.21	0.03	0.18	10.46
0.15	1.44	130.63	34.45	5.56	0.03	0.21	11.95
0.16	1.47	130.09	34.31	5.90	0.03	0.24	13.54
0.17	1.50	129.42	34.13	6.24	0.03	0.27	15.23
0.18	1.53	128.60	33.92	6.58	0.03	0.30	17.02
0.19	1.56	127.63	33.66	6.92	0.03	0.33	18.90
0.2	1.60	126.49	33.36	7.25	0.04	0.36	20.88
0.21	1.63	125.16	33.01	7.58	0.04	0.40	22.96
0.22	1.67	123.63	32.61	7.91	0.04	0.44	25.13
0.23	1.71	121.90	32.15	8.23	0.04	0.48	27.40
0.24	1.75	119.95	31.64	8.54	0.04	0.52	29.76
0.25	1.80	117.76	31.06	8.85	0.04	0.56	32.20
0.26	1.84	115.34	30.42	9.16	0.05	0.61	34.74
0.27	1.89	112.67	29.72	9.46	0.05	0.65	37.36

A portion of my simulation to calculate collector pivot speed. A time-step simulation of arm motion.



Pneumatics extended and retracted, showing arm range of movement.

Pivot Counterbalancing

The Pivot employs a method of perfect counterbalancing developed by our team a number of years ago. This year we expanded on the previously known calculations to fully analytically calculate the required spring lengths to counterbalance the arm. The counterbalancing uses latex tubing (surgical tubing) as the spring, allowing any spring constant to be created simply by cutting the tubing to different lengths.

Generalized Proof of Perfect Counterbalancing

Givens: W-Weight of Arm; L-Radius to CG of Arm; $\vec{r}$-Vector of Arm Position; $\vec{A}$-Spring Attachment Point on Solid Component; $\vec{B}$-Spring Attachment point in arm

Torque of Arm

$$T_A = L\hat{r}\ X\ W\hat{y}$$

Force from spring

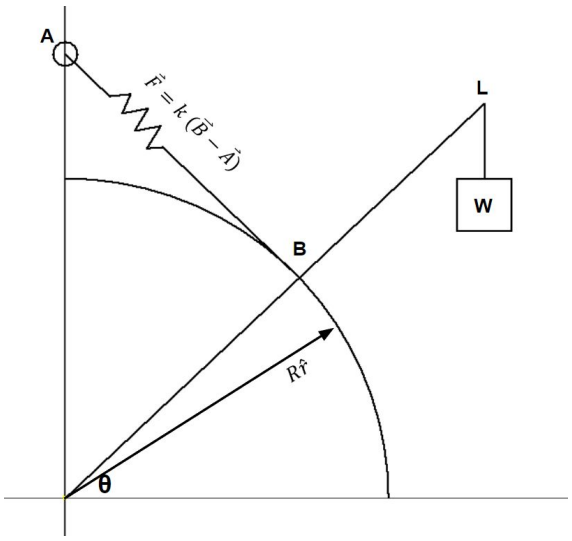
$$F_S = k * (\vec{A} - \vec{B})$$

Torque from Spring

$$T_S = R\hat{r}\ X\ F_S$$
$$T_S = R\hat{r}\ X\ k(\vec{A} - \vec{B})$$
$$T_S = k * (R\hat{r}\ X\ R\hat{y} - R\hat{r}\ X\ R\hat{r})$$
$$T_S = k * (R\hat{r}\ X\ R\hat{y}\ - 0)$$
$$T_S = k * R^2 * \sin(\theta)$$

Set Torque from Spring equal to Torque of Arm

$$\boldsymbol{k * R * H * \sin(\theta) = L * W * \sin(\theta)}$$
$$k * R^2 = L * W$$



The proof and attached drawing show the equations for perfect spring counterbalancing.

The formula above is the proof of concept but implementing the perfect counterbalancing requires more formulas. Starting with one known value, in this case desired spring stretch percent, using the below formula will solve for the spring attachment radius, and by extension the spring resting length.

Calculate Spring Starting Length

The spring has to stretch by $r\sqrt{2}$ distance (because spring goes along hypotenuse of 45-45-90 triangle of side lengths r), and with a stretch percent of S_P so starting length is stretch distance / stretch percent is:

$$L_I = \frac{r\sqrt{2}}{S_P}$$

Formula to Solve for r w/o knowing k

$$k * r^2 = L * W$$

Hook's Law

$$F_S = kx$$

With latex tubing x=stretch percent(1-stretch factor)

$$F_S = k * S_P$$

At 100 percent stretch

$$F_S = k * L_I$$

Plug into formula along with

$$L_I = \frac{r\sqrt{2}}{S_P}$$

To Get:

$$r = \frac{L * W * \sqrt{2}}{F_S * S_S}$$

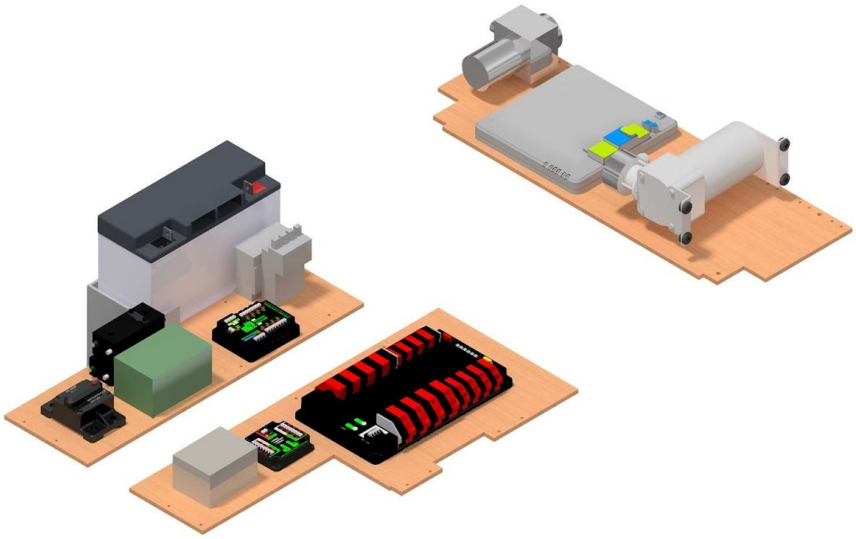
A formla that solves for the spring attatchment radius based on all the other values

These formulas solve for the spring lengths based on imputed values.

This method of perfect counterbalancing is especially elegant because it not only works in theory, but also works to near perfection in practice.

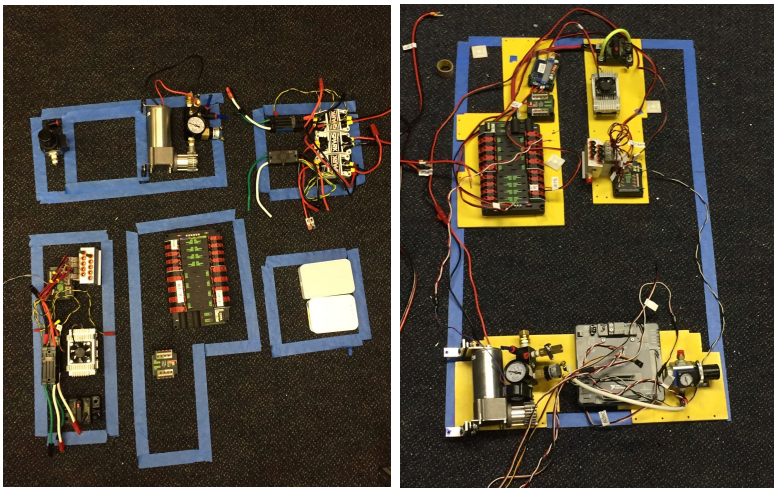
Electrical and Pneumatics

Isha Venkatesh (Sophomore) and Zachary Wu (Junior)
With Kyle He (Freshman), Anitez Gautam (Freshman), Andrew Caldwell (Freshman), Thomas Li (Freshman), Benjamin Gao (Sophomore), Claire Chen (Sophomore)



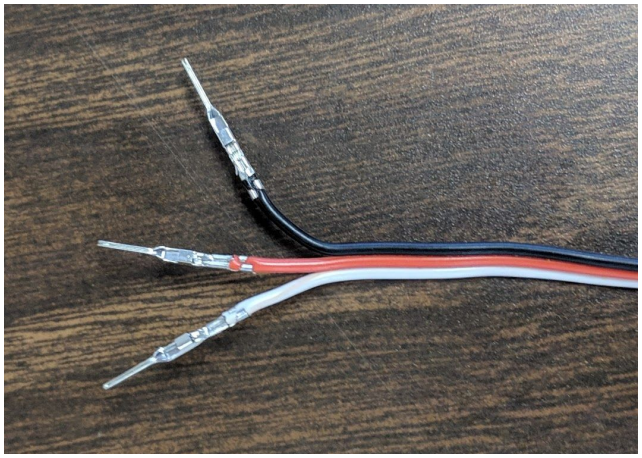
Planning

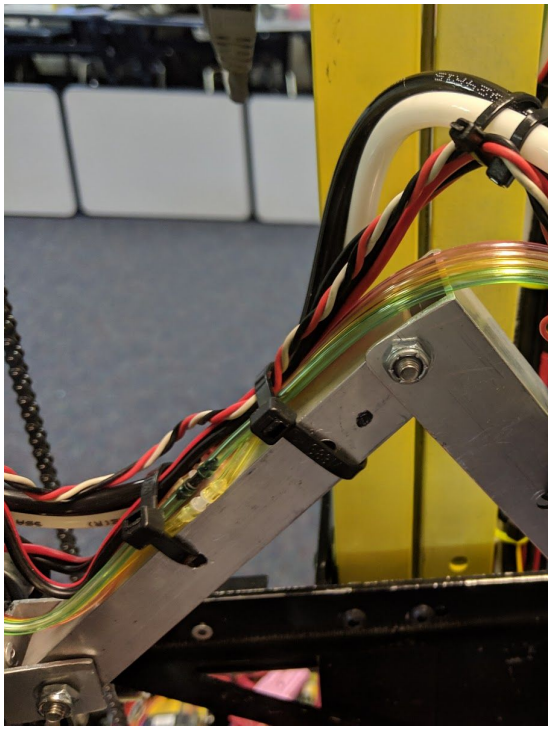
Planning ahead allowed us to cut the wires for the robot to length before it was even machined. We first used CAD to model the electrical components and then made a mock baseboard with tape to ensure that the components would fit. Before the drivetrain frame was machined, we made a mock drivetrain frame and routed wires around the robot. By the time the drivetrain frame was machined, we were ready with all the needed wires.



Wiring the Robot

To have a reliable robot, the electrical and pneumatic connections must be crimped and secured properly.





Throughout fall workshops and build season, we taught proper crimping techniques to new members so the crimps would be durable. Before putting the wires on the robot, we did a pull-test to test the strength of the crimp.

To route wires and tubing around a moving joint, we positioned the connectors on a stationary surface and gave enough slack around the joint. Securing the connectors before and after the connection provides strain relief.

All of these techniques, together with meticulous workmanship, contribute to a dependable robot.

Software

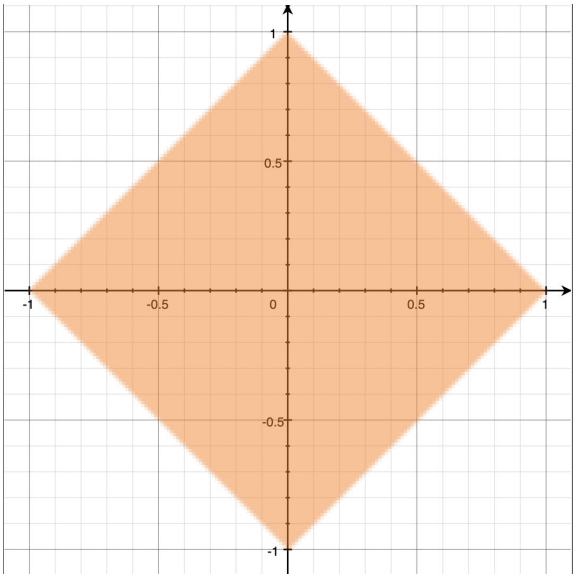
Kunal Sheth (Junior), Alvyn Wang (Junior), Andy Min (Freshman),
Nikash Walia (Senior), Sidhant Rajadnya (Sophomore), Sidharth Kannan (Freshman)

Path to Trajectory Conversion

Kunal Sheth (Junior)

When programming our robot to operate autonomously, we want to be able to have it follow a path defined by a set of waypoints. When executing a given path, we want the robot to travel as fast as possible given the physical constraints of its drivetrain.

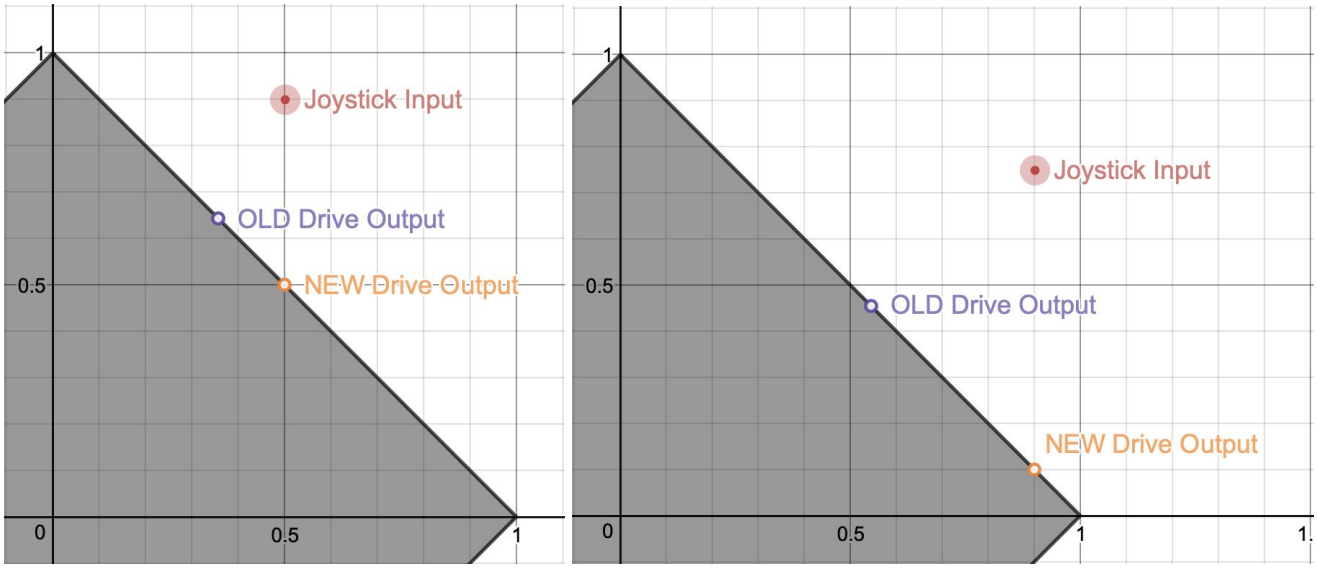
$$\begin{cases} \dfrac{\Delta\theta}{t * \omega_{max}} = \omega \\ \omega \in [-1, 1] \\ \dfrac{\sqrt{\Delta x^2 + \Delta y^2}}{t * v_{max}} = v \\ v \in [-1, 1] \end{cases}$$



We started this project by modeling all the possible states of a differential drivetrain. Our software makes the robot operate on the very edge of this region when following a path.

My favorite part of this project is that it allows us to program our robot by simply pushing it along the path we want to follow. When we want to execute a recorded path, the robot will automatically calculate how fast it can travel as it hits each waypoint.

We also apply this concept to make our driver controls more ergonomic. With more practice, I started pushing the robot to its max speed during each cycle. With our old drivetrain calculations, I noticed the robot got extremely difficult to control when flooring the joystick. Instead of spending many hours learning how to control the robot at high speed, I figured out how to fix it in software.

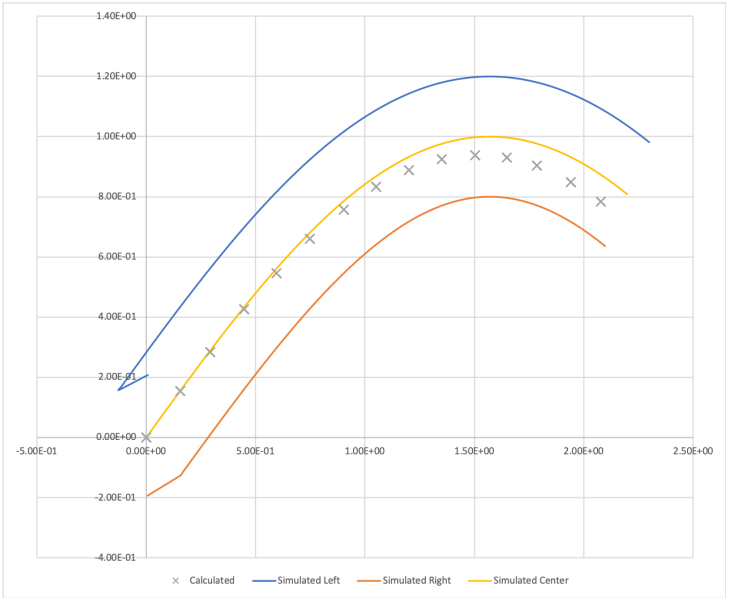


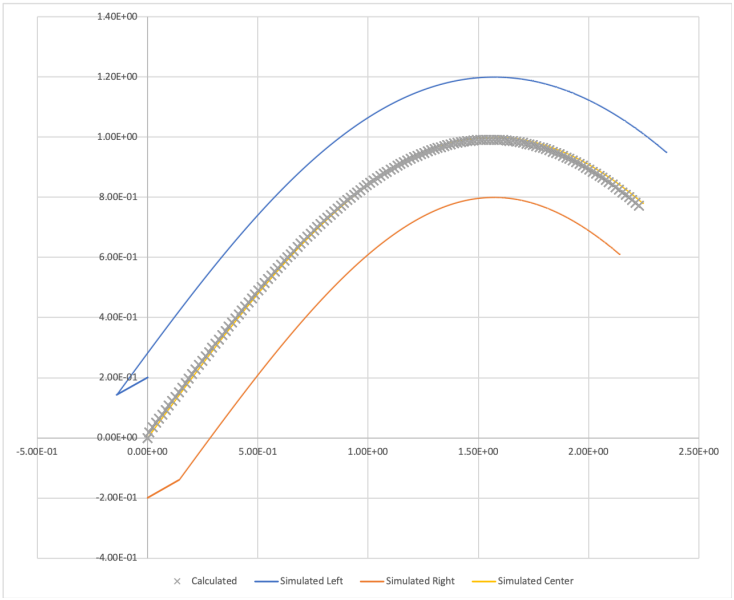
When the driver joystick inputs imply a state outside the region of feasibility, our software prioritizes the driver's steering wheel. We automatically scale back the driver's forward velocity input so steering feels consistent even at high speeds.

Offloaded Cartesian Position Tracking

Kunal Sheth (Junior), Alvyn Wang (Junior), Andy Min (Freshman)

Before it can start following trajectories, the robot needs to know its cartesian position. In past years, our cartesian tracking was consistent, but not accurate. This made trajectory following possible but time-consuming as we kept having to “fudge” our waypoints. This year, I wanted to fix this strange behavior.





I started by modeling our robot’s drivetrain and cartesian tracking algorithms in Excel. The simulation shows that a higher update frequency (say, every encoder tick) is more accurate.

Because WPILibJ does not allow users to easily write high performance interrupt service routines, I decided to wire up our encoders to a separate microcontroller dedicated to cartesian tracking.

With encoder ticks streaming in at up to 40kHz, running the trigonometric functions necessary for a naive vector addition algorithm is not possible. We tried all kinds of optimizations: using fixed-point math, storing a SIN lookup table in ROM, and even sending all the data back to the RoboRIO's more powerful processor.

To reach a frequencies we wanted, we had to take a different approach altogether — rotation matrix tracking.

$$\overrightarrow{P'_L} = \overrightarrow{P_R} + \begin{bmatrix} \cos(\Delta\theta) & sin(\Delta\theta) & 0 \\ -sin(\Delta\theta) & cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\overrightarrow{P_L} - \overrightarrow{P_R}\right)$$

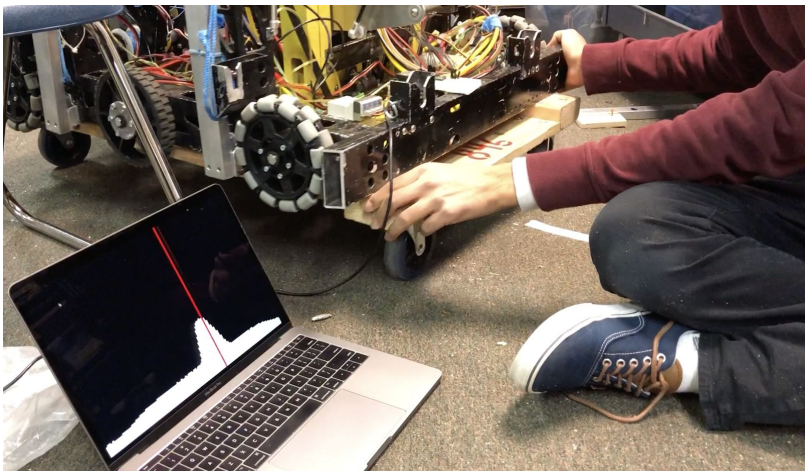
$$\overrightarrow{P'_R} = \overrightarrow{P_L} + \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) & 0 \\ -\sin(\Delta\theta) & \cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} (\overrightarrow{P_R} - \overrightarrow{P_L})$$

Rotation matrix tracking is able to keep up because we can create a small lookup table for the 4 possible values of Δθ. We use only multiplication and addition to find our position.

Line Tracking

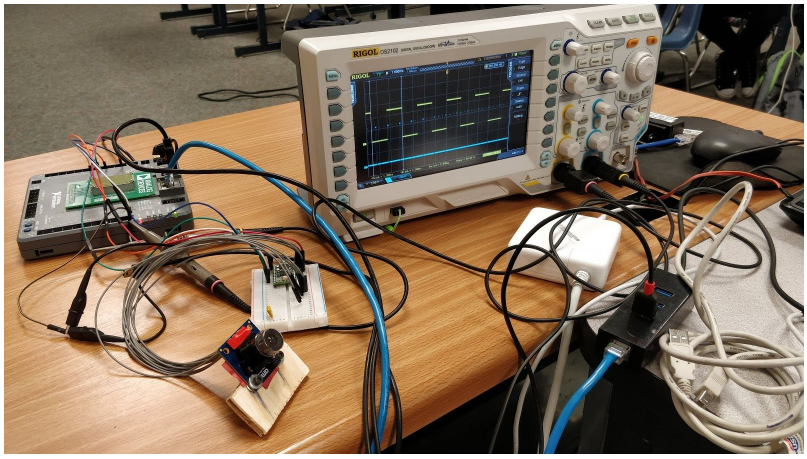
Kunal Sheth (Junior)

Side-to-side alignment is critical this year. While we could use a conventional 2D camera and some OpenCV code to follow lines on the field, there would be a lot of added complexity (coprocessor and networking). Instead, I took a far simpler approach, using a 1-Dimensional, grayscale camera.



GUI Demo of a noise-resistant line detection algorithm.

We use a microcontroller to interface with our 128-pixel photo sensor. The microcontroller detects the line and relays the information back to the RoboRIO. The simplicity of this approach makes it reliable and allows us to run real-time control loops based on the line's position.



To make exposure and detection thresholds dynamically configurable, I also developed a 2-way communication protocol between the RoboRIO and microcontroller using frequency modulation.

Absolute Steering Wheel

Kunal Sheth (Junior), Alvyn Wang (Junior)

In past years, we've used a steering wheel and joystick to control our robot on the field. The steering wheel controls our robot's angular velocity. While it is great for making broad, fast maneuvers when zipping around the field, it is not as great for smaller, more precise adjustments. This is why we've developed an absolute steering wheel which controls our robots angle as opposed to its angular velocity.

An absolute steering wheel prototype controlling the robot's angle.

Birds-Eye Transform

Alvyn Wang (Junior)

While a conventional camera system, a system in which no transformations are applied to the camera stream, is adequate for most purposes, it has weaknesses that are exacerbated by this years game. Namely, it is difficult to gauge distance through an untransformed camera stream and is not the most intuitive viewpoint to look from. I, therefore, decided to apply a bird's-eye transformation on our vision system to give our driveteam a better operating experience.

On the left is the image before the transformation and on the right is the image after the transformation.

JVM Tuning

Kunal Sheth (Junior)

When programming real-time, embedded systems, the JVM is not your friend. While the GC provides us with more safety and lowers the barrier to entry for newer members, several-hundred-millisecond stop-the-world garbage collections are detrimental to robot control. To improve the real-time performance of our code, I've been working on optimizations such as moving units.kunalsheth.info to inline classes (like Scala's AnyVal or Java's Valhalla), switching to G1GC, and inlining/specializing generic functions to prevent boxing.

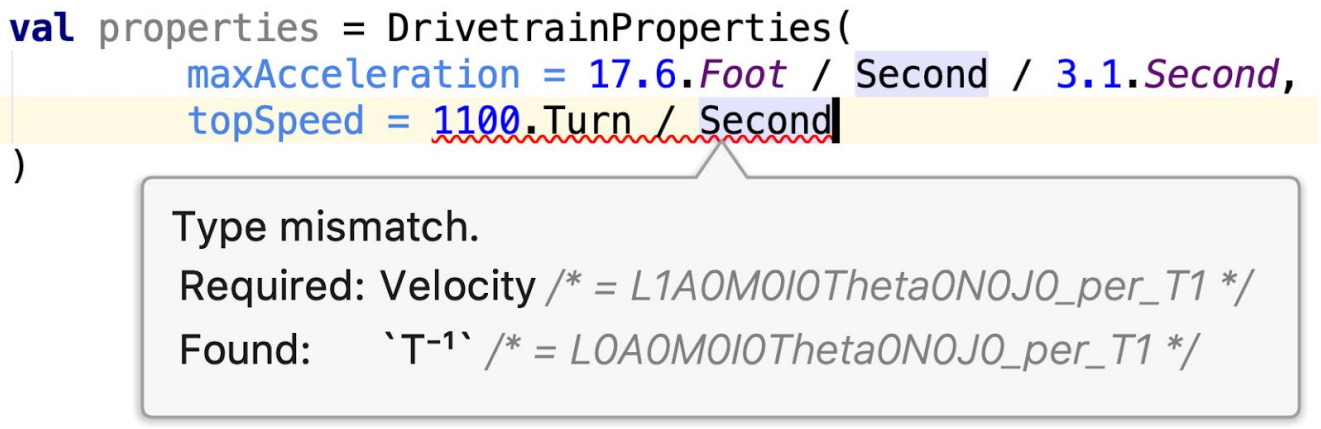
We generate logs to monitor loop timing and memory usage on the JVM. `units-of-measure` library without (top) and with (bottom) value types and inlined generic functions.

Units-of-Measure

Kunal Sheth (Junior)

With many programmers and a 15 thousand line, student-managed code base, bugs are inevitable. One category of bug is especially nasty: using incorrect units. One option may be to document the units of every single variable while enforcing even stricter code-review, but this is not always possible, especially in the heat of competition.

This is why I've developed units-of-measure, a compiler plugin for type-safe dimensional analysis and unit conversion in Kotlin. Now, instead of wasting several hours debugging a unit-related mistake, incorrect code will not even deploy in the first place.



Units-of-measure catches bugs before they even happen.