Team Dragomight

1895

Engineering Portfolio

Florida FTC
R.O.B.O.T League
2021-2022 Season

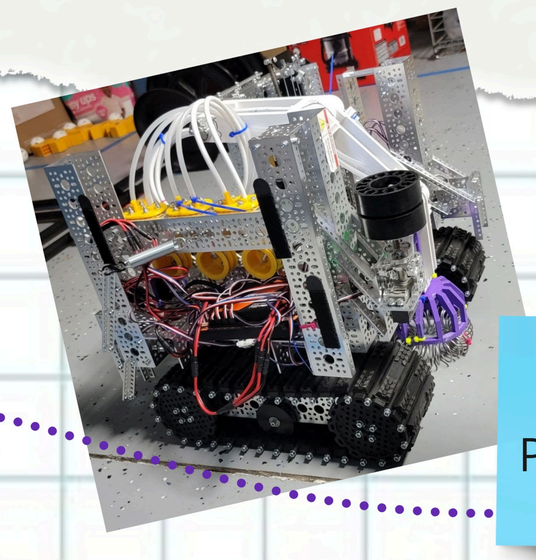
Team Member:
Colin "Valus Valoo, Ignition Bill, Valroth, Beedle"

Coaches:
Randy Huegele
Indah Huegele

19895

Introduction

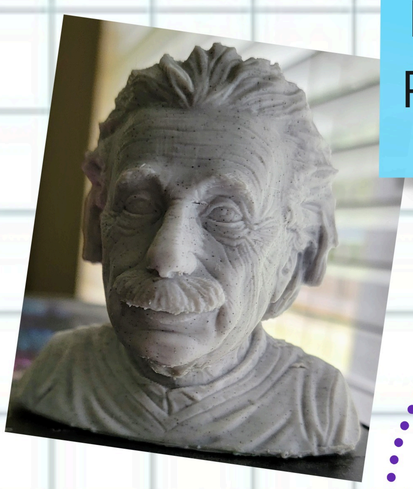
This Engineering portfolio contains a summary and illustrations of Team 19895 robot design and building process for FTC 2021 - 2022 season. It is divided into 7 major sections: Team information, Robot building process, Physics, Control, Programming Wiring , and Planning.



Team: Page 1

Robot: Page 2-9

Physics Page 10



Planning (SOAP) Page 15

Wiring Page 14

Control Page 11-12

Programming 13

Coach

Team

Future Recruit

Team Roles & Responsibilities

Sub-Teams	Strategy Team	Build Team	Programming Team	Logistics Team	Communications/Marketing Team	Fundraising/Business Team
Members	Valus Valoo	Valus Valoo	Valus Valoo	Valus Valoo	Valus Valoo	Valus Valoo
Responsibilities of Team	Decides features of the robot and how its going to be used to win the game. Tests and verifies the features of the robot and program when its delivered.	Assembles the robot according to the plan.	Programs the robot according to the plan. Maintains source control of the code. Uploads the code to the robot. Tests and verifies the code functionality.	Organization, documentation, administration, delivery, sorting. Creates high level plan for development of robot and programming.	Recruit team members. Develop team theme. Spread information and interest in FTC.	Find sponsors. Apply for grants.

Strategy

1895

In-order to win the match we must score the most points. To maximize the final score we need to pick objectives that give the most points in a short amount of time. We could estimate the points per time for each activity but they would be different for every robot design so we need to collect this data. This data could be used to determine which design elements preform the best. We can also use that data to decide which actions will be most helpful during the match. So the data decides what goes on the robot as well as the behavior.

Autonomous (30 Seconds)

Objective	Point Value	Time Estimate (Seconds)	Points Per-Time	Chance of Success	Payoff
Delivered Duck via Carousel	10	3		1	
Parked In Alliance Storage Unit	3	driving time from where ever you are	3	1	3
Parked Completely In Alliance Storage Unit	6	driving time from where ever you are	6	1	6
Parked In Warehouse	5	driving time from where ever you are	5	1	5
Parked Completely In Warehouse	10	driving time from where ever you are	10	1	10
Freight Completely In Alliance Storage Unit	2	5	12	1	6
Freight Completely On Alliance Shipping Hub	6	5	36	0.5	9
Duck used to detect Shipping Hub Level	10	15	20	0.5	10
Team Scoring Element used to detect Shipping Hub Level	20	15	40	0.5	20

Player Controlled Phase (1 Minute, 30 Seconds)

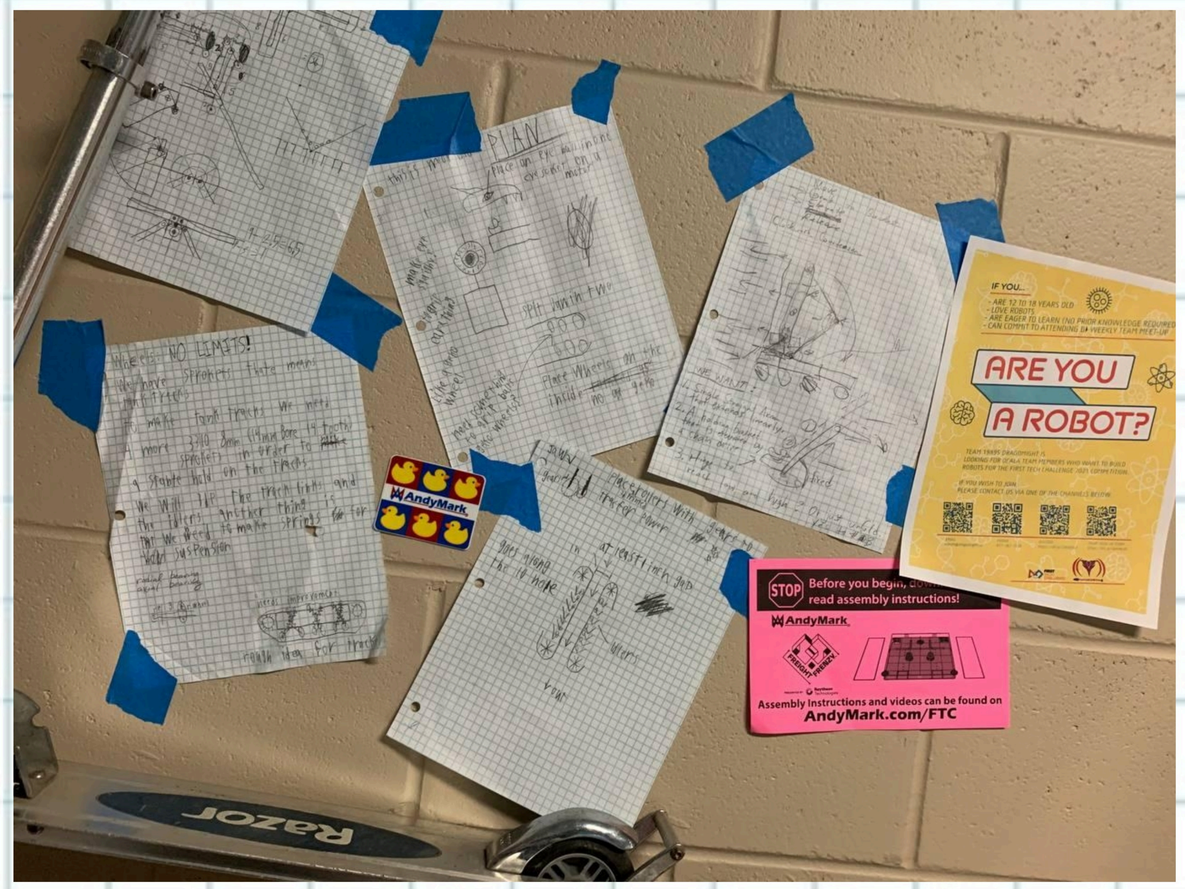
Objective	Point Value	Time Estimate (Seconds)	Points Per-Time	Chance of Success	Payoff
Freight Completely In Alliance Storage Unit	1	5	18	1	18
Freight in Alliance Shipping Hub level 1	2	5	36	1	36
Freight in Alliance Shipping Hub level 2	4	5	72	1	72
Freight in Alliance Shipping Hub level 3	6	5	108	1	108
Freight Completely On Shared Shipping Hub	4	3	120	1	120

Endgame (30 Seconds)

Objective	Point Value	Time Estimate (Seconds)	Points Per-Time	Chance of Success	Payoff
Delivered Duck or Team Shipping Element via Carousel	6	3	60	1	60
Alliance Shipping Hub Balanced	10		10	0.5	5
Shared Shipping Hub tipped toward Alliance	20		20	0.5	10
Parked In a Warehouse	3	driving time from where ever you are	3	1	3
Parked Completely In a Warehouse	6	driving time from where ever you are	6	.999999999	6
Alliance Shipping Hub Capped	15	15	15	0.5	7.5

Brainstorm

1. During the autonomous phase you can send the robot over to the warehouse so when the player controlled phases comes you start in the warehouse.
2. It is a lot easier to make the shared shipping hub tipped to your side than to make the alliance hub balanced. also the point value for tipping the shared hub is higher. the shared shipping hub is closer to the warehouse than the alliance hub.
3. During endgame it is best to do the ducks however if your allied robot is doing the ducks you can go place your shipping element on the alliance hub or you can ensure that the shared shipping hub is balanced. after you do that you can park in the warehouse.
4. During autonomous you if your allied robot goes for the ducks you can go send your robot to use the barcode however if you are alone it is best to just go for the ducks.
5. The numbers are highly dependent on our understanding of the rules.
6. Next steps we put together actions into plans and figure out payoff off plans.
7. An analysis of strategy similar to the one we are doing here is one way you can make a robot think or make decisions.
8. We don't have time to test everything against the base line so we just test the ideas we think are best first.



Product Requirements

1895

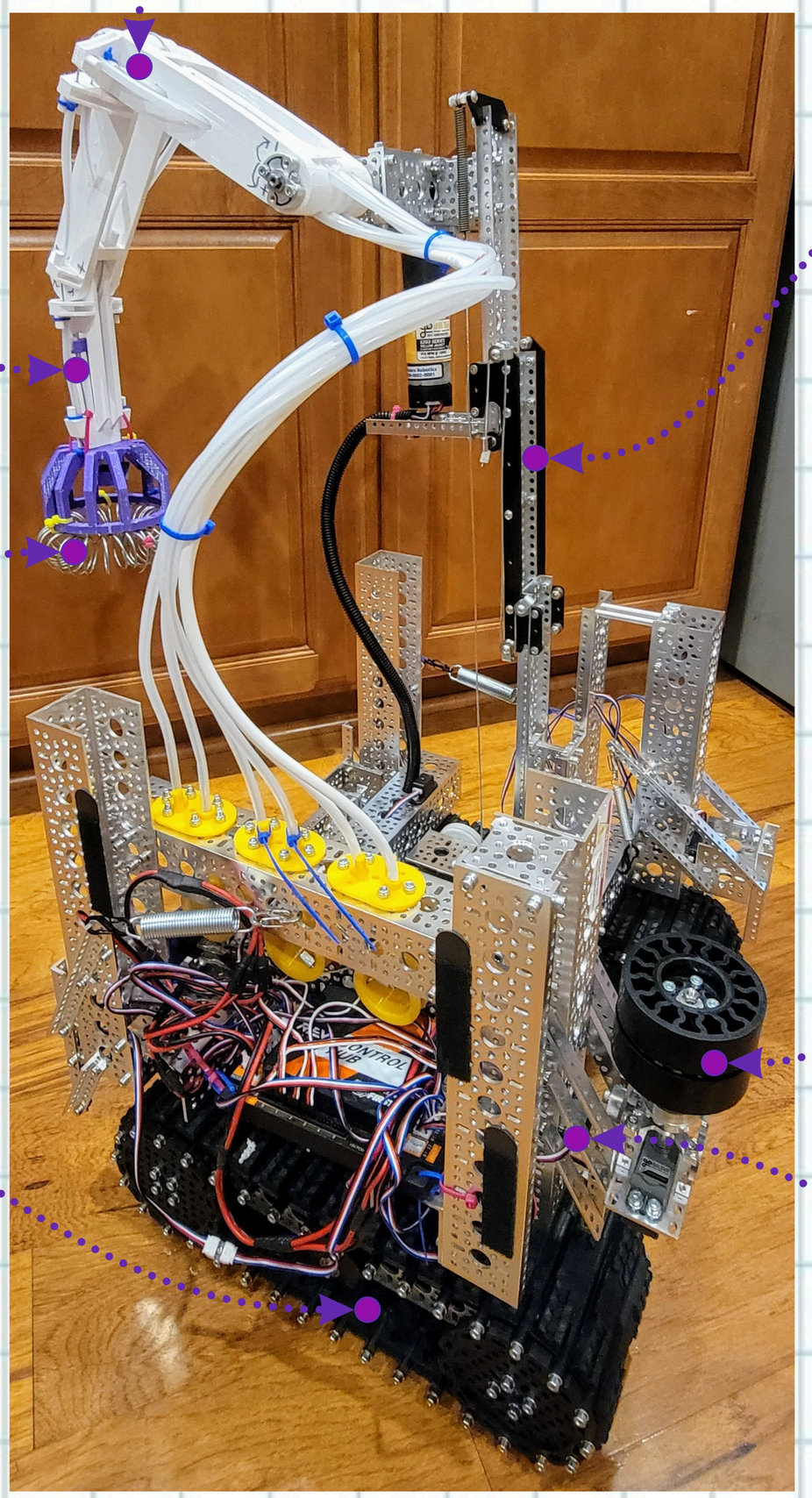
The robot needs many features to complete the challenges in the competition.

Low | Medium | High | Very Very High

Requirement	Importance	Solution	Source
Must be remote controllable by driver. The driver must drive the robot from the driver station.	High	Develop Controller Software	DADA
Must pick up a duck, ball, and weighted block.	High	Develop Grabber	
Must move quickly but under control and be able to act as the base of the robot and not get in the way of other components.	High	Develop Drive Train & Contol System	
Must be able to carry a duck, block, or ball	High	Develop Grabber	
Must release a carried duck, ball, or block at different heights	High	Develop Releaser & Elevator	
Must fit inside a 18” cubic box	High	Develop Extender	
In autonomous, must navigate to various destinations, without getting stuck, lost or breaking anyone or thing. or in other words without causing a robot uprising.	High	Develop Autonomous control	
In Autonomous, must recognize duck barcode, or shipping element, and place preloaded block.	High	Develop Autonomous Recogniton	
Must have an easily accessible on/off switch.	High	Develop On/Off Switch	RE01
Team Number must be placed on robot according to the rules.	High	Develop Team Sign	RG04
Must be built only with allowed parts.	High	Develop A CRICLE AND SQUARE	game manual 1
Must maximize game score.	High	Develop A Strategy	
Should display team themes.	Very Very High	Develop Thematic Elements	
Must spin carrousel fast to deliver a duck without tossing it off before touching the scraper.	High	Develop Saute	
Thout shalt not toss thine shipping element further than 16 feet (4.88 m) nor higher than 5 ft. (1.52 m).	High	Develop Thine Thrower Arm	RG07

Robot Overview

1895



Arm
The arm is made up of three joints, the shoulder, elbow, and wrist.

Releaser
The piston pushes cargo out of the spring to depost.

Grabber
The spring holds cargo inside. The spring was made by wrapping a length of piano wire around a pipe then zip tieing the ends together

Drivetrain
The tracks are very good at driving over the barrier at higher speed.

Elevator
The elevator moves the shoulder of the arm up and down so the arm does not need to be very long.

Duck Spinner
We took a super speed servo from Go Bilda and placed a gecko wheel on it to act as the duck spinner.

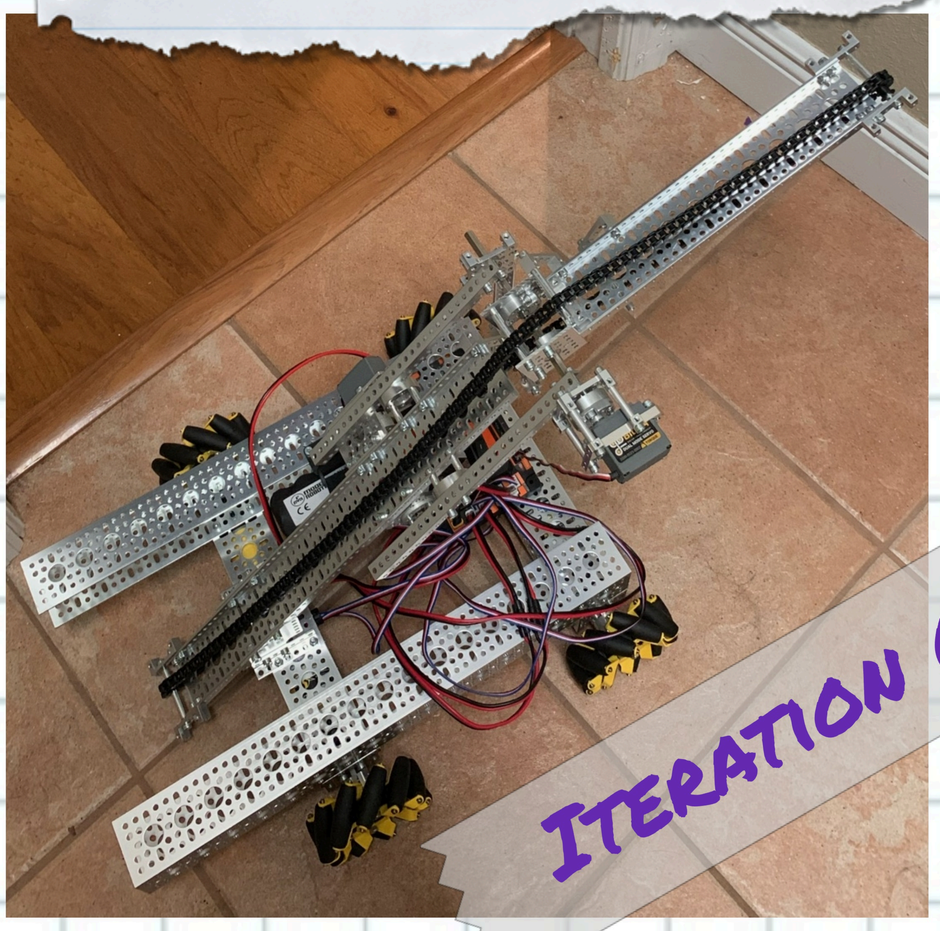
Sensors
There are four bumpers with touch sensors inside so when the robot bumps into things it stop. There is also a magnetic sensor in the elevator to stop the motor.

Improvement Ideas

- Make Bumpers Better
- Make Better Wire Management around Hub
- Move Duck Spinner Higher
- Program Arm Positions
- Make Wire Tensioner
- Add Gripper Changing Function
- Program Autonomous
- Mount Potentiometers
- Make Mount For Camera
- Make Crimping in The Pulley Easier
- Make Better Cable Management in Arm
- Make Other Grippers
- Install Microtron's Hand

Robot Evolution

1995



Iteration 0

Goals

Our goal with iteration zero was to attempt to make our own rendition of what seemed to be the concept that most other teams were doing.

Results

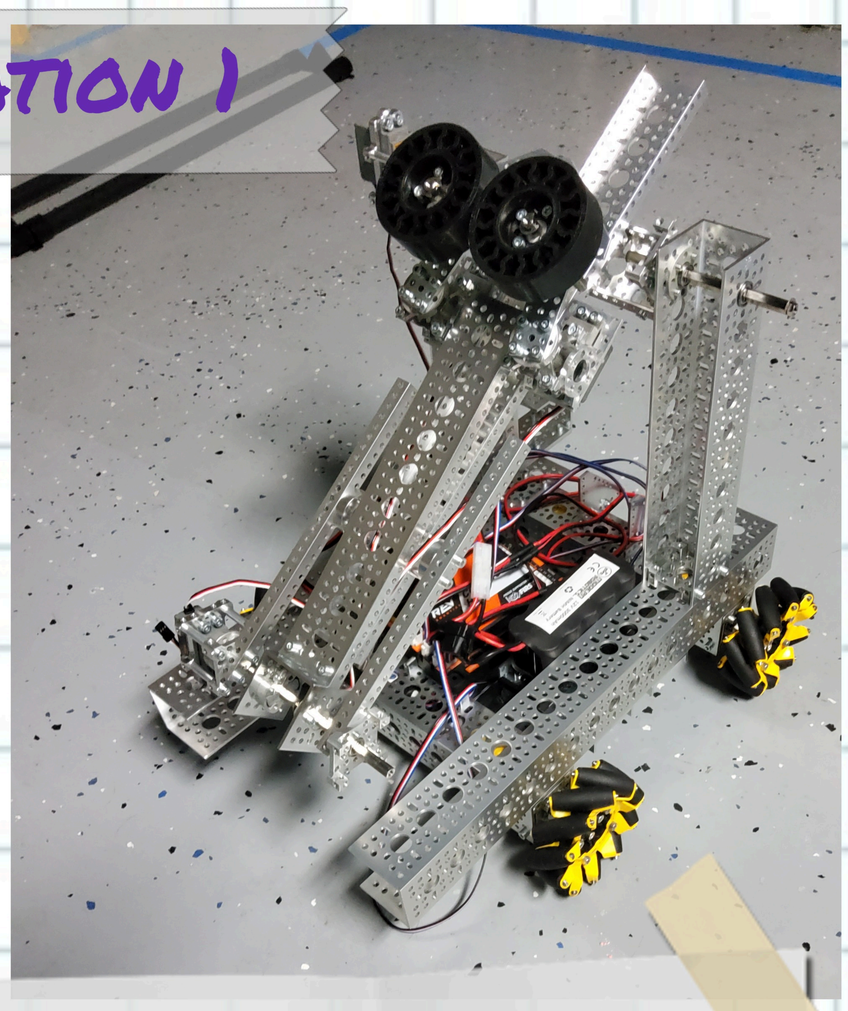
Iteration zero was our attempt to make something similar to what other teams were doing except it used a chain rather than a slide.

This iteration was equipped with a mecanum drive train and a sort of arm-ish thing that was going to have a bucket attached to a chain to make the bucket go up and down the arm.

Lessons Learned

The issue with the whole things is that the chain did not move very well and the part where the path would bend would cause the chain to change length thus making the design unusable.

Iteration 1



Goals

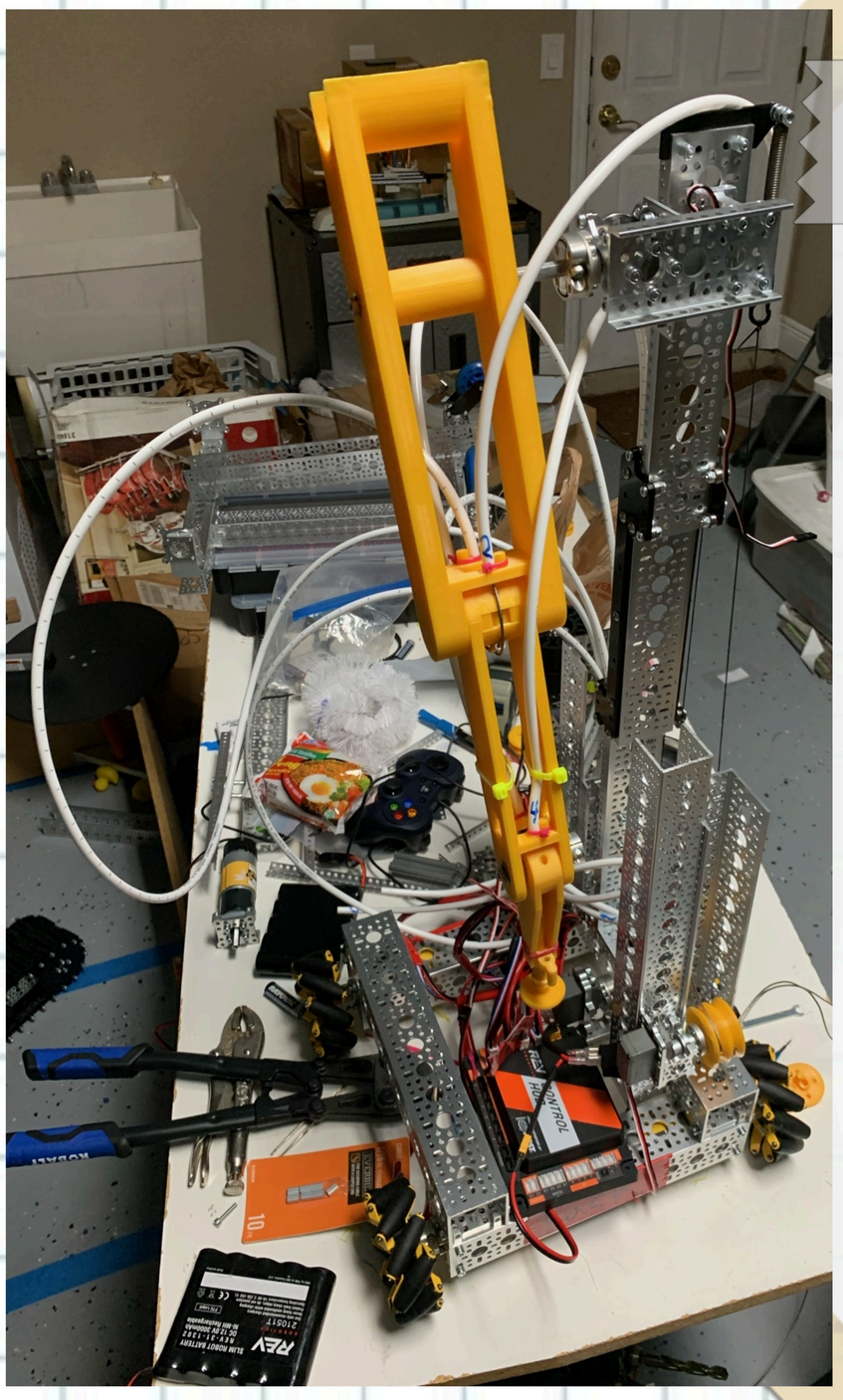
Our goal with iteration 1 was to make an arm that could pick up the blocks and place them with accuracy.

Results

Iteration one is where we began making a actual arm with a grabber at the end of said arm. Iteration one still had the mecanum drive train and it used the joint from the previous iteration in order to make the elbow joint. The grabber that was at the end of the arm used two servos that would rotate sticks with gecko wheels attached to the end in order to act like a pincer.

Lessons Learned

Now the issue with this iteration was that the servos we had mounted were too weak to hold up the arm and that the arm itself was very heavy so we decided to make the next iterations lighter.



Iteration 2

Goals

Our goal with iteration two was to make the arm lighter.

Results

Iteration two was the very first 3D printed iteration, the grabber was printed separately and then attached with some bearings and an axle and the joints were to be driven by cables attached to pulleys so that we can keep the weight of the motors out of the arm. This iteration is also the first robot we actually used in competition.

The grabber was originally going to be a "Christmas tree Branch" tied in a loop however this left debris so we switched to using a spring which did not leave debris. We were also planning on placing a counter weight on the other side of the shoulder joint. The cable that was used to drive the joints were ran through hoses which acted similarly to the brakes on a bicycle but instead of a lever on the other end there was going to be a pulley attached to a servo. We also mounted the shoulder of the arm on a linear extender so the shoulder motor did not need to turn very far in the upward direction in order to deposit cargo.

Lessons Learned

With all of these big changes came far more issues. An issue with the arm was the shoulder was not strong enough, the hoses were not gathered well and kept getting caught on things, the wire was not cable that moved the joints were not tight, the cable on the elevator kept on snaping, and the piston also snaped a few times.

Iteration 3

1895

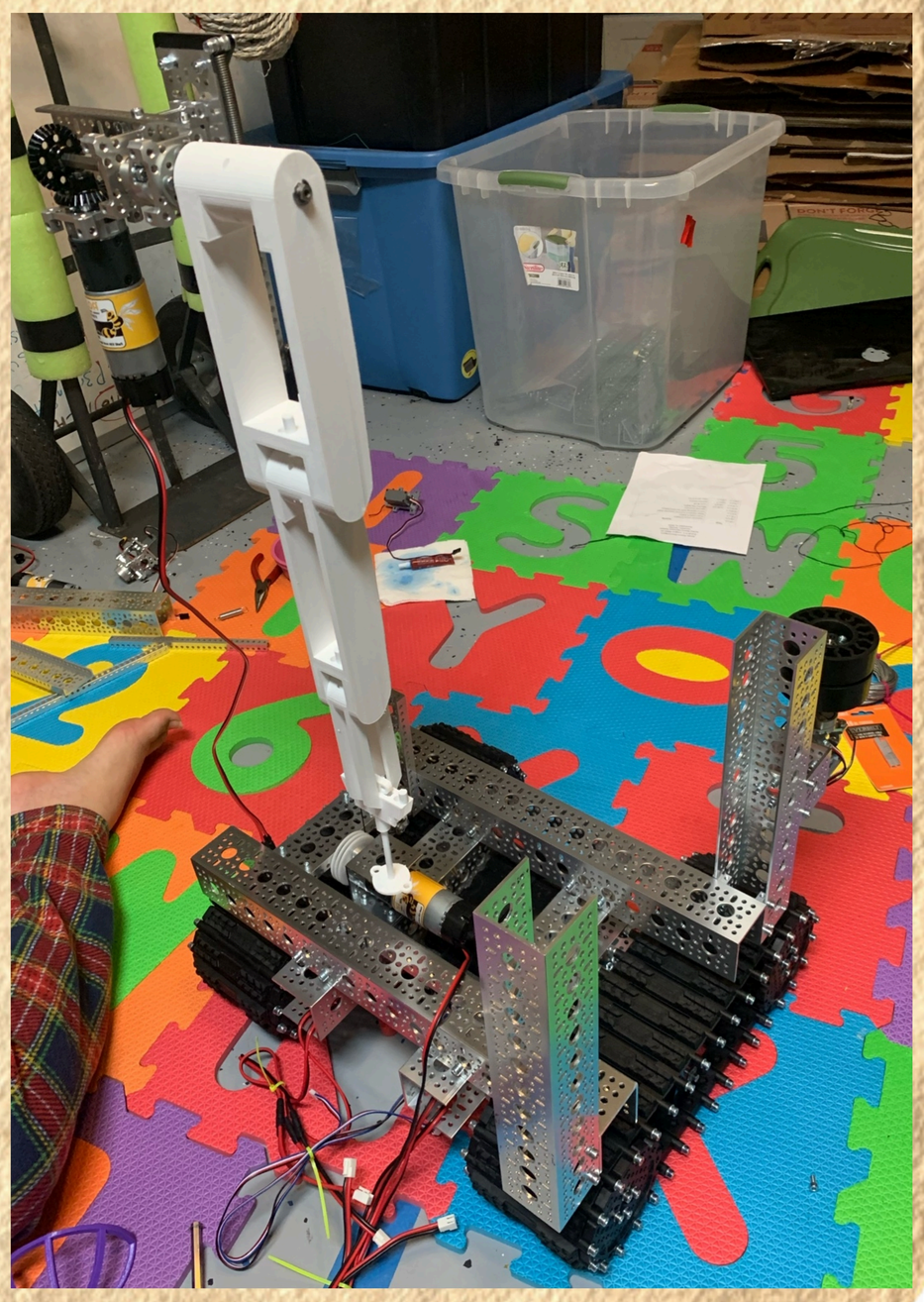
Goals

The goals with iteration three were to do the hose pathing better so that the arm can actually function. The other thing we were going to try was seeing if other drivetrains worked better than mecanum.

Results

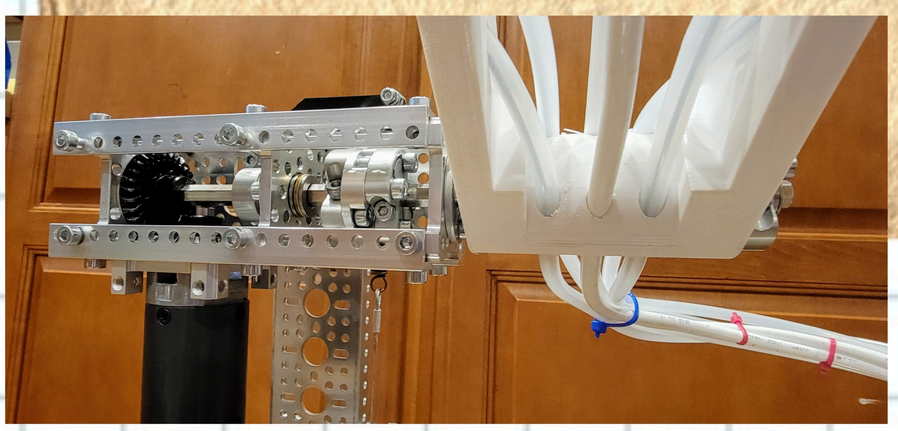
Iteration three used the same concept as two except it fixed some of the issues. For example the hose pathing was changed so all of the hoses were bundled together and could not get caught on things, the grabber was redesigned so it was more durable, and we switched to using Go Bilda tracks instead of mecanum wheels. This iteration was also used at a meet.

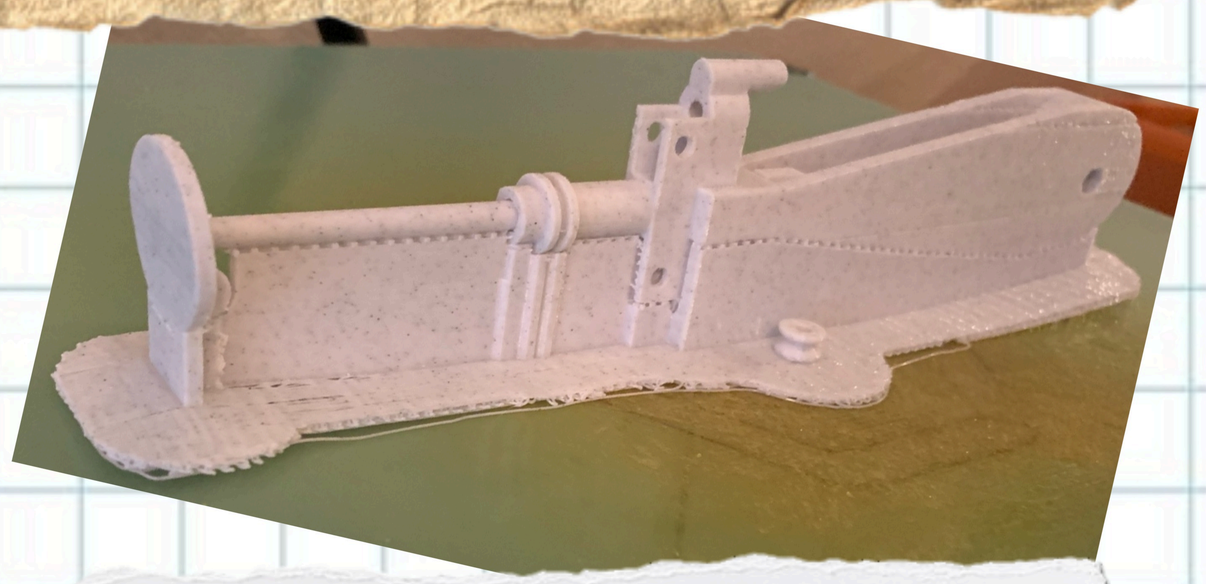
When we tested the tracks against the mecanum wheels we found the the advantage of not having to worry about getting stuck and going over the barrier faster was worth loosing the maneuverability of the mecanum wheels so we switched to tracks. We also placed small pulleys inside of the grabber so the cable could attach to all necessary parts of the piston and not snap the grabber. Another thing we did for this iteration was change the shoulder to a DC motor because a servo was too weak to lift the arm.



Lessons Learned

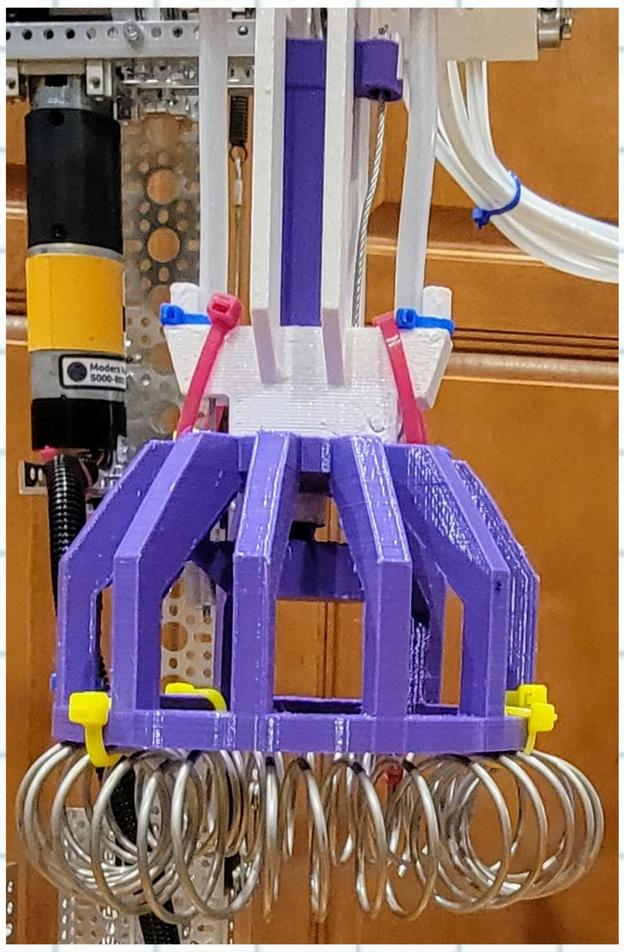
An issue with the piston that pushed the blocks out of the grabber was that it was very thin so it could easily be snapped if moved in the wrong way. However the main primary issues with this iteration was that the cable was still too loose on the joint and we could not properly secure the cable in place and the hoses that held the cables were preventing the joints from bending. Because of all this, the joints could not move but we were getting close to having a working machine.





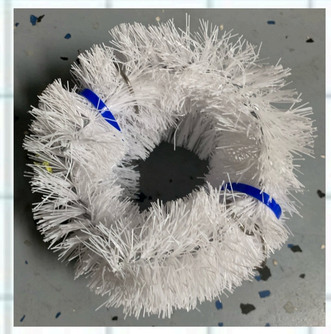
Shoulder Gear Box

We changed the shoulder from a servo motor to a DC motor because a servo was too weak to lift the arm.

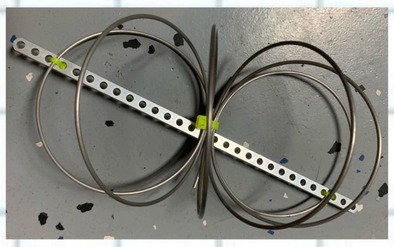


Releaser Piston Iteration 2

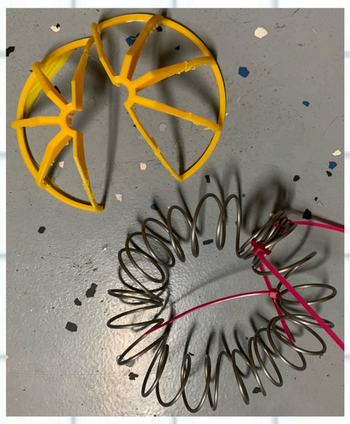
The releaser piston was very thin and easily snapped.



"Christmas tree branch"
Shed a lot of debris



Large spring loop
Unable to make a simple release mechanism



Smaller spring loop
Able to create piston which could remove blocks from loop.

Grabber Assessments

Test

1895

Mechanum Wheels

Tracks

Mode	Driver	Drive Train	Time (mins)	Path	Times Completed	Stuck	Critically Stuck	Notes
Teleoperated	Randy Huegele	Mecanum	2:00	Crossing Barrier	16	3	1	
Teleoperated	Randy Huegele	Mecanum	1:30	Crossing Barrier	24	0	1	Why is the time shorter?
Teleoperated	Randy Huegele	Mecanum	2:00	Crossing Barrier	11	0	1	Got stuck at the 1:30 mark
Teleoperated	Randy Huegele	Mecanum	2:00	Crossing Barrier	23	1	0	
Teleoperated	Valus Valoo	Mecanum	2:00	Crossing Barrier	28	0	0	I am a better driver
Teleoperated	Valus Valoo	Mecanum	2:00	Crossing Barrier	2	0	1	Nothing to see here
Teleoperated	Valus Valoo	Mecanum	2:00	Crossing Barrier	22	0	0	
Teleoperated	Valus Valoo	Mecanum	2:00	Crossing Barrier	35	0	0	Trying to go slower
Teleoperated	Valus Valoo	Mecanum	2:00	Crossing Barrier	36	1	0	still going slower
Teleoperated	Valus Valoo	Mecanum	2:00	Around Barrier	41	0	0	
Teleoperated	Valus Valoo	Mecanum	2:00	Around Barrier	29	0	0	battery died
Teleoperated	Valus Valoo	Mecanum	2:00	Around Barrier	38	0	0	Fresh battery made the robot go faster thus making it harder to drive.
Teleoperated	Valus Valoo	Mecanum	2:00	Around Barrier	42	0	0	
Teleoperated	Valus Valoo	tracks	2:00	Crossing Barrier	43	1	0	YYYEEEEEEHHHHHHHHH
Teleoperated	Valus Valoo	tracks	2:00	Crossing Barrier	41	0	0	tried being more careful
Teleoperated	Valus Valoo	tracks	2:00	Crossing Barrier	43	1	0	
Teleoperated	Valus Valoo	tracks	2:00	Crossing Barrier	42	0	0	

Observation: The GoBilda Revised strafer chassis has higher clearance and a narrower wheelbase than their basic model.

Result

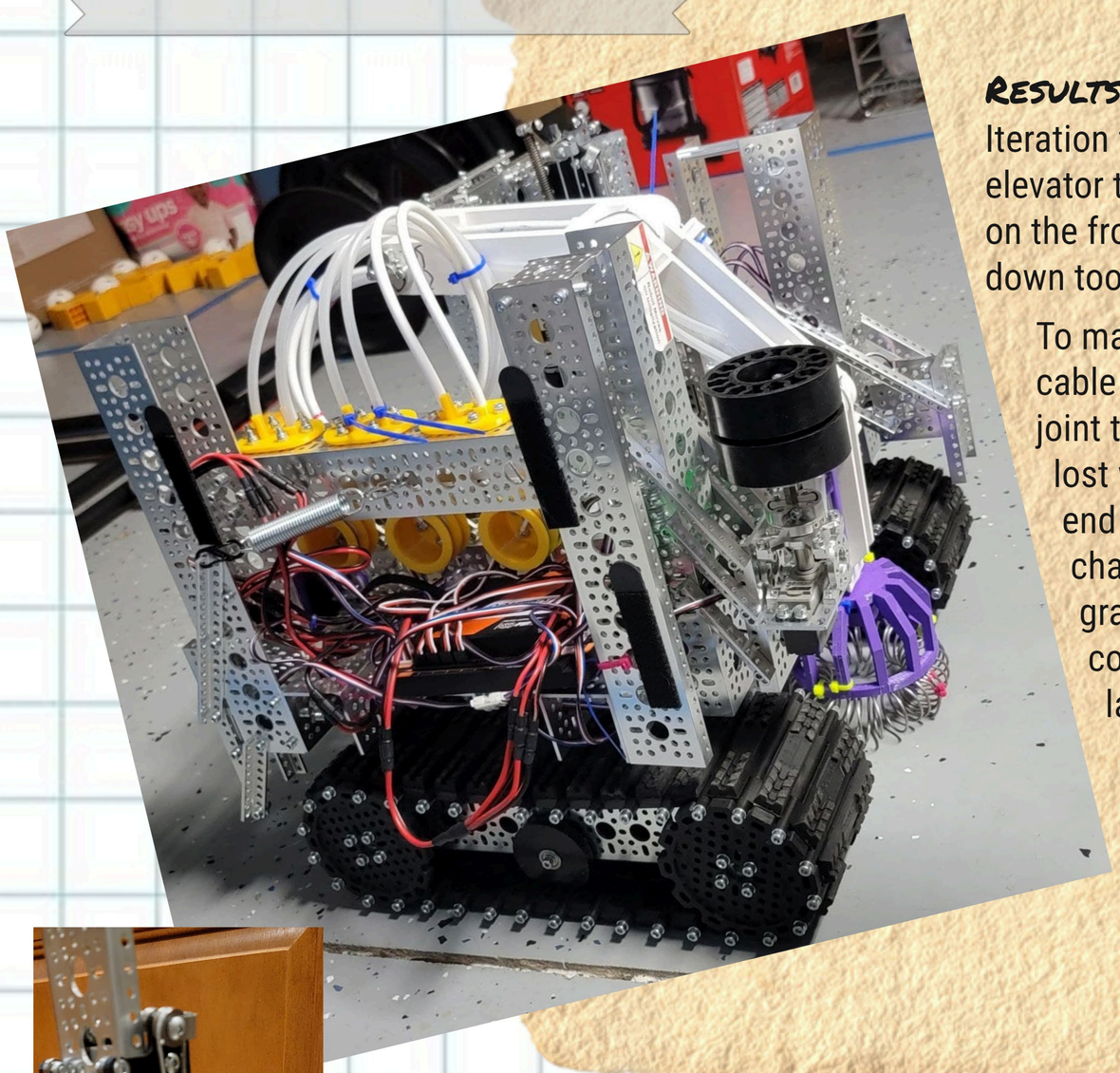
Drive Train	Time (Mins)	Average Number of Crossings	Number of Crossing Trials	Average Number of Going Around	Number of Around Barrier Trials	Average Times Got Stuck Crossing	Average Times Got Critically Stuck Crossing	Notes
Mecanum	2:00	24.6	5	37.5	4	0.2	0.2	This drive train had the weight of the robot onboard.
tracks	2:00	42.25	4	Untested also impossible	0	0.5	0	did not have any weight on the robot.

Conclusion

Tracks seem to be the better option because they cross the barrier at almost twice the rate of mecanum wheels. However improvements can be made to both drive trains.

Iteration 4

1995



Goals

The plan here was to make the grabber more robust and remove the spring force the hoses were putting on the joints, and to add some basic sensors.

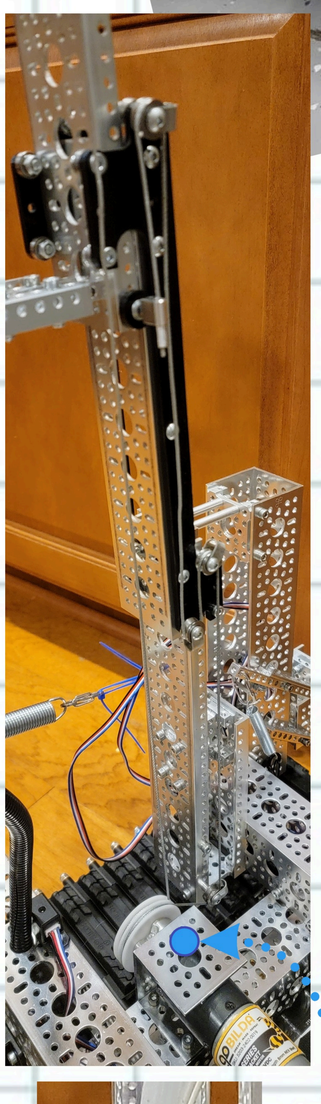
Results

Iteration four was the first iteration to actually pick up blocks with a functioning arm and an elevator that could not have the wire snap like the previous iterations. It also had bumpers on the front and back as well as a magnetic limiter switch to stop the elevator from going down too far.

To make the joints actually function we put holes that could hold the crimps for the cable so they were somewhat out of the way and we switched from using one cable per joint to using two in order to make crimping easier and to reduce the amount of cable lost when a crimp failed. Another thing we changed was making it so the hoses would end at the joints so the joints did not have to bend the hoses. The grabber was changed so it had a much larger piston that would be much harder to break and the grabber dome which connected the arm to the spring was made much thicker so it could take more force before it becomes unusable. Also the shoulder got a much larger DC motor to lift the arm with.

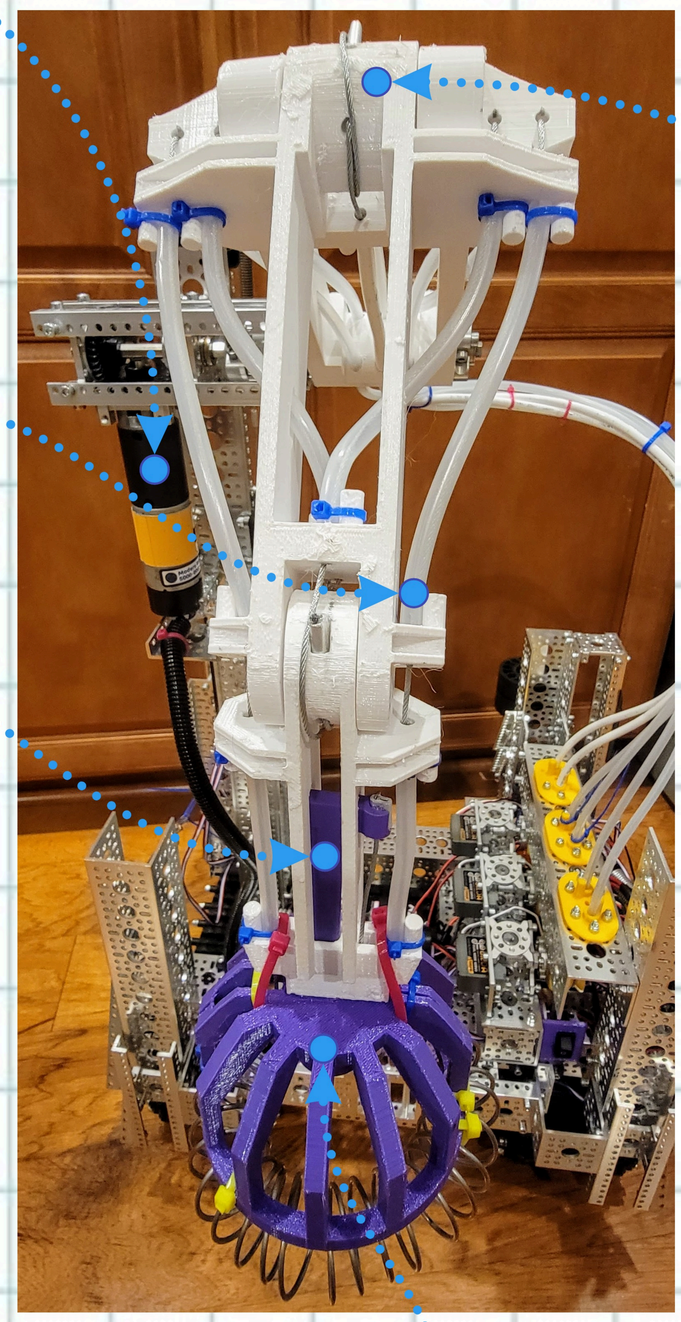
Lessons Learned

The issues with the iteration were that the crimps were not quite out of the way enough and they were still somewhat difficult to attach, this caused the cables to still be somewhat loose however the arm could still move. The bumpers were not entirely reliable and were still not included in the code.



Bigger DC Motor

During iteration 3 we used a 19.2:1 gear ratio however it was not quite strong enough for our purposes so we switched it out for a 188:1 ratio.

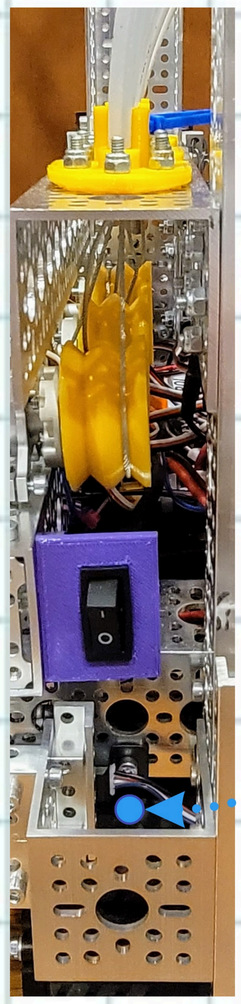


Improved Hose Routing

The hoses are cut off at the joints to allow looser movement.

Stronger Piston

The piston was made wider to prevent it from snapping.



Elevator Cable

The elevator originally came with a string but that string kept getting loose so we swapped it for steel cable and crimps like what we use for the arm.

Bumper Touch Sensor

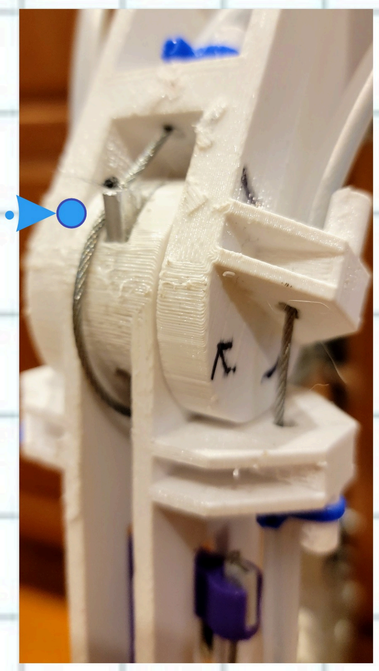
Bumpers were added so the robot can tell when it has made contact with something

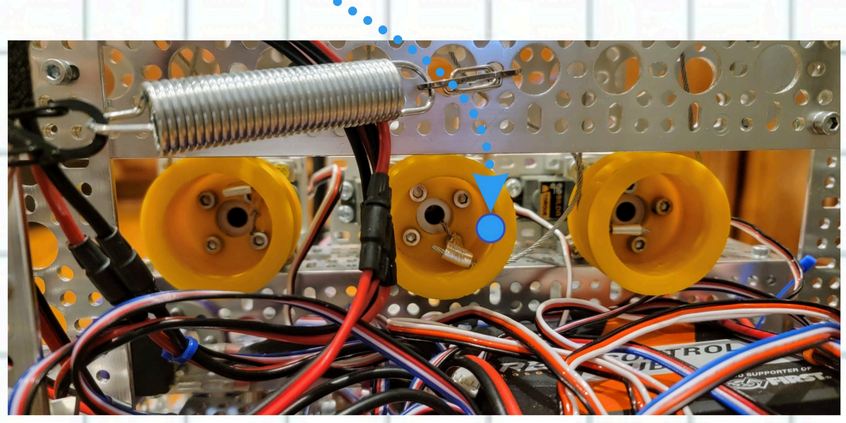
Thicker Grabber

The thickness of the grabber was a bit thin so we made it thicker.

Crimping Holes

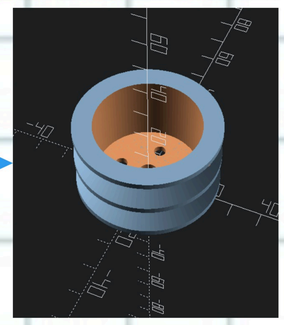
The crimping holes were designed to prevent the cable from getting caught on the crimps however they are not quite large enough and will be changed with iteration 5.





Pulley Cad Design

The team designed the 3D-printed pulley in an open-source CAD software called Open SCAD



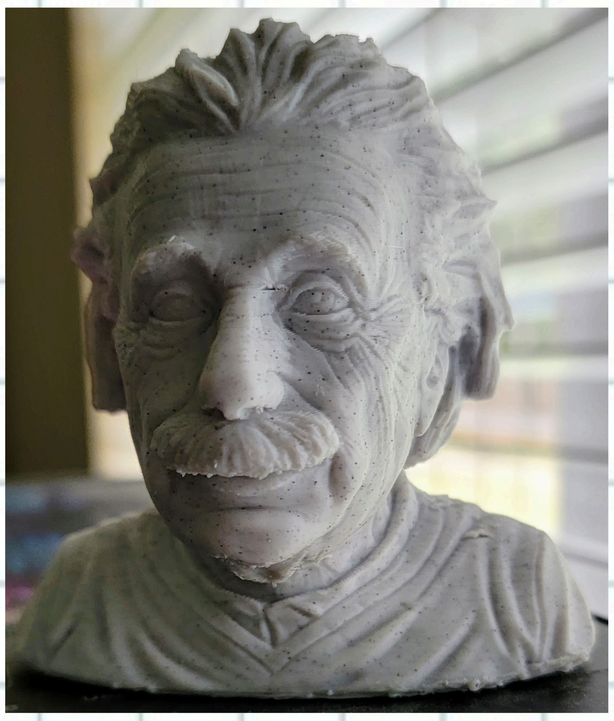
Physics

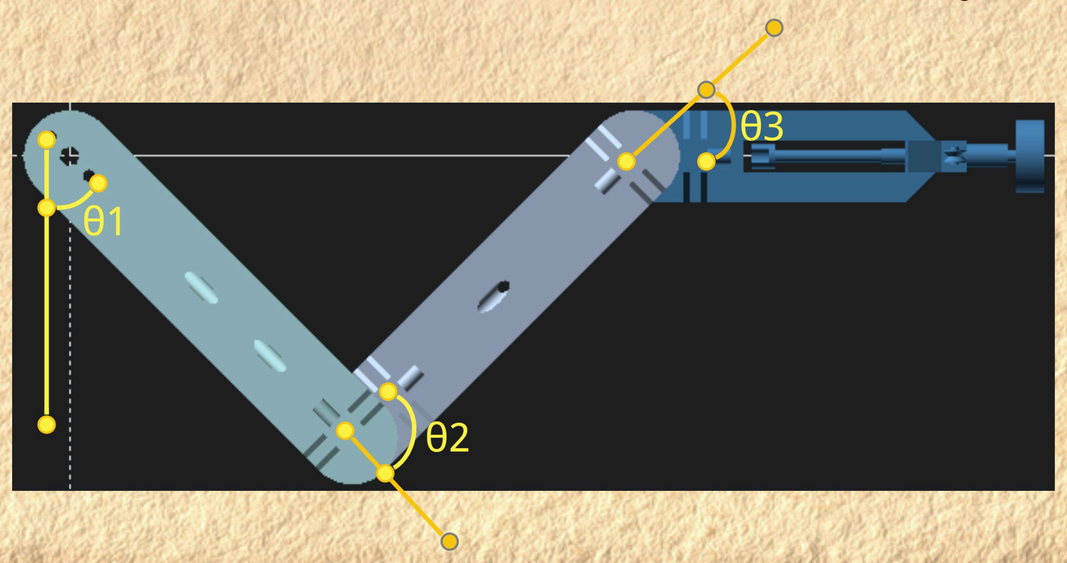
1895

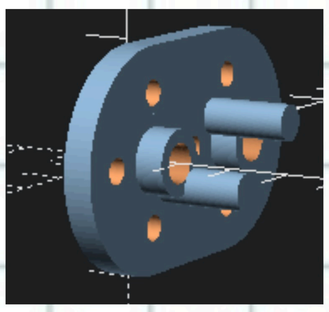
When we design an arm we would calculate if the motors that drive the arm can actually lift the load using the calculation of distance from joint to the center of mass multiplied by the weight and the sin of the angle. If you input the weight in ounces and the distance in inches the equation would output ounce-inches.

$$\tau = r * w * \sin \theta$$

Where:
τ = Torque
r = Distance from axis to the center of mass
w = Weight of segment
θ = Degree of bend







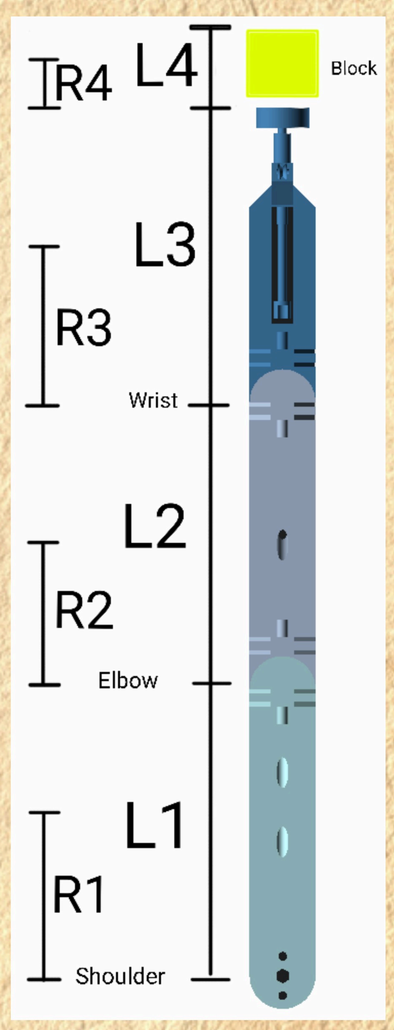
Hose Attachment
Hose attachment design in SCAD using parametric scripting

However since we require the torque when it is at its highest we can just enter theta as 90° which would make the sin 1 and that means that the equation is distance to the center of mass multiplied by the weight. We are also assuming that the center of mass is at the center of each segment so r is half the length of the segment.

$$\theta = 90°$$

$$r = L / 2$$

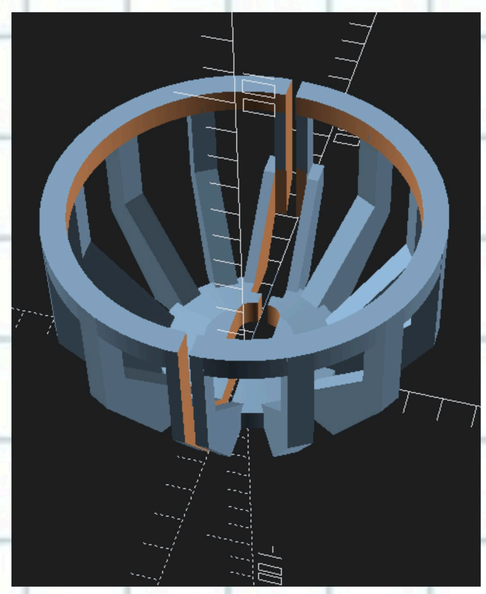
Where:
θ = Degree of bend
r = Distance from axis to the center of mass
L = Length of segment



The equation we use to see the torque requirement for the shoulder of iteration five is shown in the following function in Open SCAD.

```
function j1Torque(l5, w5, l4, w4, l3, w3, l2, w2, l1, w1) = l1
    /2*w1 + (l1+l2/2)*w2 + (l2+l3/2)*w3 + (l2+l3+l4/2)*w4 + (
    l1+l2+l3+l4+l5/2)*w5 + (l1+l2+l3+l4+l5+grabberLength/2)*
    grabberWeight + (l1+l2+l3+l4+l5+grabberLength+loadLength/2
    )*loadWeight;
```

Grabber
Grabber design in SCAD using parametric scripting



Robot CAD Design

1895

We design all of our 3D printed components in OpenSCAD and the reason why we use SCAD instead of some other cad like Fusion or whatever is that a lot of them cost money and they don't seem to have the same kind of malleability as SCAD with its functions and modules.

```
include <UtilityKit.scad>;

$fn = 100;

build();

module build() {
    shoulderAngle = 0;
    elbowAngle = 0;
    grabberAngle = 0;

    union() {
    translate([0, 0, -segmentZ/2]) upperSegment();
        translate([upSegmentX-segmentZ, 0, 0]) rotate([0, elbowAngle, 0]) translate([0, 0, -
            segmentZ/2])union() {
            lowerSegment();
            translate([lowSegmentX-segmentZ, 0, segmentZ/2]) rotate([0, grabberAngle, 0])
                translate([segmentZ/2, 0, 0]) union() {
                wrist();
                translate([hoseOuterRadius*2, 0, 0]) wristCableOffload();
            }
        }
    }
}

module upperSegment() {
    hoseHoleRadius = hoseOuterRadius+lineThickness/2;
    cubeX = upSegmentX-((segmentZ+hoseOuterRadius*2+hoseHolderDepth)+(segmentZ+hoseOuterRadius*2));
    holePos = segmentZ/2+hoseOuterRadius*2+cubeX/3*2+wallThickness;

    difference() {
        union() {
            translate([segmentZ/2+hoseOuterRadius*2, -upSegmentY/2, 0]) cube([cubeX, wallThickness
                , segmentZ], false);
            translate([segmentZ/2+hoseOuterRadius*2, upSegmentY/2-wallThickness, 0]) cube([cubeX,
                wallThickness, segmentZ], false);
        }
        //+y
        translate([holePos, (upSegmentY/2-wallThickness/2), segmentZ/2]) rotate([0, 90, 22.5])
            cylinder(wallThickness*5, hoseHoleRadius, hoseHoleRadius, true); // hose hole 1 +y
        translate([holePos-wallThickness*4, (upSegmentY/2-wallThickness/2), segmentZ/2]) rotate([0
            , 90, 22.5]) cylinder(wallThickness*5, hoseHoleRadius, hoseHoleRadius, true); // hose
            hole 2 +y

        //-y
        translate([holePos, -(upSegmentY/2-wallThickness/2), segmentZ/2]) rotate([0, 90, -22.5])
            cylinder(wallThickness*5, hoseHoleRadius, hoseHoleRadius, true); // hose hole 1 -y
        translate([holePos-wallThickness*4, -(upSegmentY/2-wallThickness/2), segmentZ/2]) rotate([
            0, 90, -22.5]) cylinder(wallThickness*5, hoseHoleRadius, hoseHoleRadius, true); //
            hose hole 2 -y
    }

    translate([0, 0, segmentZ/2]) upperAxleJoint(upSegmentY, segmentZ); //upper joint

    //positive y hose holders
    translate([segmentZ/2+hoseOuterRadius*2+cubeX-hoseHolderDepth/2+hoseHolderDepth+
        hoseOuterRadius*2+segmentZ/2, upSegmentY/2+boxY/2, segmentZ/2]) rotate([90, 180, 0])
        holderBox(); // hoseHolder 1 +y
    translate([segmentZ/2+hoseOuterRadius*2+cubeX-hoseHolderDepth/2+hoseHolderDepth+
```

Console

```
ECHO: wristTorque = 23.9579, "oz-in"
ECHO: "linearLiftWeight=", 66.3004, "oz"
ECHO: "TESTING COMPLETE"
ECHO: "********************************"
Compiling design (CSG Products generation)...
Geometries in cache: 180
Geometry cache size in bytes: 2347632
CGAL Polyhedrons in cache: 26
CGAL cache size in bytes: 7276832
Compiling design (CSG Products normalization)...
Normalized tree has 202 elements!
Compile and preview finished.
Total rendering time: 0:00:00.113
```

Error-Log

Show: All

Group	File	Line	Info

Activate Windows
Go to Settings to activate Windows.

Viewport: translate = [214.03 6.53 -1.37], rotate = [54.30 0.00 60.00], distance = 933.74, fov = 22.50 (1056x699)

OpenSCAD 2021.01

Modules

Modules are where a set of shapes can call or be called by other modules.

Parameters

Parameters are constants that are used throughout the code.

Functions

Functions take an input of a number and input said number into an equation and return the output.

ArmVer3.scad | ArmVer3Num.scad | UtilityKit.scad | GraveYard.scad | ArmVer2Num.scad | ArmVer2.scad | ArmVer1.scad

Editor

```
$fn = 100;

// variables for translating numbers
function inchesToMillimeters(inches) = inches * 25.4; // translates inches to millimeters
function millimetersToInches(millimeters) = millimeters/25.4; // translates millimeters to inches
function gramsToOunces(grams) = grams * 0.035274; // translates grams to ounces
function channelLength(holes) = holes*3*gridSize+gridSize*3;

module ring(thickness = 2, innerRadius = 0, outerRadius = 1, center = true) { // makes a ring with
    the given variables
    difference() {
        cylinder(thickness, outerRadius, outerRadius, center); // outer cylinder
        translate([0, 0, -1]) cylinder(thickness+5, innerRadius, innerRadius, center); // inner
            cylinder
    }
}

// standard Go Bilda U Channel numbers (mm)
uChannelWidth = 48; // width of u channel (mm)
gridSize = 8; // size of grid used for hole placement (mm)
plateThickness = 2.5; // thickness of u channel walls (mm)
centerHoleRadius = 7; // radius of the hole located at the center of a uchannel (mm)
smallHoleRadius = 2; // radius of small holes in u channel (mm)

// printing numbers
lineThickness = .4; // thickness of the lines made in the printer (mm)

module uChannel(holeCount){ // imports a uChannel with hole count from 1 - 18
    paths = [
    "C:/Users/colin/Documents/UChannelSTL/1120-0001-0048(1Hole).stl", //1 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0002-0072(2Hole).stl", //2 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0003-0096(3Hole).stl", //3 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0004-0120(4Hole).stl", //4 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0005-0144(5Hole).stl", //5 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0006-0168(6Hole).stl", //6 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0007-0192(7Hole).stl", //7 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0008-0216(8Hole).stl", //8 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0009-0240(9Hole).stl", //9 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0010-0264(10Hole).stl", //10 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0011-0288(11Hole).stl", //11 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0012-0312(12Hole).stl", //12 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0013-0336(13Hole).stl", //13 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0014-0360(14Hole).stl", //14 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0015-0384(15Hole).stl", //15 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0016-0408(16Hole).stl", //16 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0017-0432(17Hole).stl", //17 hole
    "C:/Users/colin/Documents/UChannelSTL/1120-0018-0456(18Hole).stl"  //18 hole
    ];
    import(paths[holeCount-1]);
}

module uChannelCutter(thickness = 10) { // cuts a holes that fit around the center hole in a go
    bilda uchannel
    translate([0, 0, 0]) rotate([0, 0, 45]) union() {
        translate([-gridSize, gridSize, 0]) cylinder(thickness, smallHoleRadius*1.25,
            smallHoleRadius*1.25, false);
        translate([-gridSize, -gridSize, 0]) cylinder(thickness, smallHoleRadius*1.25,
            smallHoleRadius*1.25, false);
        translate([gridSize, gridSize, 0]) cylinder(thickness, smallHoleRadius*1.25,
            smallHoleRadius*1.25, false);
        translate([gridSize, -gridSize, 0]) cylinder(thickness, smallHoleRadius*1.25,
            smallHoleRadius*1.25, false);
```

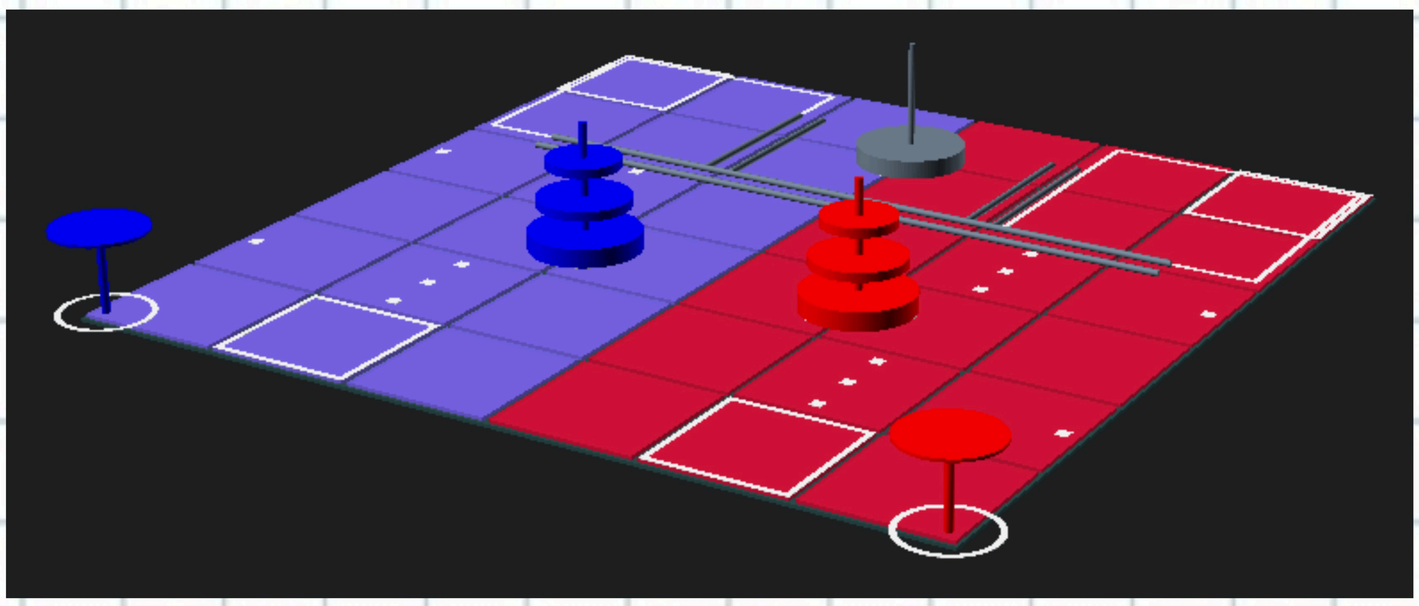
```
include <UtilityKit.scad>;

runTests();

//numbers for magnetic sensor
sensorY = 23; //(mm)
sensorYAlt = 40.5; //(mm)
sensorX = 17 - 3; //(mm)
sensorZ = 15; //(mm)
sensorScrewDistance = 35.27; //(mm)
sensorScrewRadius = 3.5; //(mm)

hoseOuterRadius = inchesToMillimeters(.25/2); // (in)
hoseInnerRadius = inchesToMillimeters(.169/2); // (in)

// CONSTANTS
g = 32*12; // gravitational acceleration (in/s^2)=(32 ft/s^2)(12 in/ft)
blockWeight = 113; //nominal weight of block with three weights, verified by measurment (g)
ballWeight = 30.1; //nominal weight of ball, verified by measurement (g)

// PHYSICAL TEST PARAMETERS FOR THE ROBOT
vmax = 16; // the vertical space under the raised tower
vmin = 11; // vertical distance (height) of the shoulder axis from the wrist joint when reaching
    forward to the floor level (in)
hmin = 11; // horizontal distance from the shoulder axis to the wrist joint when reaching to the
    floor level (in)
lowLevelHeight = 3.0; // +/-1.5 for tipping
midLevelHeight = 8.5; // +/-1.5 for tipping
topLevelHeight = 14.7; // +/-1.5 for tipping
capLevelHeight = 28.3; // 8 comes from the height of the shipping element

// TOWER
wtop = gramsToOunces(516); // weight of the tower top (oz) (whole extender)
wbottom = 1000; // weight of the tower bottom (oz)
rPulley = millimetersToInches(19); // (in)

// DRIVETRAIN
rWh = 2; // the radius of the drivetrain wheel (in)
wDT = 500; // the weight of the drivetrain (oz)
nDrivers = 4; // the number of drivetrain motors
mechanum = true; // indicates if mechanum wheels are being used
motorToDriveGearRatio = 1; // the motor to drive gear ratio

// ARM
w0 = 32; // counter weight on arm (oz)
w1 = gramsToOunces(140); // weight of the upper arm (oz)
w2 = gramsToOunces(90); // weight of the lower arm (oz)
w3 = 4; // weight of the gripper (oz)
w4 = gramsToOunces(blockWeight); // weight of the heaviest block or ball (oz)
l0 = 4; // length of counter weight (in)
l1 = 8; // length of upper arm (in)
l2 = 8; // length of lower arm (in)
l3 = 6; // length of gripper (in)
r0 = 4; // distance from shoulder axis to counter weight center of mass (in)
r1 = l1/2; // tapers, distance from shoulder axis to upper arm center of mass (in)
r2 = l2/2; // tapers, distance from elbow axis to lower arm center of mass (in)
r3 = l3/2; // distance from wrist axis to center of gripper mass (in)
r4 = 3; // distance from wrist axis to center of block or ball (in)

// MOTOR SPECS
tMotor = 338; // torque of the motor (oz in)
sMotor = 312; // motor speed (rpm)
omegaMotor = rpmToRadPerSec(sMotor); // angular frequency (rad/sec)  of motor
```

Robot Control

1895

Sensors

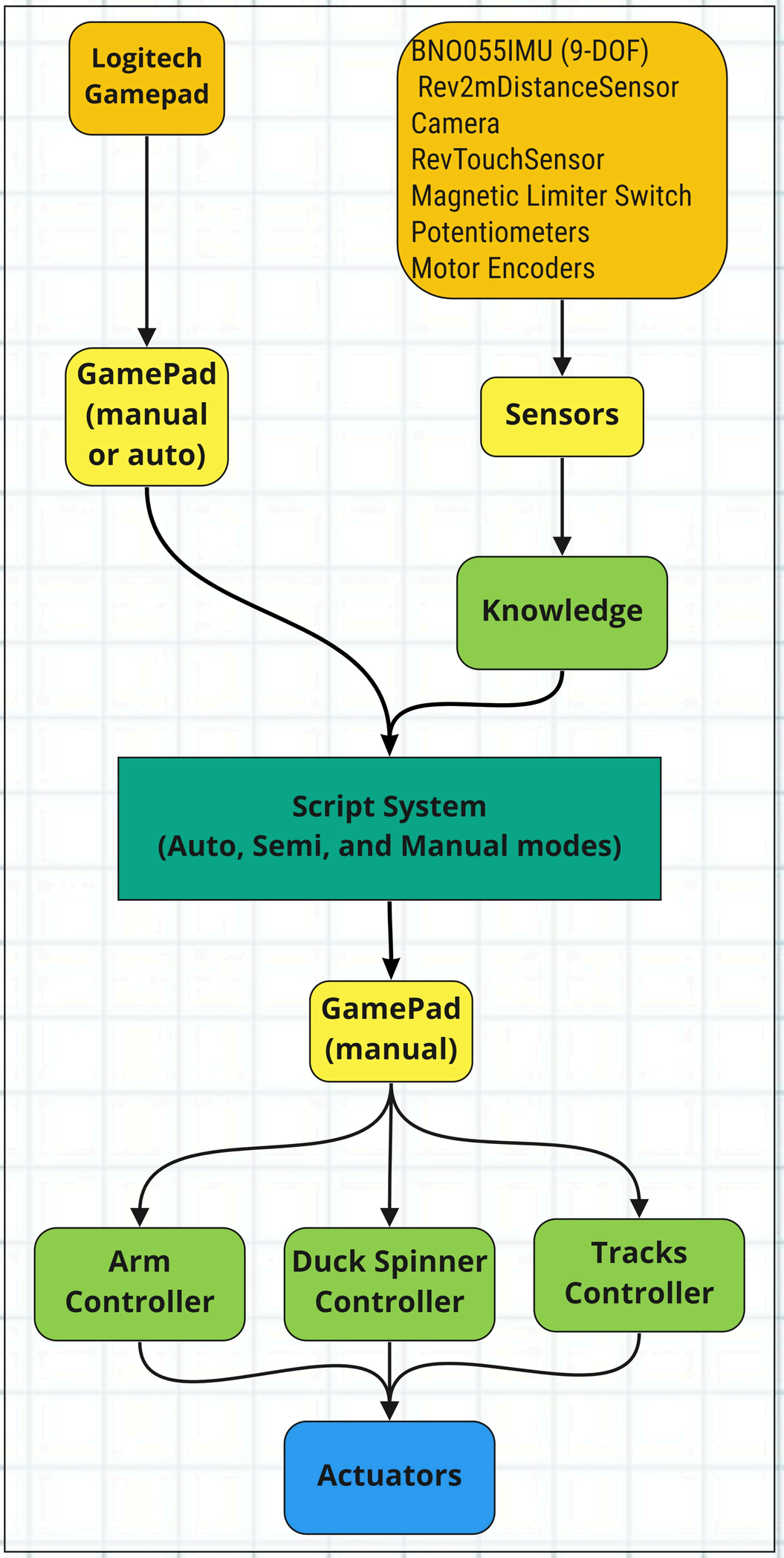
Sensor	Location	Uses + Benefits	Lessons Learned
Touch Sensors	Inside of the bumpers at the four corners of the robot.	The robot knows when it comes into contact with an obstruction and can prevent damage to said obstruction by stopping the robot.	We need a better mechanical apparatus to press and release the button.
9-DOF	Inside of Control Hub	By using the 9-DOF sensor the robot can know its orientation and position. Orientation helps the robot prevent tipping by acceleration control and helps know its facing. Position tells the robot where it is and where it needs to go.	May devolop slight errors over time needs regular correction from other sensors.
Magnetic Limiter Switch	Bottom of Elevator	The magnetic limiter switch prevents the elevator from being lowed beyond its area of movement by stopping the motor when the magnet mounted on the shoulder motor touches the sensor.	Can be unintentionally activated by magnets of other robots, however this a rare occurrence.
Motor Encoder	All motors	The motor encoders are used to tell the position of the tracks, arm, and elevator.	The encoder can be off by a degree or two at times. needs to be combined with 9-DOF with sensor fusion with a weighted average when used in drive train to determine position (still under development).
Distance Sensors	Front and back of robot	The distance sensors is for reading the bar code and seeing the distance of other objects. Distance to walls or other objects can help re-establish the robots current position and orientation.	
Potentiometers	Joints of arm	The potentiometers are for reading the arm position with greater accuracy.	
Camera	End of arm	The camera is a high resolution sensor. It can recognize objects and their positions on screen. Transforming from screen coordinates to world coordinates helps the robot move precisely toward identifiable game objects. The camera is there so the robot can see the blocks it needs to pick up and read the bar code.	



Robot Knowledge of the Field

Depiction of the Robot's knowledge of the playing field. This knowledge is used for navigation.

Data Flow from Sensors to Actuators



1995

Autonomous

The sensors assist the autonomous mode by:

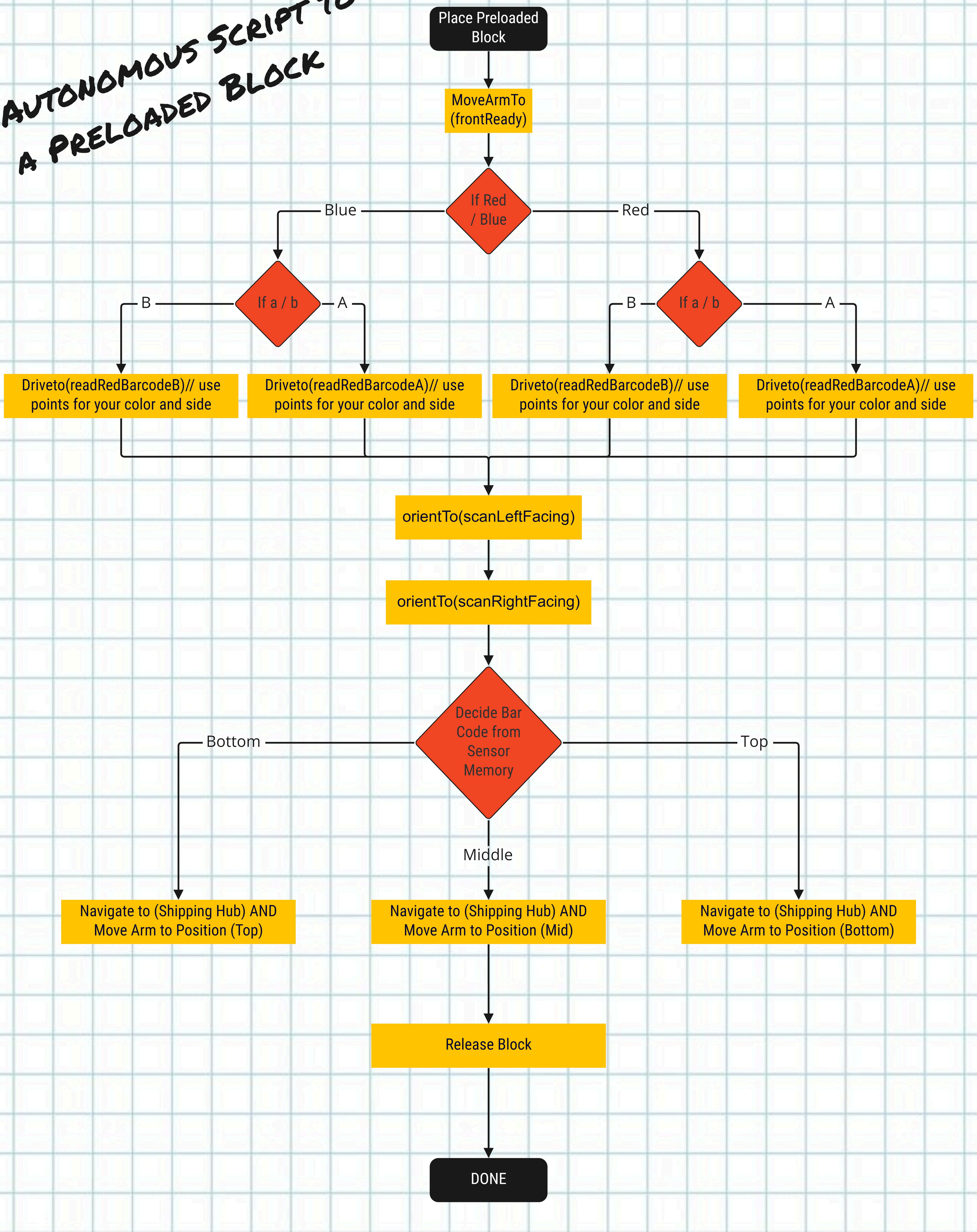
- reading the bar code
- preventing collisions and reducing accidents
- telling the robots current position
- controlling motor movements, like navigation and arm
- recognizing various game elements and updating their position in the map

Semi-Autonomous (Enhanced Tele Op)

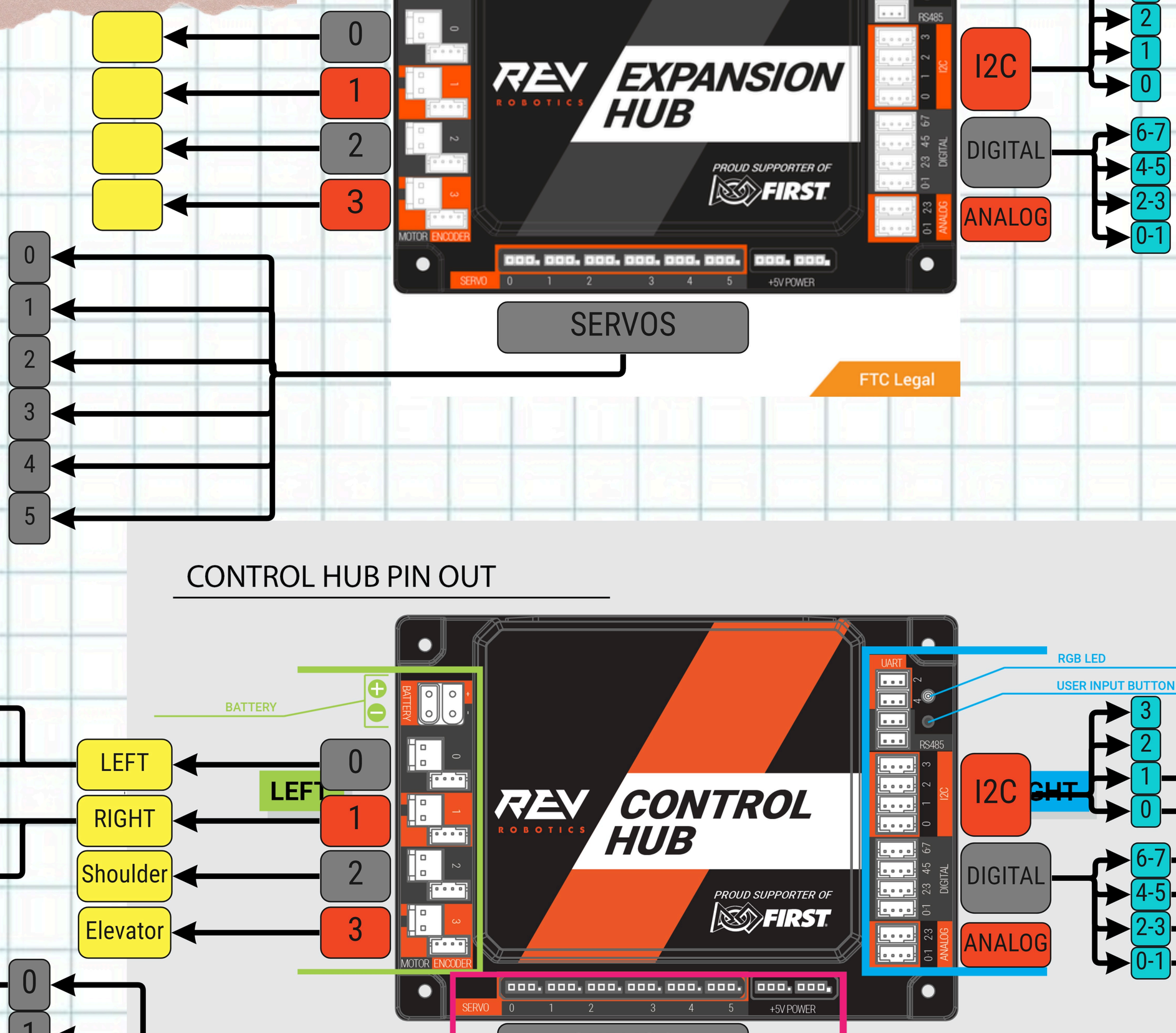
During Tele-Op the sensors can assist the driver by controlling the arm so the robot can pickup blocks with greater speed. the sensors can also prevent the robot from hitting a wall or other robots at great speed by reading the distance with the distance sensor then slowing down and eventually stopping. With knowledge of the course being updated constantly by sensors like the camera, distance sensor, motor encoder, and bumpers, the robot can quickly drive itself to all locations with the shortest possible path.

AUTONOMOUS SCRIPT TO PLACE A PRELOADED BLOCK

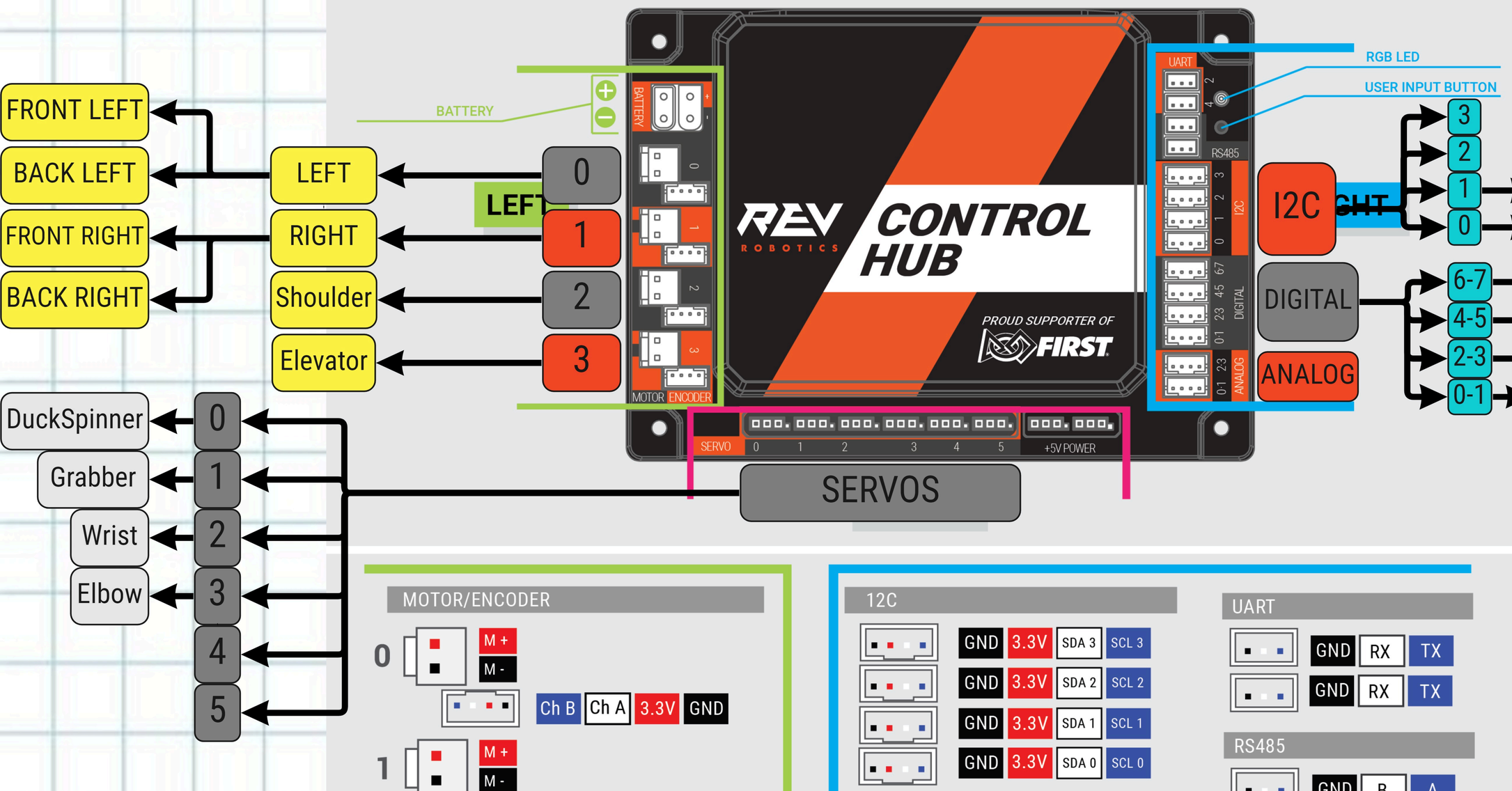
19895

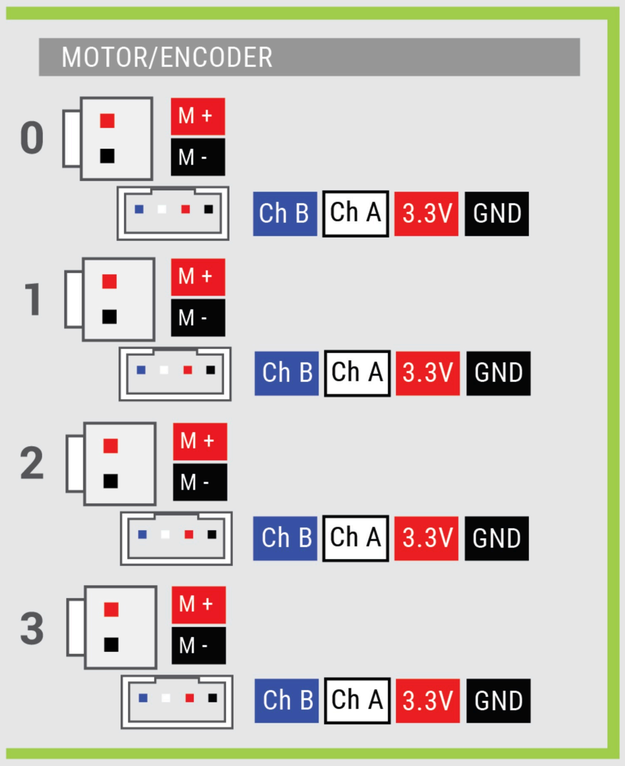


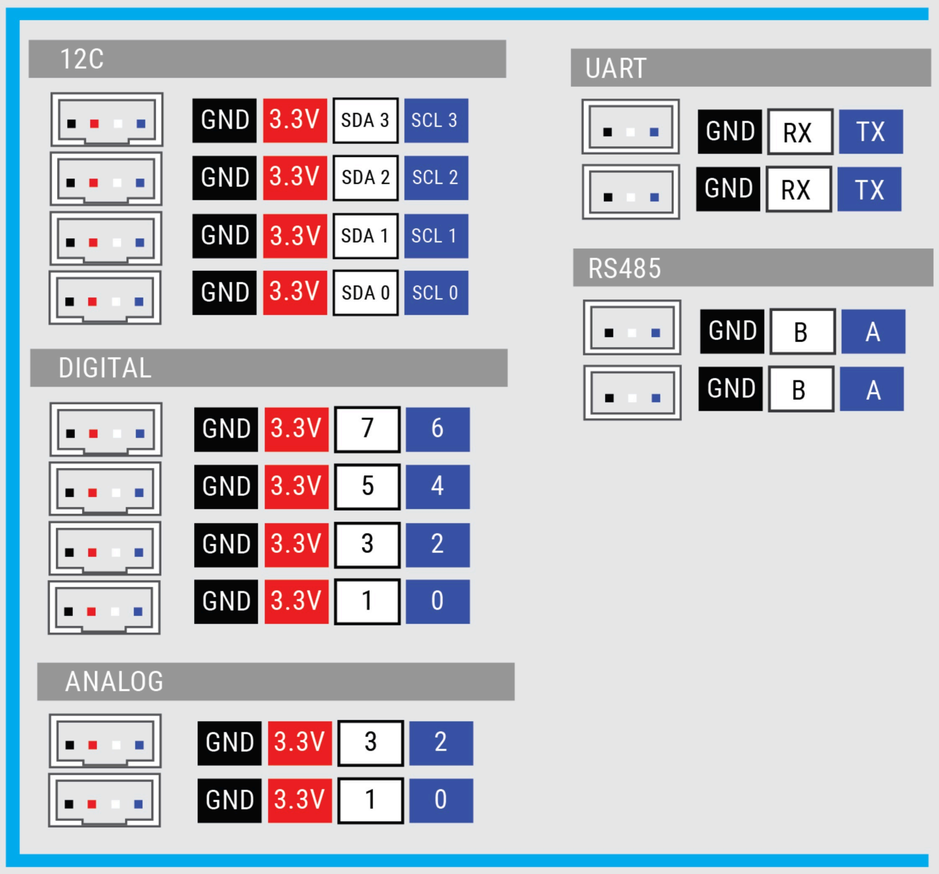
Wiring Diagram

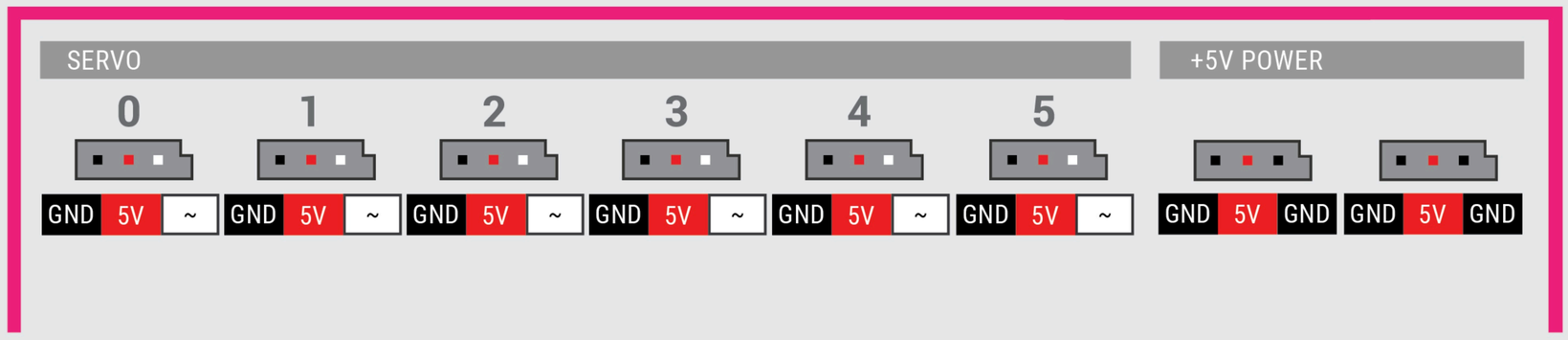


CONTROL HUB PIN OUT









Strategic Optimization of Action Planning (S.O.A.P.)

19895

A plan is composed of robot actions. A robot action is defined by a chance of failure, required state to take the action, and the state changes that result from success or failure. State changes include changes in position, time, freight, shipping hub balanced, alliance hub balanced, and points.

S.O.A.P. is where we try all possible actions on a game state to produce the highest scoring sequence of actions. By scrubbing with S.O.A.P. we can select the best possible action plan for the robot to take.

Robot Action	State Changes on Success	Required State	State Changes On Fail
Drive from Required State to New State	now at Warehouse	At Alliance Shipping Hub	Get stuck on Barrier
Drive from Required State to New State	now at Warehouse	At Alliance Shipping Hub	
Drive directly from Alliance Shipping Hub to Warehouse	now at Carousel	At Alliance Shipping Hub	
Drive around barrier from Warehouse hub to Alliance Shipping	now at Carousel	At Alliance Shipping Hub	
Drive directly from Warehouse to Alliance Shipping Hub	now at Alliance Shipping Hub	At Warehouse	Get stuck on Barrier
Drive around barrier from Warehouse hub to Alliance Shipping	now at Alliance Shipping Hub	At Warehouse	
Drive directly from Alliance Shipping Hub to Warehouse	now at Carousel	At Warehouse	
Drive around barrier from Warehouse hub to Alliance Shipping	now at Carousel	At Warehouse	
Drive directly from Alliance Shipping Hub to Warehouse	now at Warehouse	At Carousel	
Drive around barrier from Warehouse hub to Alliance Shipping	now at Warehouse	At Carousel	
Drive directly from Alliance Shipping Hub to Warehouse	now at Alliance Shipping Hub	At Carousel	
Drive around barrier from Warehouse hub to Alliance Shipping	now at Alliance Shipping Hub	At Carousel	
Grab Freight from Warehouse	now has Freight	at Warehouse and does not have Freight	drop Freight
Place Freight in Alliance Shipping Hub	No longer has Freight & gain Points	At Alliance Shipping Hub and has Freight	drop Freight
Rotate Carousel	carousel Rotated 360*	At Carousel	

Conclusion

With some experimentation we are getting closer to a better design..

What was the best part of this whole experience?

DUCK JUICE