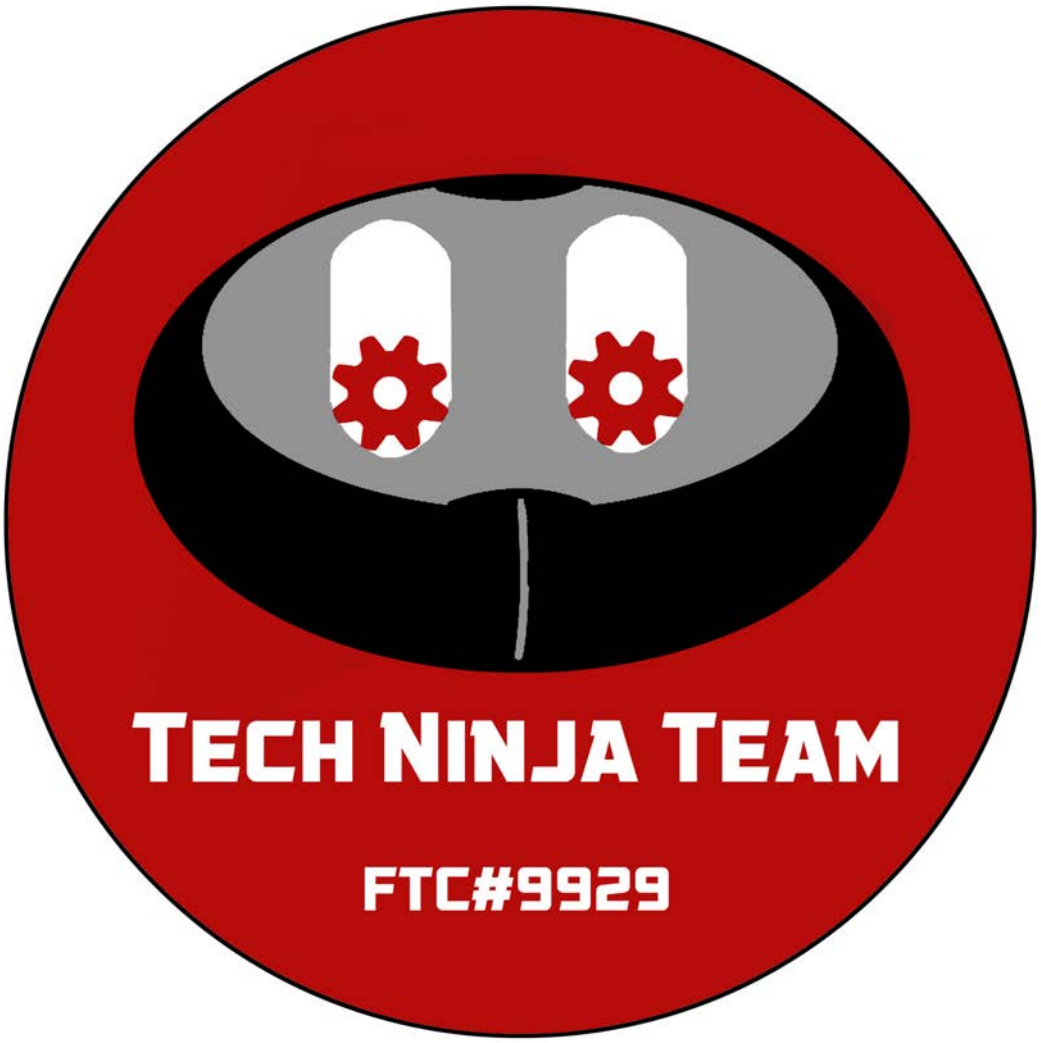


Engineering Notebook

2019-2020 Skystone

This page intentionally left blank

An Introduction to FTC Team #9929 "The Tech Ninja Team"

It all started in a basement with the Nelson family and the Matthews family. They were in the process of completing an overly complicated machine to, drum roll please, unzip a zipper. These two families had decided to complete the 2014 rube Goldberg machine challenge over a couple of weekends, for fun. After a few hours, one industrial shelf, and a ton of tape later, it finally worked. Then we went on hiatus for a few months before starting FLL. The team choose to "dominate" with education as LED (Lego Education Domination). The final robot didn't do so well but at least our name was cool.

Year two of our team's engineering adventure to becoming the Tech Ninja Team we are now brought some changes. A lot of changes. A few of the older team mates (Calvin, Kate, and Lauren) pushed the coaches to offer FTC after seeing the robots built for this competition online. They managed to convince the FLL coaches to start a FTC team. We unpackaged our first kit to build a robot in the basement of Coach Matthews. All of the team members were excited to begin building. Our team name came from a joke when we were unpacking the boxes of robot parts. We were all pretty excited about the cool parts that we unpacked and thought they were really cool, when someone said "Just wait until we unpack the ninjas". Thus the Tech Ninja Team was born, along with a cool acronym (T.N.T). after beginning work on the robot we moved to coach Nelson's garage for more space. The name for our robot, Skittlebot, came from an exercise we did to better understand programing. Shortly before the team's first qualifier we got a new space to practice in at the to be Homewood Science Center. Little did the team know at the time how lucky we were to have the space. We would end up 11th place with a Control Award when the season was over.

During our first off season we moved into our own room in the science center that is often referred to as the robot room. This space would allow us to expand more and start to learn more advanced techniques for building our robot. We are also started using Slack (a communication platform) and GitHub (a program storage platform) to better our team's communication. We researched prior seasons and worked on building mechanisms we noticed were used throughout the challenges. Later, we moved into the garage at the Science Center which gave us more space and the ability to add more power tools to use when building our robots. The team and our workshop have evolved over time to become more capable.

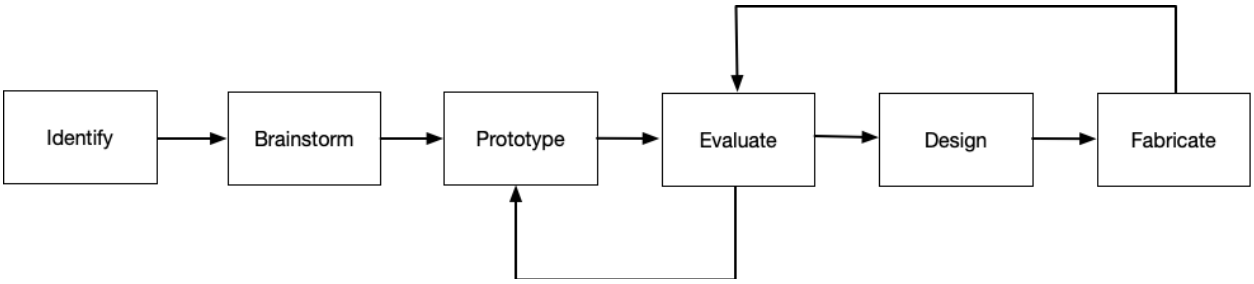
This season's robot "W2F-70" (in tribute to Woodie Flowers) has parts we designed in CAD using Fusion360 and manufactured using our CNC router and 3D printer. We continue to invest time into using and understanding motion profiles for smooth and accurate robot movement, added unit testing to our software to allow us to write software before mechanical work on the robot is complete and to ensure quality, and have started to collect and analyze real time metrics from the robot. The team has started to fill roles outside of build and programming to help the team as a whole get

better by creating processes, standards and checklists. We hope that changes we made to the team will help us move farther.

How We Work

Like many FTC teams, we have a build team, a software and controls team and drive teams. Team members choose which team they work on based on their interests and skills, but sometimes because of need or particular interest they may do work on parts of the robot that are not their "usual" team.

We follow this engineering design process:



- Identify – identify the problem to be solved
- Brainstorm – brainstorm solutions
- Prototype – quickly build physical or mathematical models to evaluate the brainstormed solutions
- Evaluate – run experiments to see if the solution works
- Design – design a solution to use on the robot based on the prototype(s) and the evaluation
- Fabricate – machine, assemble or program the solution that was designed. Evaluate whether it meets the requirement.

Our engineering notebook entries indicate which step(s) we are doing at the time.

Team-Bios

Hannah
5 years with FIRST
Junior

I enjoy reading, drawing, playing video games, and riding my bike in my spare time. I'm on the schools' Scholastic Bowl team and I'm glad to say that much of what I've learned at robotics has helped me there.

Most of the time, I help the build team, but this year I worked more with the programming team to gain a better understanding of how the virtual and physical elements of the robot work together.

I joined FIRST because I was interested in engineering and robotics, and wanted to learn more about STEM than I did in school. It's really helped me find new interests and discover what careers I might want to pursue in college.

Lauren
6 years with FIRST
Junior

My name is Lauren. I am 17 and attend HF High School. I have participated in FIRST as a whole for six years. For my first year I competed in FLL. After that season was over I, along with a few other team members, crossed over to the new FTC Team. I have always enjoyed engineering and have done other science related activities outside of FIRST. I also enjoy Olympic recurve archery, reading, drawing, and writing. I am on the programing team, have helped the build team occasionally, and am a coach for a drive team. I am excited to see where this team will go in the future years.

Habtamu
4 years with FIRST
8th Grade

Hi I am Habtamu, I am 13 years old. After doing FLL for 2 years and being the oldest student there, moving up to FTC and being one of the youngest was a big change for me. The change was good for me though because that meant I was not the one teaching others, I was the one learning from others and making mistakes. But mistakes are only actions in which something can be learned and improved. So in other words, I am going to be able to learn, make mistakes, and soon be able to teach others.

I'm a Junior at HF High School. This is my 6th year with HF Robotics, 5th year in FTC. I really like the programming and driving/operating challenges that FTC gives me. In addition to robotics I play tenor saxophone in the school bands and a basement band. I would like to go into physics or computer sciences in the future. Science is cool and good.

Calvin
6 years with FIRST
Junior

Jeremy
5 years with FIRST
Senior

I plan to attend the United States Air Force academy in CO, I play alto saxophone in band for Homewood-Flossmoor High school, I am the 2 vice president for my chapter in Top Teens of America, I am a mentee in 100 Black Men of Chicago and I am in Kappa League.

I am on the build team
My mom introduced me to FIRST one day and it was something I found special so I stuck with it because I love building and understanding how things work. I always had a thirst for knowledge so when I heard about FTC I was interested real quick.

I am in Science Olympiad, I play soccer, I like art, and I have three bunnies.

I am on the build team and a robot driver.

I joined FIRST four years ago, after I went to a LEGO robot programming event and learned about FTC.

Taylor
4 years with FIRST
Junior

Kaylin
6 years with FIRST
Freshman

I am a Freshman at HF High School. This is my third year with FTC, although I was the “intern” while I was in FLL. I participated in 3 years of First Lego League (FLL) before joining FTC. I am on the build team and enjoy constructing things and then being able to see them work to do a task. FTC has taught me valuable skills, such as teamwork, engineering, and and perseverance, that I will remember my whole life.

Outside of FTC, I enjoy playing soccer, playing my flute, and reading.

I was introduced to FIRST when my dad and some of the other coaches started a FLL team 6 years ago. I’ve stuck with it ever since.

Ernest
5 years with FIRST
Freshman

I am 14 years old. I do swimming, lacrosse, and track. This is my 5th year in FIRST. I am the electronics director. I learned CAD this year. I joined FTC because when I was in FLL it looked cool how they were building bigger robots.

Logan
5 years with FIRST
8th grade

I am 12 years old, I play the drums, have a mini root beer stand and I play soccer. This is my 5th year in FIRST and I like robots.

Tarendran

I am thirteen years old and I enjoy playing volleyball and basketball. I operate the robot during competitions and have worked on the intake and drive base design and assembly this season.

This is my first year in FTC.

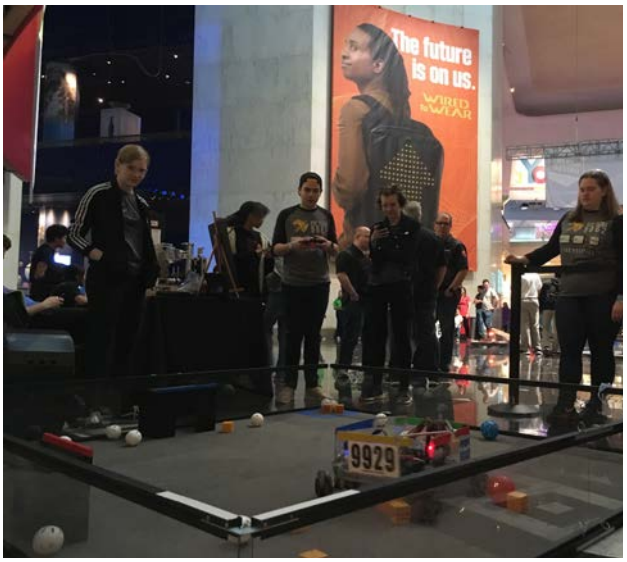
Team Outreach

Hosting league meets

We believe that in order to be successful as a team, outreach to both the community and other FIRST teams is necessary. This season we have hosted one league meet and brought fields and set up for two more to stir excitement about robotics in the community. It takes a lot of time and effort to plan these events, and we are very grateful for the many volunteers who have helped us run each meet. Hosting league meets has also allowed us to grow closer to some of our fellow FIRST teams in the area.

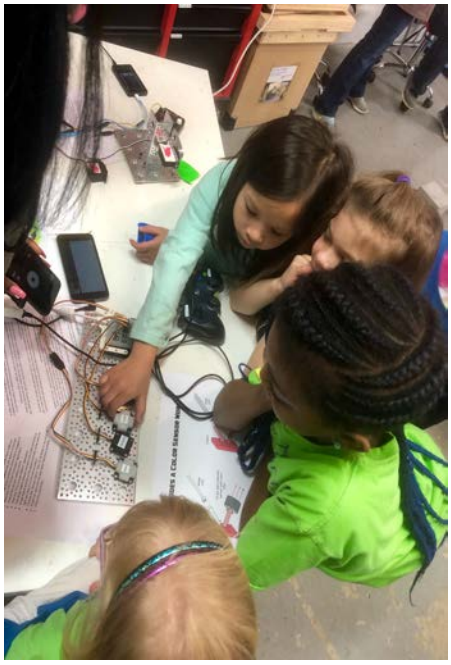
Robot Block Party at MSI

“Celebrate National Robotics Week by checking out the cool ‘bots designed by Chicago-area student and amateur teams, and interacting with some state-of-the-art robots.” Our first outreach opportunity after the Illinois State FTC tournament was in early spring, at the Museum of Science and Industry’s annual “Robot Block Party”. This was the second time the team had been invited, and we are proud that MSI wanted to have us again. This year, we decided to have a more ‘interactive’ outreach experience and made a game called “Rubbish Roundup”. We invited visitors to drive a simple robot to clear out “space junk” on the field, giving them two minutes to score as many points as possible. This idea was most definitely inspired by the team’s experience with FIRST and we were able to take a deeper look into what makes up an FTC challenge.



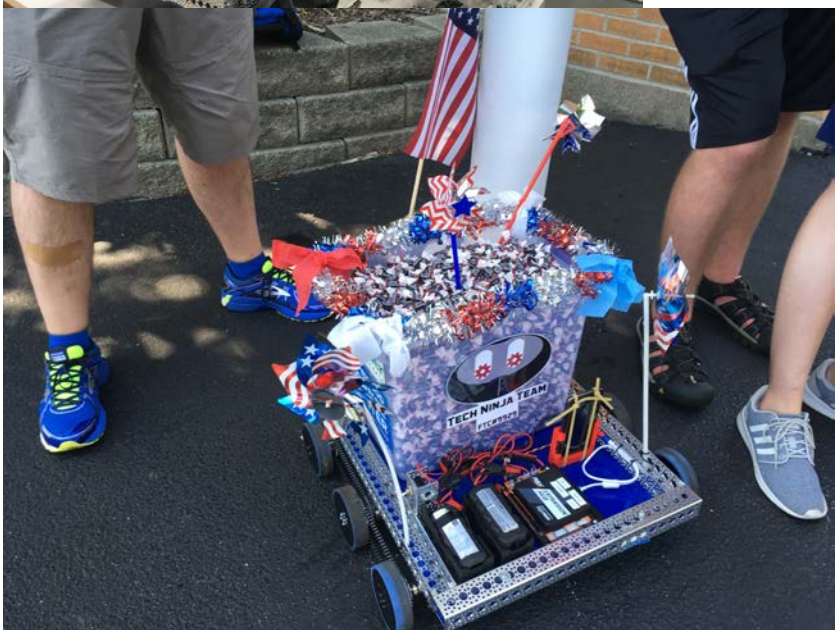
Brownie Troop Visit

A local Girl Scout Brownie Troop came to our workshop to earn a robotics badge. We had multiple ‘stations’ set up that allowed the girls to learn about and interact with sensors, motors, Skittlebot, and the basics of programming via a small game.



Homewood July 4th Parade

Last summer, we built “ParadeBot” equipped with 6000 mAhs of power and 1500 cubic inches of candy storage. This year is the second 4th of July we set out with “ParadeBot” decked in patriotic attire and marched over a mile in the Homewood 4th of July parade. We tossed candy to our adoring fans and tried to avoid getting soaked by the kids’ SWAT team and their water guns.




Milwaukee
The Black Dog

Workshop Open Houses During Wednesday Night Farmers' Markets

(200 attendees across 2 dates)

This season we had open house hours during the evening Farmers' Market and were able to reach a different audience than the morning Farmers' Market. We saw a lot more kids interested in our team. We saw about 80-100 attendees per event. We used our Mindcuber Lego Mindstorms robot to explain color sensors, servos, distance sensors and an algorithm for solving a rubix cube. We had DIY button making - with explanations of the press die in industry. We let attendees program a "robot" (another attendee) to travel from one end of our workshop to another to retrieve a bag of Skittles with our "Skittlebot" language, which is a series of cards with simple commands for the "robot" to follow. The most popular activity by far was getting to drive our first season's robot, also known as Skittlebot. We had kids and adults lined up waiting to try!


TECH NINJA TEAM
#9929

Homewood Science Center Fundraiser - Cocktail Chemistry

The Homewood Science center graciously invited us to participate in their “Cocktail Chemistry” event, which was all about the science of drinks. The team was stationed in the foyer along with other representatives from the HSC to explain what the fundraiser was all about. We even made a new robot- DrinkBot -that was able to deliver nonalcoholic ‘Cosmic Lemonade’ and root beer to visitors. Both drinks were made by members of the team, and we put a little bit of dry ice in each cup for a cool look. This was an excellent conversation starter, and it allowed us to reach out to many different people in the community!

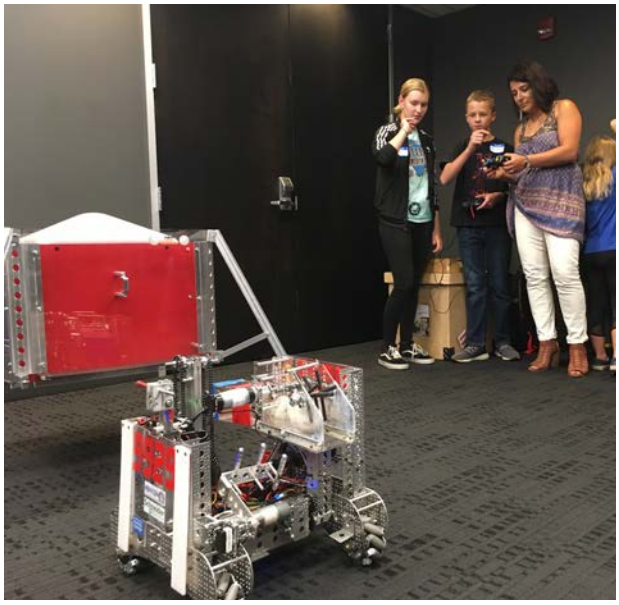
Homewood/Flossmoor Park District Day Camp Visits Our Workshop

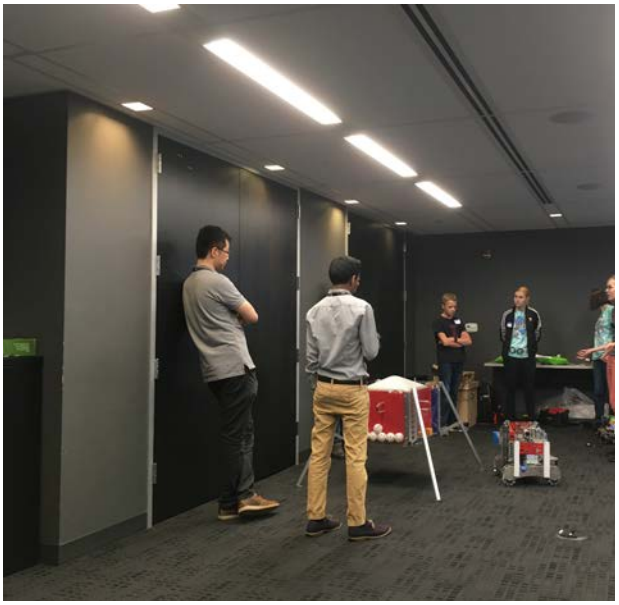
Schneider (Sponsor) Visit

One of our sponsors, Schneider Electric, invited the team to visit their office and present to some of the people who worked there and talk to them about what their sponsorship was doing. The robot from last year's challenge was brought in to demonstrate the tasks that we were able to complete, and to show off our hard work. The people at Schneider Electric were very glad to see how they helped us young adults learn more about STEM, and gave us plenty of encouragement and advice.

Schneider STEM Day

Once again, our sponsor Schneider Electric reached out to us and asked for us to attend a STEM day at the company for the children of employees. We talked to kids about what FIRST is, what we have learned, our robot, and more things about STEM. In addition, we were able to answer questions that many kids had about robots and what we do as a team.





Homewood Science Center Fundraiser - Walk Walton

Walk Walton is an outdoor fundraiser for the HSC where people can listen to ecologists and scientists along the Isaac Walton trails to learn more about the ecosystem and environment. We had a tent set up to display some of our accomplishments. Fortunately, the weather was good so we had many visitors stop by to ask questions.

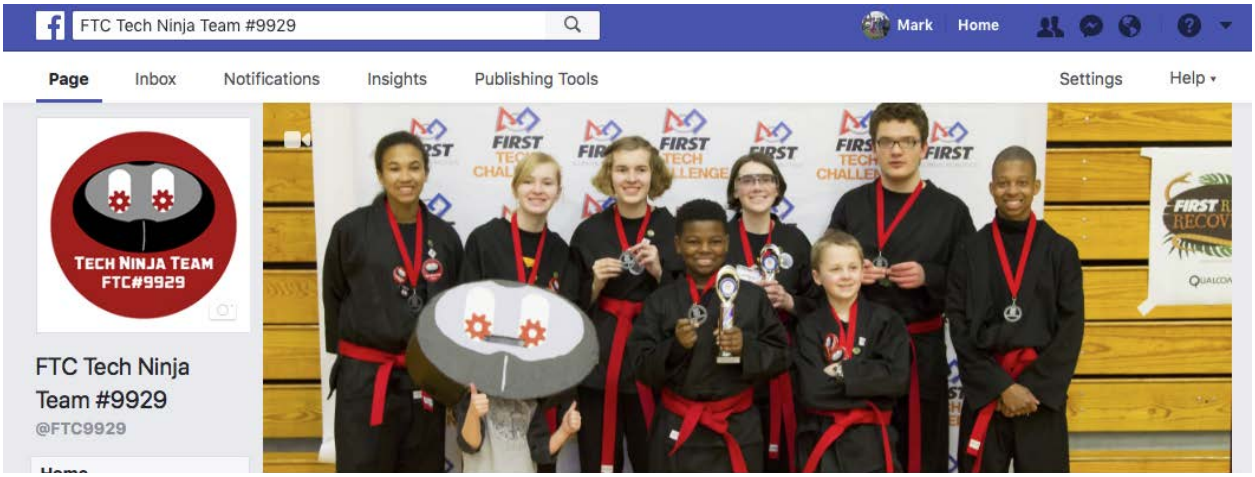
Social Media / Marketing

Website

This year is the first year that the team has its own independent website! It features pictures of events and meets, information about both the FTC FIRST program and our team, some blog entries of things we have learned, and even past seasons' Engineering Notebooks. This marks an exciting point in our teams' history as we now have a more accessible platform to reach out to potential sponsors and the community.

Facebook

We have our own Facebook Page, used more to reach local audiences (upcoming events, news). We have 150+ followers that are primarily within the state of Illinois or are family and friends of the team members.

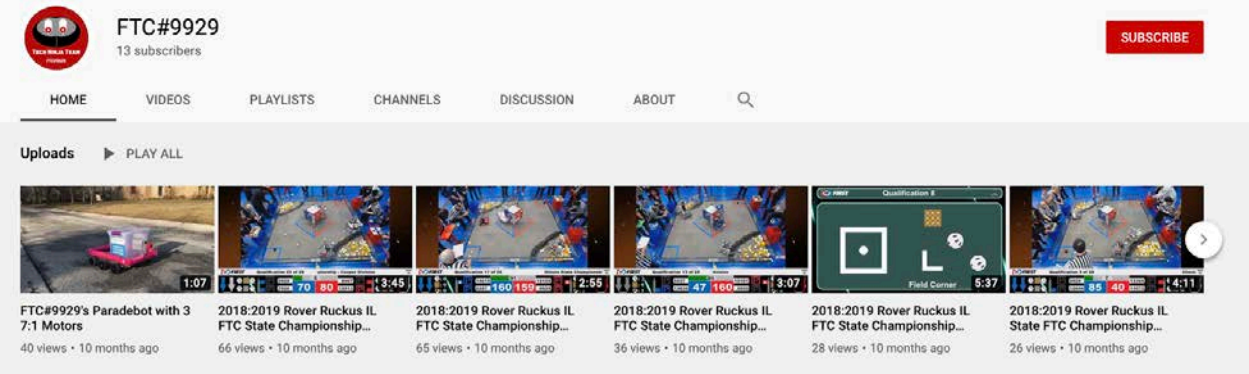


Twitter

We are @FTC9929 on Twitter. Here is where we tend to keep up with other FTC teams, sharing our successes (and experiments that don't quite work out). We have over 450 followers, and we enjoy seeing how teams around the world are having fun with robots and STEM.

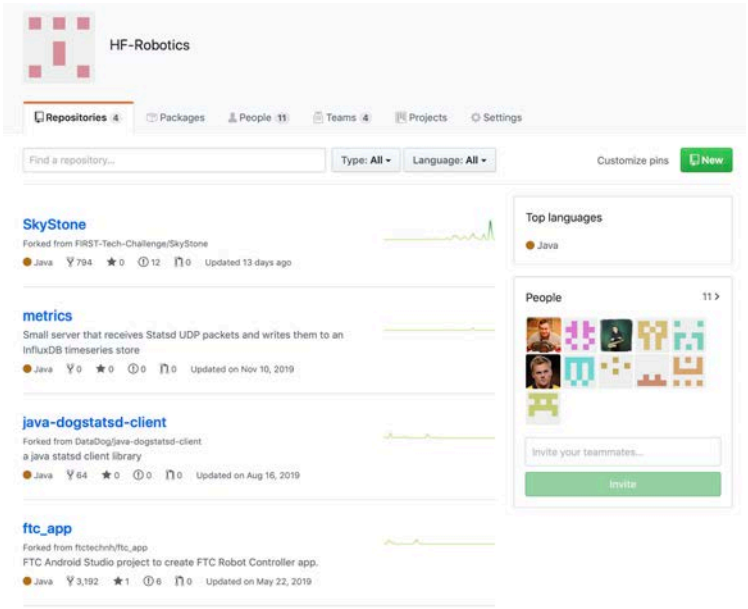
YouTube

We have many videos including footage of previous matches and tech tips such as “Friends Don't Let Friends Use KEP Nuts”



GitHub

We share all our robot code with the world as we write it at *https://github.com/HF-Robotics/ftc_app/*.



We’ve structured our program such that much of it is reusable season-to-season and we’re hoping that it may be a jumping off point for teams that need help in this area. We also publish the source code from this year’s off-season projects on Github for others to learn from: robot controller metrics client - https://github.com/HF-Robotics/java-dogstatsd-client, and metrics server and dashboards - https://github.com/HF-Robotics/metrics

Robot Overview

Skystone Analysis and Strategy

Playing Skystone well involves a strategy that succeeds in performing all tasks presented in the challenge. The major tasks associated with Skystone are detecting and delivering skystones, moving the foundation and parking during autonomous, acquiring stones and stacking them on the foundation during tele-op, and placing capstones, moving the foundation, and parking during end game.

With the objective of being able to perform competitively, at a very high level, our first goal was improving from last year's robot, place higher, and to be able to choose our alliance partners. We set out to have the highest scoring robot possible, to help create strong alliances. We tried to create a robot that was equally strong in autonomous as it is in tele-op. To accomplish this, we had a strategy and prototyping weekend, where we:

- Analyzed the game and identified the ways to score points
- Identified the constraints of the game (the rules, and dimensions of the field and game elements)
- Brainstormed and prototyped mechanisms to work with the game elements, and used existing drive bases to understand the movement around the playing field

We came up with the following list of tasks and possibilities of performing them during autonomous:

Tasks	Time estimate	Probability of completion	points
Repositioning	7 sec	100%	10
Skystones	5 s/per	100%	20 (only if initial 2)
Stones	5 s/per	100%	2
Placing	2 s/per	75%- VARIABLE	4
Navigation	3 sec	100%	5

Based on the above estimates, one robot could move 2 skystones and one additional stone in Autonomous *if* it also has to reposition the foundation.

Tele-op and end game present a set of challenges that required us to work harder at prioritizing the design for the drive base and mechanisms that would be used. Our analysis of potential scoring scenarios is below:

Scenario 1 (one block skyscraper)	Points:

Delivered stones	15
Placed stones	15
Levels	30
Total points	60

We determined that the maximum stable height is 15 blocks.

Scenario 2 (two blocks per level, changing directions)	Points:
Delivered stones	30
Placed stones	30
Levels: 15	30
Total points	90

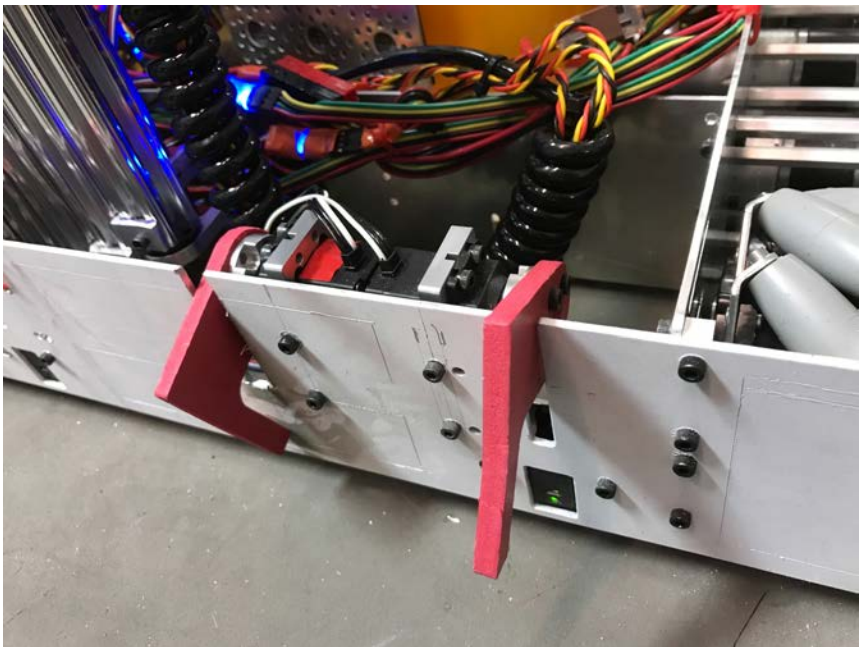
Therefore task priorities are:

- Going under the alliance specific skybridge is a must, otherwise we miss out on potential points.
- Collect, deliver, and place new stones to increase the height of the tower.
- If other alliance drops a stone, grab it instead of one from the depot (it saves time)
- Rearrange already delivered stones into tower in order to make a secure skyscraper to decrease the possibility of it toppling.
- There is no need to alternate alignment at every level. This is only a concern at higher levels of blocks because it increases the stability of the tower. Doing this at every step is unnecessary and may waste time.

Robot Design Strategy

- Stacking
    - Ideally the robot would stack blocks perpendicular to each other, but this requires much greater speed.
- Maximum height
    - The maximum skystone tower we could build by hand was 15 levels -- about five feet high.
    - But the robot had to be less than 14” tall to fit under the skybridge.

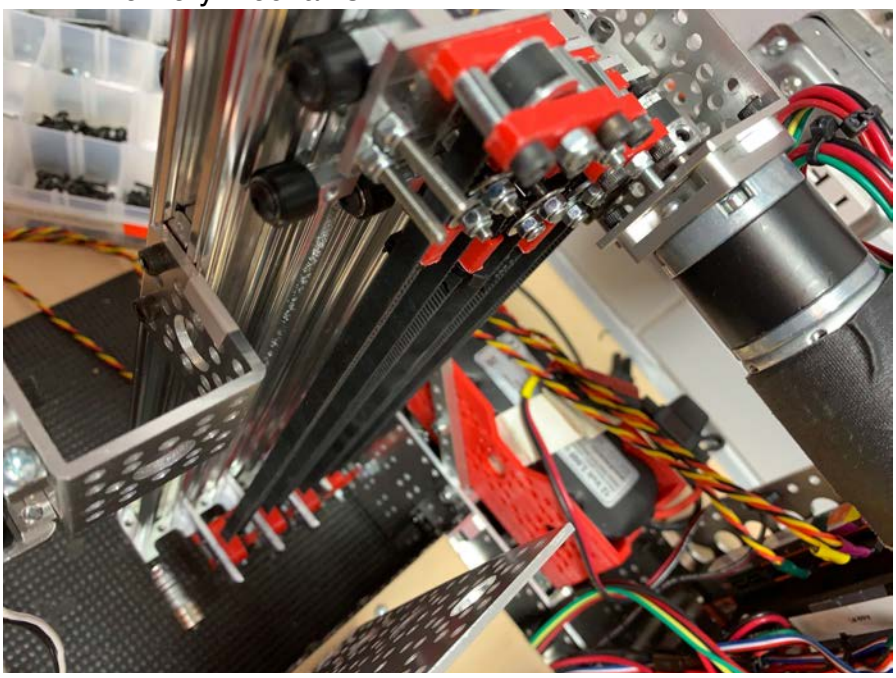
- Foundation mechanism



    - Simple, servo-driven hook mechanism which grabs onto the foundation.

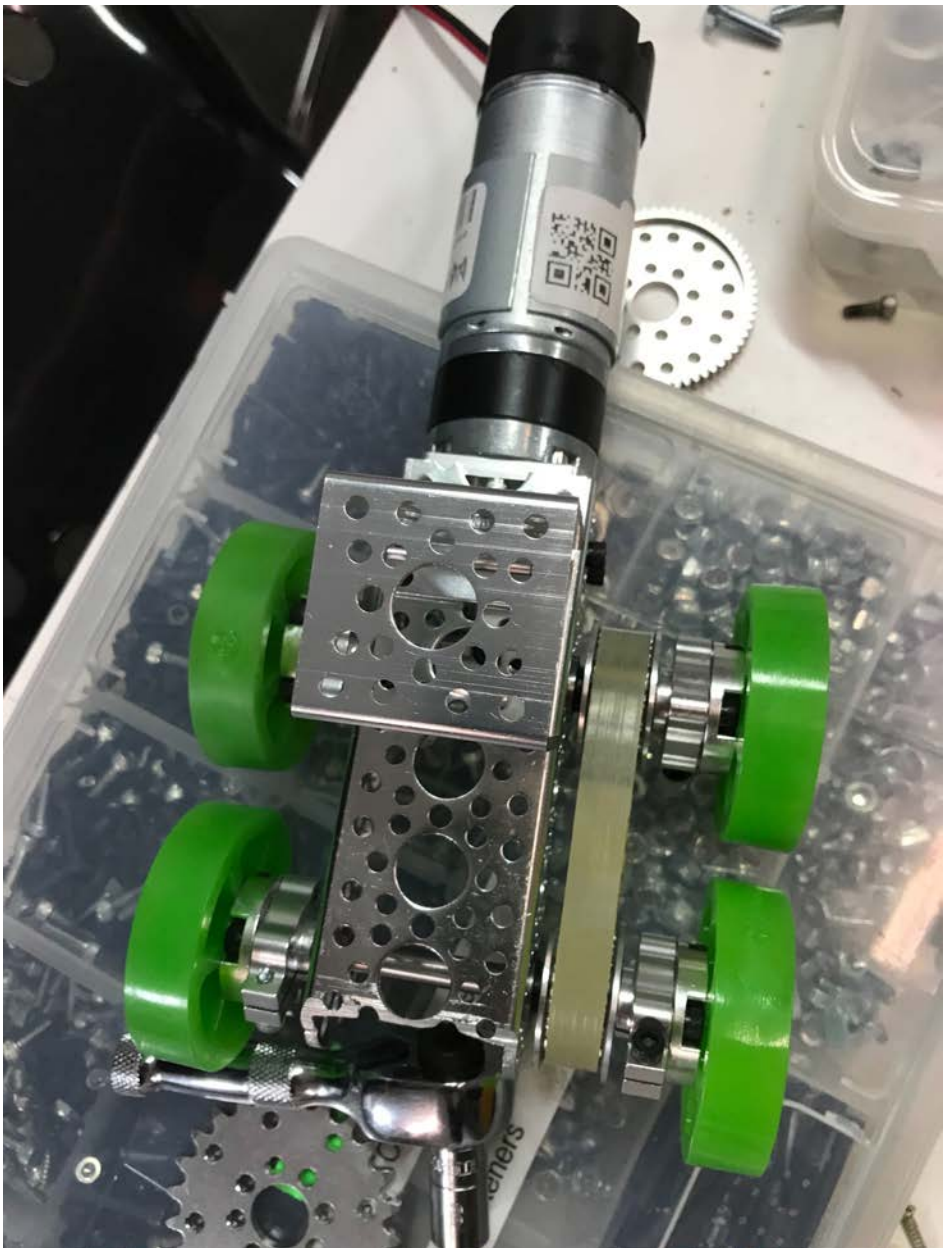
- Skystone Detection
    - The robot needs a camera to determine which stones are skystones.

- Delivery Mechanism



    - Pick up from the top

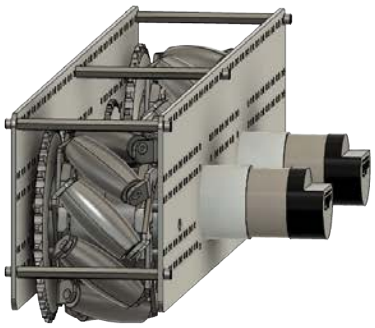
    - Able to rotate blocks 90 degrees for placing
    - Needed to be able to build a tower as high as possible
        - If double-stacked, 15 levels
    - Speed - needs to be able to deliver a new stone every few seconds.
    - Lift speed - needs to raise to full height in a second or two.
- Intake Mechanism

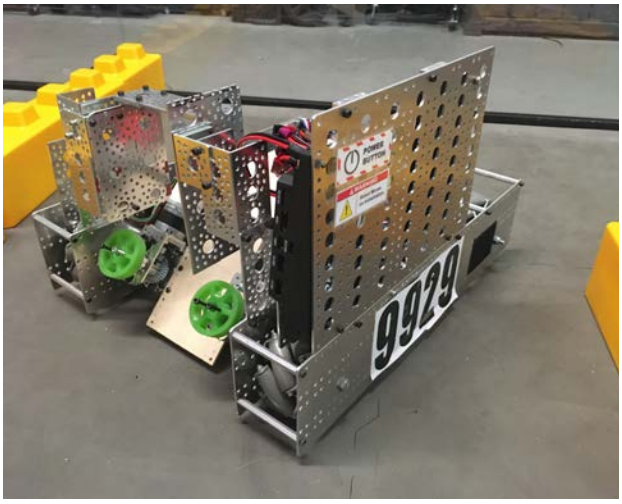


    - Wheeled intake and ramp.

Drive Base Iterations

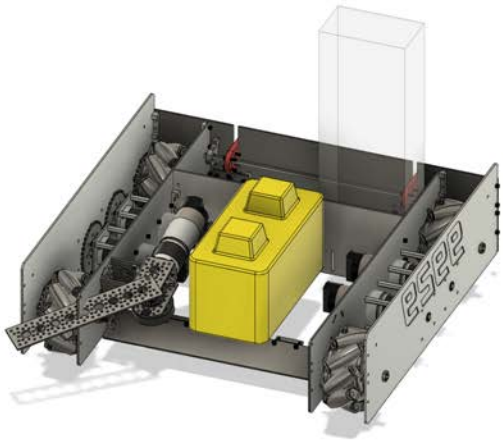
Version 1 - Prototype (Used for League Meets 1, 2)





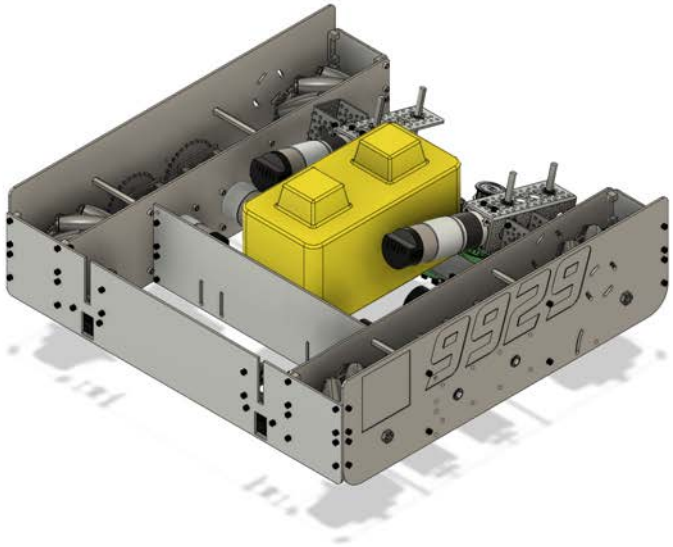
In version 1, the intake ramp was steep, a small amount of room for intake, has to get the block from a perfect angle, and was not adjustable. We had designed the drive base before we had considered the intake mechanism fully.

Version 2 - Design Ideas from Version 1, used for League Meet 3

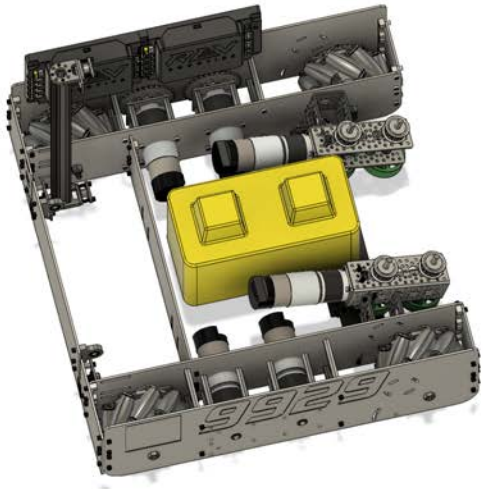


“Cowcatcher” - based on learning from “v1” of drive base. Width increased, structure to guide stones to intake wheels. We didn’t end up going with this concept.

Less cowcatcher, intakes moved forward to grab stone earlier:



Intakes still forward, experimenting placement of both REV hubs, which did end up in this configuration, but towards the intake side:



Intake Iterations

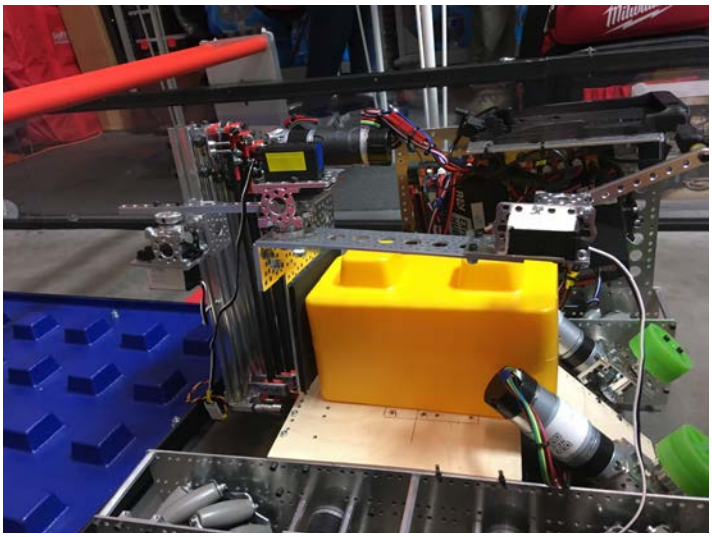
Concept #1 - Single Compliant Wheel

This intake could intake stones, and deliver them to the playing field, but not the foundation. We did not get the lift and gripper onto our robot in time for the first league meet. This mechanism was workable enough for the first league meet, but had the following problems:

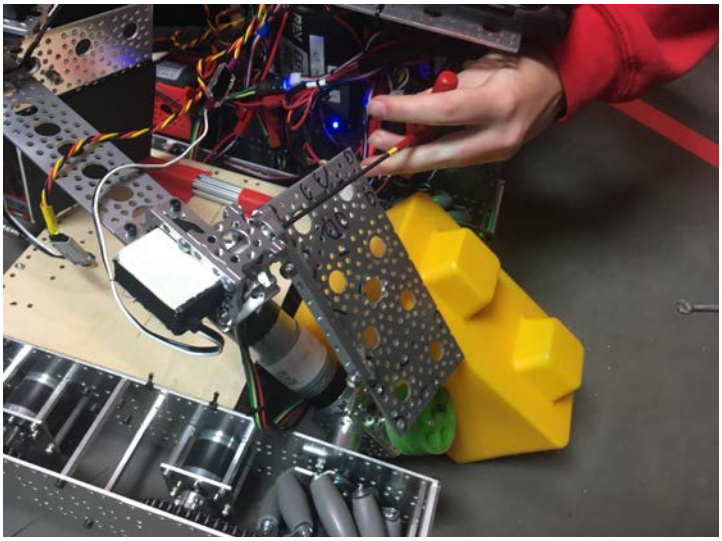
- Could only pick up stones when they were aligned straight with the robot
- The angle of the ramp (to clear the drive motors) was steep, caused stone to want to flip over
- Because of this, we used a hood to keep stones from flipping over, could not deliver to foundation
- Ramp high off ground - sometimes couldn't pick up a stone at all

Second League Meet - Delivery and Intake Working Together!

- We attached the gripper to the lift

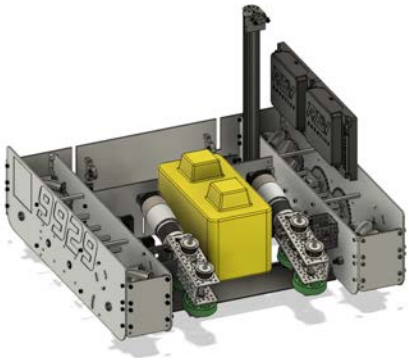


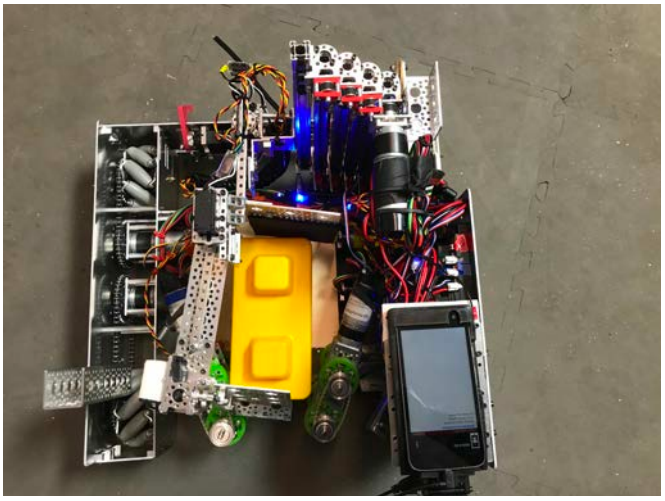
- Added a a gripping “finger” that was wide, and bent at the back to both guide to stop stone flipping over when using intake, and apply pressure to grip to deliver to foundation



- Guides added, prevented flying stones which would get stuck over intake motors

Third League Meet - Drivebase, Intake and Delivery working Together!



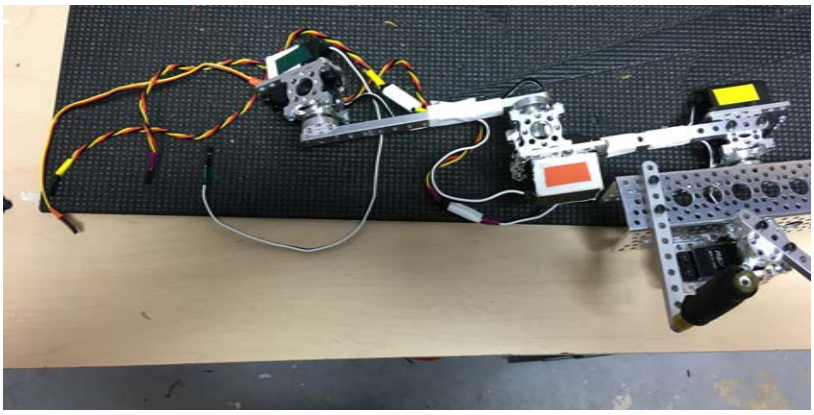


- Used four-wheeled wristed intake
    - Wristed (hinges) helps the intake grab stones that aren't perfectly aligned
    - Added wheels high and low - more wheels (lower), more grip on stones top and bottom
- Placing the floor of the robot low meant ramp is less steep, stones less likely to flip
- Room for skystone down low, more control

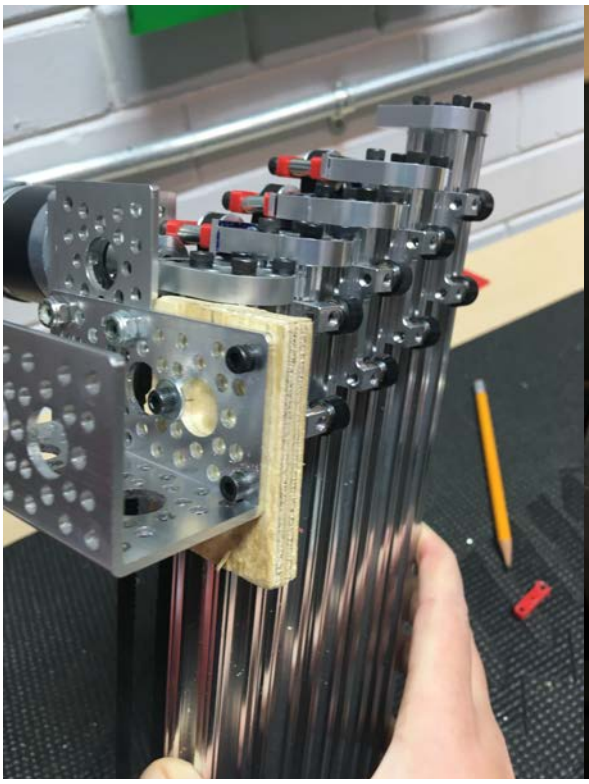
Delivery Iterations

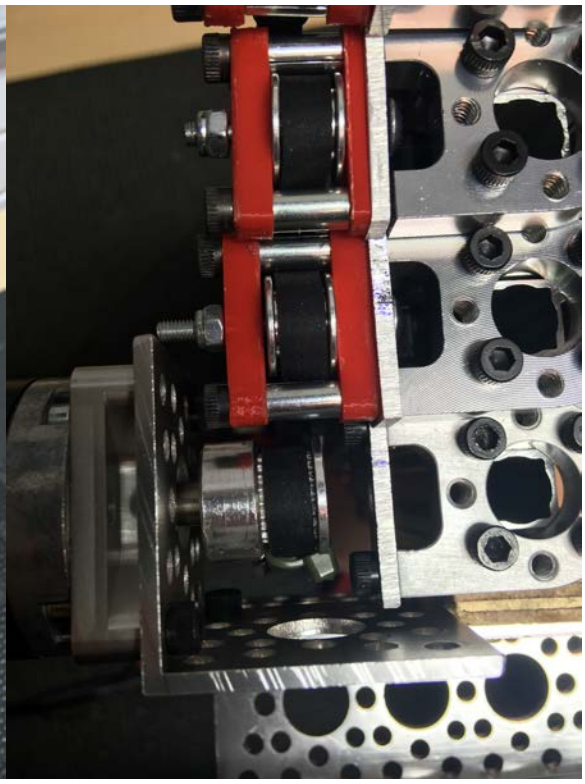
First League Meet

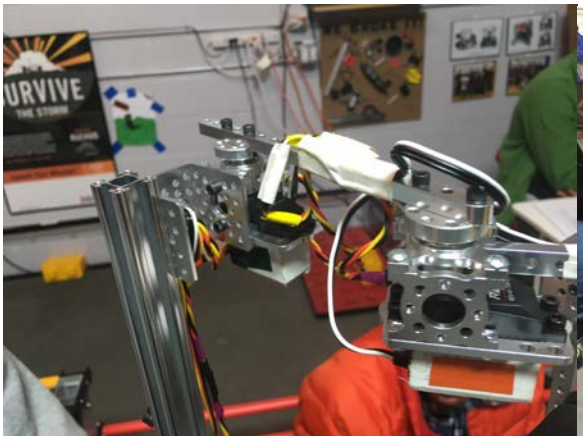
Many gripper/arm designs, building the lift, but ran out of time to get it on the robot for the first league meet. We ended up having slightly more than a pushbot, but learned a lot which would change our next iteration.



Designed and manufactured the parts needed for the belt-driven lift. Some are aluminum, some are 3-D printed:

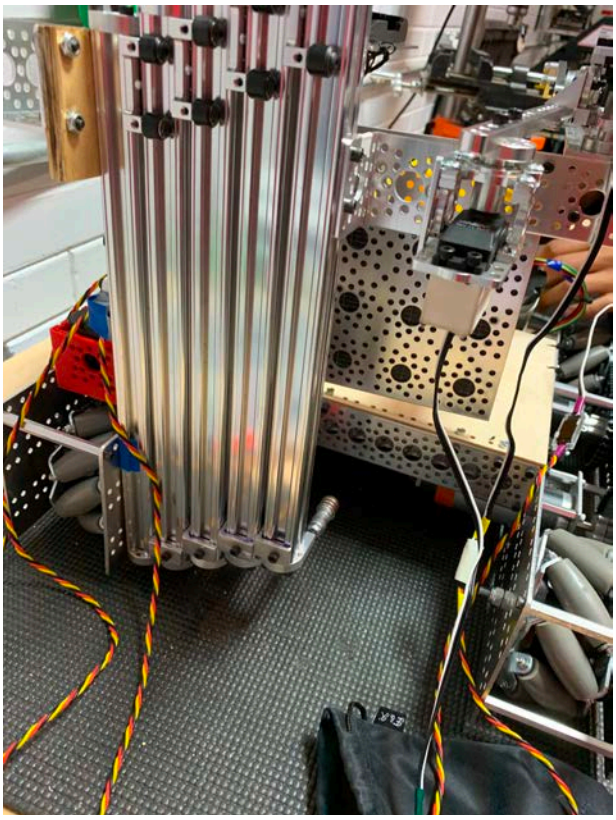






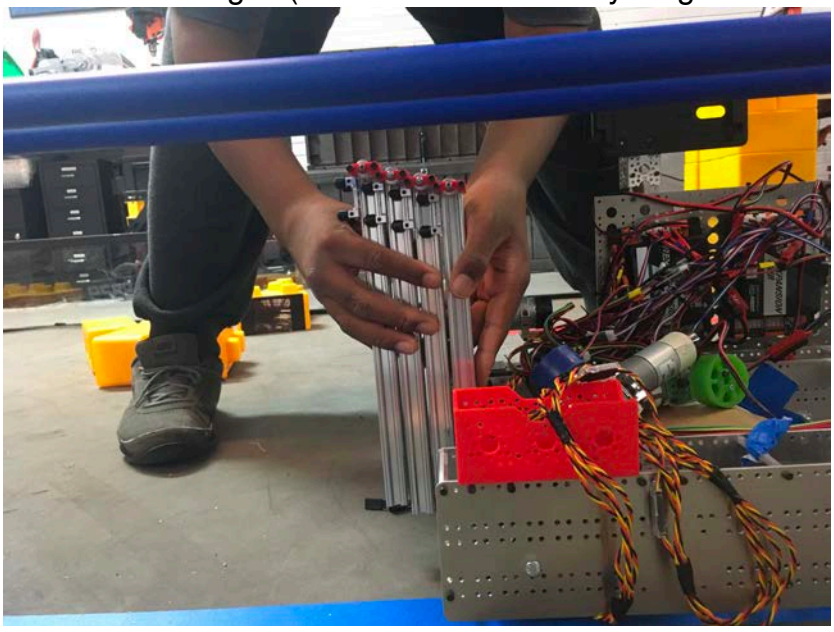
Second League Meet

We were able to combine the intake, lift and gripper mechanism into a functioning whole for the second league meet. Luckily, we were able to have 2-3 of our team members improve “v1” of our robot while prototyping and design of “v2” was being handled by the rest of the team.



The lift ended up having the following qualities:

- Short stages (to fit under alliance skybridge



- Timing belt driven:
    - No string (our team has not figured out yet how to make it reliable)
    - Couldn't use chain like last season (not enough space, too heavy)

The first gripper prototype only grabbed top of stone on one end, it tended to drop them because of the sloped sides

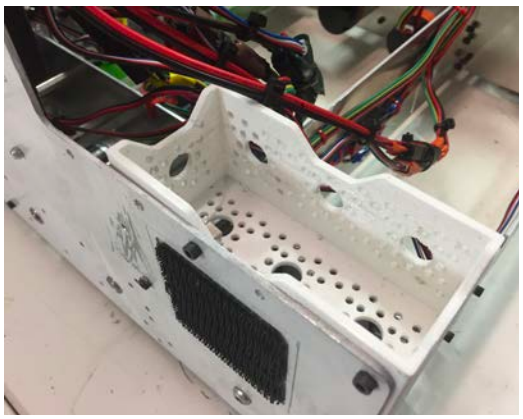
To get a more reliable gripper for the second league meet we:

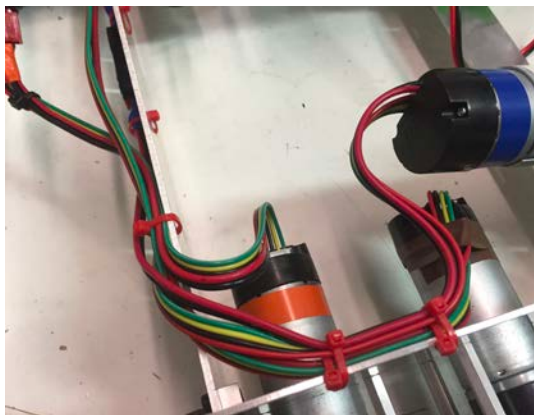
- Removed elbow from arm, not needed to build a tower (cuts down time to extend)
- Changed gripper to grab front and back of stone (see prior section)
- Added hard stops to make sure mechanism is correctly aligned when inside robot (stowed), and when delivering stones to the foundation, and save servos from breaking if arm gets hit.
- Servo cables were routed through a retractile cable, did limit height we could stack (4 stones, or 3 plus our capstone)

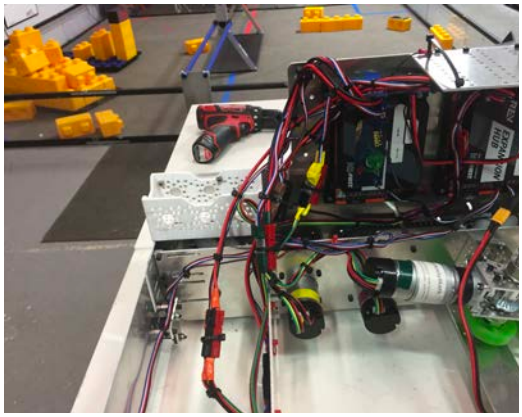
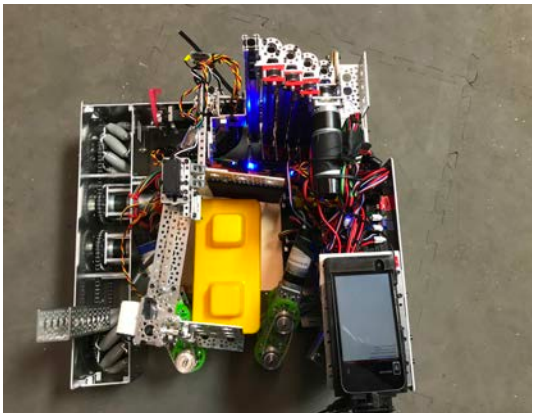
Third League Meet

We made the following changes for our third league meet to address the issues with the lift that we ran into during our second league meet, as well as the issues with the intake and ramp with version one of our robot.

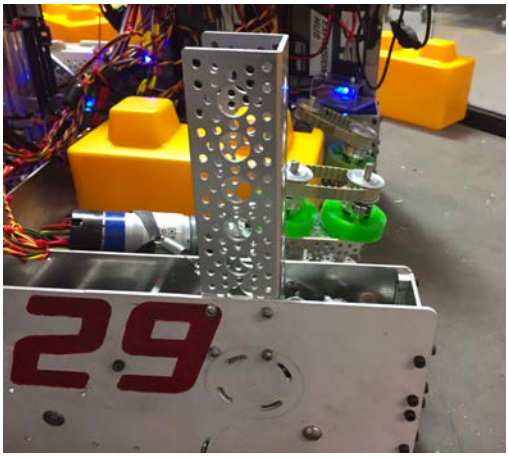
- The lift was missing pieces on the bottom of each stage, which caused it to bend, and then bind. We added those pieces, while the lift was still attached to “v1” and now the lift operates smoothly.
- We made the parts and assembled them for the “v2” drive base and transferred all of our mechanisms onto the new drive base.

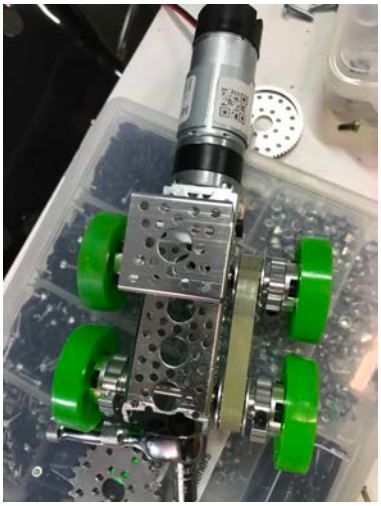




- We installed our new intake into “v2” as well.





Preparing for League Qualifier

- Wanted to stack more than four
- Noticed the wires kept the lift from going up more
- Replaced wires
- Now can stack at least six
- Added Skystone grabber

Software

“We aim for controllability and accuracy”

We run a full autonomous; including detecting and moving skystones, scoring stones, and parking. As in prior seasons, our autonomous “routes” are selectable from the drivers’ station during “init”, and we support configurable delays – which is a great help for scheduling with our alliance partners that need space to park or move the foundation.

In tele-op we started to implement code that allowed us to condense multiple repeated actions into the press of one or two buttons. This practice can be seen in our elevator (the lift that is used to stack stones on our robot). For this code we use an open loop - decisions are made based on driver controller input, and closed loop - decisions are made based on sensors and on robot functions much like autonomous. We also keep track of where the lift is in order to determine whether or not it is safe to move the arm.

Another step we take to reach better controllability and accuracy was the installation of multiple safeties. We use both hard (mechanical) and soft (code) stops that allow our drivers to operate mechanisms with much less worry of breaking them in the process.

Soft stops are one of the features that allows us to run certain functions of our robot, such as the lift and the arm, at high speeds. We also have two buttons on the controller related to the safeties; the “unsafe” button - which disables the code-based safety and is used if a failure happens with the safety, causing the robot to be less usable, and the e-stop button - which stops all automatic functions on the robot if the safeties fail to do so.

Some examples of soft stops on our Skystone robot are:

- Timeouts
- Encoder value reading
    - Switching states or functionality after a certain encoder value is reached
    - Stall detector that stops us from using a motor that is stalled (encoder values remaining constant)
- Limit switches are used to detect the minimum position of the lift

Our tele-op code also contains functionality that makes things easier for our drivers, and places less mechanical strain on the robot, such as:

- Low-pass filters
    - Smooths out driving and prevents jerky movements
- Independent throttle curves
    - Allows us to give different speeds and limits to different parts of the robot

Breaking our Robot on Purpose - Chaos Ninja

(1) We already test errors and how they are handled in unit tests

(2) Expanded to create “faulty” versions of motors and servos to be used while running tele-op

(3) Safe - enabled only via entering Konami Code (up,up,down,down, left, right, left, right, b.a)

(4) “ChaosController”, if enabled, based on configured challenge mode, randomly decides whether to fail or not, which motors or servos to fail, how (dead, slow), or to add lag, and picks a random time to do it.

(5) This is the code for “ChaoticServo” If the servo failure mode is “dead”, we don’t send the position value to the real servo, but we do store it, and return it when getPosition() is called, because that’s how it would work if a real servo failed too.

```
public class ChaoticServo implements Servo {

    public double actualPosition;

    public enum ServoFailureMode {
        DEAD,
        REVERSED,
        HEALTHY
    }

    private ServoFailureMode failureMode = ServoFailureMode.HEALTHY;

    private final Servo actualServo;

    public ChaoticServo(final Servo actualServo) {
        this.actualServo = actualServo;
        this.actualPosition = actualServo.getPosition();
    }

    protected void setFailureMode(ServoFailureMode failureMode) {
        this.failureMode = failureMode;
    }

    @Override
    public ServoController getController() {
        return actualServo.getController();
    }

    @Override
    public int getPortNumber() {
```

```
        return actualServo.getPortNumber();
    }

    @Override
    public void setDirection(Direction direction) {
        actualServo.setDirection(direction);
    }

    @Override
    public Direction getDirection() {
        return actualServo.getDirection();
    }

    @Override
    public void setPosition(double position) {
        actualPosition = position;

        if (failureMode == ServoFailureMode.DEAD) {
            return;
        }

        actualServo.setPosition(position);
    }

    @Override
    public double getPosition() {
        return actualServo.getPosition();
    }

    @Override
    public void scaleRange(double min, double max) {
        actualServo.scaleRange(min, max);
    }

    @Override
    public Manufacturer getManufacturer() {
        return actualServo.getManufacturer();
    }

    @Override
    public String getDeviceName() {
        return actualServo.getDeviceName();
    }

    @Override
```

```
    public String getConnectionInfo() {
        return actualServo.getConnectionInfo();
    }

    @Override
    public int getVersion() {
        return actualServo.getVersion();
    }

    @Override
    public void resetDeviceConfigurationForOpMode() {
        actualServo.resetDeviceConfigurationForOpMode();
    }

    @Override
    public void close() {
        actualServo.close();
    }
}
```

Developing Software Before Robot Mechanism Is Built

How we write our robot code, from our team design process manual:

> Use components, not inheritance.
>
> - Create simple building blocks (switches, servos, motors), build more complex things using those (mechanisms), and then put those into a robot class which leads to software that has the following desirable qualities:
>
> - Simple to understand - each class expresses what it needs and does in a simple way
> - Simpler to test - it's easier to write tests at each component level, as well as higher levels to make sure our code works well - or to create experiments before we build something physical
>
> - Easier to reuse - code structured this way is easier to reuse, between tele-op and autonomous on the same robot, and also from year to year on different robots.

We do this by:

(1) Communicating with the build team to come up with a rough sketch (specification)
(2) Creating a class that represents mechanism (not just code in the tele-op or auto class)
(3) Write unit tests that use “fakes” for sensors, gamepads, servos, motors, etc.
    (a) Off-season project to learn to write unit tests
    (b) During that project, implemented “fakes”, used when creating instances of #(2) in tests, use “real” servos when running on robot:

```
public class FakeServo implements Servo {
    protected Direction       direction        = Direction.FORWARD;
    protected double          limitPositionMin = MIN_POSITION;
    protected double          limitPositionMax = MAX_POSITION;

    private double servoPosition;


    @Override
    public void setDirection(Direction direction) {
        this.direction = direction;
    }

    @Override
    public Direction getDirection() {
        return direction;
    }

    @Override
    public void setPosition(double position) {
        position = Range.clip(position, MIN_POSITION, MAX_POSITION);

        double scaled = Range.scale(position, MIN_POSITION,
MAX_POSITION, limitPositionMin, limitPositionMax);

        servoPosition = scaled;
    }

    @Override
    public double getPosition() {
        double reportedPosition = servoPosition;

        double scaled = Range.scale(reportedPosition,
limitPositionMin, limitPositionMax, MIN_POSITION, MAX_POSITION);
```

```
        return Range.clip(scaled, MIN_POSITION, MAX_POSITION);
    }

    @Override
    public void scaleRange(double min, double max) {
        min = Range.clip(min, MIN_POSITION, MAX_POSITION);
        max = Range.clip(max, MIN_POSITION, MAX_POSITION);

        if (min >= max) {
            throw new IllegalArgumentException("min must be less than
max");
        }

        limitPositionMin = min;
        limitPositionMax = max;
    }


    @Override
    public void resetDeviceConfigurationForOpMode() {
        this.limitPositionMin = MIN_POSITION;
        this.limitPositionMax = MAX_POSITION;
        this.direction = Direction.FORWARD;
    }



    private double reverse(double position) {
        return MAX_POSITION - position + MIN_POSITION;
    }
}
```

Here is an example of a set of tests for the delivery mechanism (our robot's lift, intake, and gripper) with "fake" motors and servos so we can write and test our code without a real robot present.

The DeliveryMechanism code behaves the same whether it is being used by our tele-op code and calling methods that make real motors and servos move on our robot, or if it is using fake motors, servos and sensors. The DeliveryMechanism code cannot tell what is being used behind the scenes. However, our tests can make sure that the DeliveryMechanism code is telling the fake motors and servos to do what is expected by calling methods in our code, and making assertions about what should happen:

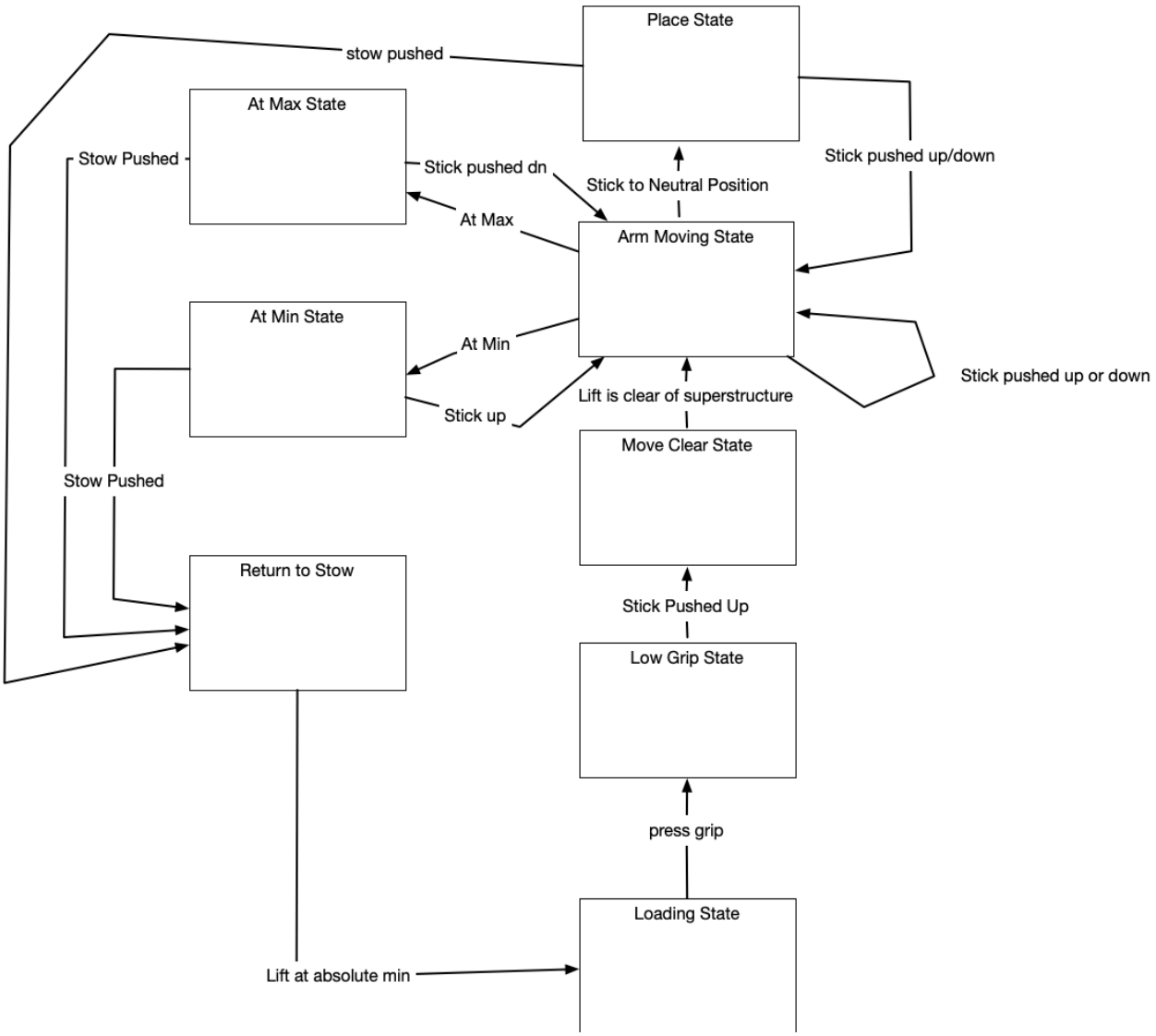
```
    @Before
    public void setUp() {
        // Create a delivery mechanism using fake components
        // (some code removed to simplify for engineering notebook)
        ticker = new FakeTicker();
        ticker.setAutoIncrementStep(5, TimeUnit.SECONDS);

        hardwareMap = new FakeHardwareMap();

        liftMotor = new FakeExtendedDcMotor();

        fingerServo = new FakeServo();

        deliveryMechLowLimitSwitch = new FakeDigitalChannel();

        hardwareMap.addDevice("liftMotor", liftMotor);
        hardwareMap.addDevice("fingerServo", fingerServo);
        hardwareMap.addDevice("deliveryMechLowLimitSwitch",
deliveryMechLowLimitSwitch);

        FakeTelemetry telemetry = new FakeTelemetry();

        deliveryMechanism = new DeliveryMechanism(hardwareMap,
telemetry, ticker);

        gamepad = new FakeNinjaGamePad();

        controls =
OperatorControls.builder().operatorsGamepad(gamepad)
                .deliveryMechanism(deliveryMechanism)
                .build();
    }
```

A simple DeliveryMechanism unit test - testing that the grip finger responds correctly to the gamepad button press:

```
    @Test
    public void testFingers() {
        gamepad.reset();
        FakeOnOffButton gripButton = (FakeOnOffButton)
gamepad.getYButton();
        FakeOnOffButton ungripButton = (FakeOnOffButton)
```

```
gamepad.getAButton();

        // Run the mechanism's state machine one time
        controls.periodicTask();

        // Test that the beginning state has the servo in the un-grip
position
        Assert.assertEquals(DeliveryMechanism.FINGER_UNGRIP,
fingerServo.getPosition(), 0.001);

        gripButton.setPressed(true);

        // Test that the servo in the grip position after pressing
the button
        controls.periodicTask();
        Assert.assertEquals(DeliveryMechanism.FINGER_GRIP,
fingerServo.getPosition(), 0.001);

        // Test that the servo in the un-grip position after pressing
the button
        gripButton.setPressed(false);
        ungripButton.setPressed(true);
        controls.periodicTask();
        Assert.assertEquals(DeliveryMechanism.FINGER_UNGRIP,
fingerServo.getPosition(), 0.001);
    }
```

A slightly more complex example, testing the state machine of our delivery mechanism, and whether it goes to correct state when responding to sensors)




```
        // ---------------------------------------------------------
-
        // Test transition from max to min
        // ---------------------------------------------------------
-


        {
            gamepad.reset(); // "let go" of the lift throttle
            Assert.assertEquals(
               DeliveryMechanism.AtMaxState.class.getSimpleName(),
               deliveryMechanism.getCurrentStateName());

            liftThrottle.setCurrentPosition(1);
```

```
            deliveryMechanism.periodicTask(); // one to transition
            deliveryMechanism.periodicTask(); // the other to
actually do what the state does

            // Lift is still moving down
            Assert.assertTrue(liftMotor.getPower() < 0);

            // Use the fake to put motor in a position below the
lower limit
            liftMotor.setCurrentPosistion(
               DeliveryMechanism.LIFT_CLEAR_SUPERSTRUCTURE_POS -  3);

            deliveryMechanism.periodicTask();
            deliveryMechanism.periodicTask();

            // Delivery mechanism should now be at "AtMinState",
which is
            // as low as it can go without stowing.

            Assert.assertEquals(
                DeliveryMechanism.AtMinState.class.getSimpleName(),
                deliveryMechanism.getCurrentStateName());

            // When not moving, the delivery mechanism code should
            // send a feed-forward voltage to the lift motor so that
            // it does not sink due to gravity or the coiled servo
cable

            Assert.assertEquals(
               deliveryMechanism.LIFT_HOLD_FEED_FORWARD,
               LiftMotor.getPower(), .0001);

            // Even though the joystick is pushed down, test that
            // the state machine stays at min position
            // and keeps the feed-forward power that keeps the lift
            // from sinking

            liftThrottle.setCurrentPosition(1);

            deliveryMechanism.periodicTask();
            deliveryMechanism.periodicTask();

            Assert.assertEquals(
               DeliveryMechanism.AtMinState.class.getSimpleName(),
               deliveryMechanism.getCurrentStateName());
```

```
            Assert.assertEquals(
               deliveryMechanism.LIFT_HOLD_FEED_FORWARD,
               liftMotor.getPower(), .0001);
        }
```

On-Robot Metrics and Analysis

> "Why can't we have real time performance data from the robot the way F1 teams do from their cars?"
>
> Kaylin (FTC#9929 Pit Crew Chief and F1 Fan)

The FTC SDK itself can send real-time data to the driver station phone via the Telemetry class, but that data is unstructured and is not stored anywhere. Android's logging facilities do store what is logged, but there is a rate limit to the amount of data that can be sent to the logs, and the log itself is unstructured. Because of this, our robot OpModes have used the log for significant events, but not performance data.

The team decided to spend some time before SKYSTONE kicked off to research some solutions to this problem, and come up with something we felt could be used to analyze the performance of our robot and operators in seasons to come. We decided that the solution should have the following characteristics:

(1) It should be possible to analyze the data in (near) realtime.
(2) Collection and transmission of the data should not hurt robot reliability or performance
(3) The data should be saved for later analysis
(4) The tools used should allow users to ask "what-if" questions easily
(5) Data is structured to make it easier to analyze
(6) We should not reinvent the wheel, if possible

We chose the Statsd protocol, it's simple, text-based ("name:valueItype"), and started by writing a parser, and unit tests for the parser:

```
public class MetricsParserTest {
   private MetricsParser metricsParser = new MetricsParser();

   @Test
   public void entirelyWrongData() {
       // Can't parse, don't return any guages
       Assert.assertEquals(0, metricsParser.parseMetric("the answer
is 42").size());
```

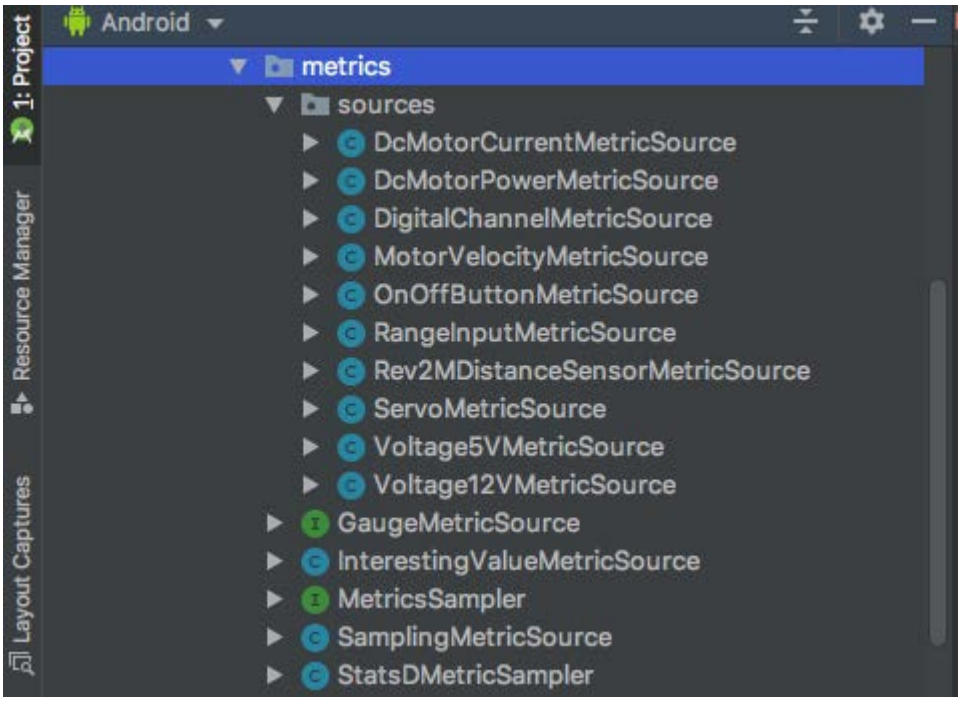
```
        Assert.assertEquals(0,
metricsParser.parseMetric("foo1c").size());

        List<Metric> parsed =
metricsParser.parseMetric("s_pos:100|g\ngarbage\ns_pos:200|g");
        Assert.assertEquals(2, parsed.size());
    }

    @Test
    public void validGaugeData() {
        List<Metric> parsed =
metricsParser.parseMetric("s_pos:100|g");
        Assert.assertEquals(1, parsed.size());

        Metric metric = parsed.get(0);
        Assert.assertEquals(Gauge.class, metric.getClass());
        Assert.assertEquals("s_pos", metric.getName());
        Assert.assertEquals(100, ((Gauge)metric).getValue(), 0.01);

        parsed =
metricsParser.parseMetric("s_pos:100|g\ns_pos:200|g");
        Assert.assertEquals(2, parsed.size());

        metric = parsed.get(0);
        Assert.assertEquals(Gauge.class, metric.getClass());
        Assert.assertEquals("s_pos", metric.getName());
        Assert.assertEquals(100, ((Gauge)metric).getValue(), 0.01);
    }
}
```

We then wrote a small server that we run on one of our laptops, connected to the WiFi Direct network used by the RC phone. It accepts the data over the network, parses the data into values and stores it into a database:

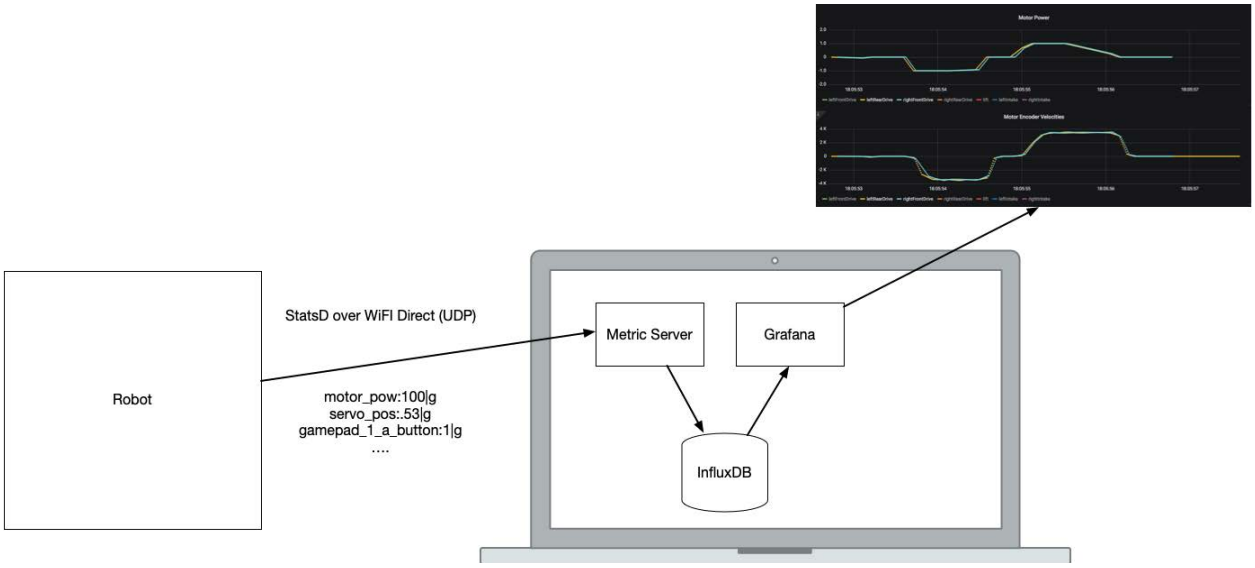
```
public class Ingest {
    public static void main(String[] args) throws Exception {
        InfluxDB influxDB = InfluxDBFactory.connect(
                "http://localhost:8086",
                "...", "...");
        influxDB.setDatabase("metrics");
        influxDB.enableBatch(BatchOptions.DEFAULTS);

        int port = 8126;

        final NioEventLoopGroup group = new NioEventLoopGroup();

        try {
```

```
            NetworkInterface ni = NetworkInterface.getByName("en1");
            Enumeration<InetAddress> addresses = ni.getInetAddresses();

            InetAddress localAddress = null;

            while (addresses.hasMoreElements()) {
                InetAddress address = addresses.nextElement();
                if (address instanceof Inet4Address){
                    localAddress = address;
                }
            }

            final Bootstrap b = new Bootstrap();

            b.group(group).channel(NioDatagramChannel.class)
                    .option(ChannelOption.SO_BROADCAST, true)
                    .handler(new ChannelInitializer<NioDatagramChannel>() {
                        @Override
                        public void initChannel(final NioDatagramChannel ch) throws
Exception {

                            ChannelPipeline p = ch.pipeline();
                            p.addLast(new IncomingPacketHandler(influxDB));
                        }
                    });

            // Bind and start to accept incoming connections.
            System.out.printf("waiting for messages...");
            b.bind(port).sync().channel().closeFuture().await();
        } finally {
            System.out.print("In Server Finally");
        }
    }
}
```

Code in our OpModes uses classes we wrote that look at the devices in the hardware map, and read values from them. Here is a screenshot of our project that shows what the structure of this code looks like, and what devices we can collect data from:



Our mentor helped us install Grafana, an open source graphing tool to show the data and InfluxDB to store the metrics data on the laptop that would receive the metrics data from the robot. We then created dashboards in Grafana for the data that was being sent from our robot:



As we added more metrics to the robot, we soon noticed an issue:

> *Today we expanded what our server intakes for data by adding the driver and operator buttons. We used two types of metrics in the code for it. One dealing with ranged inputs and one with on-off buttons.*

One problem we came across was that we took in a lot of data which slowed down the computer we were running the server on. We were taking in 350 KB a second, which is about equivalent to streaming a high def video.

One idea to reduce overload of the computer or server is to only log when the value has changed which would remove all the unchanged zero values that we intake. These unchanged zeros are most of our data intake and we don't need to log unchanged zeros for all inputs. With the method of only giving data when the value had changed is we would only log if we had a change in value and assume it has remained the same as the last one we received.

Engineering Notebook Entry – September 6, 2019 – Lauren

We did change the code to use our idea (called "InterestingValueMetricSource" in the screenshot above), and it reduced the amount of data that was being sent by over a factor of 10.

An example of putting this code to use is a robot issue that the team had leading up to our first league meet, and how real time performance data collected from the robot made it possible for the team to solve a hardware problem.

While practicing for our first league meet, our driver Taylor noticed that the robot would not strafe in the way that was expected , in tele-op and during autonomous. This is our third season with a Mecanum drive base, so while the software team was pretty confident the issue was not with the code. Even so, we could not rule that out, because we've built robots before that did not have this issue from a mechanical point of view too.

We put together a quick tele-op that used the kinematics from our regular tele-op that would only allow inputs for strafing to be sent to the drive base.

We then put the robot up on some gold minerals left over from Rover Ruckus so that the wheels could spin freely. We used the controls to send different power levels to the drive base and collected the power levels and velocities of each drive motor over time. This is one of the resultant charts:

setPower() values for both leftRear and rightFront is the same over time

The rightFront motor responds to power changes more slowly, and does not reach same speeds as the leftRear motor over time.

The two series in each chart are motors on opposite corners of the robot that should be turning in the same direction, with the same velocity resulting in a strafe that does not drift.

The collected data shows that for the exact same power levels, there is one motor on the robot that both lags in velocity change to power input and never reaches the same velocity as the motor that is diagonally opposite. Based on this data, Lauren inspected the drive train, checking chain tension, ensuring the wheel and sprocket were not binding on anything and then decided to replace the motor and test again.

After the motor was replaced, Lauren and Calvin re-ran the test, and this is the data that was collected, which looks more like what is expected:

These motors were directional pairs (should have the same direction and velocity). Because of that, if the inputs are near identical, the outputs should be as well.

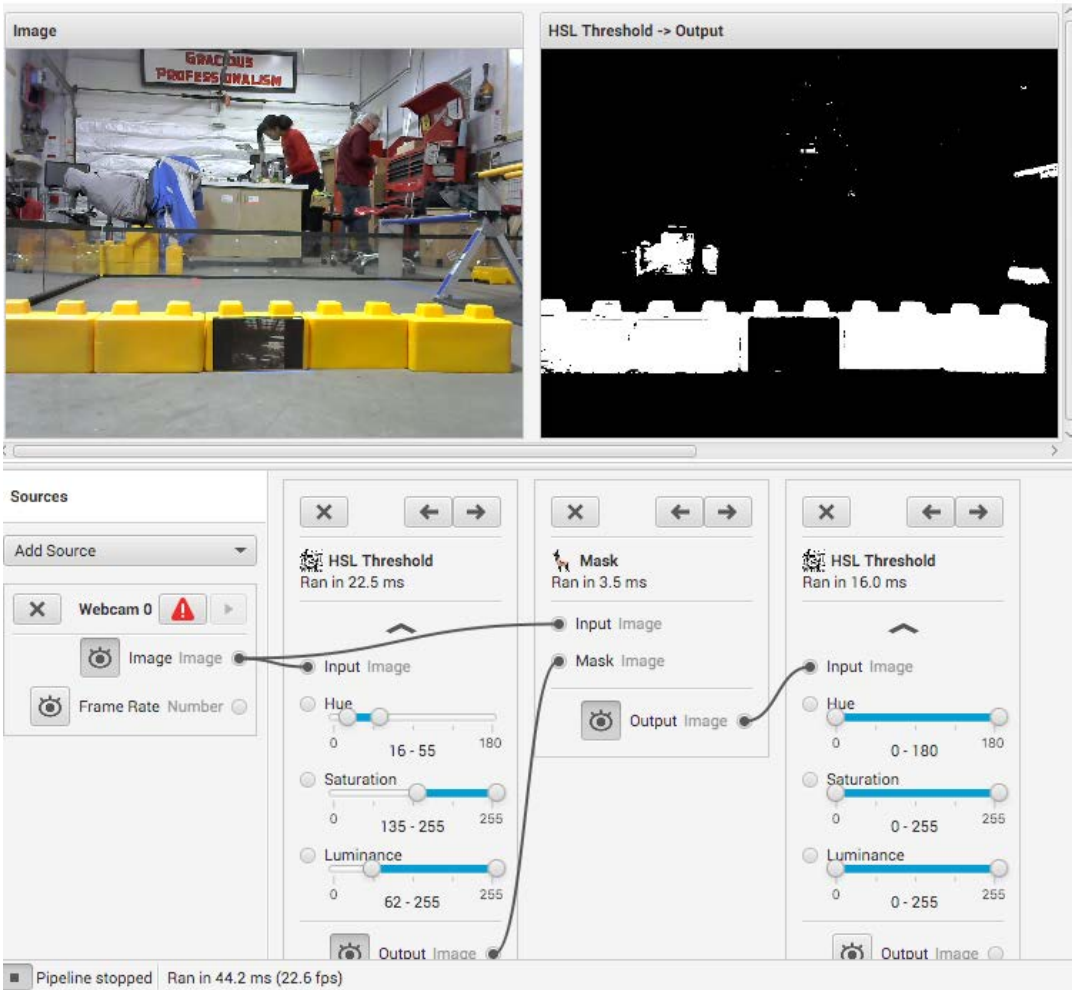
Using OpenCV Instead of Tensorflow or Vuforia

We used a program called GRiP to understand how the OpenCV pipeline works, and do some experiments with identifying the Skystone. We found that

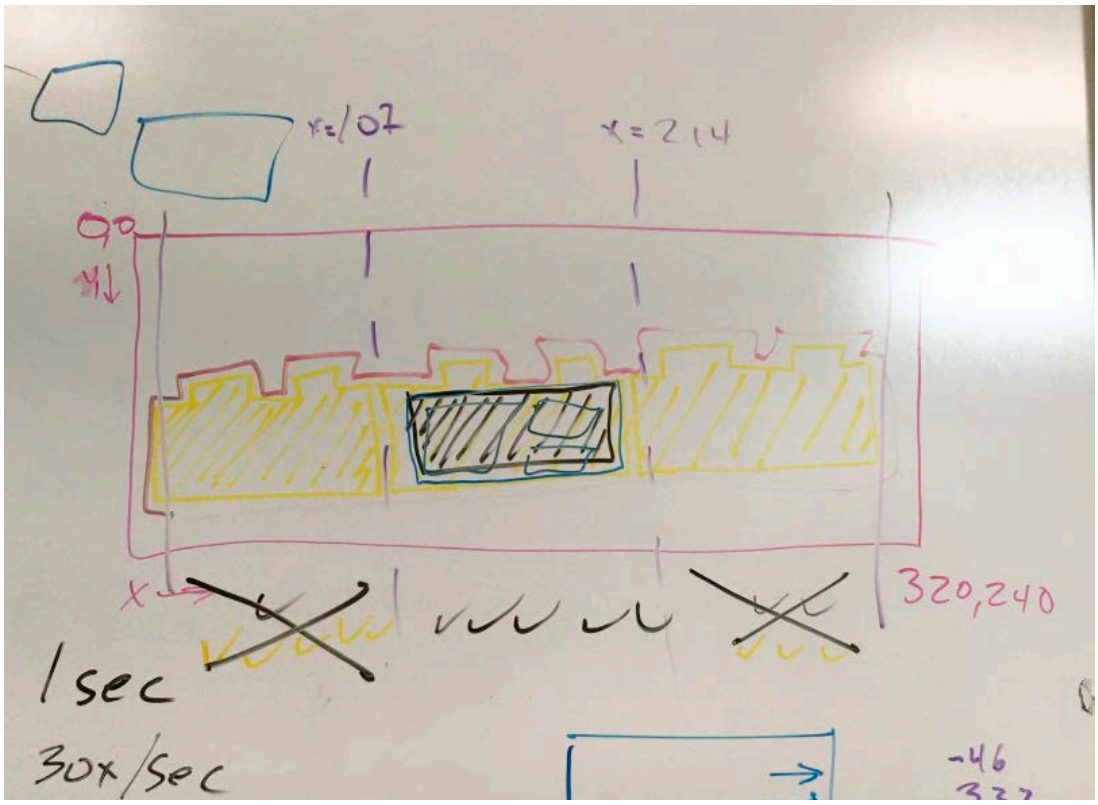
- The range for hue needs to be roughly 16 -55 to pick up the blocks ONLY
    - this is because having the starting value lower would cause the image to

pick up red as well. In the RGB scheme yellow includes red, so you want to see some but not enough that you also see pure red when you want to see yellow

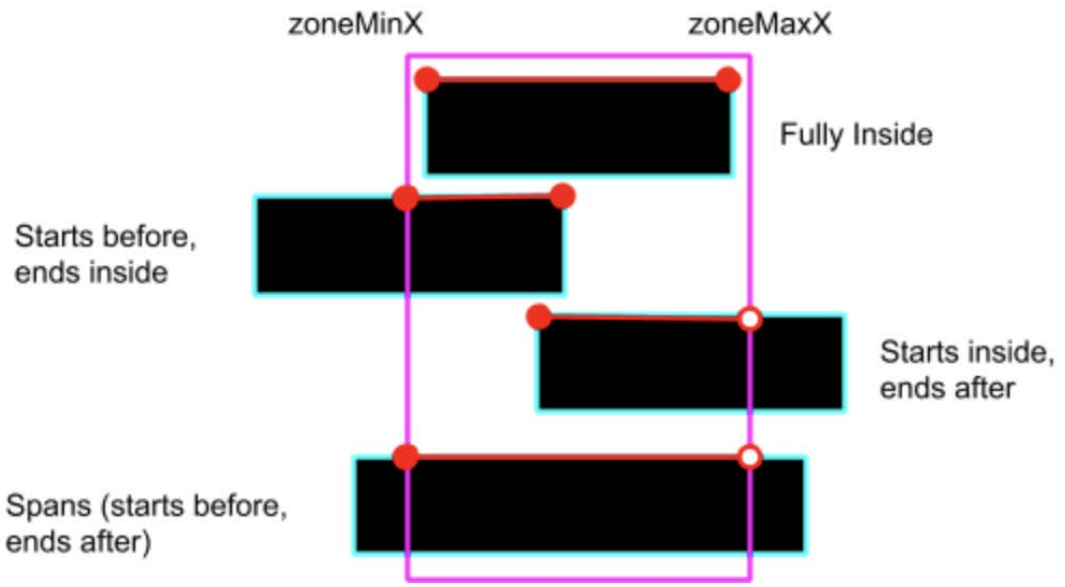
- The camera needs to be about 10 inches above the field and a hood may also be needed to block out the stack of blocks that will be by the human player.
    - possible mask could be a image with a blank background that we import that has the top masked off. we would have to test this. it would be added as a source when we take inputs.
- Luminance is 62-255
- Saturation is 135- 255



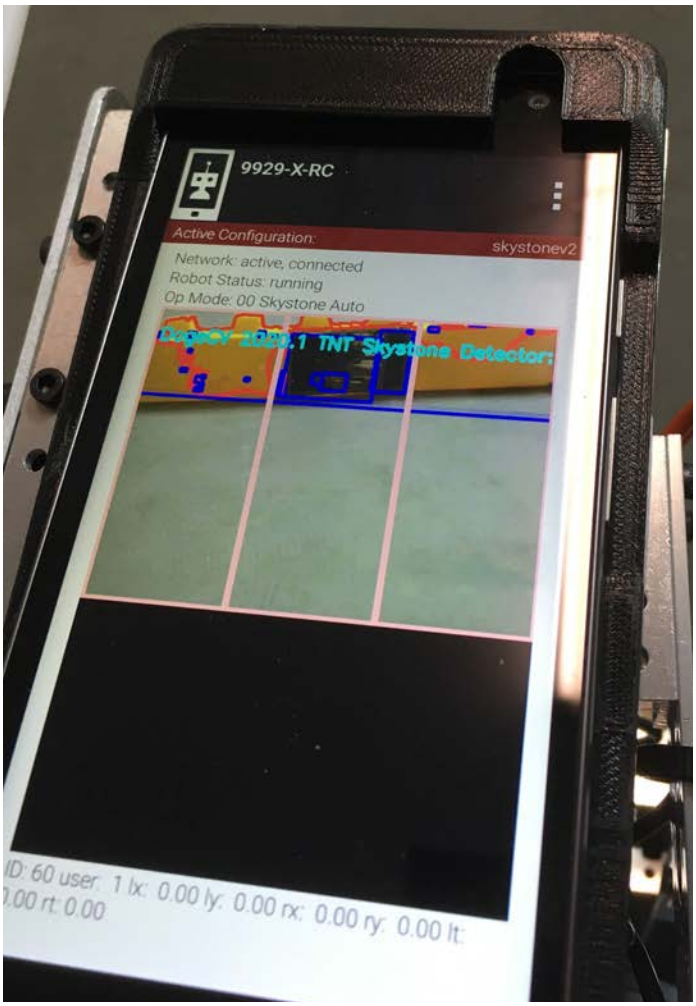
Our final system finds areas of black in the input from an on robot camera. We then filter out areas too small to be the image on the skystone. Next, we split the image into three sections of left, right, and center. From there we calculate which segment has the most black. This zone is the zone with the skystone in the eyes of the robot and it's code.



Our original idea on the whiteboard



How we allocate dark contour areas to a detection zone



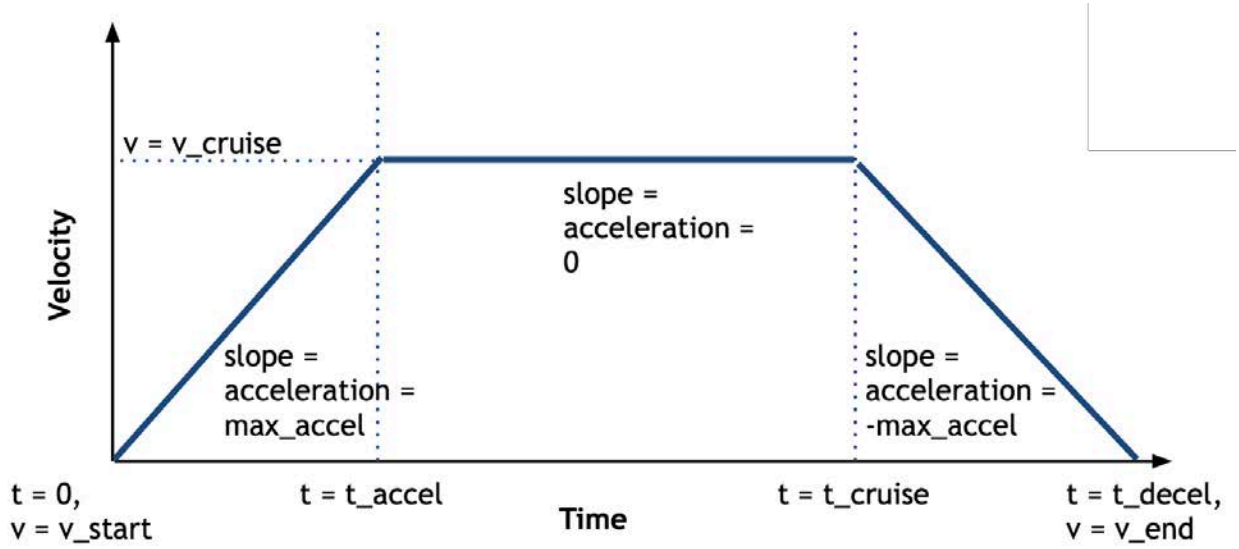
Our OpenCV Detection Pipeline Running during Autonomous

Usage of Motion Profiles During Autonomous

This is our second season using motion profiles during autonomous. Motion profiling uses trajectories, which are a path plus where we expect the robot to be on said path at any given time, to decide how fast and where to go. Roadrunner, the motion profile system we use, uses both feedback - where our sensors (encoders in this case) think they are - and feed forward - where we expect to be on the trajectory at that moment. The feed forward is determined by an equation using maximum acceleration and velocity measurements measured with our robot at the beginning of the season.

This method allows the robot to run with smooth, accurate motion during autonomous as the speed changes based on how far off the feedback is from the feed forward rather than tracking a single end goal like a traditional PID would do.

Before the motion even begins, the control system pre-plans a "motion profile" that describes the robot's position and velocity over time, accounting for acceleration, “cruise” and deceleration.



(Images from FRC#254's Motion Planning and Control for FRC presentation)

The motion profile "plan" starts with calculating the acceleration/deceleration ramps (segments #1 and #3) using the earlier measured maximum velocities and accelerations for our robot:

$$\Delta T = v_{start} - \frac{|v_{start} - v_{max}|}{a_{max}}$$

Roadrunner can compute where the robot should be, and the velocity at any point in time during the acceleration and deceleration states (v is 0 during acceleration, and max_velocity at the start of deceleration):

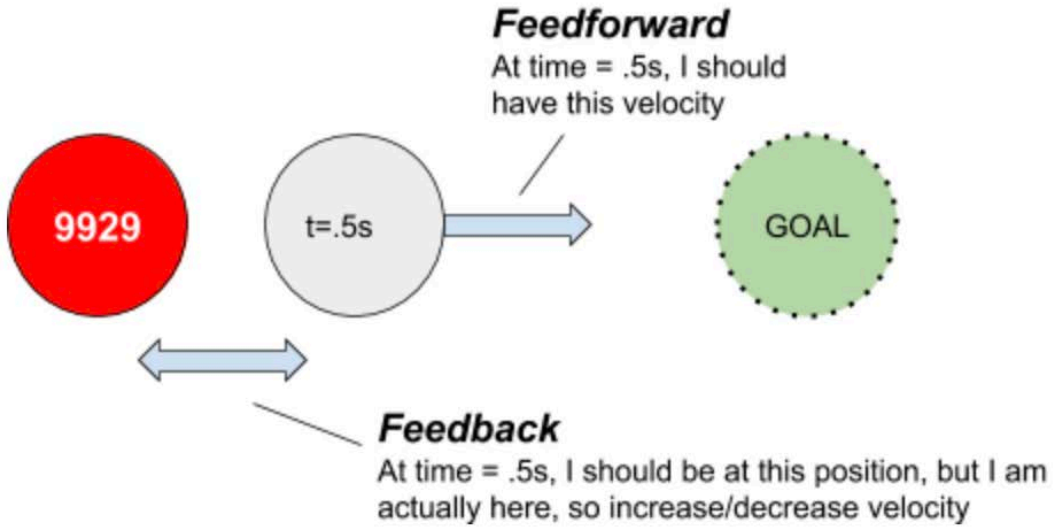
$$position(t) = vt + \frac{1}{2}at^2$$

$$velocity(t) = v + at$$

Roadrunner then uses the acceleration/deceleration calculations to determine how far/long to "run" at max velocity (segment #2) by using the value of delta-t from computing the "ramps" and using that amount of time with the above two formulas to calculate how far the robot will have moved during the acceleration/deceleration segments. It then subtractings that distance from the goal distance, which gives the distance and time that the robot should "cruise" at maximum velocity.

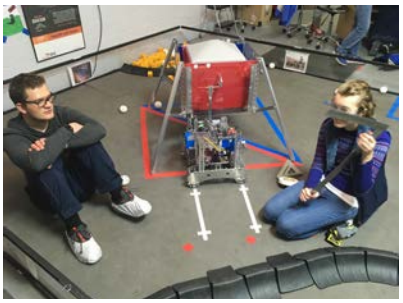
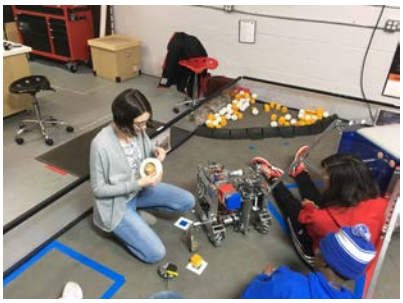
All of the data about the amount of time it takes to accelerate or decelerate, and the amount of time the robot will be "cruising" at a fixed velocity is then used with the above formulas in RoadRunner's "MotionState" class to calculate feed-forward values for the expected velocity and position at any point in time during the movement.

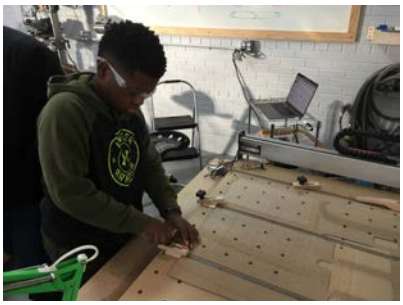
Instead of instantly trying to reach the goal (in most FTC cases, the goal position is measured by the encoders on the motors driving the wheels), the PIDF controller tracks the motion profile and the computed trajectory (where the robot should be at a given point in time) instead as a feedforward term and odometry (encoders) for position as feedback, leading to more consistent, smoother motion:



Engineering

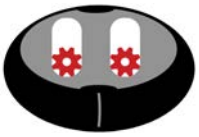






FTC#9929 Tech Ninja Team Engineering Notebook 71

	**Friday, April 5, 2019**	07:00 PM-9:00 PM

Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Taylor

Entries

1. A New Game for MSI Outreach

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Last practice, we designed a game for people to participate in for the Museum of Science and Industry's Robot Block Party. We had agreed on the title "Rubbish Roundup". In the game, the player would control an old robot of ours, nicknamed Skittlebot, to round up pieces of gold and silver "space junk", as well as bringing "satellites" into their corresponding docking stations. The movable components were all recycled game elements from past seasons, and the only thing we needed to manufacture was our docking stations.

Today we manufactured our docking stations out of plywood and got everything put together. We laid out our game on our 8'x8' field, and we started testing it. Everything seemed to work well with Skittlebot, and all we needed to do was change around the scores for completing tasks. The final ruleset is shown below, along with a picture of the game.

Orbit **- square area defined by red or blue tape**

Satellites **- large red and blue spherical shaped objects**

Docking Stations **- black boxes marked with red or blue tape**

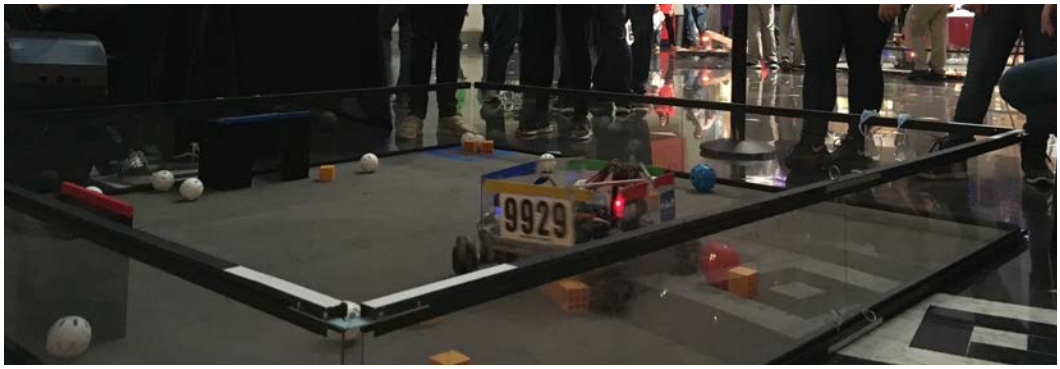
Game Play

Scoring ***Space Junk*** -

- Each piece of ***Silver Space Junk*** scored into the red ***Orbit*** earns five (5) points
- Each piece of ***Gold Space Junk*** scored into the blue ***Orbit*** earns five (5) points
- Each piece of ***Space Junk*** scored into the incorrect ***Orbit*** will not earn any points
- Each piece of ***Space Junk*** scored into the "mix" red and blue ***Orbit*** earns two (2) points

Scoring ***Satellites*** -

- ***Satellites*** may be scored in the last thirty (30) seconds of the game.
- Each ***Satellite*** scored into the correct ***Docking Station*** earns seventy-five (75) points.
- ***Satellites*** scored into the incorrect ***Docking Station*** will not earn any points



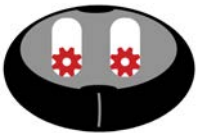
[Liam]

2. MSI Robot Block Party Plan

Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

Additionally to the game, we worked on our plan for the block party on Sunday. We ended up deciding on having last season's robot, Zaphod Beeblebot, on full display on our turntable on top of the folding table we were given. We put our poster board with a brief overview of the robot to the side, as well as several pictures and informational sheets. We would keep the spare batteries and chargers on the ground so that we could switch out the batteries on Skittlebot or Zaphod at any time without getting in the way of anything we wanted to display.

	[Liam]

	**FRIDAY, MAY 10, 2019**	07:30 PM-9:00 PM
Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Logan		

Entries

1. Teamwork and Organization Discussion

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We discussed the processes the team should have during competition. One idea was that we have a team chant right before a match to help the drive team stay confident and focused. Then, we sorted the ideas and categorize them into three categories: Physical, Mental, and Teamwork.

We decided to develop this into a formal drive team practice plan and a Crew Resource Management plan:

Drive Practice Plan

Spatial Awareness

- Drive the robot through a maze without touching any walls
- Drive the robot into a box, just barely big enough for the robot, without hitting the sides
- Accuracy/precision
  - "Parallel parking"
  - Drive while picking objects up, without losing speed
  - Drive the robot through a route moving at, or close to, the top speed (time trial)
- Have rewards for the winners (Dairy Queen, soda, chips, etc. as prizes)

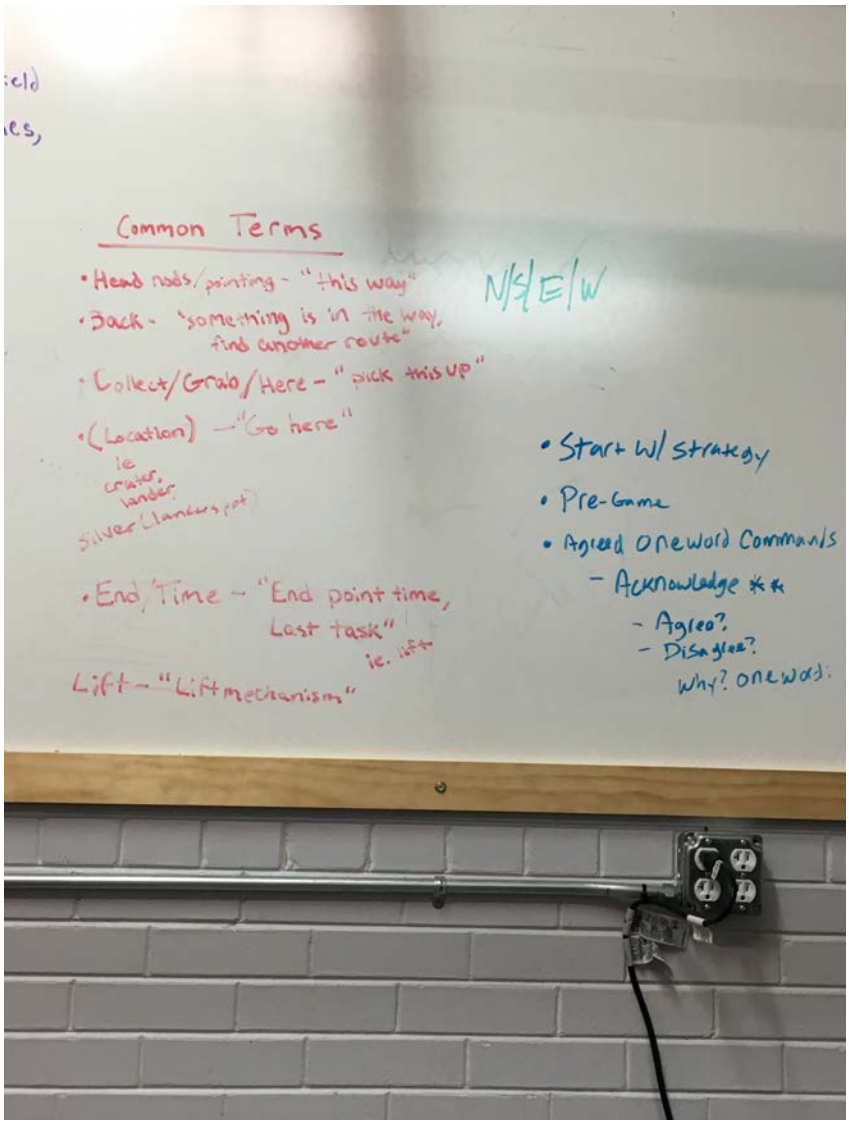
Crew Resource Management

- Focus on the big picture
- Use prior experience
- Pre-match checklist

Communicating for the Drive Team

- Head nods/pointing- "this way"

	● Back- “something’s in the way, find another route ● Collect/grab/here- ”pick this up” ● (Location)- “go here” ● End/time- “end point time, last task” ● Lift- “lift mechanism” ● Start with strategy ● Pre-game ● Agreed one word commands ○ acknowledge** ○ Agree? ○ Disagree? ○ Why? One word **[Logan]**



2. Researching needs

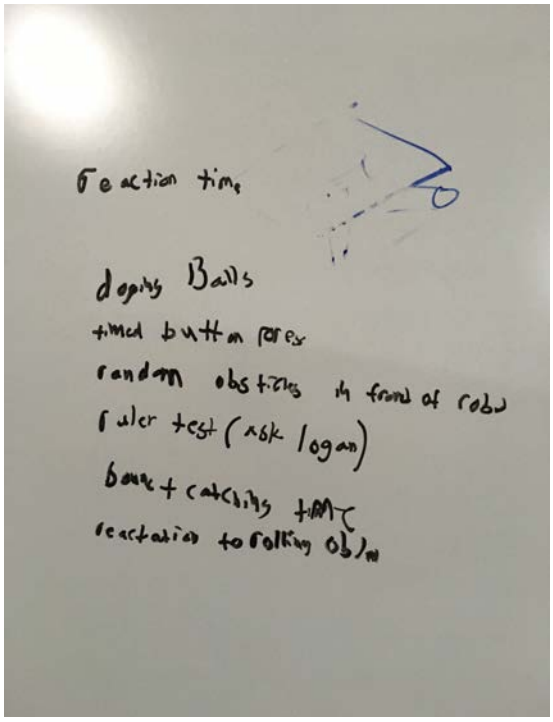
Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

We split up in groups to look into the details and try to figure out how we would test things like: Reaction timing,group greeting, strategy talk, auto set up, or troubleshooting. At the end, we came together and figured out what ideas we would use for the up-coming season.
[Logan]

- Strategy talk
- Auto set up
- Team cheer/visualization
  A)Who are we
  T)Tech Ninjas

A) what are we going to do
T) Drive well
A) what are going to have
T) an awesome match

- “It is like us to do well” repeats if nervous
- Match
- End signal
  - TBD



	Friday, July 12, 2019	07:30 PM-9:00 PM
Contributors: Cal, Ernest, Hannah, Kaylin, Lauren, Liam, Logan		

Entries

1. Prepared software for Skystone/SDK 5.0

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Official start to programming work on Skystone. We collectively decided to import the previous years' code into the SDK version 5.0. We made sure it could function (by having a test run of last year's autonomous) in order to have a strong foundation for this year. However, as no details have been announced and as no robot has been built, we are unable to go any further than that.
[Calvin]

2.

Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

We split up to look into the details and tried to figure out how we would test things like: Reaction timing, or troubleshooting. We also discussed strategy and planning for driver practice. The entire team wanted to keep the basic movement controls the same. The drivers also expressed a desire to have a "half-speed" setting on the robot to help prevent any collisions or accidents. The idea of an obstacle course was brought up, but we are not sure how helpful that will be, considering the field is not announced yet.
[Hannah]

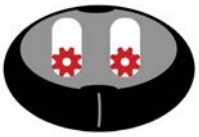
	Sunday, July 28, 2019	05:00 PM-8:30 PM
Contributors: Liam, Logan		

Entries

1. Calibrating CNC Machine	Our CNC machine, after a season of use, was becoming somewhat unreliable. The wasteboard wasn't completely flat anymore, and the spindle wasn't quite perpendicular to where the wasteboard should have been in all planes, meaning that our cuts were coming out inconsistent and sometimes jagged. When we first started work on the machine, we flattened the wasteboard by drilling a small fraction of an inch into it around the whole board. However, when we did this, the cut lines were visible and tangible, confirming our suspicion that the spindle was misaligned. To fix this, we built a jig out of a block of wood and one hole on either end (shown below) that would help us retram the machine. By putting the drill bit on the machine through one hole in the wood and another loose drill bit through the other hole, we were able to assess just how off-level the machine was. 

As shown in the pictures above, the bit on the end of the wood was hitting the wasteboard at some points and being held above it at others. This was further evidence of the spindle being off-axis, leaning a little towards the front and left. To fix this, we had to check the x- and y-axis belts. We took the beams holding the belts, unscrewed the bolts on the ends a small amount, and realigned them one by one. Eventually, we were able to relevel the spindle, and we flattened the wasteboard with it. Now the machine is ready for use in the upcoming season, and we shouldn't have to do something like this for about a year.

[Liam]

	**August 16, 2019**	7:30 PM-9:00PM
Contributors: Calvin, Kaylin, Kyla, Lauren, Logan, Jermey, Hannah		

Entries

1.Telemetry server - Gauge builder

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	**Fabricate**

Today we worked on a gauge builder for our server that we plan to use to track telemetrics of the robot at practice. We are building this server to help better track outputs and inputs involved in the robot.

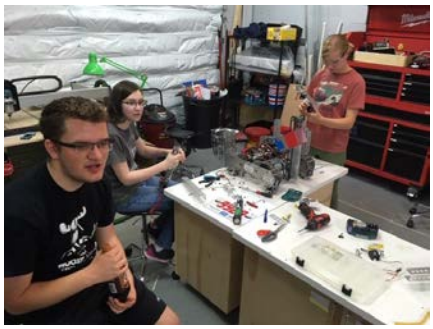
Today we worked on a type of metric called a gauge. A gauge is a metric with a timestamp. We also learned how to use Foo which is used as a filler to put in fake inputs when there are no real ones.We used Foo to complete a test run of our server to make sure it worked.

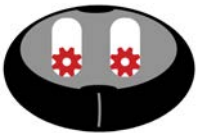
[Lauren, Kyla]

2. Taking Apart Zaphod

Identify	Brainstorm	Select	Prototype	**Evaluate**	Design	**Fabricate**

This practice, the build team started to disassemble our robot from last season. We feel it is important to do this to better evaluate our mechanisms from last season and make sure we have enough parts to start our new season. Zaphod Beeblebot. also had many expensive parts. We wanted to make sure we removed these parts with no damage for future use.

	 **[Kaylin]**
3. Measuring Velocity	Today I programmed a method to track the velocity of each individual motor on the robot, will greatly help us plan our autonomous based on time. **[Calvin]**

	**SEPTEMBER 6, 2019**	7:30 PM-9:00PM
Contributors: Calvin, Kaylin, Kyla, Lauren, Logan, Jermey, Hannah		

Entries

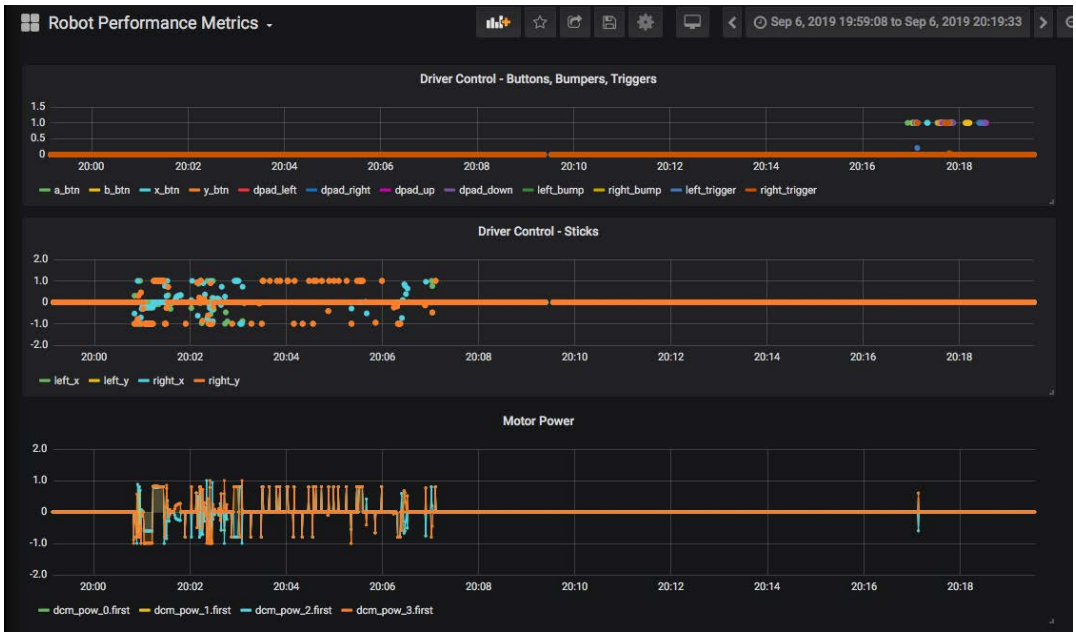
1.Telemetry server - Gamepads

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	**Fabricate**

Today we expanded what our server intakes for data by adding the driver and operator buttons. We used two types of metrics in the code for it. One dealing with ranged inputs and one with on-off buttons.

One problem we came across was that we took in a lot of data which slowed down the computer we were running the server on. We were taking in 350 KB a second, which is about equivalent to streaming a high def video.

One idea to reduce overload of the computer or server is to only log when the value has changed which would remove all the unchanged zero values that we intake. These unchanged zeros are most of our data intake and we don't need to log unchanged zeros for all inputs. With the method of only giving data when the value had changed is we would only log if we had a change in value and assume it has remained the same as the last one we received.


Robot Performance Metrics
Sep 6, 2019 19:59:08 to Sep 6, 2019 20:19:33
Driver Control - Buttons, Bumpers, Triggers
1.5
1.0
0.5
0
20:00
20:02
20:04
20:06
20:08
20:10
20:12
20:14
20:16
20:18
a_btn
b_btn
x_btn
y_btn
dpad_left
dpad_right
dpad_up
dpad_down
left_bump
right_bump
left_trigger
right_trigger
Driver Control - Sticks
2.0
1.0
0
-1.0
-2.0
left_x
left_y
right_x
right_y
Motor Power
dcm_pow_0.first
dcm_pow_1.first
dcm_pow_2.first
dcm_pow_3.first

[Lauren]

	SEPTEMBER 14, 2019	8AM-8PM
Contributors: Hannah, Occie, Taylor, Kyla, Ernest, Liam, Logan, Lauren, Kaylin, Malcolm		

Entries

Autonomous Strategy

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

We discussed tasks and strategies for autonomous. We agreed on the following priorities:

1. Teach the robot to park under the skybridge during the last few seconds of autonomous. This is worth 5 points, and should be the easiest task.
2. Teach the robot to identify Skystones, pick them up, and move them to the foundation. This is worth 28 points total (10 for each skystone delivered, and 4 for each skystone placed on foundation). This is not the easiest autonomous task, but the basic "robot skills" needed are needed for a successful season. To accomplish this we will need:
    a. Computer Vision or Color Sensors for detecting skystones
    b. Autonomous Path Planning and Programming
    c. Mechanisms for picking up and placing skystones
3. Teach the robot to move the foundation into the building zone. This is worth 10 points in Autonomous, but also enables an additional 15 points in endgame.
4. Teach the robot to pick up additional regular stones and move them to the foundation after it places the skystones, if there is time. Each stone is worth 6 points (2 for delivery, and 4 for placing).

Our analysis of these tasks is below:

Tasks	Time estimate	Probability of	points

		completion	
Repositioning	7 sec	100%	10
Skystones	5 s/per	100%	20 (only if initial 2)
Stones	5 s/per	100%	2
Placing	2 s/per	75%- VARIABLE	4
Navigation	3 sec	100%	5

Based on the above estimates, one robot could move 2 skystones and one additional stone in Autonomous *if* it also has to reposition the foundation.

Questions:

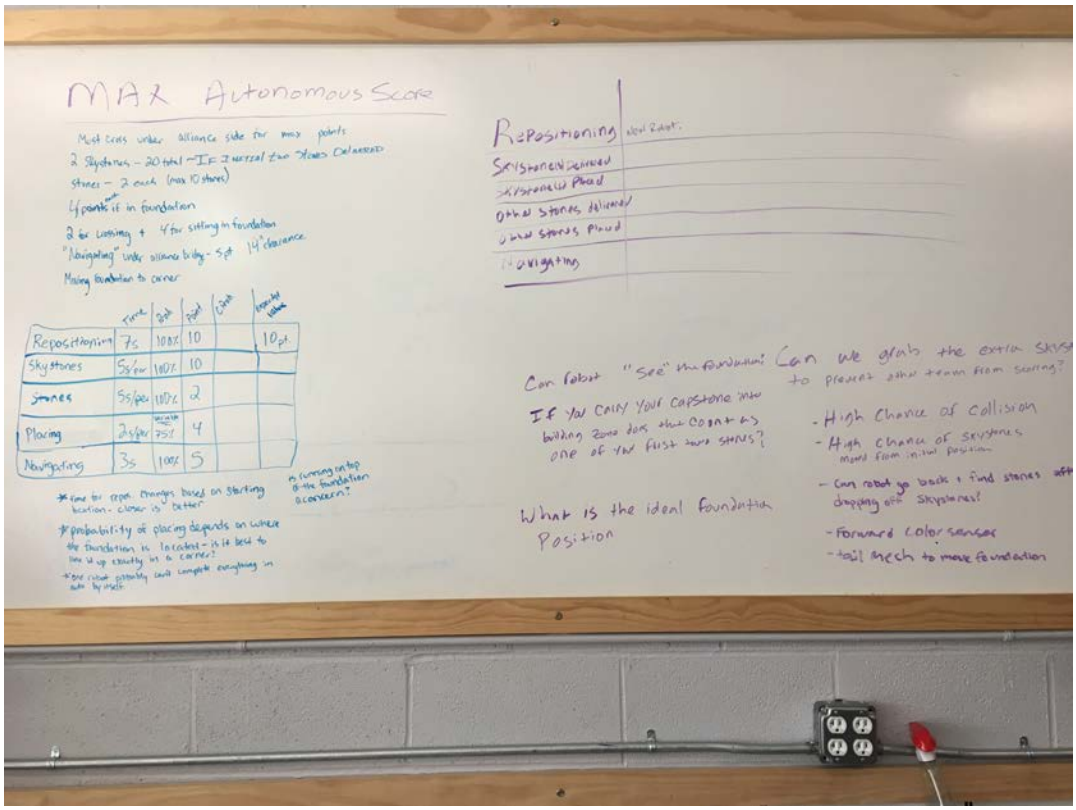
- Can the robot "see" the foundation using computer vision?
    - If the foundation gets moved, by an alliance partner or by an opponent, our autonomous program will need to adjust if we want to place the skystones / stones.
- If you carry the capstone into the building zone does it count as an initial stone?
    - We were unable to find a clear answer to this in the rules. For now we would assume it does count against us.
- What is the ideal foundation position?
    - If we put it all the way in the corner, it may be more stable, *but* in endgame we'll have to pull it farther.
- How much of a concern is tipping the foundation? How plausible is it to accidentally drive on top and tip it over?

Other Notes:

- We will receive a major penalty if we interfere with the other alliance's stones or skystones in the quarry during autonomous.
- Time for repositioning the foundation changes based on starting location- closer is better.
    - But it may be impossible to see the skystone positions from near the foundation
- Probability of placing stones depends on the position of the foundation.
- One robot most likely cannot complete all tasks by itself.
- High chance of robot collision during autonomous

- Both robots may be heading to pick up stones
- Both robots could be trying to place stones in the foundation at the same time
- We could have an autonomous path that is not compatible with their routes

- High chance of stones being moved from the initial position, making them hard to grab.
  - We need to pick up the stones with as little interference as possible
  - We need to be able to pick up stones that are knocked over (though maybe not in autonomous)



Teleop Strategy

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

We looked at tasks and strategies for TeleOp. The task priorities are:

- Rearrange already delivered stones into tower in order to make a

secure skyscraper to decrease the possibility of it toppling.

- Collect, deliver, and place new stones to increase the height of the tower.
- If other alliance drops a stone, grab it instead of one from the depot (it saves time)
- There is no need to alternate alignment at every level. This is only a concern at higher levels of blocks because it increases the stability of the tower. Doing this at every step is unnecessary and may waste time.
- Going under the alliance specific skybridge is a must, otherwise we miss out on potential points.

Our analysis of potential scoring scenarios is below:

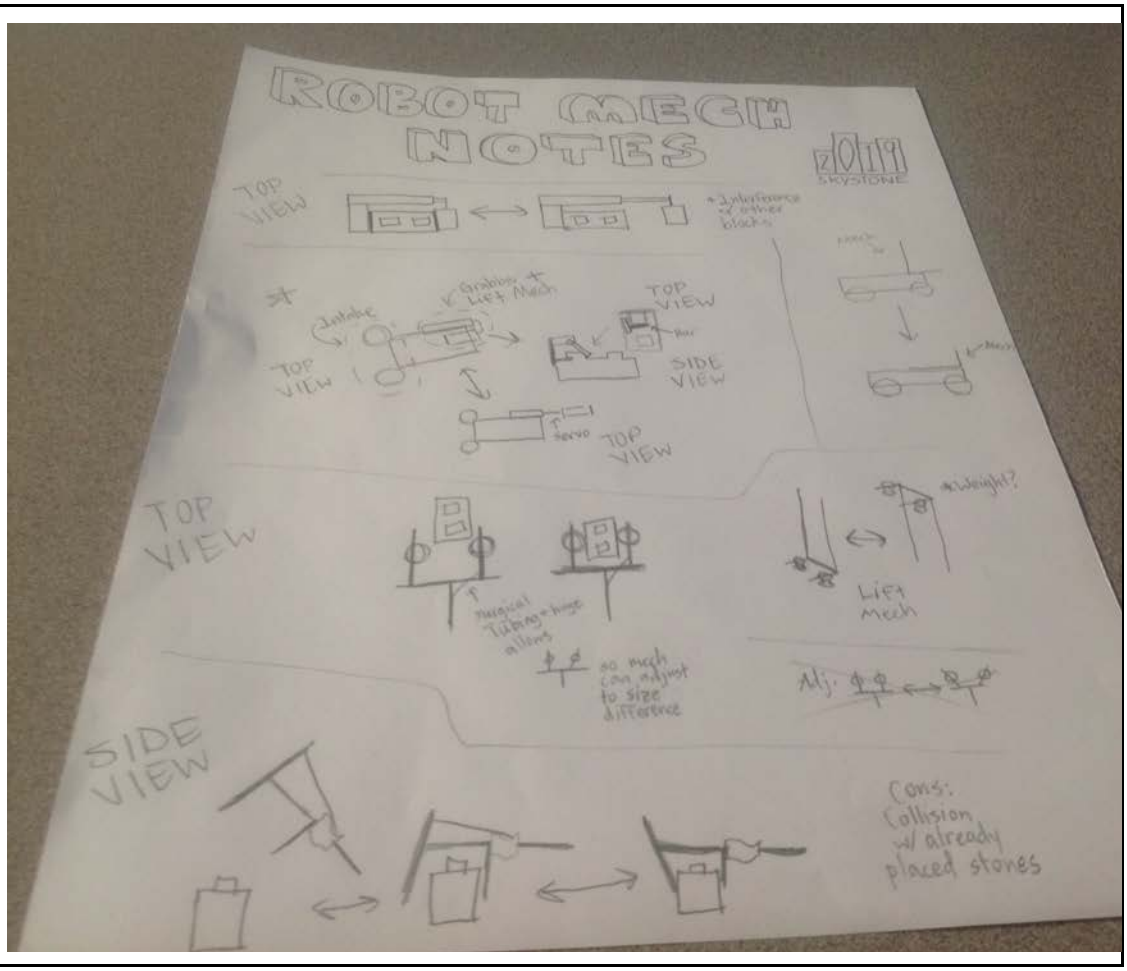
Scenario 1 (one block skyscraper)	Points:
Delivered stones	15
Placed stones	15
Levels	30
Total points	60

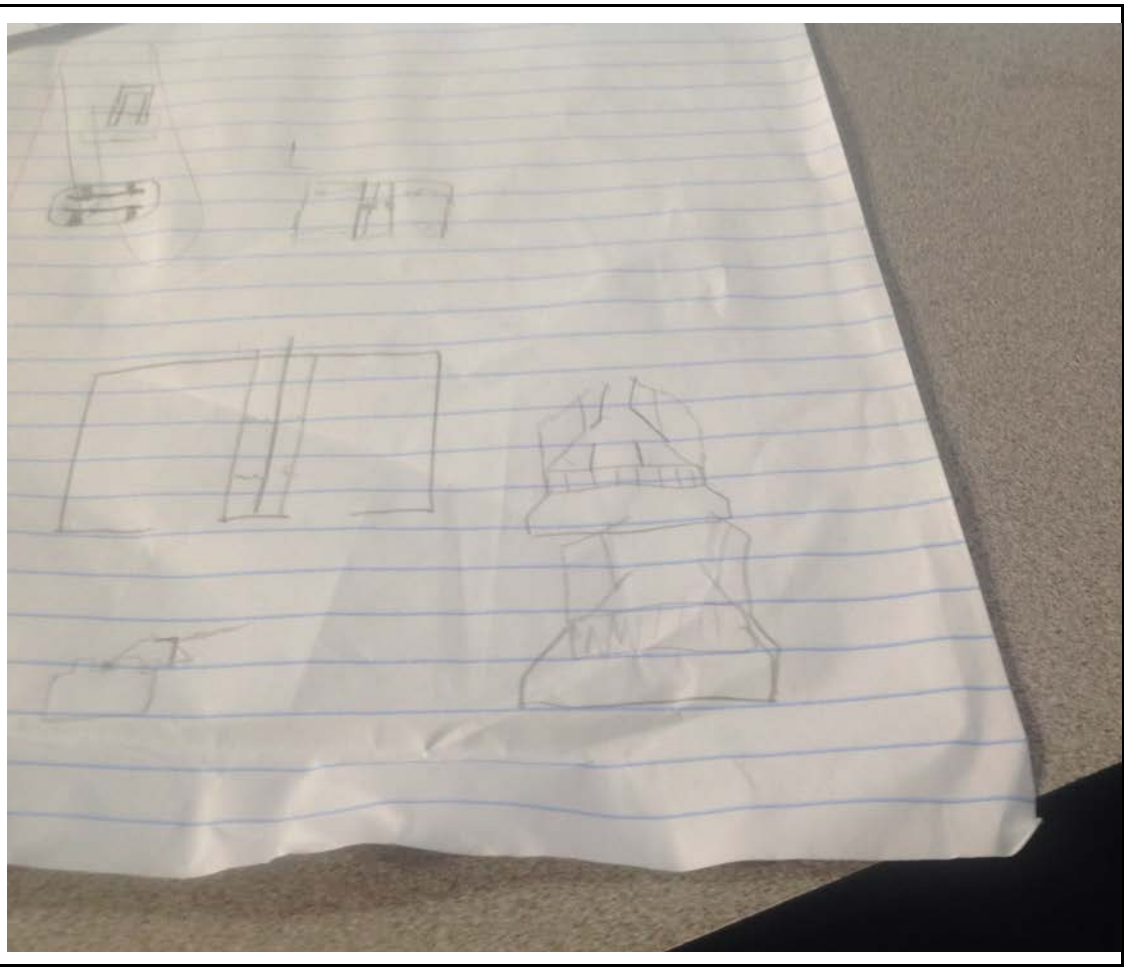
We determined that the maximum stable height is 15 blocks.

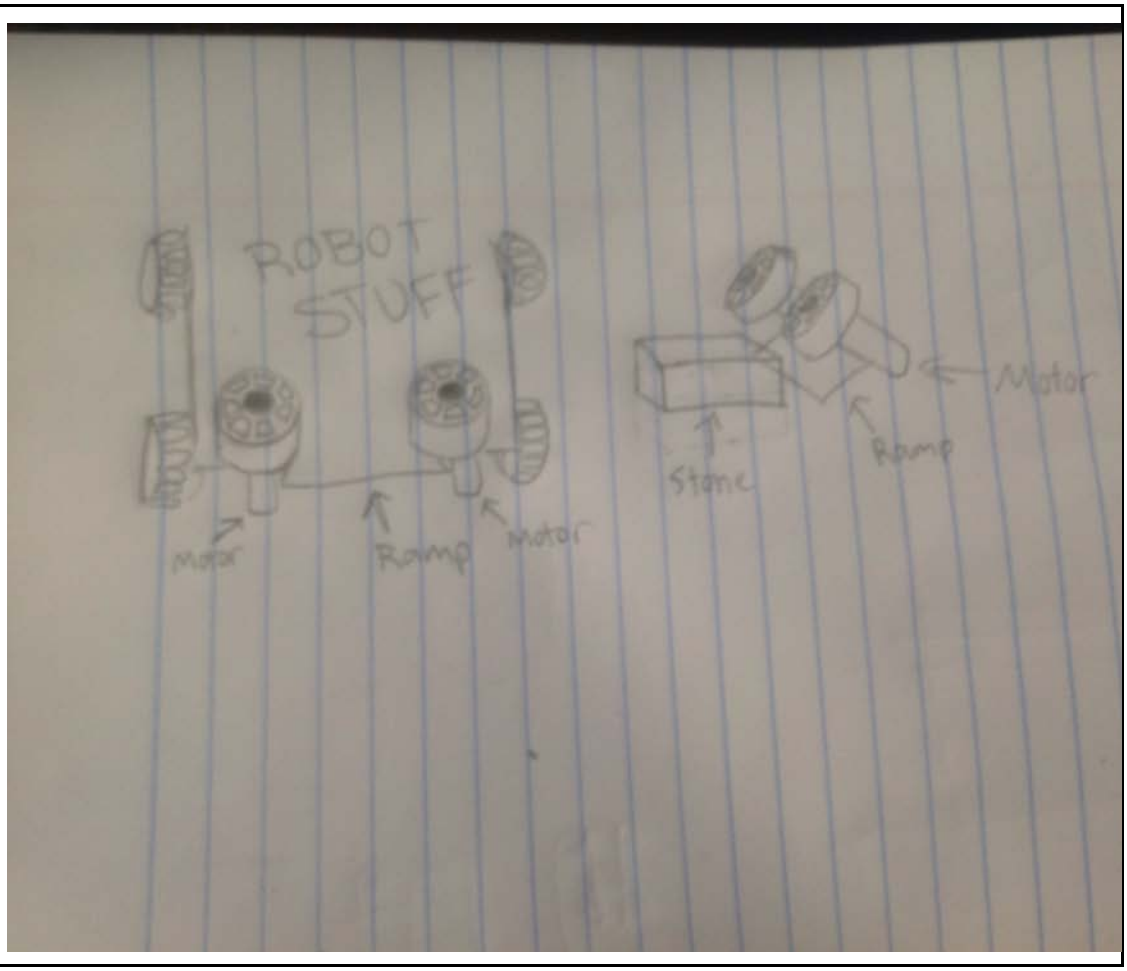
Scenario 2 (two blocks per level, changing directions)	Points:
Delivered stones	30
Placed stones	30
Levels: 15	30
Total points	90

With this design, we still top out at 15 levels, because we run out of stones. But it is much more stable.

	
Tasks and Rankings	
Intake strategies	We chose a wheel intake design because it is effective and proven to be fast by previous seasons. The reason we chose this design over other designs is because of the minimal contact with other non targeted blocks


ROBOT MECH NOTES
2019
SKYSTONE
TOP VIEW
TOP VIEW
SIDE VIEW
TOP VIEW
servo
TOP VIEW
Lift Mech
TOP VIEW
SIDE VIEW
Cons:
Collision w/ already placed stones


ROBOT
STUFF
Motor
Ramp
Motor
Stand
Ramp
Motor

	September 15, 2019	8AM-8PM
Contributors: Hannah, Calvin, Kaylin, Occie, Lauren, Liam, Logan, Taylor, Ernest		

Entries

Foundation Moving Mechanism

Identify	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate

We discussed ideas and designed prototypes for the foundation moving mechanism. In a smaller subgroup, we had a discussion about potential ideas for the mechanism and narrowed it down to the best ideas. The initial mechanism tested consisted of a servo attached to a small actobotics plate. The servo would rotate to move the plate, so it would hook over the side of the foundation. Following some basic tests of the mechanism, the team made some important observations:

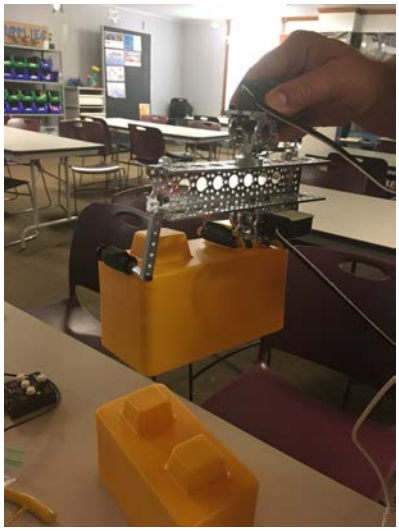
- The mechanism would often slip off of the plastic. This was solved through the use of rubber bands stretched around the actobotics plate. As a result, the higher friction of the rubber lessened the probability of the actobotics “hook” slipping.
- The robot would need to get incredibly close to the foundation for it to work. Even a half inch of position inaccuracy was enough to stop the mechanism from working properly.
- The actobotics plate had a small length across and would often cause the foundation to be pulled at an angle instead of in a straight line.

The team decided that this “turning grabber hook” prototype was good conceptually, but needed changes in its execution. Coach Beezie suggested that we make the “hook” wider in order to cover more of the foundation. This was accomplished by connecting small L-shaped plates to each other via standoffs. This was then attached to the servo, held about an inch and a half away by a standoff. This prevented the servo from being too close to the mechanism and causing the same alignment issues as in the previous prototype.

The next step of the mechanism was to upgrade it to something more

<table>
<tr><td></td><td>secure. We machined new hooks out of old materials from previous season's game parts. These were attached to clamps on an axle in addition to a gear. The gear is connected to a servo in order to turn it and hook on the foundation.
This redesign is more secure and will reduce the probability of it breaking.
[Hannah]</td></tr>
<tr><td>TensorFlow</td><td><table><tr><td>Identity</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table>
After carefully analyzing the game journal, I realized that the patterns could be easily analyzed by observing 3 of the 6 blocks on the sides. Due to this realization, I am making plans to program tensorflow and build the camera in the right place to only analyze the 3 blocks closest to us.
[Calvin]</td></tr>
<tr><td>Delivery mech</td><td><table><tr><td>Identity</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table>
Today we worked on a prototype for the delivery mechanism for the stones used in this year's game. The current version of the prototype has one grip servo that puts pressure on one of the nubs of the stone. We did this because we wanted the sides of the stone and the bottom of the stone to be open for more options for stacking. We originally had two servos for this part but found the forces at play would cause the servos to be pushed open.

We also added a servo on the top to allow turning of the stone to complete a bricklayers' pattern for building in a more efficient manner. The 90 degree turn allowed by this servo will help with that efficiency. A lot of this design is focused on mobility and being lightweight</td></tr>
</table>

	  We plan to add a part to this mechanism that will allow for back and forth motion from the robot. This addition will allow us to reach farther into the foundation (which may as we look at strategy will be more advantageous). The back and forth mechanism segment will be what attaches to the lift we will use to lift the stone to the proper height. **[Lauren, Kaylin]**

	September 20, 2019	7PM-9PM

Contributors: Habtamu, Hannah, Calvin, Kaylin, Kyla, Lauren, Logan, Taylor,

Entries

Refactor tele-op to components

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we worked on a system to create an object that we are at the moment called “robot”. This object works to the side of OpMode and it’s functions will be bridged over to OpMode when needed. This allows an easier means of reusing functions, like our mecanum drive, from OpMode to Opmode. Instead of implementing the code of said functionality into each year’s code we can simply ask for the robot object. This allows us to call functionalities that we reuse from year to year to be implemented with less work, and less possibility for errors.

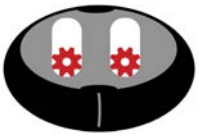
We also wrote a test for the “robot” object mecanum drive. This was to ensure that the mecanum drive was still functioning how we expected it to after we moved it into the “robot” object.

[Lauren, Hannah, Calvin]

Delivery Mech

Identity	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate

	We added the prototype delivery mech to a chain lift in order to explore how the lift interacted with the prototype delivery mechanism. This allowed us to envision how the mechanism might sit on our robot this year. This brought up questions about alignment and positioning on robot. Furthermore, it highlighted issues with the chain lift we used in the past, such as space, and weight **[Kaylin]**

	**September 22, 2019**	4PM-6PM
Contributors: Calvin, Ernest, Hannah, Kaylin, Kyla, Lauren, Logan, Malcom, Occie, Taylor		

Entries

Refactor tele-op to components

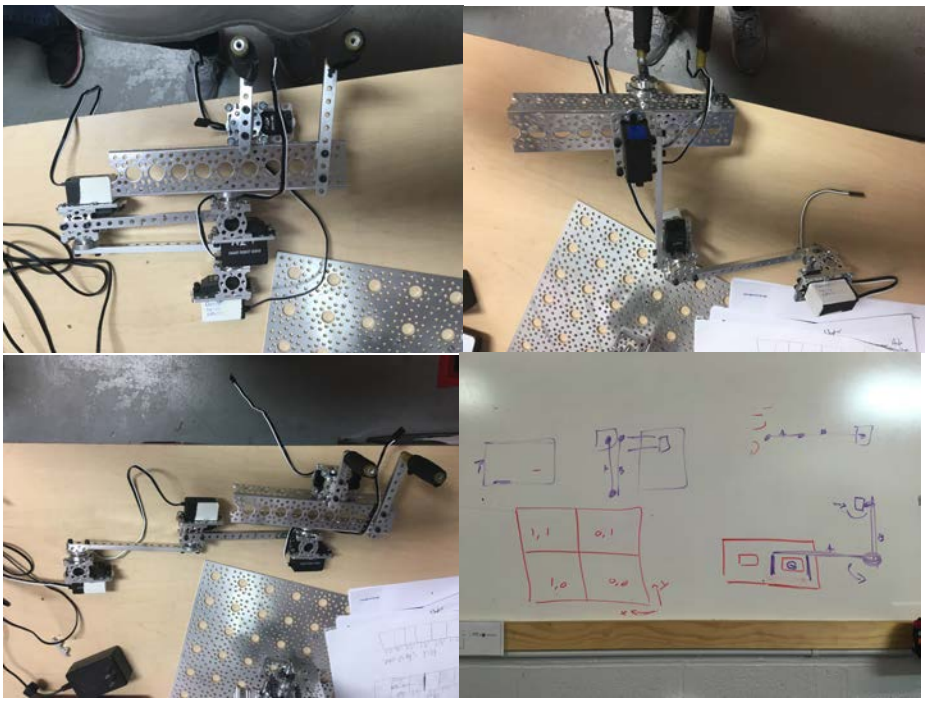
Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we remade teleop for this year's challenge. We made it so that we can switch between a velocity (encoder) based drive and a power based drive. The velocity based drive is for our normal or "slow" mode as the velocity drive is more precise. We believe that in this year's game precision will be of high value. In our fast mode we switch to a power based mode as it allows us to have more speed than the encoder-velocity drive does.

While doing this we found a bug. We found in testing that the robot was moving in the direction that was opposite of what we expected for the direction we were moving the controller. We found out that this was due to a patch we had copied over from a past season for a bug we had back then. The old bug was that the motor would turn in the opposite direction than what we expected. We managed to fix this bug in a more permanent fashion as our code advanced. But when we moved over the tele-op code for this season we brought along the patch that was made to fix a bug that no longer exists.

Overall, it wasn't that hard, we simply copied over from our various tests and from previous years' code, and then troubleshot a few errors. It is now working as it should, and things are looking up for the season ahead. We did this because we are using the same drivebase as last year, because it works, and there isn't really a better option available for us.

We also set up a debug mode that connect a computer. This debug mode allows us to tweak the pid control we have on the robot to create pid that has an error closer to zero. This is for the reasons of finding the best pid as we want. The pid that has closer to zero error will lower the amount of jerky

	movement while driving and the oscillation that can occur in autonomous. We want to avoid these things as they can cost us time and precision. **[Lauren, Calvin]**
Delivery Mech	<table><tr><td>Identity</td><td>Brainstorm</td><td>Select</td><td>**Prototype**</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table>  Tonight we experimented with servos to cut down on translations needed to build a 2x2 tower without moving the robot. The configuration of these servos would allow the delivery mechanism to extend out and place the blocks in the best position to build a sturdy tower. These servos allow the mechanism to work like an arm, with a shoulder, elbow, and wrist.Using these, we can place the stones in any configuration we need to to build a tower. **[Kaylin]**
Lift Mechanism	Tonight we continued to build the lift mechanism. Along the way, we encountered several problems as some of the pre-existing lift pieces had been built incorrectly. In addition, we lacked the parts needed to fully complete the mechanism. While we were able to find spare parts for some

	of the components, there were still several components missing. This was resolved by making the lift temporarily shorter by using only the amount of stages that there were parts for. Also, the incorrectly built stages were fixed. **[Hannah]**

	September 27, 2019	7PM-9PM
Contributors: Calvin, Ernest, Hannah, Kaylin, Kyla, Lauren, Logan, Malcom, Occie, Taylor		

Entries

ToF sensor test and color sensor test

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

Today we tested how the REV robotics 2m Distance sensor and the rev color sensor works. We did this as we plan to use sensors for localization in relation from the block as to avoid collision with blocks and skyscrapers during the season. We find the second important as running into a skyscraper could cost you a match. We also wanted to test out different options for localization as it is a stretch goal to build a system in our code that can avoid collision with other robots during the autonomous period.

For the time of flight sensor where we moved a skystone away from the sensor. *Figure 1* is a table of the values, *figure 2* is a graphed version of the data for the yellow side of the stone, f*igure 3* is a graphed version of the data for the images side of the skystone.

Expected (mm)	Yellow side (mm)	Image side (mm)
30	18	32
40	29	52
50	40	59
60	62	80
70	82	75
80	98	86
90	110	100

100	122	110
200	225	220
300	320*	299*
Expected	Yellow side	Image side
400	440*	390* / 8191 [1]
500	570*	490*/8191 [1]
1000	980/8191 [1]	1046*/8191 [1]
2000	8191 [1]	8191 [1]

** the values with this symbol by it displayed a greater fluctuation in value compared to the other values measured*

[1]the value of 8191 can be viewed as seeing nothing in the case of this sensors. The entries with both this value and another occasionally displayed this value. The ones with only this value displayed that value alone

Figure 1

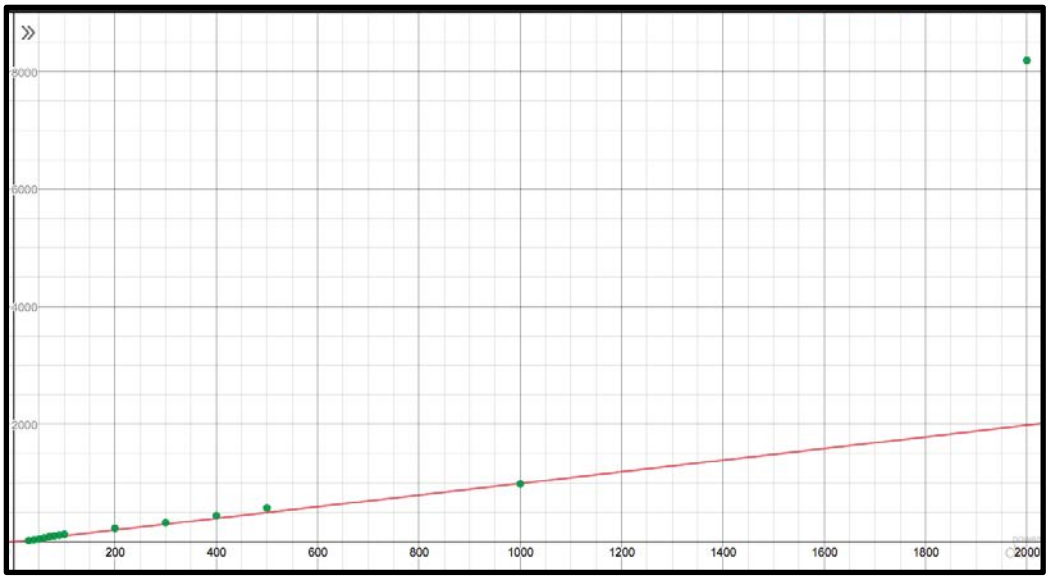


Figure 2

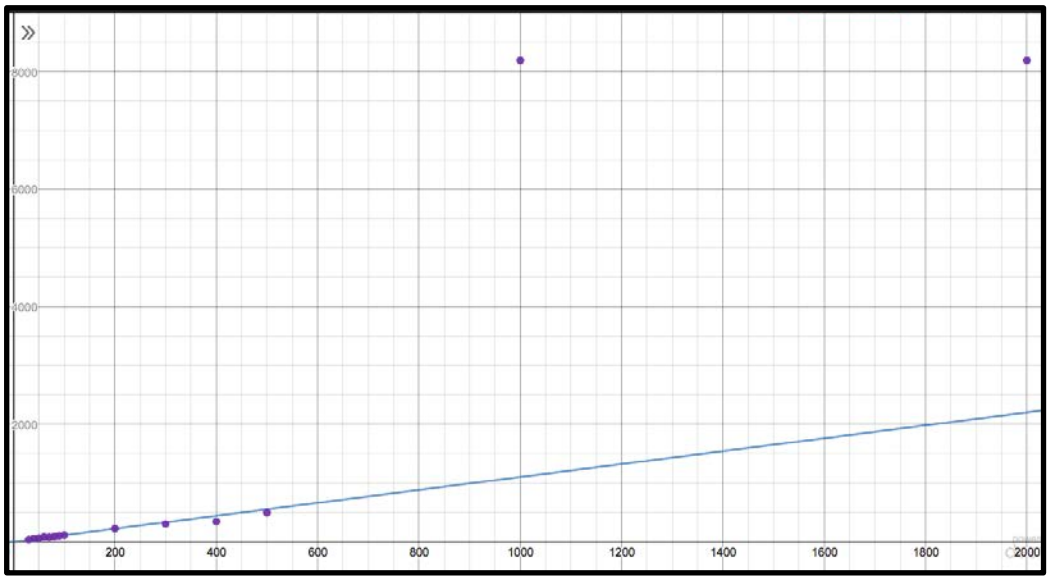


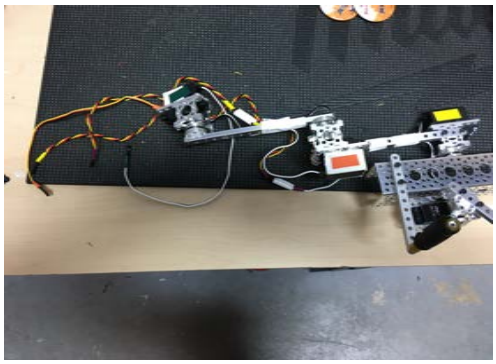
Figure 3

From the data collected from the sensor we can expect the sensor to be most accurate between 50 and 200. This result matches the spec that can be found on the REV website. We also learned that the sensor sees the distance of the different sides as different distances even if they were placed at equidistant.

[Lauren, Calvin, Hannah]

Wired Up Delivery Mech.

Identity	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate



Tonight we started wiring up the delivery mech prototype. We did this so we could experiment with all the servos working together, allowing us to see the full potential of the mechanism. It was difficult to do because there were many servos in a small space. The wires were color coded to make further work on wiring easier and to keep track of what wire connected to which servo.

[Kaylin]

	SEPTEMBER 29, 2019	4PM-6PM

Contributors: Calvin, Ernest, Hannah, Kaylin, Kyla, Lauren, Logan, Malcom, Occie, Taylor

Entries

ToF sensor and color sensor test on foundation

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

Today we tested both sensors: the time of flight (ToF) and color sensors, from the robot to the foundation to see what sensor was more accurate and easily tell the robot when it got close to the foundation to slow down.

ToF sensor:

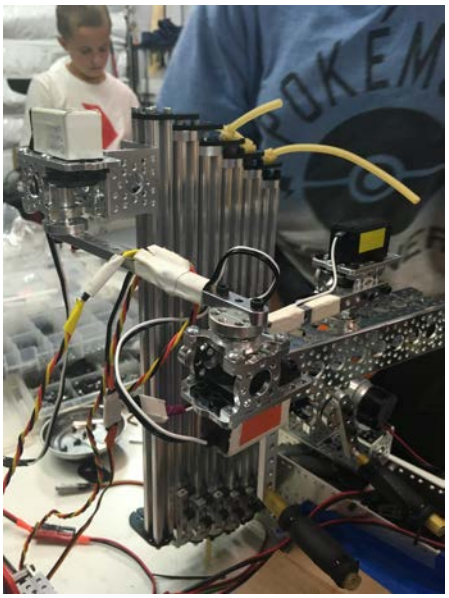
Actual Cm: Mm:

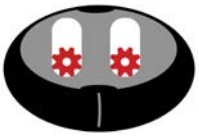
Actual CM	Measured CM
5	2.0
10	8.5
15	17.0
20	20.0
25	25.0

Color Sensor:

Cm:	Raw Cm:	Calibrated Cm:
5	130	0.13
10	120	0.12
15	110	0.10
20	100	0.10
25	100	0.10

	*The ToF sensor is way more accurate the closer it gets, but it could become a problem if there are already yellow blocks on the foundation because the measurement can change drastically if the robot isn't close enough. The color sensor loses its range after 15 cm. Compared to the ToF the numbers were not accurate and had a wide range, which can be tricky for the robot to know when to slow down. So the ToF is better to use for the robot as long as it isn't measuring under or over the foundation. **[Kyla]**
Gripper mech. Abstraction and Method Set Up	Identity | Brainstorm | Select | Prototype | Evaluate | **Design** | Fabricate Today we set up the class that sets the values for each part of the gripper at different positions. The positions for the first two joint (we will be calling them the shoulder and elbow joint) are close and far in reference to the methods, which can be will be able to be called by both autonomous and teleop. The wrist joint(the third closest joint to the lift or the second to last joint depending on your preference of wording) have four positions available. Those positions being rotated far, rotated close, natural (the term used in the code to describe the opposite state of the rotated position) far, and natural close. The finger joint (the fourth joint from the lift or the out most joint) has two method states. Gripped and ungripped. The job of this joint is to hold the stone whereas the other joint jobs are to move the position of the stones. **[Lauren]**
Button Assignment to Gripper Mech. Positions	Identity | Brainstorm | Select | Prototype | Evaluate | **Design** | Fabricate Today we worked on prototyping a gamepad for the operator(which is what we call the second driver). We based it off of the current prototype for the robot's arm, which we will use to grip the blocks to build with. We simply assigned the actions we wanted to perform (grip, ungrip, move the mechanism to positions A, B, C, and D, and moving the lift) to the buttons that we wished to assign them to. This is only a prototype, so it is very likely that we will change this all later. **[Calvin]**
Attached Delivery Mech.	Identity | Brainstorm | Select | **Prototype** | Evaluate | **Design** | Fabricate

	 We attached the delivery mechanism onto the robot. This allowed us to explore how the mechanism interacts with the robot. For instance, we found out how much space the mechanism might take on the robot. We discovered that with the mechanisms current positioning on the robot, it is unable to pick up the stones from the intake mechanism. We also noticed that the servo wires we used were too short to reach the REV hub. Finally, we programed the servo positioning so that the programming team will have an easier time programming the different delivery positions. **[Kaylin]**

	**October 4, 2019**	7PM-9PM

Contributors: Hannah, Taylor, Ernest, Occie, Tarendran

Robot

Identify	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate

Today we continued to put together the first prototype of the robot. The phone holder that was previously attached was too shaky and unstable, so it was replaced with a more secure holder. Surgical tubing was put inside each stage of the lift to keep tension on the lift. We also worked on wiring the robot. At first, the wires were just plugged into any port they fit into, but we realized this was wrong. One team member consulted the Hardware Map and was able to figure out the right place to plug in each wire so that the programming worked correctly. We also continued to work on a second lift prototype in case the original design is not satisfactory.

[Taylor, Occie, Tarendran, Hannah, Ernest]

	October 5, 2019	7PM-9PM
Contributors: Lauren, Calvin		

Entries

Tying together controls and delivery mech. Method code.

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	Fabricate

This session we planned on adding the code for the arm mechanism we had designed. To accomplish this, we had decided on a plan of attack: to wire up the arm mechanism in the code; to tie the arm to controls; to get possible positions for the servos on the arm; and to test the arm mechanism with the controls.

We started by connecting the delivery mechanism object to the controls. We added the derived buttons (aka the names of tasks the physical button will do when pressed) and tied them to the physical buttons of the controller. The button and what functions they are tied to are listed below:

Left bumper = arm in x button = wrist turn right-Joy = lift throttle
Right bumper = arm out b button = stow left-Joy = intake throttle

We only manage to tie the turn wrist button to the object today. We also debugged the button. We used debugging feature of coding software. At next practice we plan to connect the rest of the methods from the object and add the methods for the lift to the object.

When we started testing the screws there were multiple problems with the wiring of the robot. The wiring was extremely messy to the extent that we could not run most of the mechanism with the risk of pulling out wiring or entanglement. Multiple servos were unwired also (shoulder and wrist). The elbow servo had one of the wire connectors reversed which caused it to be inoperable. The two hubs were also not wired together.

After we connected the controls to the object in the code and began testing the code, we found a bug. At this point we had only connected the wrist functionally and nothing else. We tested it to find that it was not working. This

	was due to a missing = sign in our conditional statement of the rotateToPos() method. It might be a good idea to look at the other conditionals in the deliver mech object next practice. The wrist works as far as we can test today. In summary, today we wired and unwired servo, fixed a backwards connection, told the build team to wire better, fixed our code with the debugging feature, and added a missing equals sign. We learned how to properly utilize the debug feature, and incorporated our delivery mechanism method into the control system. After we had finished with all of that, we tested it again, and it all worked well. We plan to incorporate the rest of the Delivery mechanism methods into the control system, debug the rest of the control systems, and test positions and mechanisms with new code. **[Lauren, Calvin]**

	October 6, 2019	7PM-9PM
Contributors: Hannah, Taylor, Ernest, Occie, Lauren, Calvin		

Entries

Tying together controls and delivery mech. Method code cont.

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	Fabricate

Today we continued to connect our abstract methods that run the different functions of robot operability to the physical controls. In the last practice we had completed the functionality tied to the X-blue button on the controller. Below are the buttons and the methods with what they do:

- The arm “in” or close position to the left bumper
  - turnToPos() - turns wrist to proper position
  - armClose() - moves arm to close position
- The arm “out” or far position to the right bumper
  - turnToPos() - turns wrist to proper position
  - armFar() - moves arm to far position
- Stow [values of this operation are wrong at the moment]
  - stow() - turns all servos to stow positions

We also added the motors and variables necessary to create a min and max methods for the delivery mechanism. We tied these methods to the left joystick. The positive values on the joystick being the “go to max” or upwards functionality. The negative values on the joystick being the “goto min” or downwards functionality. Our min and max have a feature that prevents from going above or below the max or mins respectively.

In the controls have we added the derived grip and ungrip buttons. We tied grip to the y-yellow button and the ungrip to the a-green button. The grip button calls the gripBlock() method and the ungrip button calls the ungripBlock() method.

We have yet to test the added methods. We plan to test and fully debug all the buttons as soon as we have access to the robot as the programing team.

	This will most likely be after the wiring is a bit more finalized and the robot is a bit more fortified. (Hopefully in the next practice or two) We also need to finish the code for the intake mechanism, which to our code is part of the delivery mechanism code. **[Lauren, Calvin]**

	**Friday,October 11, 2019**	07:00 PM-9:00 PM
Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Logan		

Entries

1. Robot drive base/ mechanism repairs and mock up

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Redesign of side plate

- New Drive plates
- We reangled the wheels to form a “X” from above

Repositioning delivery mechanism

- Positioned servos for delivery arm

Intake mock up

- We decided to change the intake by moving the motor inside of the robot and use belts to initiate the intake
- We mounted a motor on the inside of the drive base and the belt leading to the front of the robot

2. Autonomous Planning

Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

Note: The section below also serves as the notes taken on the discussion we had for planning autonomous.

Definitions of autonomous task options:

Note: cycle is a termed used by our software team to describe one complete task (I.E delivery of a stone) in which said task may be repeated through the code weather in auto or tele-op.

Note: “auto” is out shorthand for autonomous

1. Delivery - moving one (1) stone/skystone from the loading zone to the building zone

1.1. For cycle one and cycle two, a skystone delivered counts as ten (10) pts.
1.2. For cycle one and cycle two, a stone delivered counts as two (2)pts.
1.3. For cycle three and onward, a **stone or skystone** delivered counts as two (2)pts.

2. Repositioning - Alliance Foundation moved into Alliance build site and not in contact with robot at the end of auto
   2.1. Ten (10)pts
3. Placing - stone in foundation (for our case preferable buildable upon) at the end of auto
   3.1. Four (4)pts each cycle
4. Navigating - robot parked over the tape that separates the building zone and the loading zone.
   4.1. Five (5)pts

Desired final auto

1. Auto split into segments of steps to help optimize working with alliance partner
   1.1. Possible segments :
      1.1.1. collect skystone 1/ stone 1
      1.1.2. move foundation ?
      1.1.3. deposit skystone/ stone;
      1.1.4. collect skystone 2/stone 2
      1.1.5. deposit skystone 2
      1.1.6. park / navigate
      1.1.7. Idle - stay out of the way of the partner task (only to be used if the main/ typical tasks are not being run.)
2. Ability to start at any possible starting location and still run all auto tasked listed above
3. Accurate

This will end up, best case scenario with a menu at the bottom of the auto in screen that looks like below.

Note: the idle option will be true when certain conditions are met. We think those conditions will be when running only the foundation and or the navigate tasks.

```
ALLIANCE : [RED; BLUE]
COLLECT: [0;1;2]
DELIVER: [0;1;2]
FOUNDATION: [YES; NO]
NAVIGATE: [YES;NO]
```

	WAIT @ START: [1;... 10] TIME ESTIMATE: [CALC.] **[Lauren,Calvin]**

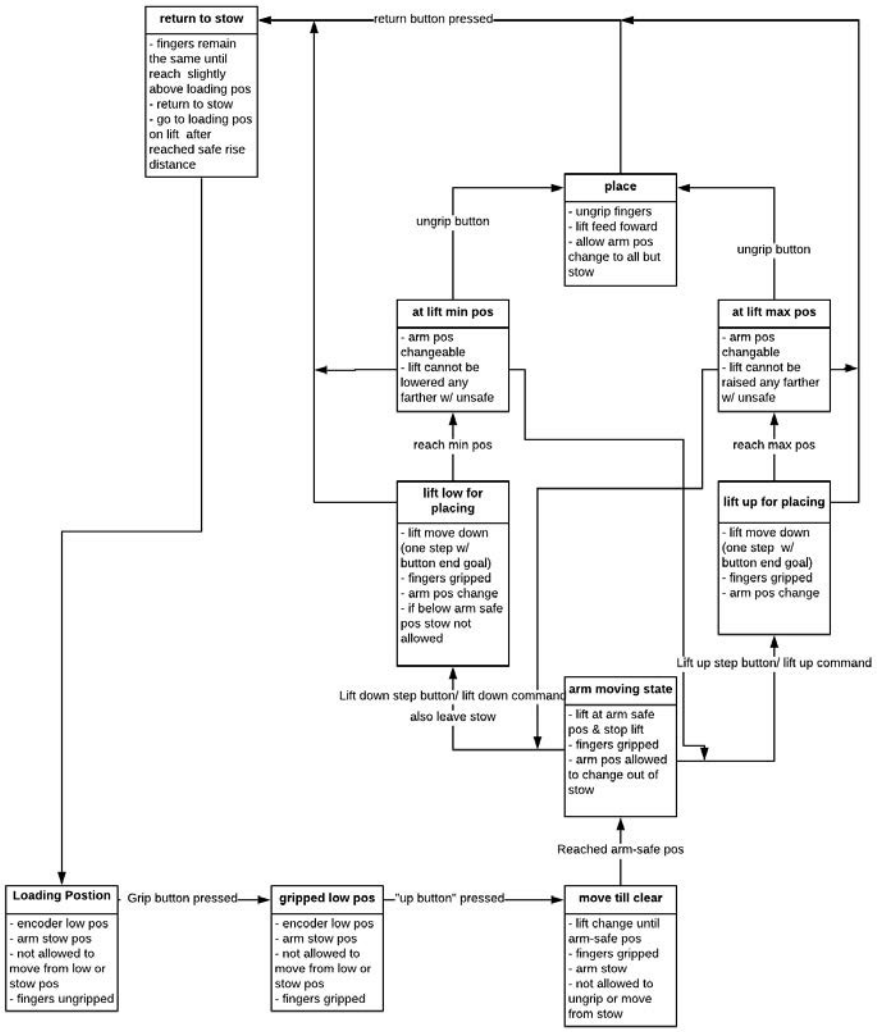
	October 13, 2019	4PM-6PM

Contributors: Hannah, Calvin, Kaylin, Occie, Lauren, Liam, Logan, Taylor, Ernest

Entries

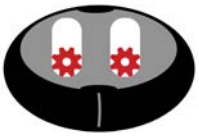
Elevator Lift State Machine

Identify	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate



This is an image of a state machine we are going to use for the lift mechanism. A state machine is an open loop code that consists of various

	parts called “states.” A state is a different form of functionality within the code. For example, how the arm-lift mechanism act when we are loading a stone (and thus in the loading state) has different requirements of functionality from when we are in when we are raising the lift (the lift up for placing state). In short a state represent a step in a much larger task that helps avoid mishaps from occurring by not allowing certain functionality of the mechanism at certain state and preventing the skipping of steps in the larger task. The states are represented by the boxes in this image, with the top part with the bold text representing the name of the state, while the box below this represents the actual state itself. However, state machines also change between states. This is represented by the arrows. These states are changed when certain conditions are met, and due to the fact that this is open loop, these change based on input from drivers. The conditions for these changes are listed on the arrows. The reason we are doing this is because it always helps us programmers make better code by using state machines. But without planning a state machine can be as good as, or worse than no state machine. State machines also allow us to put multiple functionality on one button that changes based on conditions met. Thus allowing for less human error with complex mechanisms and functionalities **[Calvin, Lauren]**

	**October 17, 2019**	7PM-9PM

Contributors: Connor, Ernest, Hannah, Occie, Lauren, Malcom, Taylor, Tarendran

programing; servo value changes, SDK update, ran tests

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	Fabricate

Today we changed some values of the gripper serovs as they had been moved/altered and need repositioning. This was partially due to a slight redesign in the arm during this time that removed one serovo. The change to the gripper design was due to movement in the mechanism when it was used. We did this so we could test the gripper mechanism without having to account for errors in the servo positions. We also ran test on the servos in the gripper mechanism and updated the SDK to version 5.3. We updated because there was a bug fixed that latency of reading of I2C sensor, which we use on this robot.

[Lauren]

	October 18, 2019	7PM-9PM
Contributors: Connor, Ernest, Hannah, Occie, Lauren, Malcom, Taylor, Tarendran		

Entries

Arm lift state machine

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	Fabricate

Today I (Lauren) was the only programming team member present. I worked on the skeleton for our delivery mechanism (or arm-lift mechanism). We are using a state machine as we find it is one of the most reliable ways put multiple step process on less buttons. We do this as we have found when we put repeated steps, in which they may not be in the same order but have clear paths of where they can go, it is more reliable than humans pressing buttons repeatedly. It can also allow us to complete tasks faster than a human could (this is due to the fact of human response time versus computer or sensor response time).

Today I completed connecting the states to each other and creating the names within the code for each state. The plan going forward is to complete the if statement conditions that run within the states (and allow us to go from one state to another) and the functions of each state. After that, the hope is to have a mostly functioning state machine.

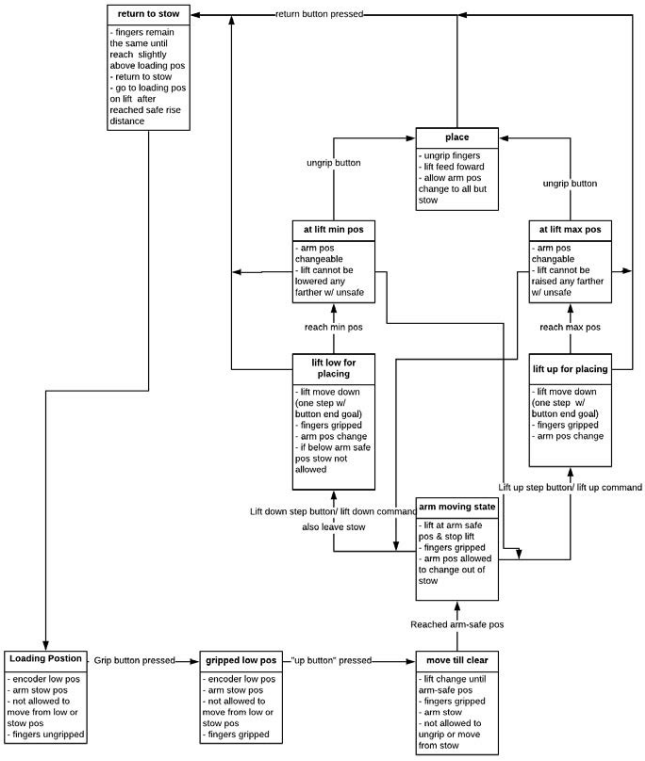
[Lauren]

	October 25, 2019	4PM-6PM
Contributors: Hannah, Calvin, Kaylin, Occie, Lauren, Liam, Logan, Taylor, Ernest		

Entries

Elevator Lift State Machine

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**



The image above is a repeat from the october 13th entry. I thought it would be good to include it as it applies to the fabrication of the state machine since this chart was used in making sure it was set up correctly.

Today we set up how the state machines relate to each other. In the prior set up, which was temporary as we need to make sure the other code compiled before going on to the proper way of setting up the order. We set up the order outside the state themselves, using methods to call the state in each case

	they were used. We did this due to our circular nature in the state machine above. Because of this it would be impossible to initialize the state in a way that they would run properly. **[Lauren]**
Belt Drive for Elevator Lift	<table><tr><td>Identity</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table> • **Tiny belt attachment plate have been added** **[Hannah]**
Drive Base Design	<table><tr><td>Identity</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table> Today we re-evaluated the drive base design and made changes to hole and cut-out placement in order to better suit the needs of the now-developed mechanisms. Previously, the drive base was designed without consideration of mechanism placement, because no mechanisms or completed prototypes existed. One of the problems that were discussed today was that of the intake mechanism placement. While testing the intake, the team noticed that the best angle to place the mechanism at was affected by the height at which it sat. The team felt that finding the exact best placement for the current intake would not be a worthwhile change, as future iterations might have a different optimal placement. Eventually, we chose to cut out arc-shaped pathways around the motor mount position for the mechanism, so that the intake angle could be adjusted for optimal performance. This made future changes much easier and faster, because the base plate could accommodate changes, eliminating the need to machine a new side plate every time. **[Hannah]**

	October 27, 2019	4PM-6PM

Contributors: Hannah, Taylor, Ernest, Occie, Tarendran, Kaylin, Lauren, Calvin

Entries

Fabricated Brackets

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we fabricated the brackets for the belt lift. These brackets will help the lift move up. We first measured out where the holes should be. Then we punched where we wanted the holes so that we would be able to drill accurately. After that, we moved to the drill press and drilled all the holes needed to attach these brackets to the lift. We cut off the excess material and sanded the rough material down. After cleaning up the dye-chem with acetone, we proceed to tap the holes that we needed to be threaded. Finally, we mounted the brackets to the lift so that our lift would function properly.

	[Kaylin]
	<table><tr><td>Identify</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table> Today we actually got the state machine that was previously mentioned in this journal started. It now responds to our button pushes. Also, we managed to troubleshoot a fair amount of problems [problem list]. We also added an interface to our state machine. We did this because it would help lower the amount of lines of code we have to write overall. With the interface we were able to create a special kind of for loop that runs for the length of a list instead of off integer values. **[Lauren,Calvin]**

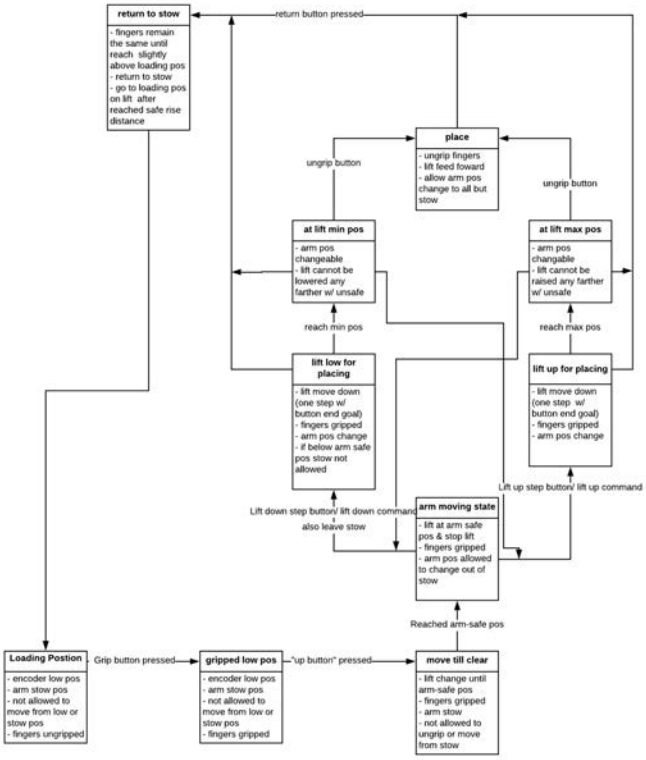
	November 1, 2019	4PM-6PM

Contributors: Calvin, Kaylin, Occie, Lauren, Logan, Taylor, Ernest, Tarendran,Habtamu

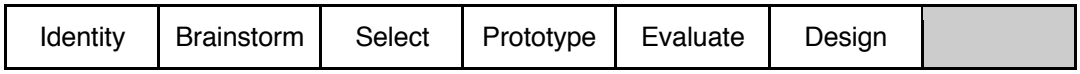
Entries

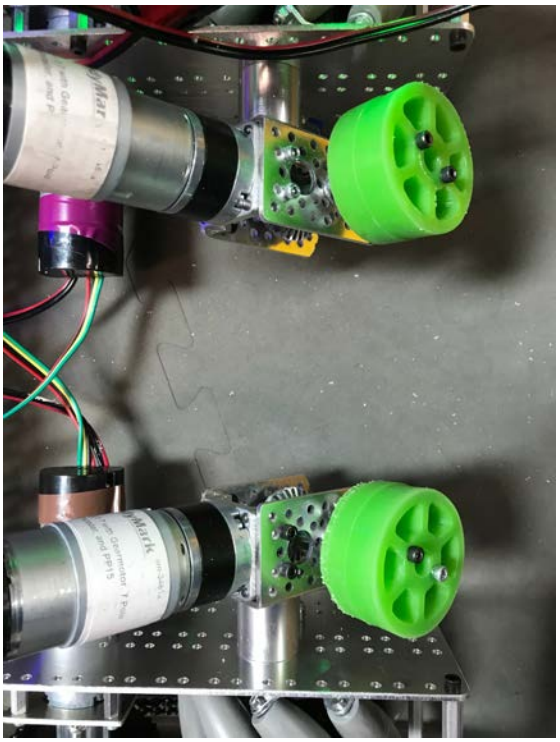
Elevator Lift State Machine

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**



Today we continued to write tests for our state machine. We did this as it will help us get a clearer prediction of how the robot will act when we add the code without taking the robot up for testing while building still is in sesion. During our test we found that most of our code acted as we expected (after dealing with the bad naming system for comparison our testing system has, thank past selves). We did find a section of code that did not work as we expected. We believe there is a bug of some sorts from leaving the at min. State back into the arm moving state. We were unable to find the bug in this

	meeting but it is on our radar to fix soon. (between next day to next meeting depending on availability). Another thing to note is that we are using a simplified version of the state machine because we were unable to get the testing needs to make the original state machine design work in the time we had to the league meet. We still have the code of the more advanced state machine and plan to implement it later. The current state machine only focusing on stowing, loading, extreme positions (min. and max.), and moving up and down (armMovingState). It does not have the step incrementer that was planted in the initial state machine design and thus is less circular. During driving we also came across a minor bug in steering. The left and right direction from the control to the robot are flipped from what we find to be intuitive. This is most likely to be fixed with a sign change in the code. It will be fixed when we fix the other bug as well. [<--- note to future programing selves] **[Lauren, Calvin]**
Mounting Intake Wheels	 Today we flipped the wheels on the intake from beneath the motor to above the motor.We flipped what side the motors were on. We did this to lessen the likelihood of the stone hitting the channel before the wheels. We also trimmed the channel. We made sure that the robot wasn't over the size limit. We tested and figured out that the way that we previously controlled the motors wasn't that effective.



When we tested the intake mechanism we learned many things:

- The motors on the intake spun too fast
- Couldn't pick up blocks if not completely aligned
    - Going to try some kind of funnel
- May need to build chute for blocks

[Logan]

Lift

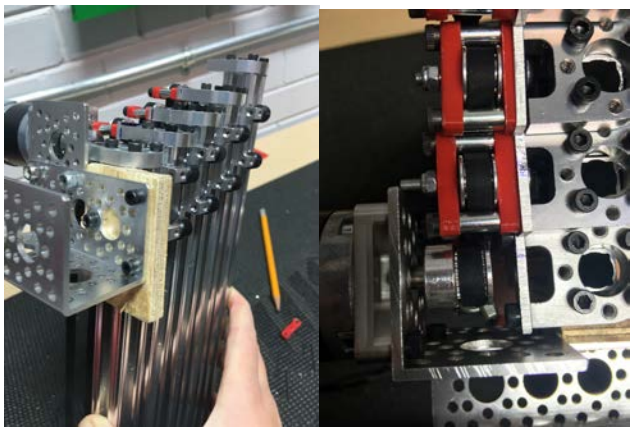
Identity	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We fabricated new brackets for the left because the ones we had machined earlier were too short and got hung up on the wheel. We measured, punched, and drilled the new brackets. Finally, we attached them to the lift.

[Kaylin]

Lift Motor

Identity	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**



Tonight, we attached the motor to the lift. We decided to put the motor at the top of the lift to save space. We used wood to pad out the channel we used to attach the motor, so the motor shaft and the belt would line up. At first, we tried using a smaller piece of wood to attach the motor. We soon found out that we would not have enough places to securely screw in the channel. We settled for a slightly wider piece in order to have a more secure motor.

[Kaylin]

	November 3, 2019	4PM-6PM
Contributors: Calvin, Kaylin, Occie, Lauren, Logan, Taylor, Ernest, Tarendran, Habtamu, Hannah		

Entries

Elevator Lift State Machine

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we did some debugging and polishing of the delivery mechanism code. We discovered the values that work for the pid control on the lift and the feedforward value. We may return to re evaluate these values if we find that they are not the most optimal. We also found that we need to heavily debug the gripper mechanism before we continue with the full debug of the delivery mechanism.

[Lauren, Calvin, Hannah]

Mounting cowcatcher

Identity	Brainstorm	Select	Prototype	Evaluate	Design	

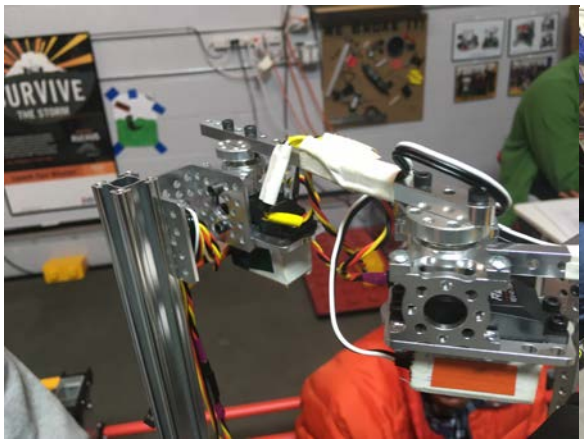
To get the blocks onto the ramp without the block spitting back out, we made a prototyped cowcatcher that forces the blocks down a chute and into the intake so that it is easier to collect. When we tested our idea, we found that it wouldn't work, so we widened the cowcatcher and curved the edges. When we tested it again, it worked way better and gave us what we needed to load the block successfully with ease.

We also made a ramp for the intake, so that it can be loaded on to the delivery mechanism. We took a piece of plywood and cut it to 4 by 12 inches. We then saw that it was too long so we cut 4 inches off to make the ramp 4 by 8 inches. We then copied a actobotics hole pattern to then drill holes to fit a butterfly plate hole pattern. Then, mounted the plywood with button head screws.

[Logan, Ernest Woods]

Lift

Identity	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**



The lift was mounted onto the side of the robot. We had to move the channel in the middle of the robot to make room for the lift. The build team and the programming team tested the lift with the drop-off mech. We finally got to see how these two very crucial elements of the robot work together.

[Kaylin]

	November 10, 2019	4PM-6PM

Contributors: Calvin, Kaylin, Occie, Lauren, Logan, Taylor, Ernest, Tarendran, Habtamu, Hannah

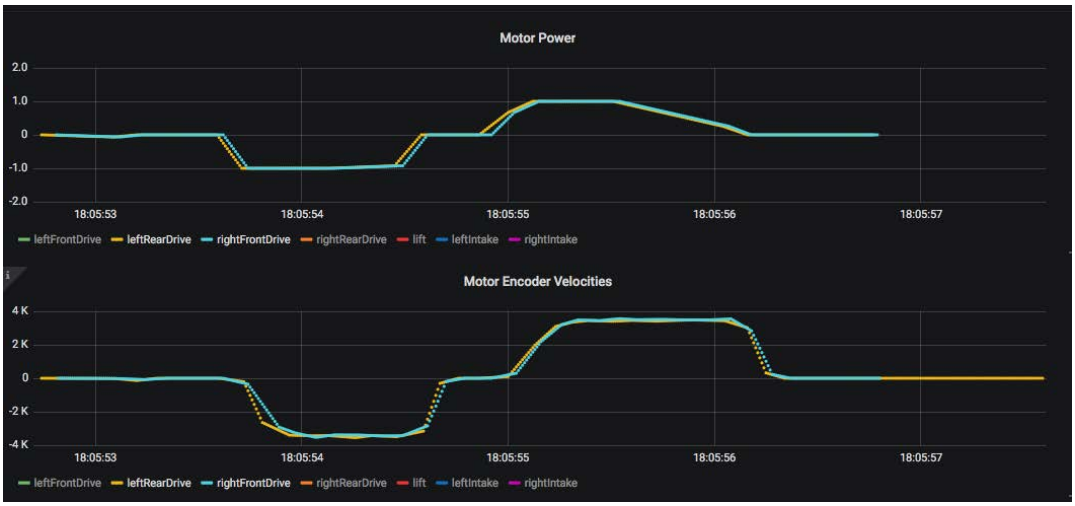
Entries

Debug Strafing Accuracy Issue

Identify	Brainstorm	Select	Prototype	**Evaluate**	Design	Fabricate

Today we tried to troubleshoot a problem we had noticed: our strafing was all wrong. We first used a program to measure how our motors were working individually (pictured above), and it turned out that the front right motor was working in a completely different manner than every other motor. The right motors are supposed to mirror the left motors, and while the motor power for the motors seemed to be in order, the front right motor's velocity was not mirroring the front left motor's velocity. The two back motors seemed to be in order. We looked through the code, but we found that nothing in it would lead to such issues. That lead us to the conclusion that the motor itself was the problem.

Here is the time series after replacing the suspected motor:

	 **[Lauren, Calvin]**
Gripper Mech	Today we tried to brainstorm some improvements to the block-lifting grabber. The current design was not reliable and often dropped the blocks. This was partially due to the fact that we were attempting to grab slanted surfaces, which caused the block to slip. This was solved by lengthening the stationary “finger” downwards in order to cover a larger surface area of the block to reduce the problem with angles. When this failed to completely fix the problem, we decided to extend the moving “finger” piece in order to grab the end of the block and not just the protruding part. This drastically reduced the angle of the block and came with the added benefit of extra stability.

	**November 17, 2019**	4PM-6PM
Contributors: Calvin, Kaylin, Lauren, Logan, Ernest,, Habtamu, Hannah, Kyla		

Entries

Debug Strafing Accuracy Issue

Identify	Brainstorm	Select	Prototype	**Evaluate**	Design	Fabricate

Today we worked on rewiring jackie-bot and tuning the pid controls within roadrunner. Roadrunner is the application we use to drive during autonomous using feedforward and feedback. Much of our practice was spent rewiring the robot as all the wires had been removed priorly and were poorly labeled. We had to rely on our hardware map and motor tester program to rewire the robot because of this because of the poor labeling. One thing that we found was that when encoders were wired wrong the robot would not drive straight or turn in unexpected ways. One we fixed the issue of encoder being plugged in wrong these problems were no longer present.

After this we worked on tuning pid. We used multiple test during this practice. Such as an oscillation test and a track width running test. When we were testing we had an issue with the distance test we found that no matter how we changed the method we would always travel the distance. We decided to try a slow the acceleration of the robot to see if it fixed it but ran out of time to test if that changed help. We plan to continue to solve the bug in this test and tuning the pid next time.

[Lauren, Calvin, Kyla]

Planning/Testing Routes for Auto

Identify	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate

Tonight we planned new autonomous routes. We worked on identifying the

	sky stones and talked about how many stones we needed to scan to be able to determine where the sky stones are located after the randomization. We need to be able to see 3 because there will always be 1 sky stone in the first 3 that will determine the position of the other sky stone. We also worked on how we can move the foundation in autonomous. We started to program from a different starting position on the other side of the bridge. We used Road Runner to make up an “S” pattern. Once there, we switched to teleop to see that if we had gripped the foundation, how we could move it. We discovered that we can probably do a 90-degree turn and then drive forward to be able to score in the building zone. Now we just need to program these new autonomous paths. **[Kaylin, Calvin]**
CAD Intake Model	In CAD we a version of our new intake that was worked on yesterday.

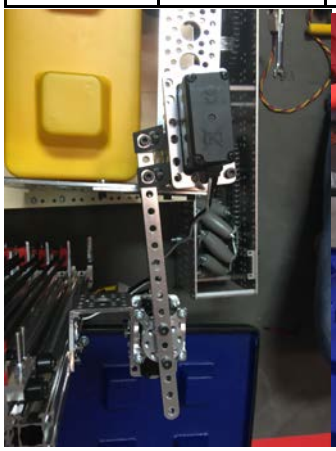
	Friday, November 22, 2019	07:00 PM-9:00 PM

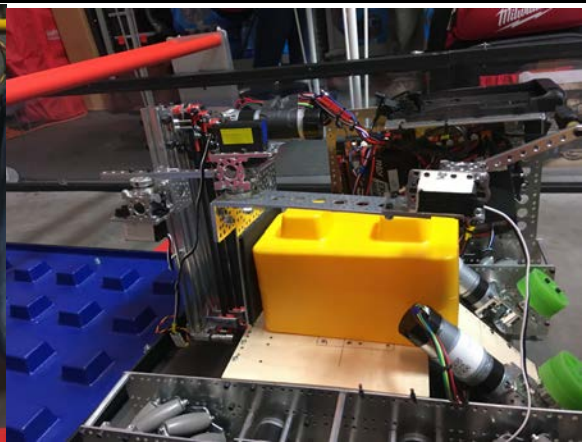
Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Taylor, Occie, Tarendran,

Entries

1. Simplified Arm Design

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**





Tonight, we took apart our complicated arm design and then used some of the old servos to create a much simpler arm. We figured out how to attach the new arm to the robot while maintaining the same geometry of the old arm. We then added “rails” for the block in order to ensure the block does not move out of place while the arm is trying to grip. Finally, we discussed adding a plate to the finger servo to act as a hood for the block.

We also simplified the code for the arm as this version of the arm has one less servo. The missing servo was referred to as the elbow servo in the code. We removed all code that reference said servo as it is no longer used and was just making our code messy.

2. Reclaim parts.	

Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

We removed the REV Hub from Paradebot to use on V2. We need the Hub because it gives certain parts of the Robot power. The Hub translates the programs from the phone into a code the robot can understand.

aWe also worked on finding a spot to put a battery mount on Robot V2. The Mount has to be put in a spot where it will not be in the way of the intake mechanism, and it's weight distribution has to be equal throughout the robot so that the robot moves straight. We also put in bearings into wheels, but after a period of time, realized that the shipment was wrong so we took out the bearing and put them pack in the package.

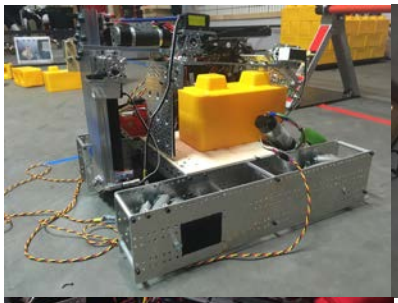
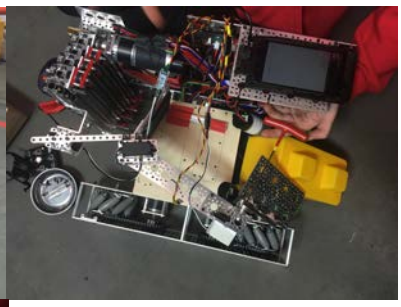
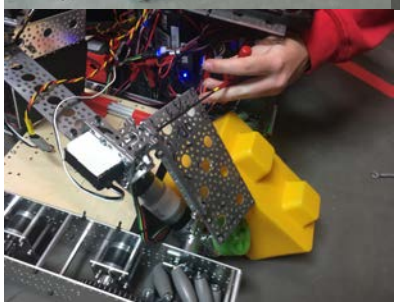
	Sunday, November 24, 2019	04:00 PM-6:00 PM

Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Taylor

Entries

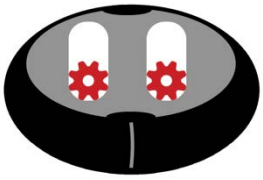
1. Continued Arm Design

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We continued work on the arm design. We reprogrammed all the servos needed to put the mechanism in the 2 positions needed, stow and placing. We had to remove the servo horns and then reposition them in order to allow the servos to reach the right rotation needed to obtain the positions we wanted. We wired the servos so that the cords would reach the REV hub. We still need to figure out what path is needed for the wires so they will be out of the way of the mechanism. Finally, we tested the arm and gripper with the intake mechanism. We found out that we need more guides for the stone because it often gets in an awkward position from intake, making it difficult to pick up with the gripper. Finally, we made sure the drop-off process worked. It still needs iteration and practice, but the basics work.

[Kaylin]

	**Sunday, December 1, 2019**	0:00 PM-6:00 PM

Contributors: Ernest, Hannah, Kaylin, Lauren, Liam, Taylor

Entries

1. Chaos ninja set up.

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we worked on creating the inner workings of Chaos ninja. Chaos ninja is a version of tele-op that can be activated by the konami code and set up with different settings (boolean for metric collection and challenge level of Chaos Ninja). Today we set up the state machine to sense if the konami code was entered via the controller. We did this via a time out state that checks if the *d-pad up, d-pad up, d-pad down, d-pad down, d-pad left, d-pad right, d-pad left, d-pad right, "b" button,* and *"a" button* are pressed and in that order without other buttons or long waits between those buttons. We also created the landing page for chaos ninja that allows us to change the settings before activating.

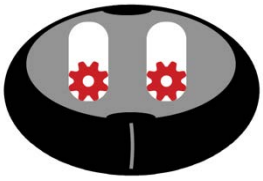
[Lauren and Calvin]

2. Chaos ninja challenge level planning

Identify	**Brainstorm**	Select	Prototype	Evaluate	Design	Fabricate

- **Level 0**
  - Metrics only
- **Level 1**
  - Low battery mode
  - Poor connection mode (lag mode)
  - Bad motor mode (1 motor)
  - Dead servo (1)
- **Level 2**
  - Multiple failure ([level 1,1])
  - Bad motor mode (1 dead and one dying on same side)
  - Bad motor mode(2 dying at different rates on opposite sides)
  - Dead servos (2-3 depending on the amount of servos and location on robot)
  - Servo fixed wrong (reverse servo)
- **Level 3**
  - Multiple failure (mega failure mode [2,2][1,1,2])
  - Dead mech (mechanism on robot stops working completely)
  - Robot temp disconnect
  - Static failure

	[Lauren and Calvin]
New Driver Station	<table><tr><td>Identify</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr></table>  Tonight we set up a new driver station with our new controllers and phones. We layed out the wires so they would not get tangled up. We added strain relief to the important connections in order to ensure they would not become unplugged during practice or competition. This is important because a functional drivers station allows productive drivers practice to occur **[Kaylin]**
Intake Mechanism	Today we made a small adjustment to the intake mechanism. A small “wall” was added in order to stop the stones from turning from side to side. As of right now, only one side is complete but it works. The wall ended up having to be trimmed down because the original attachment was longer than the allowed size of the robot. We also added a bend at the base of the hood that also serves as the “finger” of the placing mechanism. When stowed it acts as part of the intake mechanism to prevent the stone form launching out of the robot. The bend was added because it make intake easier. We also switched attachment method of the hood to make it a smoother surface and thus easier to grip the block. We found in past testing that the old attachment method of the gripper/finger/hood caused it to slip across the block due to the screws used protruding from the surface.A note for the future, we may need to move the hood farther back on the finger servo as the block may slip from it when it is used as the finger of the delivery servo. **[Hannah and Lauren]**

	**FRIDAY, DECEMBER 6, 2019**	07:00 PM-9:00 PM
Contributors: Lauren, Calvin		

Entries

1. Delivery Mechanism

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we tested the delivery mechanism. We tried changing servo parameters on the new arm, which went as planned.

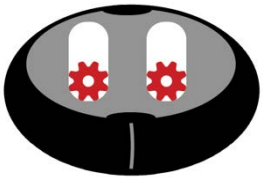
[Calvin, Lauren]

2. Debugging for Days

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today I debugged the current autonomous because it was experiencing multiple failures of a game-ruining kind. The errors mostly stemmed from conversion errors: it was interpreting degrees as radians. Those two things are, of course, two radically different measurements, and resulted in the robot spinning for almost 10 seconds without end. It now works much better, and will definitely earn us more points during matches. We were lucky this error happened to us here and didn't cost us millions of dollars like certain NASA projects (i.e. Mars Climate Orbiter).

[Calvin]

	**Thursday, December 13, 2019**	07:00 PM-9:00 PM
Contributors: Lauren		

Entries

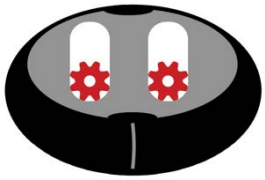
1. Auto test and alteration to lift state machine

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today we worked on a basic auto that allows us to park under the skybridge. We tested it on our field to get the proper distance. Today was mostly adjusting of those values. We have it so it parks slightly under the skybridge. We have four parking positions: a left near, left far, right near, and far.

We also made an alteration to a value in the lift state machine. We made the min height value the same as the move clear height, because of limitation of the lift at the time, we cannot make the min position normally. This causes issue in running the lift when we lowered it.

[Lauren,]

	**December 14, 2019 (League Meet #2)**	7AM-2PM
Contributors: Calvin, Conner, Ernest, Habtamu, Hannah, Kaylin, Lauren, Logan, Occie, Tarendran, Taylor		

This entry was created by information collected during the league meet, and authored together as a team at our normal practice scheduled for the following day.

Entries

1. Lift Mechanism Failed to Operate Correctly

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

What Happened

1. **After 20secs the lift jammed going up.**
2. **The list would go up, but then would make grinding sounds, and would be jerky when coming down.**
3. **When trying to push the lift down, it would feel like trying to push a cat into a box**

How did we fix it then?
We took off one of the V-wheels and that fixed the problem of it going up and down for a little bit. And we were then limited with height.

Why did it happen?
The robot was driven hard on Friday.

How will we try to make it not happen again?

Having more control over driver practice the night before a meet.
After every meet, we need to make sure to test drive the robot.

2. Driver / Operator cannot See Capstone Placed by Human Player

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

What Happened
The capstone was placed at an angle and position in the depot that disallowed the driver to see it when trying to collect it.

	How did we fix it then? We didn't get enough time to fix it during the meet. **Why did it happen?** Capstone was placed in an insufficient position. **How will we try to make it not happen again?** The human player needs to be apart of drive team practices. Make the ends of the capstone more contrasted.
3. Warned Multiple Times About Autonomous Late Starts	**Identify** | Brainstorm | Select | Prototype | Evaluate | Design | Fabricate **What Happened** Coaches started the autonomous late, due to standing before the start of the game, and unawareness of when autonomous started. **How did we fix it then?** We got a warning and didn't have time to fix it during the meet. **Why did it happen?** We were not ready for the start of autonomous. **How will we try to make it not happen again?** Put a sign or gripper on the remote control case, or just kneel down by the remote controls.
4. Room to work in the Pits.	**Identify** | Brainstorm | Select | Prototype | Evaluate | Design | Fabricate **What Happened** There were too many people in the pits, and it got busy and hard to work on the robot. **How did we fix it then?** We had people go and talk to other teams. **Why did it happen?** Teammates didn't have jobs to do. **How will we try to make it not happen again?** Get people jobs.
5. Bowl for buttons.	**Identify** | Brainstorm | Select | Prototype | Evaluate | Design | Fabricate **What Happened**

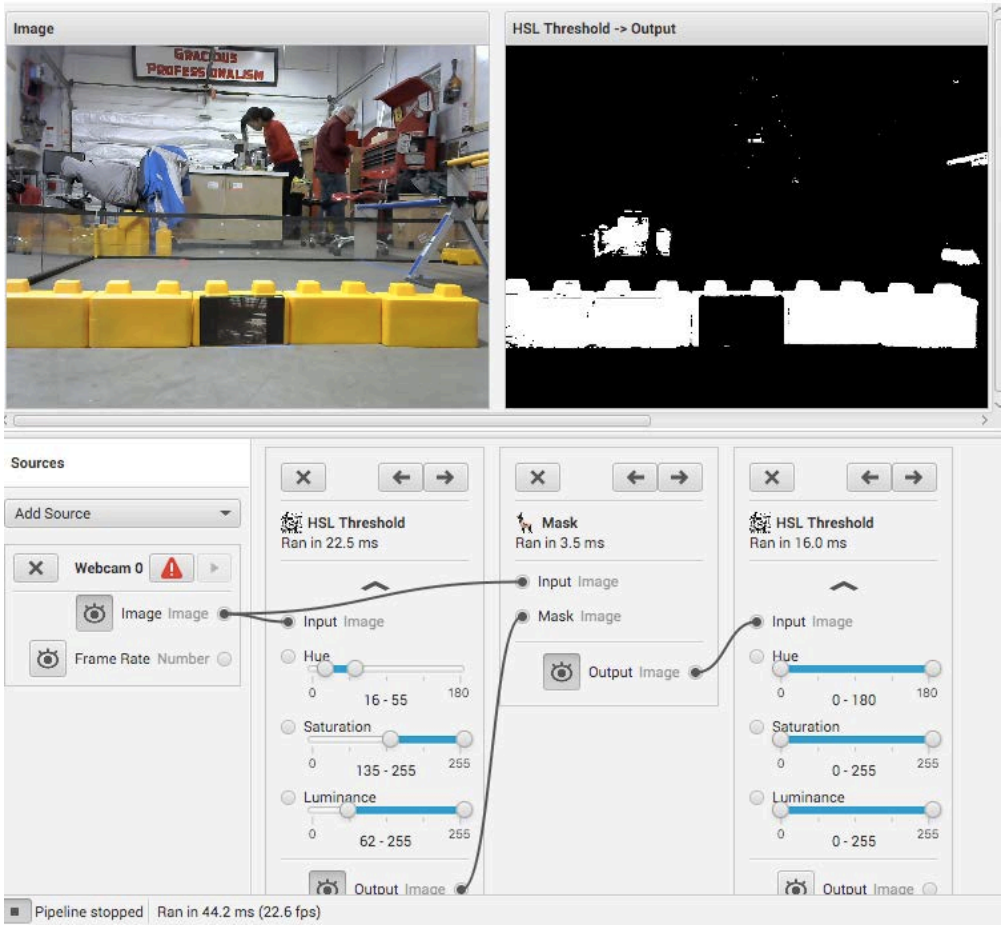
	Buttons on desk look to boring maybe putting them in a jar would help. **How did we fix it then?** **Why did it happen?** **How will we try to make it not happen again?**

	DECEMBER 20, 2019	7PM-9PM
Contributors: Calvin, Hannah, Kaylin, Lauren, Liam, Logan, Occie, Taylor		

Entries

1. Learning the OpenCV Pipeline with Grip to Detect (Sky)Stones for Auto

Identify	Brainstorm	Select	**Prototype**	Evaluate	Design	Fabricate



- The range for hue needs to be roughly 16 -55 to pick up the blocks ONLY

<table>
<tr><td></td><td>

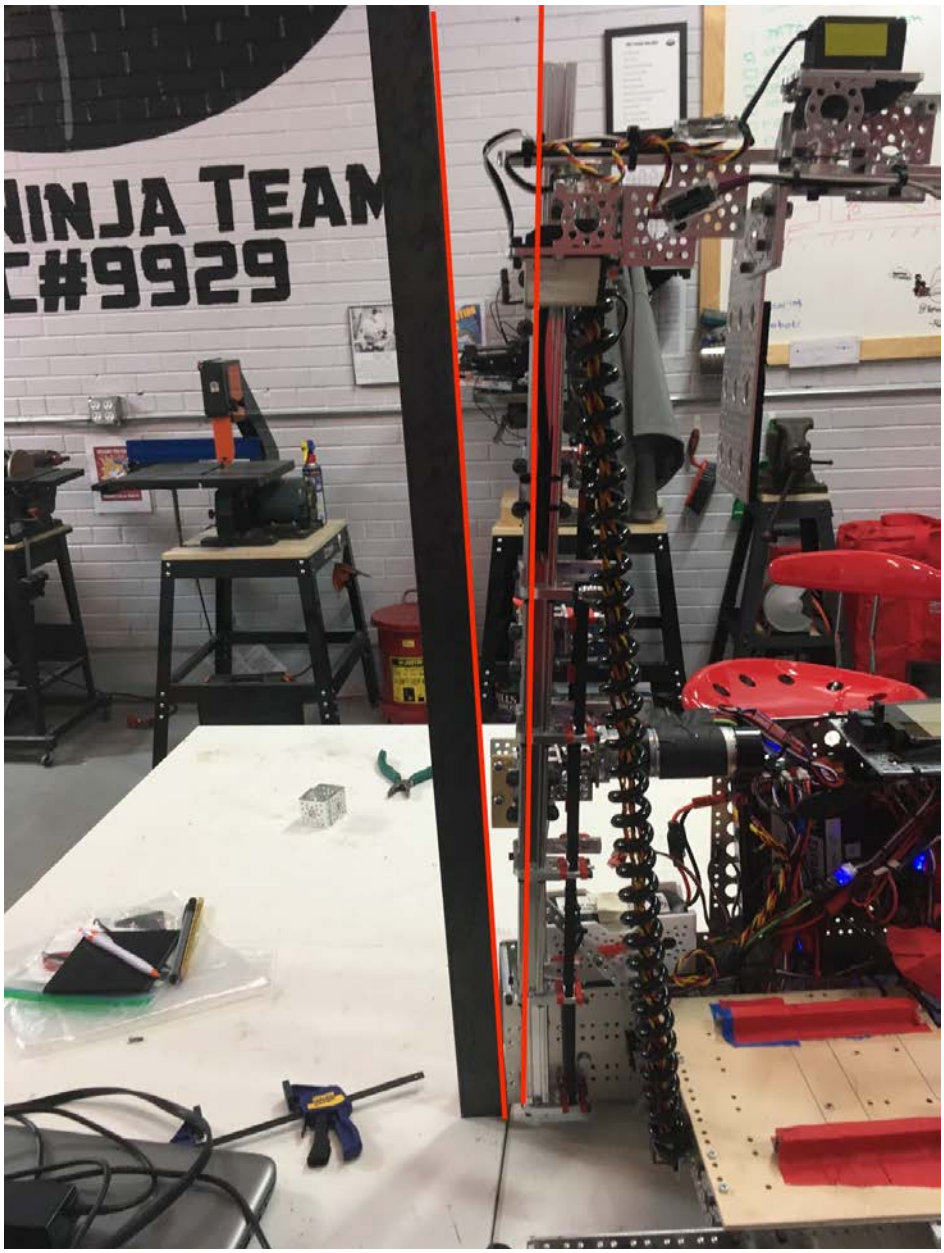
this is because having the starting value lower would cause the image to pick up red as well. In the RGB scheme yellow includes red, so you want to see some but not enough that you also see pure red when you want to see yellow

The camera needs to be about 10 inches above the field and a hood may also be needed to block out the stack of blocks that will be by the human player.

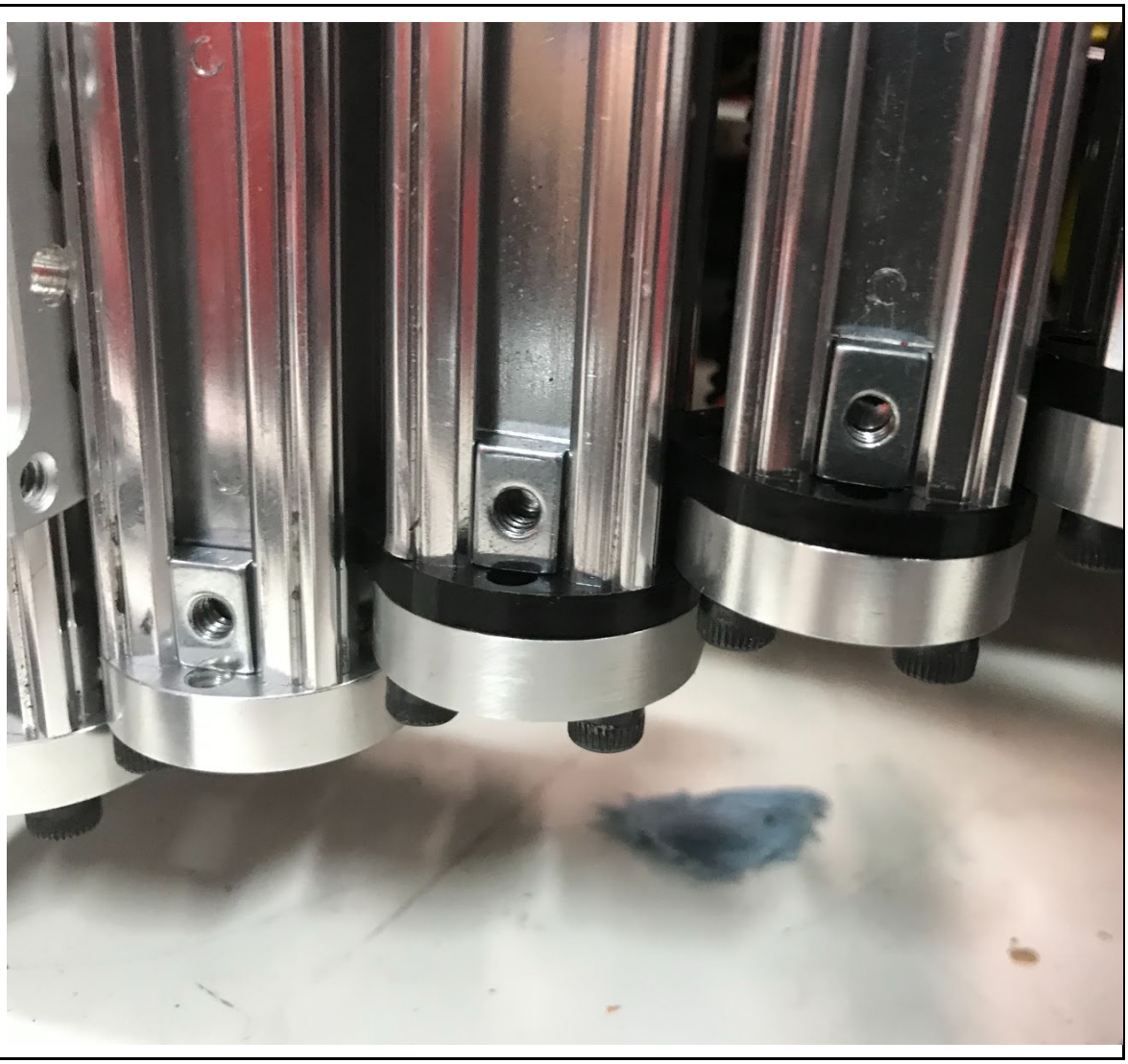
possible mask could be a image with a blank background that we import that has the top masked off. we would have to test this. it would be added as a source when we take inputs.

Luminance is 62-255
Saturation is 135- 255

[Lauren]
</td></tr>
<tr><td>Making the Lift more Stable/Reliable</td><td>
<table>
<tr><td>Identify</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>Design</td><td>Fabricate</td></tr>
</table>
</td></tr>
</table>


INJA TEAM
C#9929



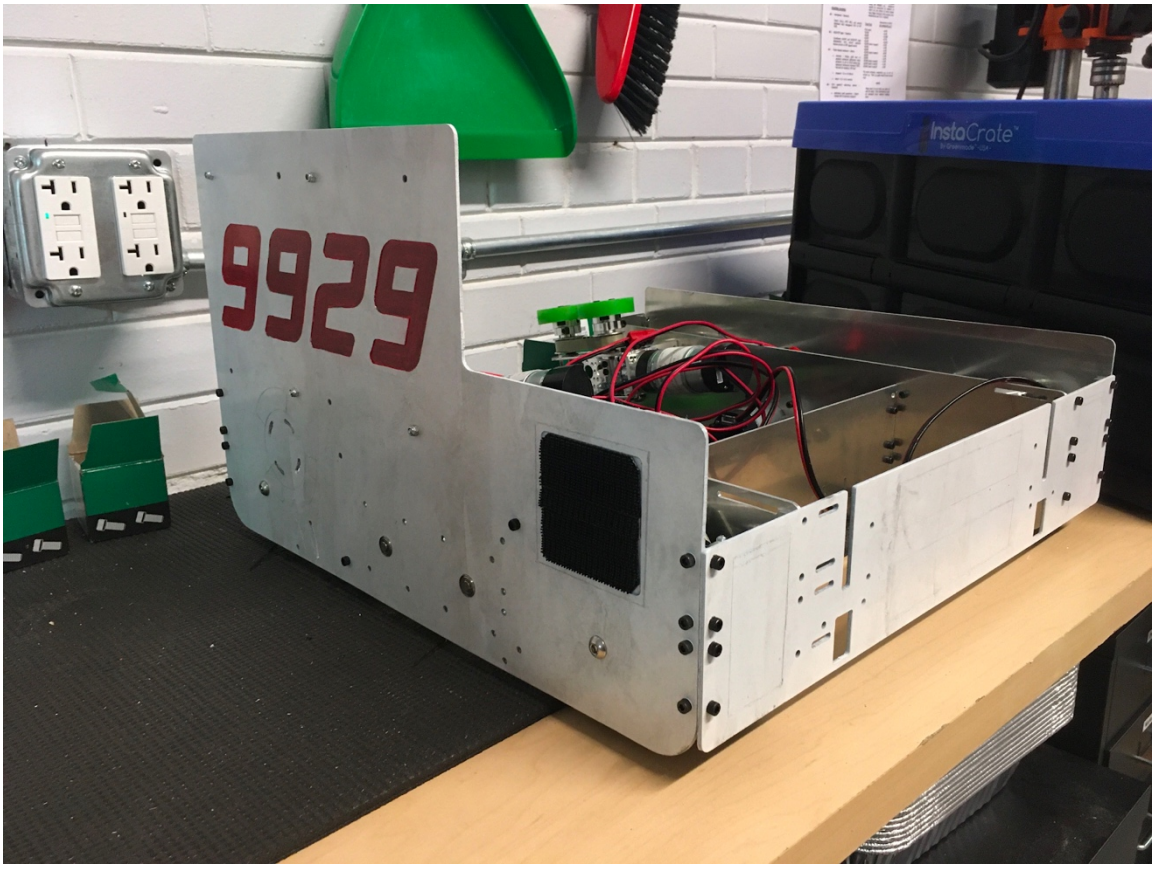
	DECEMBER 26, 2019	3:30PM-5PM
Contributors: Calvin, Hannah, Kaylin, Lauren, Liam, Logan, Occie, Taylor		

Entries

1. Assemble V2 drivebase after Painting

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

This afternoon we re-assembled V2 of the drive base after painting the outside of the structure - which is designed to look like a droid from Star Wars.

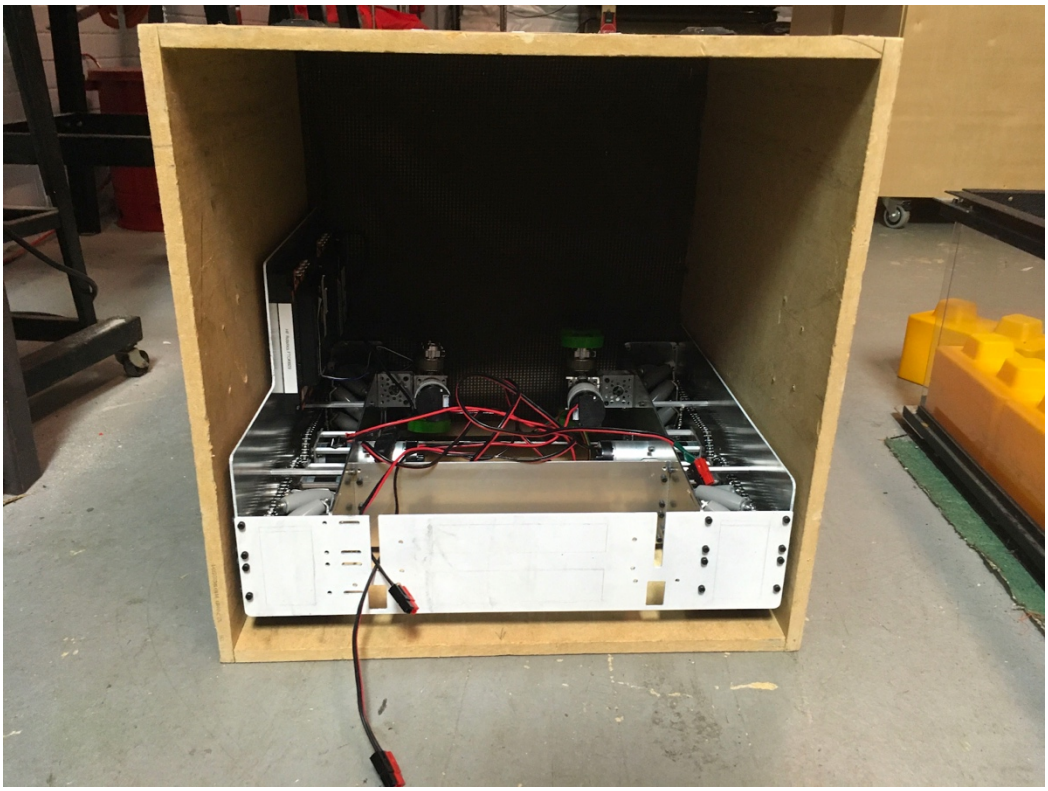


A few issues were found during assembly:

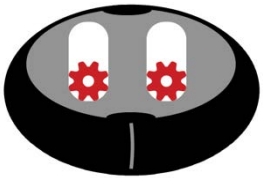
- There was a screw for the intake that was rubbing the left rear wheel, this was moved to match the way it was installed on the opposite side.
- The flanged bearings on two wheels were not assembled completely, and were not flush. We noticed this because the number of spacers required as shown in the CAD model would not fit. This happened on one side on one wheel, both sides on the other:

- We did not have enough ½" spacers, so some were cut from spacer stock.

We were concerned that the robot might be too wide, but it *just* fits in the sizing box:



[Kaylin]

	**JANUARY 3, 2020**	3:30PM-5PM
Contributors: Calvin, Hannah, Liam, Logan, Occie		

Entries

1. Put on new Servos and program them in

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Today I began the process of adding something to move the foundation. I added two servos to the hardware map as the first step, then proceeded to test the two servos for the correct alignments for our needs. After finding that the two servos were in working order, I then got to work on programming a button push (the A button on the driver gamepad) to move the two servos into positions. This has yet to be finished, but will be soon, for I will be working on it over the weekend. This will be achieved by creating a foundation grip mechanism class.

[Calvin]

	JANUARY 5, 2020	4:00PM-6PM
Contributors: Calvin, Hannah,Lauren,Kaylin,Taylor, Jermey, Tarendran, Logan, Occie		

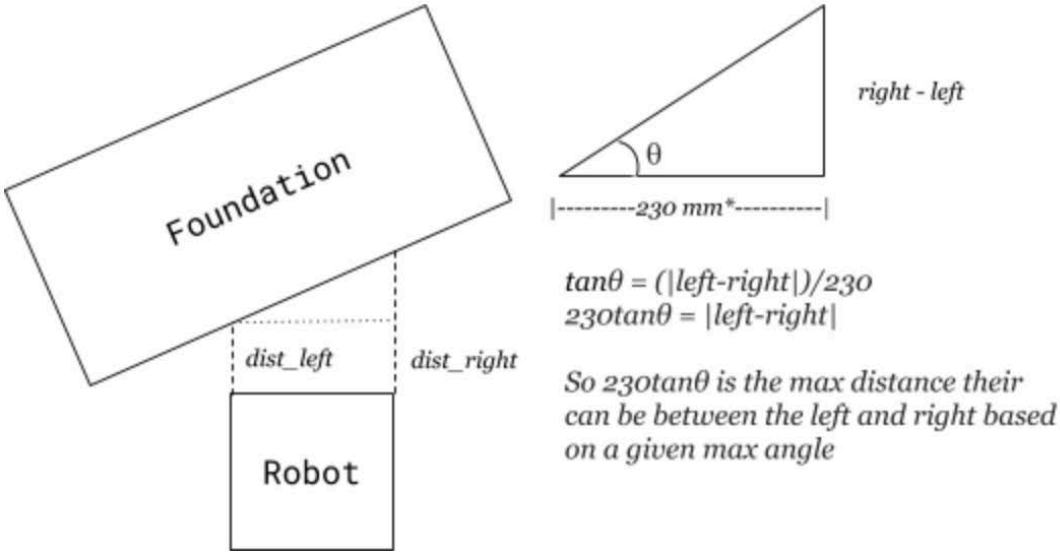
Entries

1. Programed a foundation alignment tool and tested via software

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	**Fabricate**

Today we wrote a class called stationKeeping, that allows us to line up with the skystone foundation. To to this we used trig to calculate the max difference between the left and right side that we would allow based on a max angle. We did this because we found that the sensor has inconsistencies at extremities of distance. We want to only operate this section of code if we are in the optimal range of the sensors and are within a certain angle of the base. (it would be more efficient for the driver to line up a bit better before using this software over a certain angle, currently 45 degrees from parallel) It is meant as a fine tuning tool.

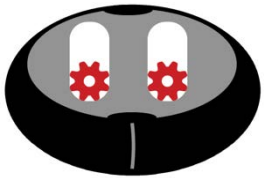
The math we used is shown below:



**230 is distance between the two sensors*

	The max angle is calculated so we can tell the robot to not bother running this set of code to avoid running it when we are not positioned by a foundation. We also have min and max distance that are not reliant on the angle because of the limit of the sensors found via testing. **[Lauren,Calvin]**
2. Gripper positions	<table><tr><td>Identify</td><td>Brainstorm</td><td>Select</td><td>Prototype</td><td>Evaluate</td><td>**Design**</td><td>**Fabricate**</td></tr></table> Today the programing team also iterated on the foundation gripper code. We added an additional position of the gripper. This position is called initPos in the code. It is what the old “up” position was. We added this because we need this mechanism to get out of the way of the cables. The new “up” position would allow us to be clear of the cables without infringing on the size limit that we are required to meet at init. **[Lauren]**

3. E.N. Strategies	<table><tr><td>Identify</td><td>**Brainstorm**</td><td>Select</td><td>Prototype</td><td>**Evaluate**</td><td>Design</td><td>Fabricate</td></tr></table>
	Tonight we discussed more effective ways to write E.N. entries. We talked about the idea of having people who did not work on a specific thing write an entry on that thing in order to help team members work on their listening skills and note taking skills as they interview others to write their assigned entry. We also talked about some possible writing prompts to make sure that everything we want to say is captured in each entry. The prompts discussed were:
	• “What did we talk about?”
	  ○ “Why did we talk about this?”
	• “What went wrong?”
	  ○ “How did we fix this?”
	  ○ “What steps did we take to fix this?”
	• “Why did we change x?”
	• “What did we do?”
	  ○ “Why did we do this?”
	  ○ “Why did it need to be done?”
	[Kaylin]

	**JANUARY 10, 2020**	7:00PM-9PM
Contributors: Calvin, Hannah,Lauren,Kaylin,Taylor, Jermey, Tarendran, Logan, Occie		

Entries

1. Foundation moving auto and pre-meet tuning [4:30-6]

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	**Fabricate**

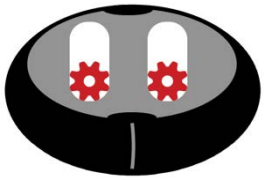
Before the meeting time today one of the programing team members came in to complete an autonomous with the capability of moving the foundation. Most of the distances were already measured before this by team member, Kaylin. Thanks to our sequential state code, which allows us to add states in a much quicker fashion than the normal java method. The building of the state machine went very quickly. We also made a runnable state, which allowed us to call one method from another class into a state instead of re-implementing the entire class into autonomous.

In the end we had to use only strafing and driving instead of our original design due to complication of time and code. Because of this we had to modify some values. The amount to do so was determined by testing of the code on robot. We also had to use untrue turn values as we didn't have time to calibrate the gyro sensor.

We also chose to fully implement the alignment code as, when we tested it it worked well. As the programing team we decided that it would be less time consuming to do the minor pid control alterations then to disable the code for this section. Also it was really cool to us when it worked, so that might have been a factor.

We also made an alteration to the smoothing feature of the drive code. We told the curve to smooth the slows less as the robot is heavier and has a slight natural slowing to it because of inertia. We did this as the driver mentioned a feeling of the robot "lagging" which was making driving less efficient.

[Lauren]

	**JANUARY 11, 2020 (LEAGUE MEET #2)**	7AM-2PM
Contributors: Calvin, Conner, Ernest, Habtamu, Hannah, Kaylin, Lauren, Logan, Occie, Tarendran, Taylor		

This entry was created by information collected during the league meet, and authored together as a team at our normal practice scheduled for the following day.

Entries

1.

Identify	Brainstorm	Select	Prototype	Evaluate	Design	Fabricate

What Happened

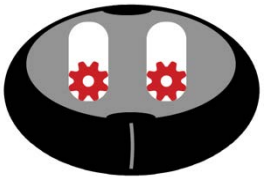
Capstone weight was too heavy to stack with a tower of three.

How did we fix it then?
We just capped on two.
Why did it happen?
The material we used for the capstone was too heavy. Lift doesn't reach far enough.
How will we try to make it not happen again?
Use longer wires in the lift, and use lighter material for the capstone.

What happened?
Auto started on tele-op.
How did we fix it?
Cleared wires away from phone.
Why did it happen?
???
How will we try to make it not happen again?
Check for wires on phone.

What happened?
Stone was tipped over when the alliance delivered it.
How did we fix it?
We collected the tipped over stones and tried to stack them.
Why did it happen?
Alliance partner was unaware that stones needed to be right side up.

	How will we try to make it not happen again? **Talk to alliance before, while scouting, and during the match.**
	What happened Knocked over completed towers before endgame. **How did we fix it** **Why did it happen** Operator/driver error--Moving foundation--rushing for the capstone-- **How will we try to make it not happen again** Practice certain scenarios-- capstone redesign--software team help with moving foundation.
	What happened Servo cables came unplugged **How did we fix it** Plugged it back in **Why did it happen** ??? **How will we try to make it not happen again** Design a tool to make sure all servos are plugged in--before match make sure that all servos are plugged in.
	What happened Non GP towards alliance **How did we fix it** **Why did it happen** We were partnered with teams that we though their not as good. **How will we try to make it not happen again** Do the best we can with whatever team we are partnered with.--Team needs a well thought out plan that includes auto and teleop.

	**JANUARY 17, 2020**	7:00PM-9PM
Contributors: Kyla, Conner, Occie, Malcom, Tarendran, Hannah, Habtamu, Kaylin, Lauren, Liam, Logan, Calvin,Ernest, Taylor, Jermy		

Entries

1. Continuing Chaos

Identify	Brainstorm	Select	Prototype	Evaluate	**Design**	**Fabricate**

Today software continued working on chaos. We made the functional side of what we call “Level one” Chaos Ninja. Level one chaos ninja cause the robot to occasionally fail in minor ways. It is the easiest level of Chaos that causes “errors” to occur when run. Note, that these “errors” are intentional and are pieces of software we have created to mimic failure we have seen on the field. We created them this way as we want the drive team to practice with Chaos to prepare for failures during competition and simulate some of the stress felt at competitions

We also added some messages to go along with the selected errors. We did this so we could identify what “error” was run so we can avoid missing actual errors that have occured when a chaos “error” was run if the real error or failure looked similar to that of a Chaos “error”

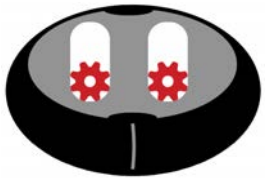
[Lauren, Calvin]

2. V1 Control Hub Testing

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Tonight, we got V1 back up and running in order to make our drive practice more realistic and challenging. Being able to have V1 on the field with V2 makes practice more realistic because there is another robot trying to collect stones on the field. It also ensures that two drivers can practice at one time. In order to do this, we needed to rewire the drive motors and hook up the control hub. Unfortunately, the color coding on the wires and the hardware map did not line up, so we had to figure that out. We also had to figure out how to get our software onto the control hub, as well and wire it up. This is important because we might want to utilize the control hub for next year’s game.

[Kaylin]

	**FEBRUARY 14, 2020**	7:00PM-9PM
Contributors: Occie, Malcom, Tarendran, Hannah, Habtamu, Kaylin, Lauren, Logan, Calvin, Ernest, Taylor, Jermy		

Entries

Refactored Autonomous Skystone Trajectories

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We refactored the skystone autonomous to improve the trajectory values.

[Lauren]

Improved Skystone Detection Time

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We found way to improve detection time (such as looking for a difference between the first and second greatest total of black calculated by our algorithm). We implemented these changes to make our autonomous more efficient since time is very limited during this time.

[Lauren]

Added Inner/Outer Routes for Skystone

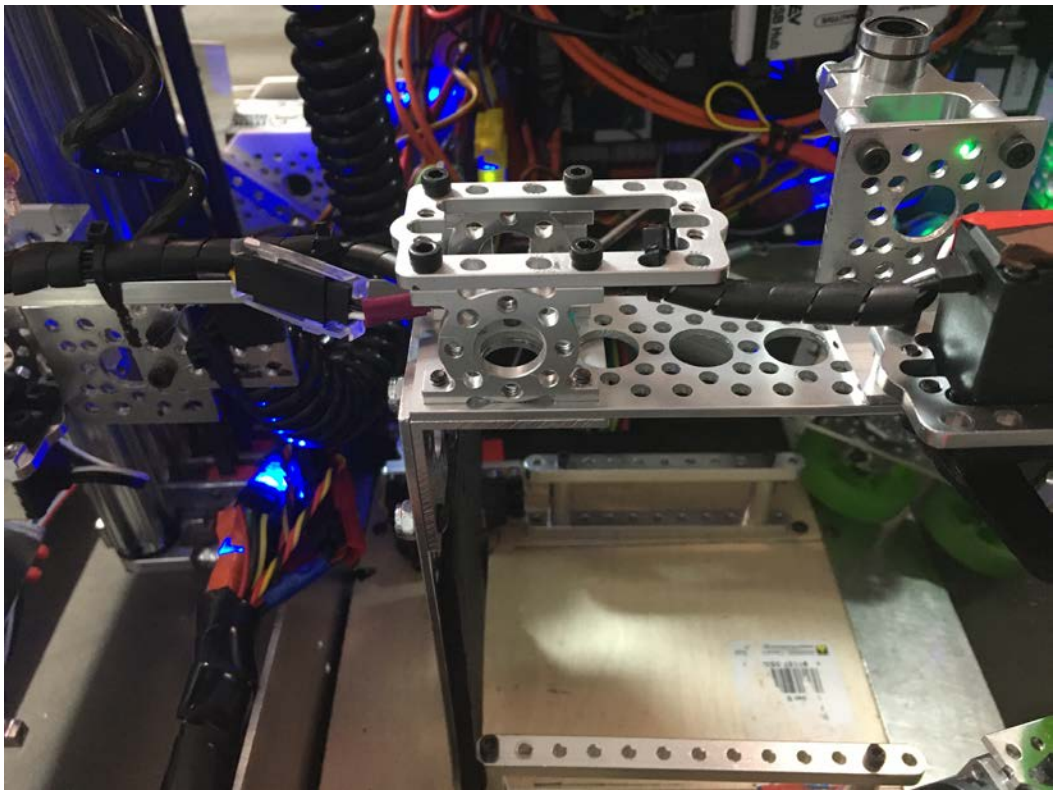
Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

We added the option of inner and outer routes, also called wall and bridge in reference to where they travel closest to. We did this to make our robot more compatible with alliance members going forward.

[Lauren]

Removed Wrist Servo and Code that Referenced It

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

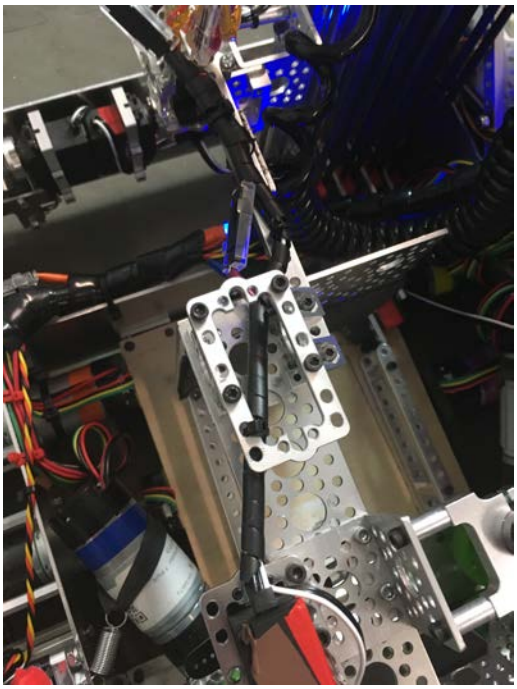


The original delivery arm had a wrist to turn the stone 90 degrees to do a brick laying pattern. We haven't used this technique, and needed the wires for the capstone dropping servo. We left the servo in the mechanism dead, and the wrist mechanically locked for the league qualifier, but removed it to simplify the structure.

[Kaylin, Lauren]

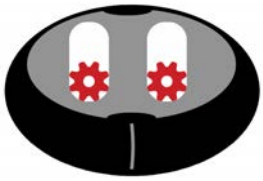
Started to improve wiring with spiral wrap

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**



As the season progressed, we realized the wiring was a mess. We discussed different methods to control the wiring. We settled on using spiral wrap to organize the wires. We chose this method because it is a quick and effective fix. Putting this spiral wrap on the wires made the robot look cleaner and improved safety.

[Kaylin]

	**FEBRUARY 16, 2020**	4:00PM-7PM
Contributors: Occie, Malcom, Tarendran, Hannah, Habtamu, Kaylin, Lauren, Logan, Calvin, Ernest, Taylor, Jermy		

Entries

Improved Autonomous Foundation Move - Added Parking

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Added parking to our foundation move program to gain additional 5 points during autonomous. We made the turning more precise, and added parking near the wall or the bridge.

[Lauren, Calvin]

Added Support for Ejection Servo

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

This servo didn't exist before, so it needed code. We wrote the code in such a way that the servo is operated automatically when the intake spins in reverse, and we added safety code so that the servo only moves when it won't damage the lift or the servo.

[Lauren]

Improved Safety of Stone Ejection

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

Don't get penalties during matches (eject slowly):

```
@@ -153,11 +153,28 @@ public class DeliveryMechanism {
     }

     public void setIntakeVelocity(double velocity) {
+        if (velocity < 0) {
+            if (!unsafe.isPressed()) {
+                velocity = -0.2;
+            }
+        }
+
```

Don’t allow stone ejection when lift is not in a safe state/position:

```
        if (velocity < 0) {
-            ejectorServo.setPosition(EJECTION_EJECT_POSITION);
+            // Safety - there's only certain states
+            // where it is safe to eject, namely LoadingState,
+            // or when position is above some certain height, let's
say 1/2 of the way
+            // up?
+
+            if
(getCurrentStateName().equals(LoadingState.class.getSimpleName()) ||
+                    liftMotor.getCurrentPosition() >
LIFT_MAX_HEIGHT_POS / 2) {
+                ejectorServo.setPosition(EJECTION_EJECT_POSITION);
+            } else {
+                telemetry.addData("DM", "Eject servo - not safe");
+                ejectorServo.setPosition(EJECTION_STOWED_POSITION);
+            }
```

[Lauren]

Improved Cycle Time of Delivery Mechanism Stow Code

Identify	Brainstorm	Select	Prototype	Evaluate	Design	**Fabricate**

While practicing for the state tournament, the drive team noticed that the delivery mechanism would cycle the gripping finger while stowing, when really it could just leave it extended. This was a bug left over from v1 of the physical robot, where we actually needed this buggy behavior to clear a part of the superstructure of the robot:

```
@@ -1233,7 +1252,7 @@ public class DeliveryMechanism {

        @Override
        public State doStuffAndGetNextState() {
-            gripBlock();
+            ungripblock();

            stowed();
```

[Lauren]

Business/Sustainability

This season we were fortunate to be awarded a grant from Schneider Electric once again.

We have a lot of costs that Schneider helped cover:

- Power Tools ($400)
- Robot parts and spares ($1400)
- Competition Registration ($275)
- Robot Game Elements ($450)
- Food and travel ($300)
- Pit-banner sign ($25)
- Stickers/buttons ($75)

The Homewood Science Center has been invaluable to us for many seasons. They provide us with space to work and have introduced us to our two sponsors. We give back by volunteering at their events, which in turn helps us with STEM outreach.

We continue to reach out to our sister FLL teams, helping them with programming, listening to dry-runs of their research presentations and adding functionality to their space. We believe it is important to maintain a relationship with our sister FLL teams because over 50% of our FTC team are former FLL participants from those teams.