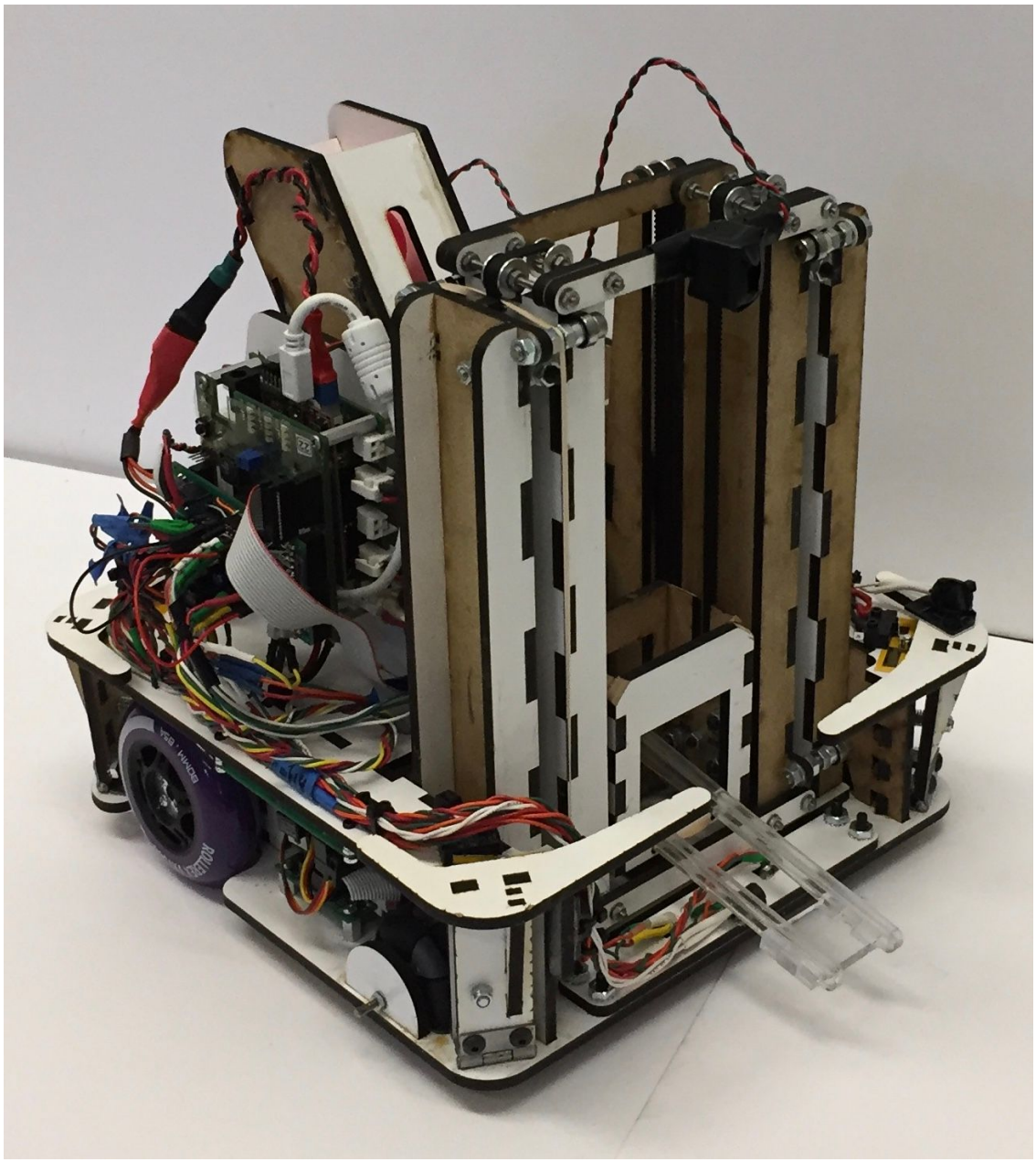
CMPE118
Final Project Lab Report 12/14/17



Team 13: **THE RESISTANCE**
Talmor Kliman
Joe Loughboro
David Kooi

Table Of Contents

Pg 3 Objectives and Constraints

Pg 4 Robot Strategy

Pg 6 Mechanical Design

Pg 11 Electrical Design

Pg 23 Bill Of Materials

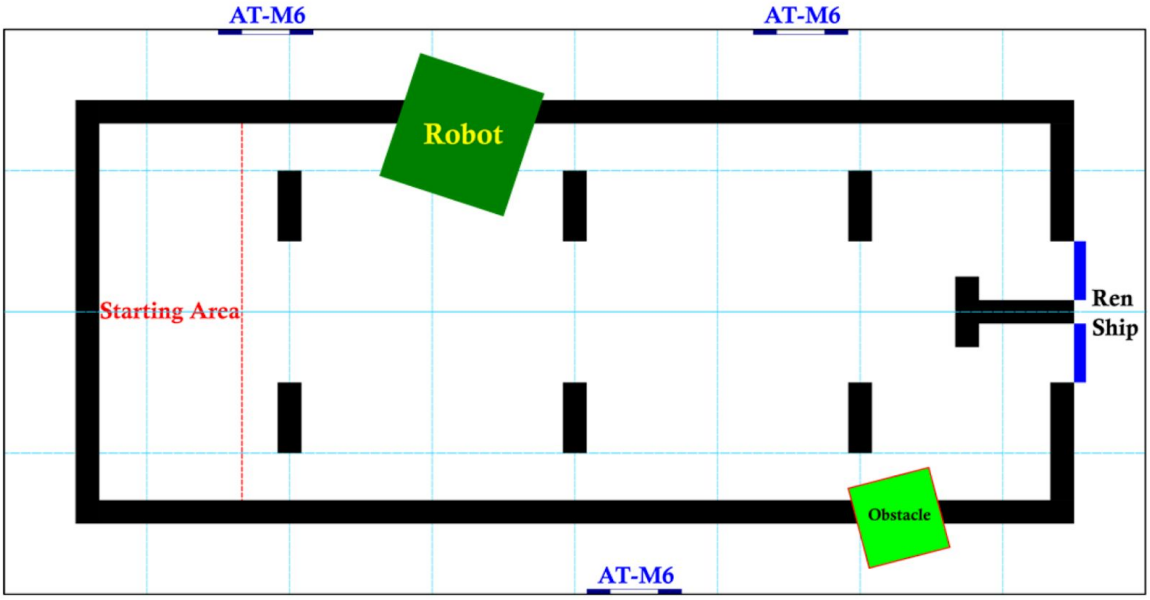
Pg 25 Software Design

Pg 29 Integration and Test

Pg 30 Conclusion

Objectives and Constraints

The objective of the project was to locate and ‘destroy’ three AT-M6 targets and one Ren Ship target. The three AT-M6 targets are randomly placed along the perimeter of the field. The ren ship is located opposite of the starting area. There are different ways to two ways to identify the targets. The AT-M6 targets contain an electrically oscillating wire. The ren ship hosts a infrared beacon. We used an inductive sensor and a phototransistor to detect these two targets.



Each robot needed to operate within the bounds of the black tape. Our team used short range photo detectors to identify the areas of black tape and of white open board. The field also contained a randomly placed obstacle. We detect the obstacle through actuating four switches. One switch sits on each corner of the bot.

The robot’s objective was to kill three AT-M6 targets and finally the ren ship target. The robot is constrained to finish within two minutes. At the start of the game the ren ship’s infrared beacon is turned on for fifteen seconds. After this the beacon turns off. The beacon turns back on when all three AT-M6 targets are destroyed. Our team has time constraints to complete the robot within 5 weeks.

Robot Strategy

There are many ways to complete this task with the given constraints, but as a team we decided on the most practical and optimal approach. The team collaborated and brainstormed solutions to completing the course in the allotted 2 minutes. How are we going to complete the objective in a robust and efficient manner?
The following is our time frame breakdown of how we wanted to execute the mission:

Time Frame	Time Amount	Task
0 - 15 sec	15 sec	Find the Beacon by spinning
15 - 25 sec	10 sec	Reverse directly onto the the tape and align to it
25 - 80 sec	55 sec	Find and insert balls into first 3 targets
80 - 90 sec	10 sec	Spin to find the beacon
90 - 105 sec	15 sec	Drive directly to the final target
105 - 120 sec	15 sec	Align with tape and insert ball to the final target

Elevator and Geneva Wheel

Now that we had an articulated mission to reference, we could systematically design the features on the robot so that it conforms to our intended performance. There were many considered designs, but the first decision we had to make was if we wanted to have two seperate ball-deployment mechanisms for the two different targets. The team decision was to use one mechanism for both heights. We decided on using an elevator mechanism powered by a stepper motor. This turned out to be our secret weapon for robustness and as well as giving us a unique design feature. One of our team member, Joe, has knowledge of building an elevator from his past Vex competition experience. We considered other options such as: a scissor lift, blank, blank, etc. but the elevator had the most votes. it was a robust solution for reaching both heights accurately, but it was a risk because only one teammate had experience with designing elevators.

To get the ball onto the elevator we decided a geneva wheel powered by a servo motor was the most simple and reliable solution for deploying the ball. To assist the ball out of the Geneva wheel, a fixed wedging piece was inserted into the design. This would allow for the ball

to propel forward and ensure it gets onto the ramp without uncertainty. Naturally, a ball hopper directly above the wheel would be needed to reload the ramp with a ball for all the targets.

Driving Mechanism and Bumpers

Rear wheel drive was chosen for a few reasons: to increase the distance between the motors and the track wire circuit to avoid tripping the sensor, to accommodate the strategy of reversing onto the tape, and for accurate tape following because the tape sensors are in front.

The bumpers would need to be robust to handle repeated testing and strain so we chose a design that bottoms out on a strong component to avoid stressing the switch. A micro limit switch would suffice for this application and is a low cost part to entertain the $150 budget. Each of the four corners of the robot needed a switch to avoid barriers. The rear bumpers seem unnecessary because there aren't any moving barriers, but since our strategy involves reversing onto the tape at the start, we needed them. Later it was realized that the barrier could not be placed in the starting position so the rear bumpers did not have a purpose.

Reflective Optical Sensors, Track-Wire Sensors, and Beacon Sensor

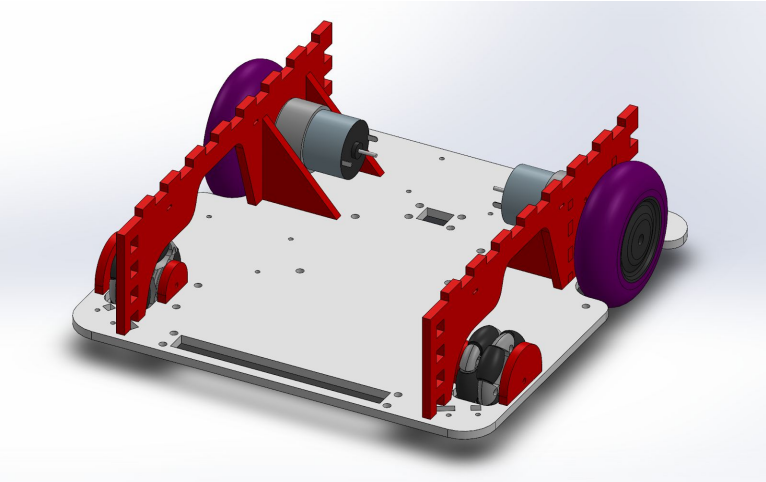
For an optimal, linear, and accurate way to follow the tape, we placed 3 reflective optical sensors in a triangle formation with one point in facing the forward direction directly in the center of the robot. The other two "tape sensors" were placed at a distance to be directly off of the tape on both sides. By having this placement of sensors we can program the robot to immediately detect when it needs to swerve left/right based on which two sensors are on the tape. The fourth "tape sensor" is placed in the rear-center of the robot directly in between the wheels for an accurate and simple tape alignment maneuver at the start of a round.

The beacon detecting sensor placement was a simple decision because it was used for one simple task; finding the direction of the beacon. Therefore, anywhere on the front of the robot was customary, but to add to the robustness of this part we placed it on the elevator to account for an edge case; the barrier blocking the beacon detector.

The track wire sensor placements were tricky because the orientation of the magnetic field lines produced by the track-wire must be parallel to the center of the circle inductor coil to induce a magnetic field and inherently trip the sensor. We considered using a 45 degree angle on one sensor to detect the track-wire and align, but we decided to use two sensors to increase robustness. One inductor is placed on the front left corner (on the bumper) of the robot to be as close as possible to the track-wire and a second inductor directly in the front-middle of the robot to assist in the alignment to the target. Each inductor is in the optimal orientation to detect the magnetic field of the track-wire.

Mechanical Design

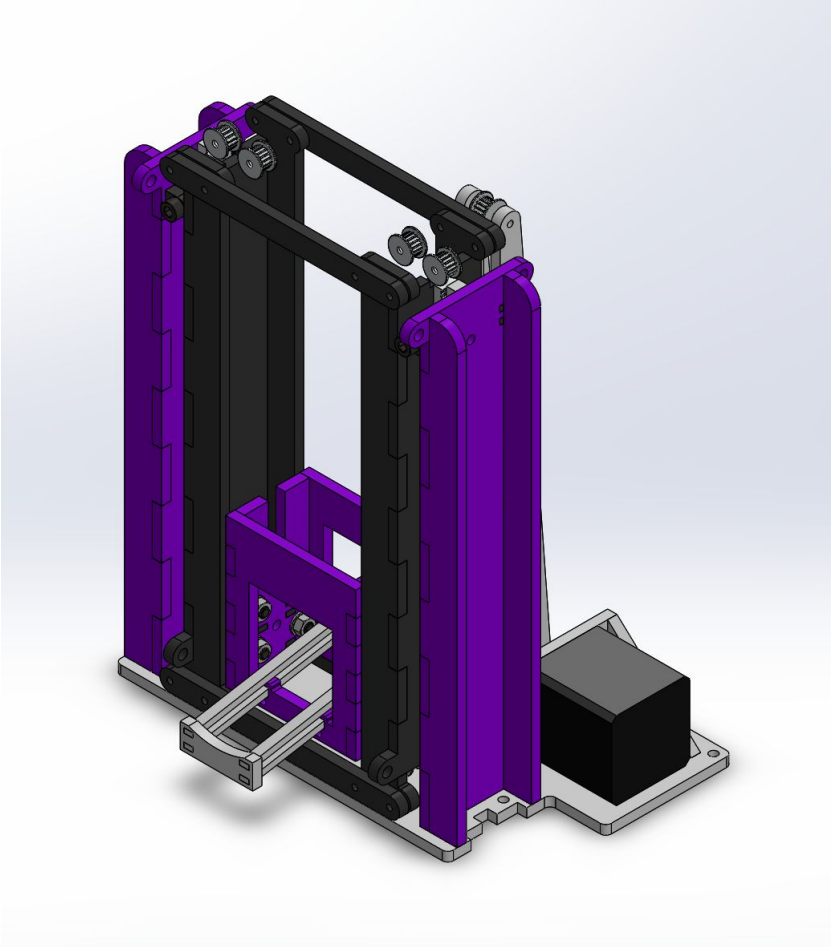
Drive System



Based on our collective previous experience building small mobile robots, and on collective class wisdom, we decided on a rear-wheel drive with three-inch rollerblade wheels powered by 200rpm 12V-rated gearhead motors, with omni-wheel casters in the front to minimize turning resistance. We integrated this design choice with our tape sensors, placing three tape sensors in the front of the robot and one in the rear exactly between the two drive wheels, optimally locating it to remain stationary as the robot pivots.

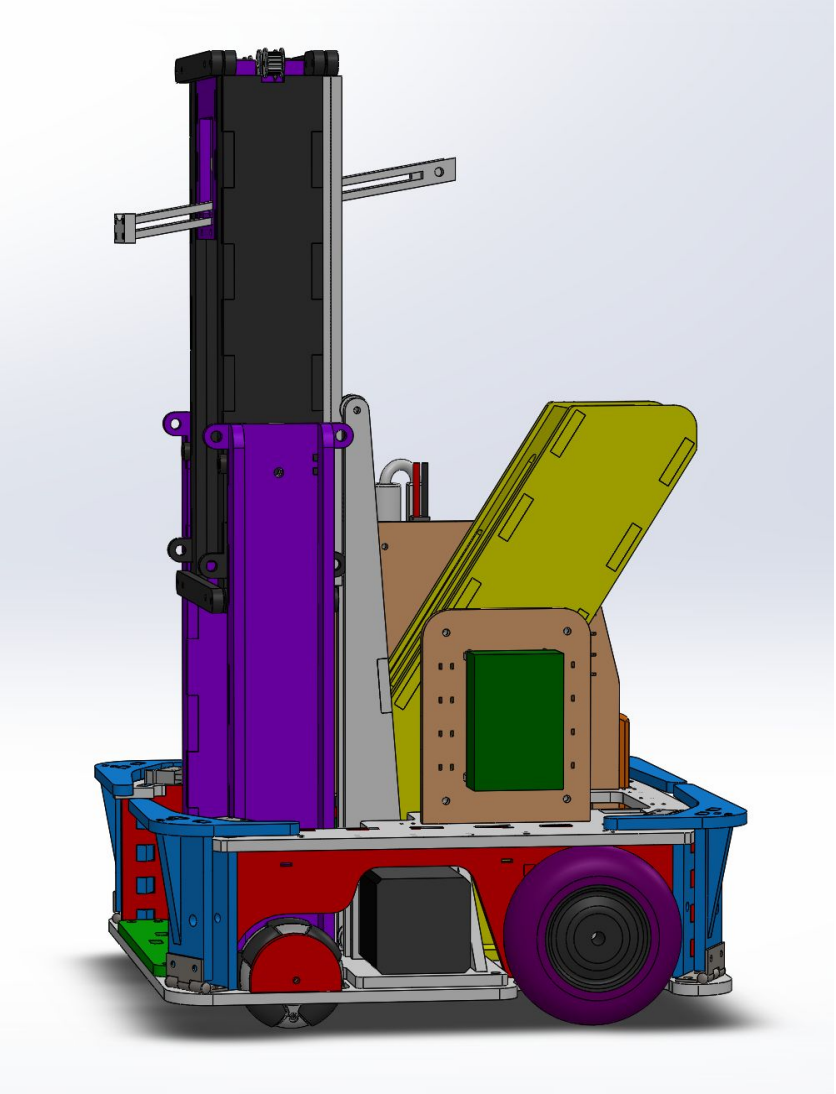
While this drivetrain design performed exceptionally well at robust tape-following, it proved less effective at enabling our robot to maneuver easily while orienting itself to the stationary targets on the field. While robots with a center-wheel drive could simply pivot in order to face the AT-M6 targets, our robot required a more complex series of maneuvers in order to “park” itself in front of the targets. We were fortunate to have good software tools in our arsenal to enable us to execute these maneuvers without too much trouble, however using these same tools with a robot that could simply pivot to face the target would have made our robot that much more effective.

Two Stage Cascade Elevator

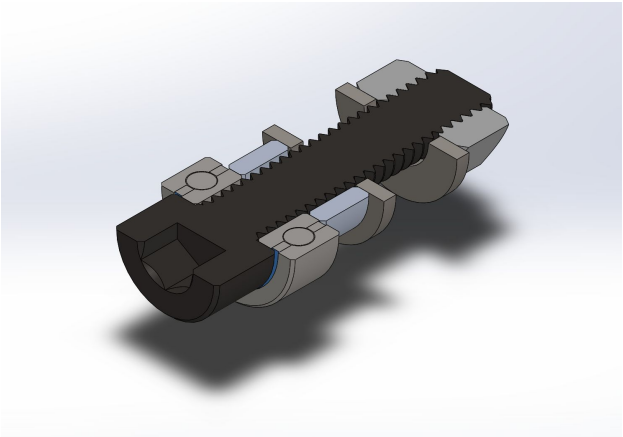


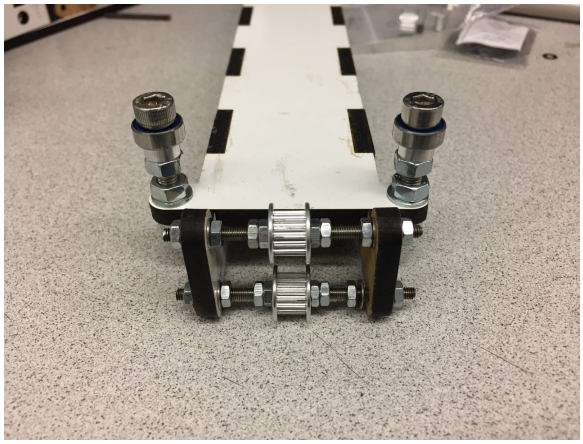
After strategizing, we believed it would be necessary to design a single mechanism capable of scoring at the height targets of both the AT-M6s and Kylo Ren's spaceship, rather than having two distinct subsystems to achieve similar tasks. Additionally, we thought that the horizontal depth variance of a single-joint arm or a four-bar mechanism would complicate the process of aligning the robot to the scoring targets. Our brainstorming process selected a two-stage cascade elevator as the best response to these design considerations, while simultaneously taking advantage of our collective mechanical experience.

As the name implies, a two-stage cascade elevator is an elevator mechanism consisting of *two* distinct sliding stages, wherein the motion of the lower stage is *cascaded* through a system of drive belts to move the upper stage, effectively doubling the range of motion of the mechanism over a single-stage elevator. There also exist two-stage non-cascading elevator designs (common in many fork-lift mechanisms), however these designs to not constrain the stages relative to one another, which can cause problems if the stages are not constructed extremely precisely or machined to tight tolerances.

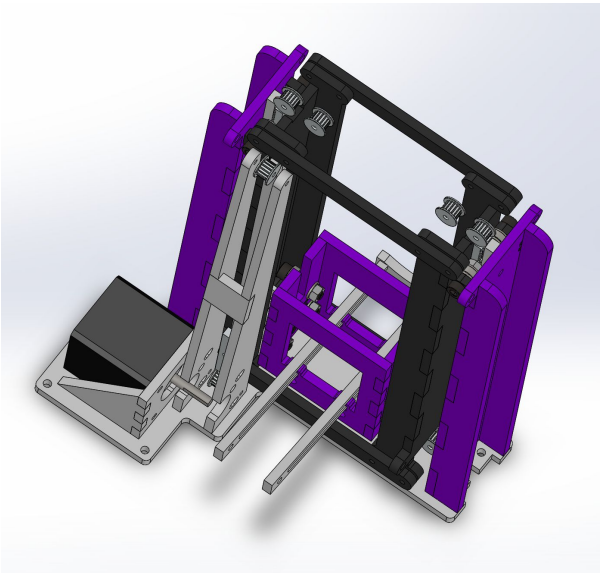


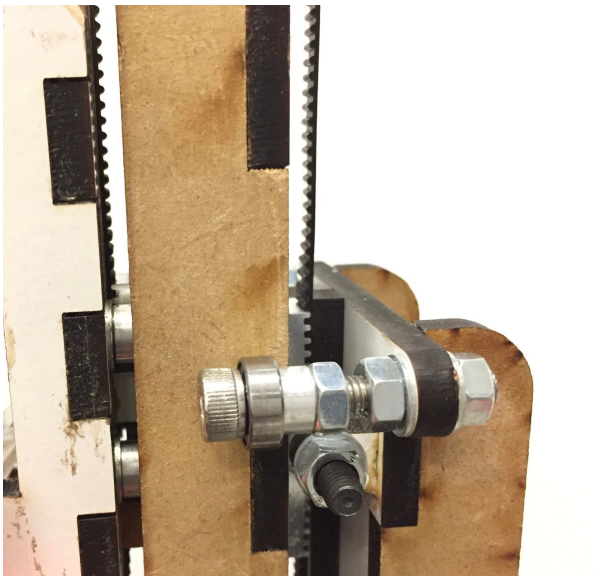
Again, our initial design required this additional range of motion because we believed it would be necessary to elevate our entire hopper and dispensing mechanism to the height of each goal, and the only way to fit this entire subsystem into the allowed starting volume while still reaching the height requirements would be to design an elevator with an exceptionally large range of motion. Later on in our design process we realized this reasoning was flawed in several aspects, however the elevator's design was so mature by that point, we reasoned that a strategic switch to a simpler elevator would cost us more time than keeping the elevator we had already designed.





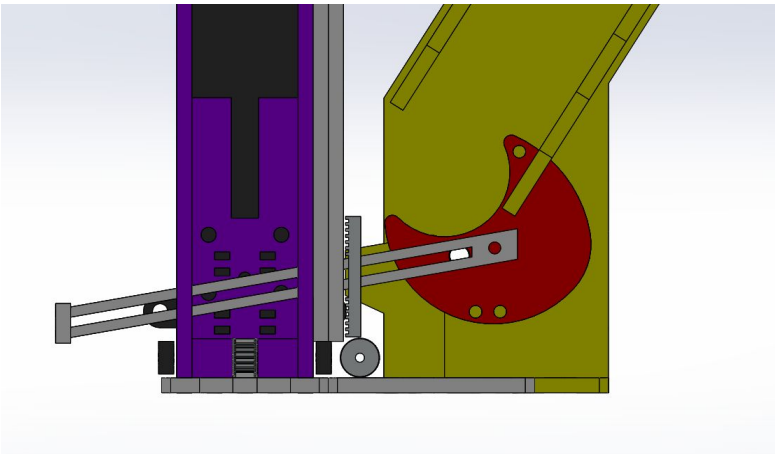
The heart of any multi-stage elevator is the design of the sliding interface. Rather than employ a simple sliding-friction interface which could bind easily and jam the elevator, we designed our own inexpensive cam followers using socket-head cap screws and miniature radial ball bearings. These were arranged in such a way as to fully constrain the mechanism and maintain its robustness through a variety of loading conditions.



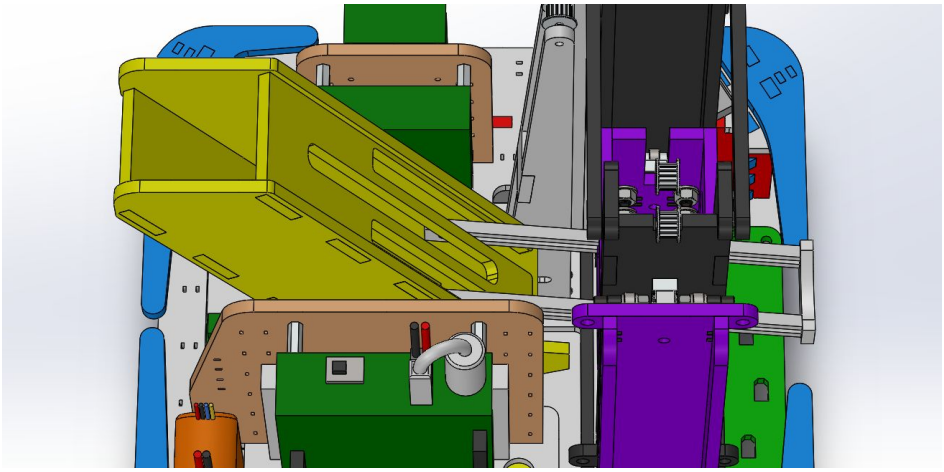


To achieve the linear (rather than rotary) drive required for an elevator, we decided on the Gates GT2 2mm belt drive component system due to the ubiquitous availability of inexpensive matching parts that are available through online distributors for the homebrew 3D printing market. We selected a belt drive over lead screws in order to make the most of the stepper motor's available power, and also to take advantage of the fact that we would already need a flexible drive system to cascade the motion of the first elevator stage to its second stage.

Ball Hopper and Deployer



In order to consistently dispense exactly one ping-pong ball at a time, we prototyped and designed a geneva-wheel inspired mechanism to dispense and deploy a single ball onto our ramp. We used Solidworks' sketch tool to ensure good geometric performance at the assembly level, and once the geometry was designed, the actuation of the wheel performed perfectly on the first attempt. To drive the wheel, we chose a servo to provide convenient and consistent position-based actuation.



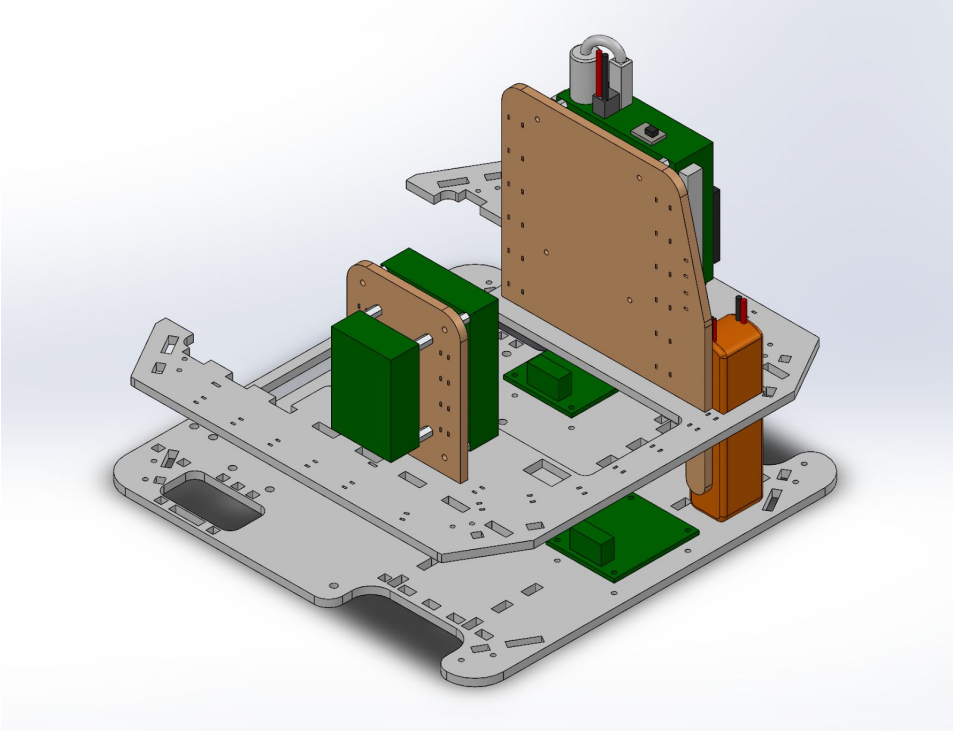
Our strategy to align to Kylo Ren's spaceship would require driving up very closely to its front face, depressing our ramp far into the robot. This added one more design consideration to the layout of our hopper, namely that our elevator would have to lift the ramp "through" the hopper in order to make the final kill! To enable this seemingly impossible geometry, we created slots in the front face of our hopper to allow the ramp to pass through on the way up. Testing revealed this slight modification worked effectively, even allowing two spare ping pong balls to remain in the after the final kill. The interface between the ramp and the ping-pong ball magazine through these slots is shown above.

Electrical Design

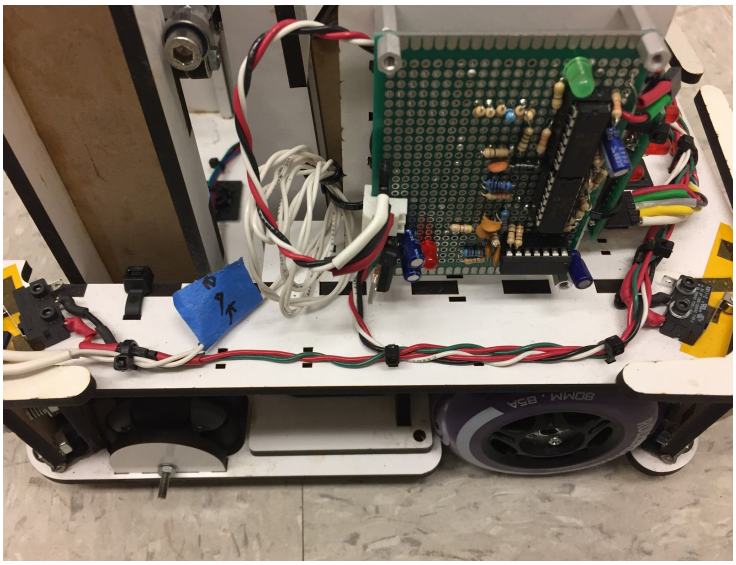
Electrical Components and Specifications

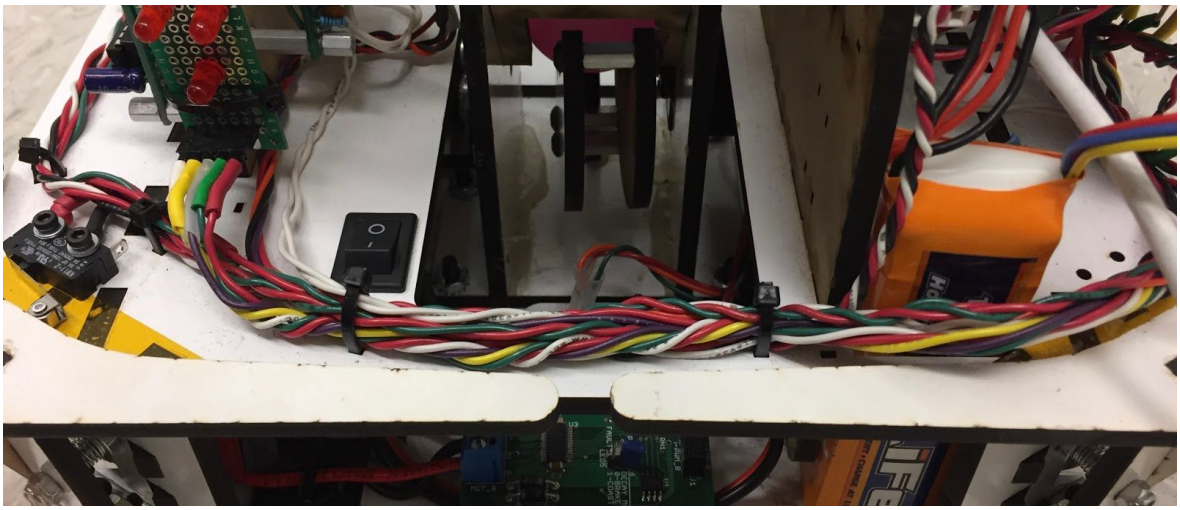
Component Name	*Description*
Track Wire Detector and Signal Filter	Single-stage multiple-feedback 2nd-order bandpass filter.
Beacon Detector and Signal Filter	Two-stage multiple-feedback 4th-order bandpass filter.
Servo Power and Stepper Signal Board	Distributes 5V power to our servo and provides pin routing for servo and stepper motor on bottom level.
Custom IO Board 1	Directly mounted to Uno, includes pin routing for bumper switches, beacon, H-Bridge.
Custom IO and Power Board 2	Directly mounted to Uno, includes regulated 3.3v power distribution, reflective optical Sensors and indicator lights, and Trackwire pins.
DRV8814 H-Bridge Module	DC motor bidirectional control with PWM for our drivetrain.
DRV8811 Stepper Driver Module	Used for driving the stepper motor that powers our elevator.
ChipKit Uno32 Power & I/O Stack	The brain and brawn of the operation: runs our state machine, performs computations, reads sensor data, generates control signals, and supplies power to all subsystems.
HobbyKing LiFe TX Battery	1500mAH 9.9V battery, 1C 15Wh. Connected to the power distribution shield on the Uno via the provided cable.

Electrical Layout and Wire Harness

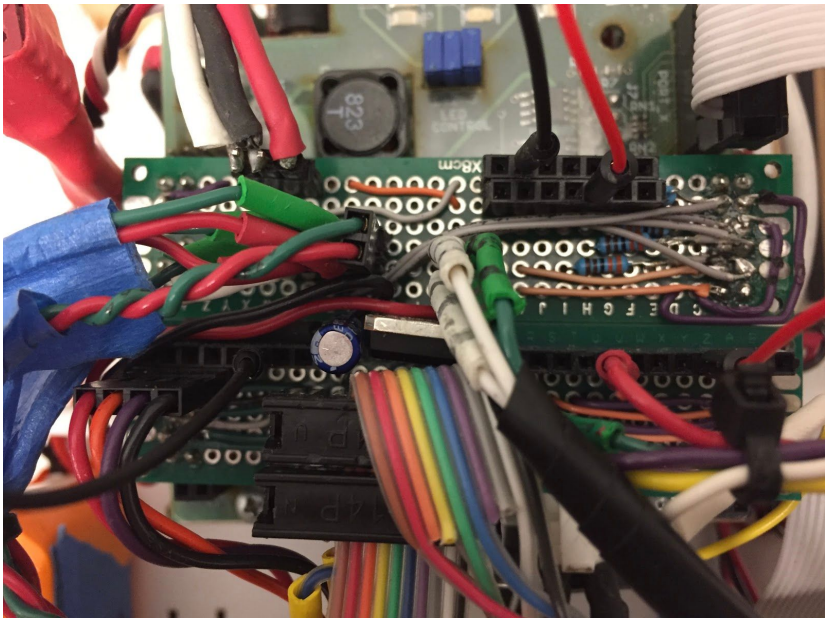


With two large mechanisms in the center of our robot, space for electronics was was very limited. In order to make the most of the available robot volume while still leaving key components easily accessible, we opted to mount the Uno I/O Stack and our main sensor interface filter boards vertically on racks, rather than horizontally on our robot. We developed crude models of each of these components in Solidworks to ensure that there would be ample room for these components without putting undue strain on the wire harness.





In order to manage the complexity of our electronic subsystems, we took care to plan the layout and wire harness for most of our electronic components in Solidworks before assembling each system on the final robot. Aside from making our robot simply look more clean and professional, this design choice made our robot much easier to troubleshoot and maintain in the long run.

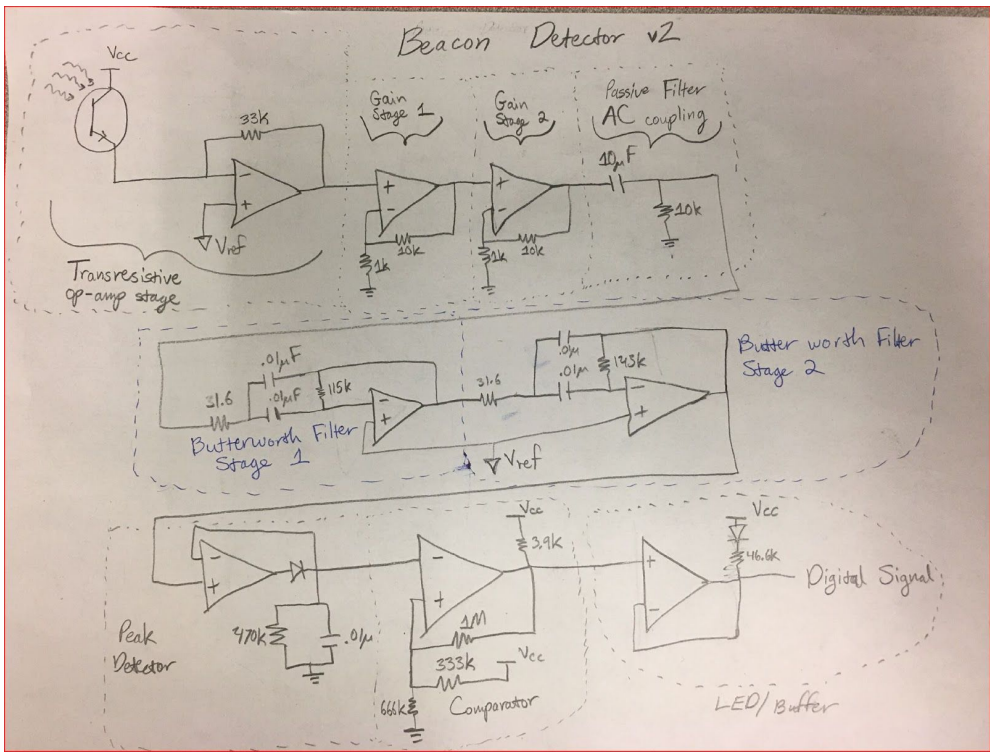


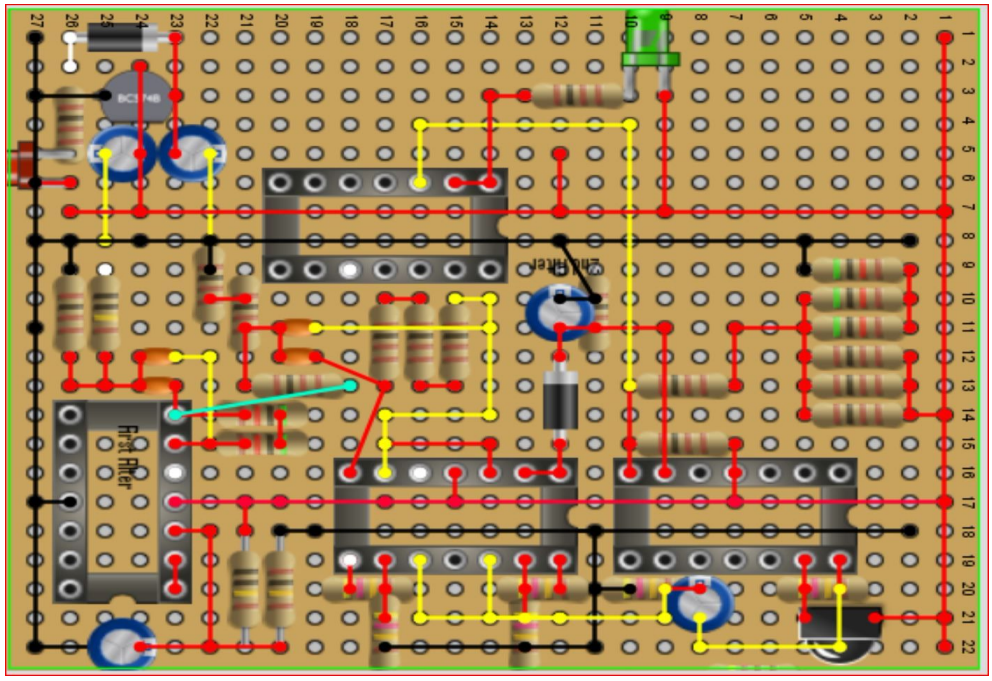
To augment our wire harness design, we fabricated I/O boards to interface directly to the Uno. These greatly simplified the process of removing the Uno I/O stack from our robot, as in most cases we needed only remove these boards as units, rather than meticulously remove and replace the wires for each individual sensor and actuator. We were extremely grateful for having taken the time for this extra layout step each time we had to replace the fuses on the Uno32 power and I/O stack.

Electrical Filters

Beacon Detector

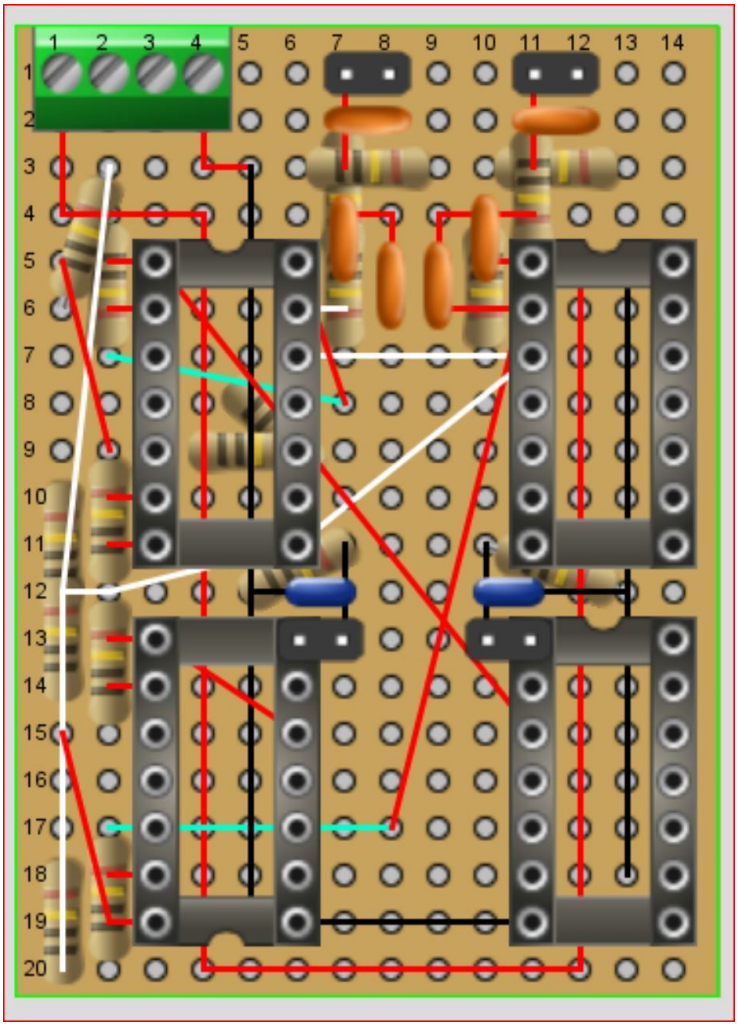
- Center Frequency 2kHz
- PassBand 400Hz
- 40dB Attenuation 2.5kHz and 1.5kHz



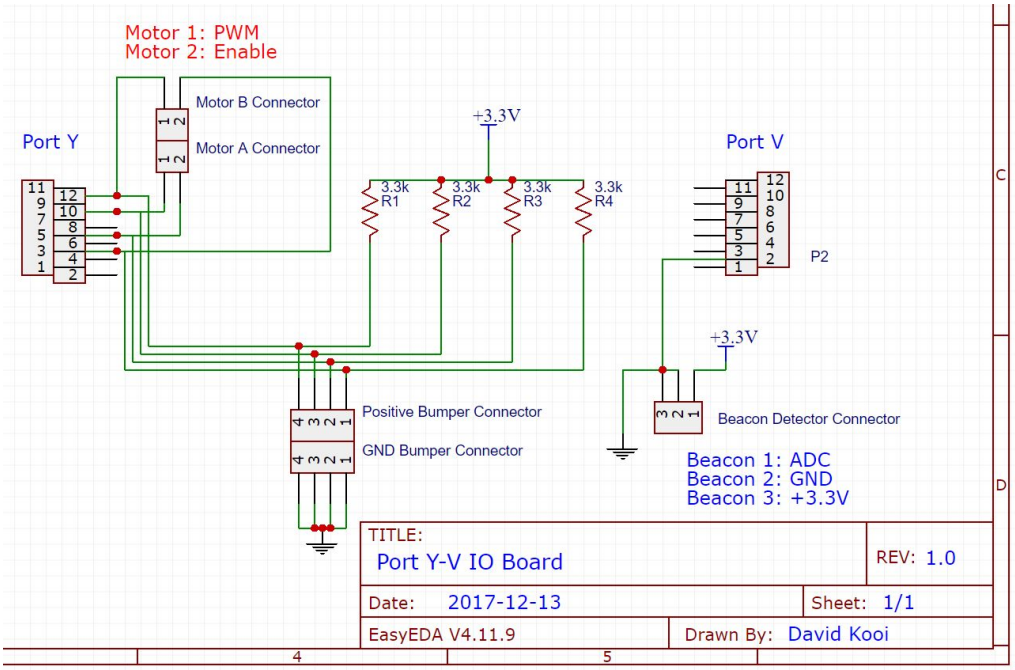


TrackWire Circuit

- Center Frequency 25kHz
- PassBand 400Hz
- 40dB Attenuation 27kHz and 23kHz

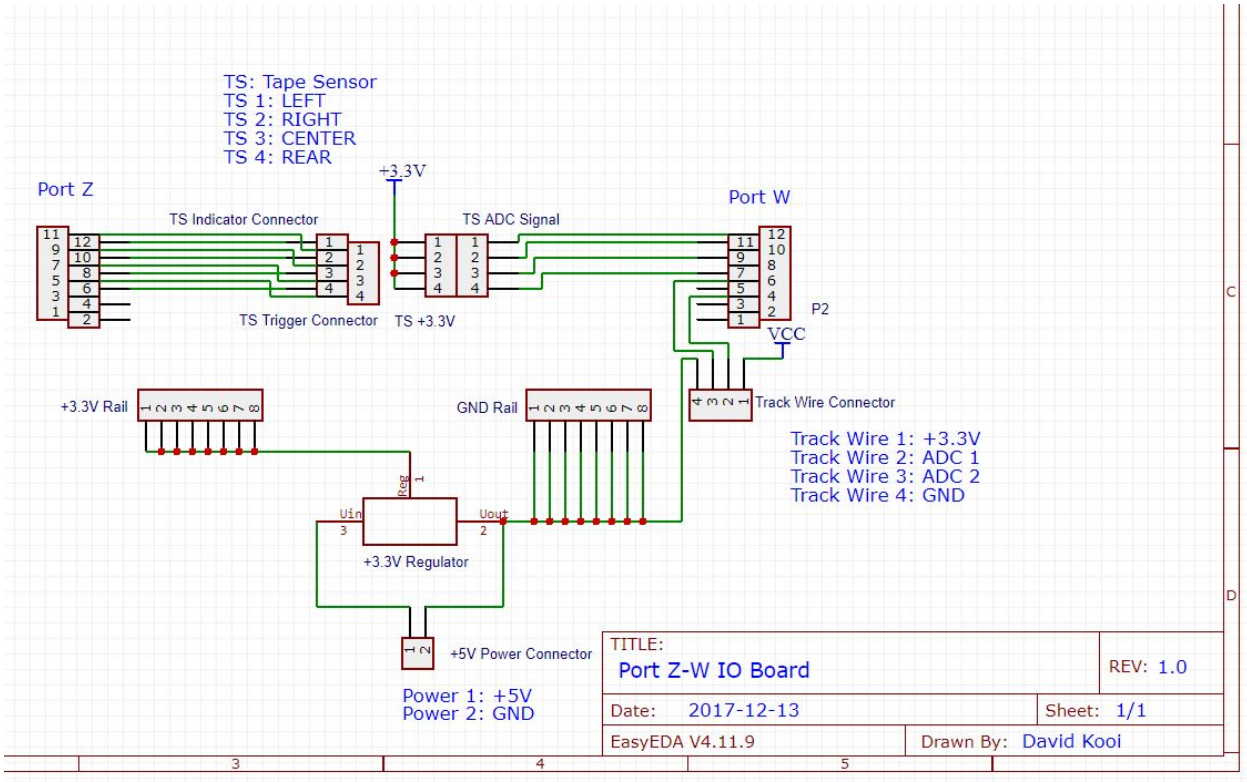


Custom IO Board 1



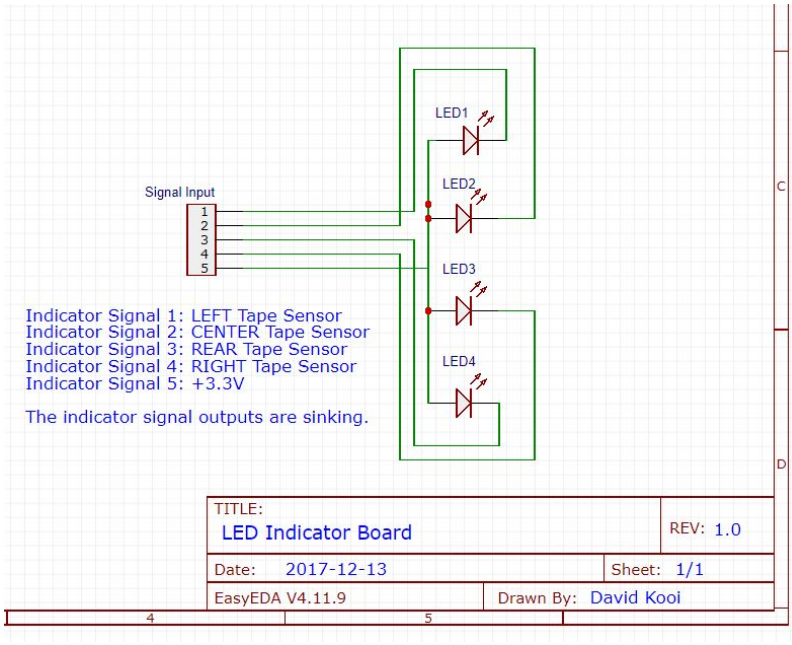
This custom board fits onto the Y and V ports of the Uno IO stack. The board provides connections for all bumper switches, the beacon detector, and motors. The motor connector aggregates the motor pins into a 2x2 header. The beacon connector does the same by providing power and ground to the beacon detector board and connecting the beacon's output signal to an ADC. The bumper sensors are using in a sinking configuration. The Uno input is pulled up to +3.3V. The switches are single pole, single throw. One side if connected to the pulled up input and the other side is connected to ground. When the switch is closed the input pin is pulled low.

Custom IO and Power Board 2



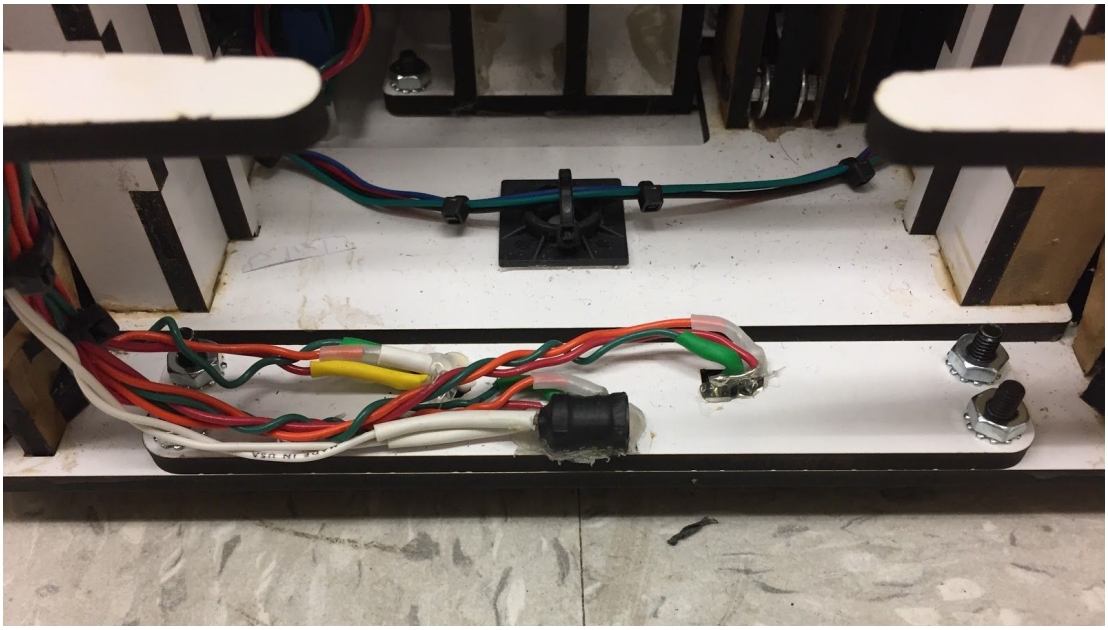
This custom board provides connections for the four tape sensors, for the track wire board, for the indicator board, and for the +3.3V and GND rails. The tape sensor connector provides a standardized interface for each of the tape sensors. We made a ribbon cable connector to ease the connection of the sensors. The tape sensors require +3.3V, a trigger signal, and an ADC connection for the photo detector output.

Tape Sensor Indicator Board



The indicator board provides a visual information about the state of the tape sensors. This has proved useful for testing. LED states are triggered in the tape sensor event checker. This allows the indicator board to operate independently of higher level processes.

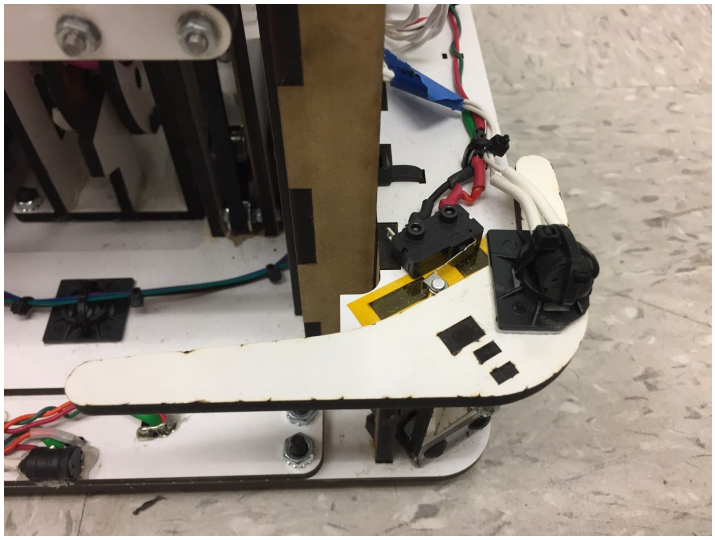
Tape Sensors



We used four Vishay TCRT5000 reflective optical sensors to detect the tape. The sensor works by exciting the base of a phototransistor with the reflection of a photo emitter. An applied voltage over the collector and emitter causes a current to flow in proportion to the base current. When on white a lot of light is reflected and the phototransistor drives a higher current than the less light is reflected on black tape. We digitized the output signal and set the high and low level thresholds in software. We drove the photo emitter sinking current into a digital output pin. The PIC32 IO board can sink 12mA maximum. This results in 40 mW of driving power. The sensor spec sheet tested at 75mW. Using a TIP Darlington transistor would provide 75mW to the emitter, but after testing, we decided the simplicity of driving the emitter without a transistor was worth the decrease in power.

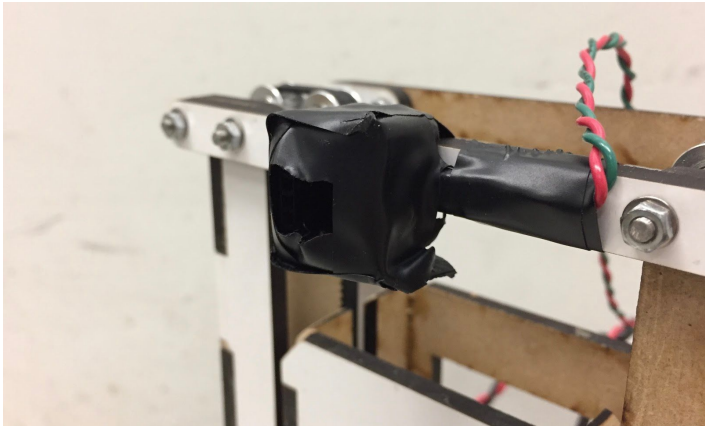
First we connected the photo-transistor emitter directly into the ADC. But because the ADC is high impedance the current from the photo-transistor could not drain. A pull down resistor applied to the ADC input allowed for changes in voltage to be detected.

AT-M6 Trackwire Sensors



The AT-M6 targets contain an electrically oscillating wire. The wire oscillates at 25kHz. We upgraded from the10 mH inductors supplied in the kit, opting for larger 33mH inductors we happened to have on hand in order to get better reception of the trackwire signal. We used this inductor in an RLC tank circuit to generate current when the inductor came close to the AT-M6's magnetic field. The RLC tank circuit has a resonant frequency of 27kHz. The output of the tank circuit is filtered, then amplified. The amplified signal is input into an ADC pin of the PIC32. We set high and low thresholds in software to implement hysteresis.

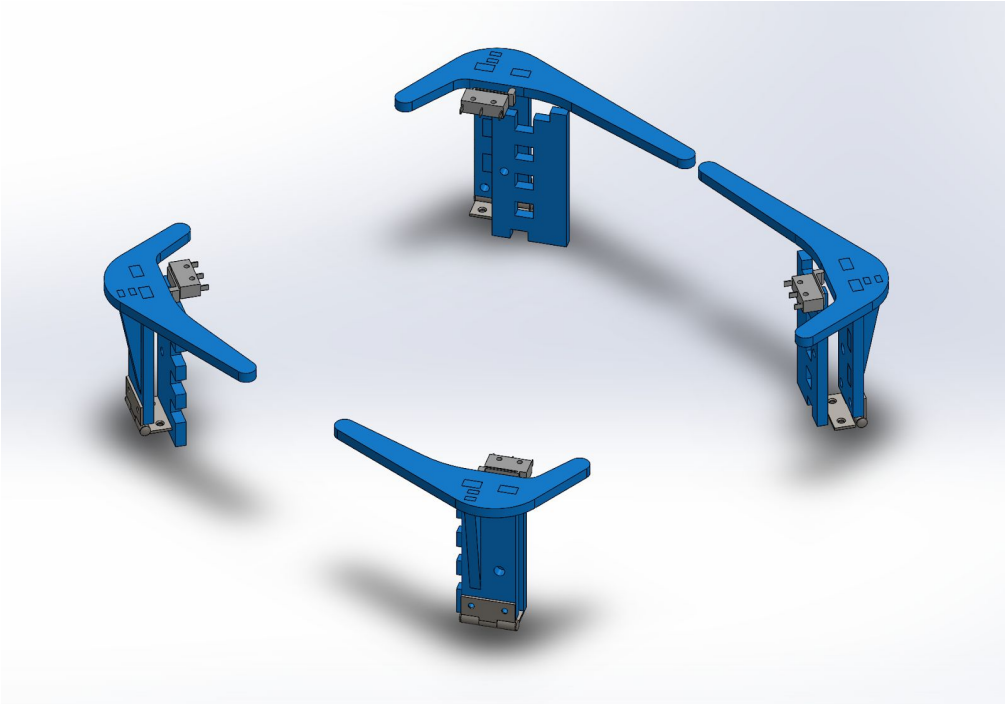
Ren Ship Beacon Sensor



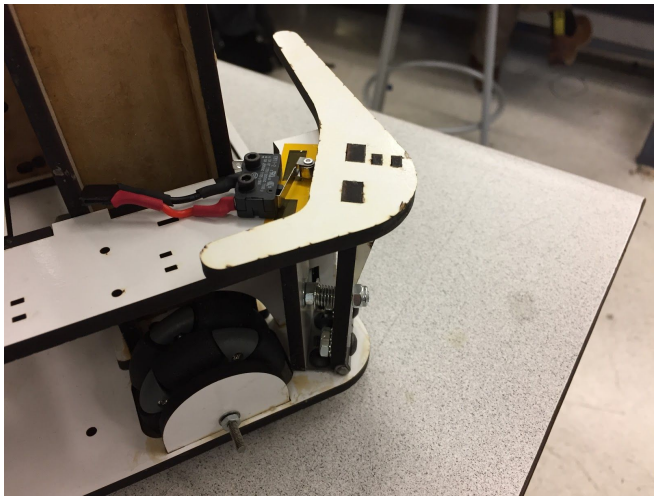
The ren ship houses a flashing infrared beacon. We used a phototransistor to receive the signal. The phototransistor output is run through a trans-resistive amplifier, filtered, and amplified again. This signal is input into an ADC pin of the PIC32. We set high and low thresholds in software.

Bump Detectors

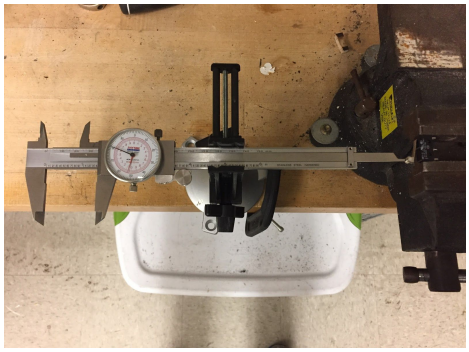
There is a randomly placed obstacle on the playing field. We use four switches to detect physical contact with any obstacle.The bump detectors are action limited by MDF hardware. This ensures the delicate switches do not take load. The switches were used in a sinking configuration. One end of the switch is connected ground and the other end is connected to a PIC32 digital inputs. The inputs were pulled up to 3.3V. When the switch is actuated a circuit is completed and the input pin is pulled low. Circuit details can be seen in our electronic diagrams.

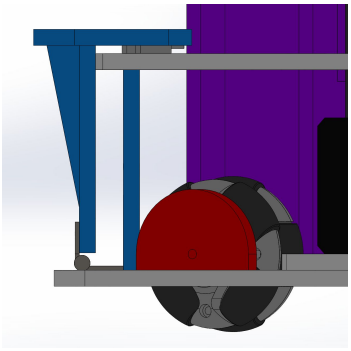


As a fundamentally mechanical sensor, the bump detectors would require a robust and effective mechanical interface between the sensors themselves (small microswitches) and the bump events they would need to detect. In particular, the bumper interface would need to serve three key requirements: (1) isolate the delicate microswitches from the comparatively extreme forces involved in bump events, (2) supply the spring force required to release the strike plate to its resting position after a bump event, (3) redirect bump forces to actuate the microswitch, enabling the bumpers to consistently detect bump forces from a variety of directions and locations across the face of the strike plate by converting these forces into one consistent motion at the position of the microswitch lever. By satisfying these requirements, we were able to avert all of the mechanical problems we observed with the bump sensors on the roaches in Lab 0.



After brainstorming the mechanism, we decided on a remote-hinge pivoted strike plate. Initially we desired a well-constrained sliding interface, but we quickly realized that a sliding interface that met our requirements would either be difficult to implement or ineffective. The hinge pivot design served as a simple and inexpensive way to approximate the sliding motion we desired at the height of the microswitch, and effectively redirected bump forces into the requisite sliding motion. A small spring halfway up the “diving board” provided the force necessary to de-actuate the strike plate after each bump event.





In order to shield the microswitches from bump forces, the mechanism was designed to bottom-out on itself rather than the microswitch. This simply required placing the mechanical stop at the appropriate location to guarantee that the microswitch lever would always fully actuate, but never come to rest on the housing of the switch itself. To measure the actuation distance necessary to achieve this, we placed one of our microswitches in a vise in BE138 and used dial calipers to measure the requisite actuation distance.

In the end, our bump sensors met all of our requirements, effectively shielding the microswitches from bump forces, ensuring consistent spring-return to the released position, and converting a variety of bump force scenarios into one consistent motion to be read by the microswitch.

Bill of Materials

When our robot was mechanically complete, we tallied up the individual costs of the components we purchased in order to complete it, and discovered that we had exceeded our $150 budget by about 33%. The budget overrun did not owe itself to expensive components in any one subsystem, but rather stemmed from the high complexity of our robot and the many components it required.

In fact, at nearly every phase of the design process, we worked to minimize costs where appropriate while still meeting requirements, and nowhere was this process taken more seriously than the elevator, which turned out to be our most expensive subsystem. We could have opted for an easier-to-build design using more modular off-the-shelf components, which would have simplified the engineering process considerably, but instead we chose to break the problem down as far as we possibly could in order to cut down on component cost. (For what it's worth, Joe would definitely opt for this easier approach in any future project involving a cascade elevator and disregard the cost overruns to save time.) In the end, the most expensive line-item in the elevator subsystem was the roller bearings, however if one were to remove these from the cost of the elevator, the robot would still be over budget.

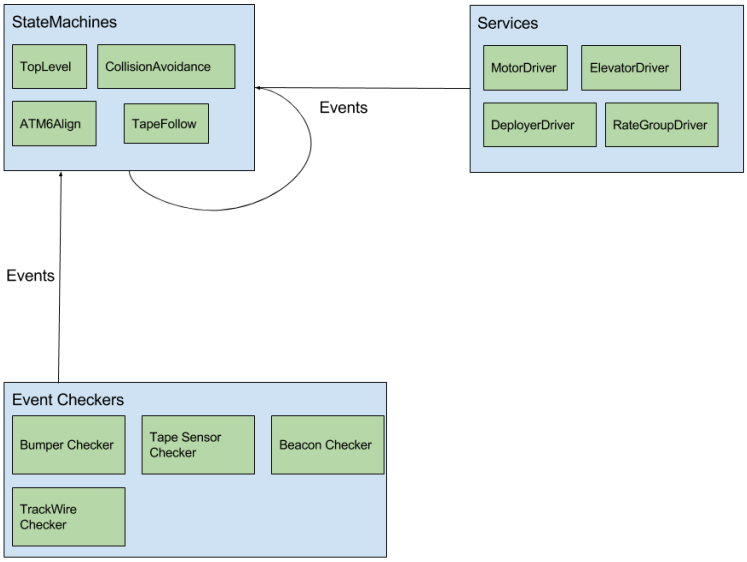
Taking a step back, the cost overrun makes sense in that it matches the time-complexity overrun of the robot's design. Premature optimization in several parts of the robot's design resulted in an unnecessarily complex design, which may have performed beautifully, but ultimately did not well-enough respect the constraints of the final project.

As a side note, if the CMPE118 staff wishes to keep any part of our robot as penance for exceeding our budget, we will gladly comply, asking only that the mechanism in question be kept as an example for future classes of the engineering process, whether that reflects well on our choices or not.

(Insert BOM PDF here [delete this page from final report PDF {to keep page numbers consistent}])

Software Design

General Software Overview



Our software was composed of three main parts: Event checkers, Services, and State Machines. The software is built on the provided Events and Services framework. This framework provides and event driven programming interface. An event driven system is useful for running multiple processes asynchronously. Instead of blocking calls, events are sent between modules. The Event and Services library maintains an event queue and distributes the events to the proper modules. In effect the services, state machines, and event checkers run as threads.

State Machines and Services are outlined below.
The event checkers contain ADC thresholds and emit events to the state machines and services.

State Machines

Top Level

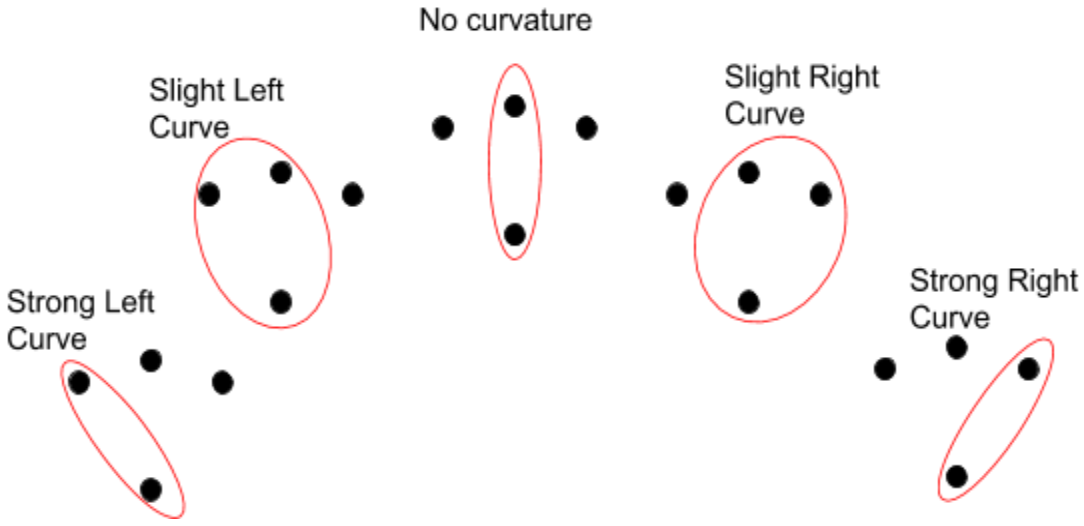
The top level state machines controls the high level logic of the robot. These include states like: ATM6 Search, Ren Ship Search, and Collision Avoidance.

ATM6Align

This state machine handles the maneuvering and alignment of the robot to the ATM6. This state machine is initialized when a track wire even is detected. After detected, the robot maneuvers to make its front face center aligned with the target. Once in this position the robot scans through the magnetic field of the track wire. While scanning a discrete derivative is taken from the measurements. When the derivative is close to zero we know we are at the peak point. The robot stops rotating and moves forward to deposit the ball.

Tape Following

The tape following state machine operates on received tape sensor values. We from our three forward tape sensors we created five weight states. Each state affected the robot's curvature of trajectory.



Collision Avoidance

There are two obstacles the robots encounters:

- The randomly placed box
- The Ren ship

The robot's collision avoidance behavior depends primarily on the robot's state of achievement. When the robot has not completed it's ATM6 goals, the ren ship and the random obstacle are equivalent: The robot needs to traverse around the obstacle and identify the tape sensor.

After the ATM6 goals are accomplished it is necessary to distinguish the ren ship from the random obstacle. This is done by raising the elevator upon bumper contact. If a beacon event is received the robot knows to attempt a ren ship alignment. Otherwise the conventional collision avoidance maneuver is executed.

Services

Rate Group Driver Service

The rate group driver is a service that provides different ‘rate groups’ to periodically run functions like event checkers or the motor drive service.

Motor Driver Service

This service periodically services the motor pwm signals to follow user generated commands or trajectories.

Deployer Driver Service

The deployer driver asynchronously deploys a ball from the ball hopper onto the elevator ramp.

Elevator Driver Service

The elevator driver asynchronously moves the evenator between height states.

Enhancements and Cool Features

In order to better align to the AT-M6 targets, we interpreted the analog data from our “strong” trackwire sensor by taking a digital derivative of the signal, using this derivative to detect a peak, and stopping the sweeping motion upon detection of said peak. Since we lacked experience in designing digital feedback estimators such as a Kalman filter or other state-of-the-art techniques, we instead opted for a simpler open-loop approach, taking a least-squares best-fit line of the previous 64 ADC samples, and using the slope of this best-fit line as the estimate for our derivative. This technique proved extremely robust, however it did introduce a delay of approximately half the sampling period, and this, along with the robot’s momentum during the sweeping motion, required us to execute a five-degree pivot trajectory in order to return back to the center of the AT-M6 target.

A snippet of least-squrares derivative estimation code is shown below:

```
int32_t estimateDerivative(Derivative d, int32_t y) {
    d->preMean -= d->yData[MOD_LENGTH(d->frontIndex)];
    d->yData[MOD_LENGTH(d->frontIndex)] = y;
    d->preMean += d->yData[MOD_LENGTH(d->frontIndex)];
    int32_t SSty = SStx(d);
    d->frontIndex = MOD_LENGTH(d->frontIndex + 1);
    d->v = (1000 * SSty) / d->SStt;
    return d->v;
}
```

Integration and Testing

We iteratively performed our integration and testing. After each major software addition we tested the code. If the code was satisfactory we made a version control commit. If the code was not satisfactory we rethought our approach. For mechanical components we designed prototypes to achieve a proof of concept. For electrical design we designed and tested on a breadboard before moving the design to a perforated circuit board. An example is when we achieved a proof that our photo detectors would work by sinking current into the PIC32 Stack.

Conclusion

Fortunately for our robot's performance, the engineering of our robot, through mechanical, electrical, and computational subsystems proved to be our strength. The mechanisms we designed worked effectively in concert with our electrical subsystems; we rarely experienced the interfacing problems we saw plaguing other teams such as electrical interference with our sensor readings, and despite some strategic missteps, we had the technical prowess and know-how to solve the problems we created for ourselves, having enough software tricks in our back-pockets to solve nearly any problem that presented itself to us, from our use of a least-squares derivative estimator to interpret the signal from our trackwire to the use of motion-profiled kinematic trajectories to elegantly choreograph our robot's motion across the field.

However, our greatest weaknesses were our executive skills; at many points in our project we lacked the wisdom and foresight to understand the complexity we would encounter along the path we had chosen for ourselves. While we were consistently able to meet our technical milestones, we were consistently unable to deliver them on-schedule. Perhaps our biggest takeaway from this project will not be the electrical, mechanical, or software engineering skills we learned along the way, but the brutal lessons these tasks taught us in time management.