List of Practices

1. 9/9/17
2. 9/17/17
3. 9/22/17
4. 10/16/17
5. 10/20/17
6. 10/28/17
7. 11/3/17
8. 11/11/17
9. 11/17/17
10. 11/19/17
11. 11/10/17
12. 11/20/17
13. 11/21/17
14. 12/27/17
15. 12/28/17
16. 12/29/17
17. 12/30/17
18. 12/31/17
19. 1/2/18
20. 1/7/18
21. 1/8/18
22. 1/11/18
23. 1/12/18
24. 1/13/18
25. 1/17/18
26. 1/20/18
27. 1/22/18
28. 1/25/18
29. 1/27/18
30. 1/30/18
31. 2/2/18
32. 2/7/18
33. 2/10/18
34. 2/15/18
35. 2/16/18
36. 2/17/18
37. 2/20/18
38. 2/20/18
39. 2/21/28
40. 2/22/18
41. 2/23/18

2017-18 Notebook

September 9 Kenny, Ethan, Andrew, Myles
Observed the FTC launch at UVM Billings Lecture Hall.

Team made some preliminary goals of being able to place some glyphs, drive on and off the balance board.

9/17/17-Ethan, Andrew, Myles, Kenny
Mark ran a session where the team watched the launch video.

9/24/17 Ethan, Andrew, Myles, Kenny
Mark ran a session where the team worked to trouble shoot PTC Creo.

10/16/17-Andrew, Kenny, Luke, Myles
Team worked on robot dissection and learned some basics of brazing.

Dissection Introduction
https://youtu.be/rlPnx8uRZ4g

Dissassembly of External Power P1
https://youtu.be/bZpR1YIXc8o

Disassembly of Extern Power Part 2
https://youtu.be/LpG1nLymG5g

Disassembly of Extern Power Part 3
https://youtu.be/dOn1v2YXZSQ

Removeal of Hopper Assembly
https://youtu.be/b6HdUoQmaq0

Removal of Shooter Assembly
https://youtu.be/3EsSq2R1uqE

Removal of Wheel Assembly
https://youtu.be/tqDF_ADQsvQ

Dissassembly of Sweeper
https://youtu.be/pHvTOCv17w4

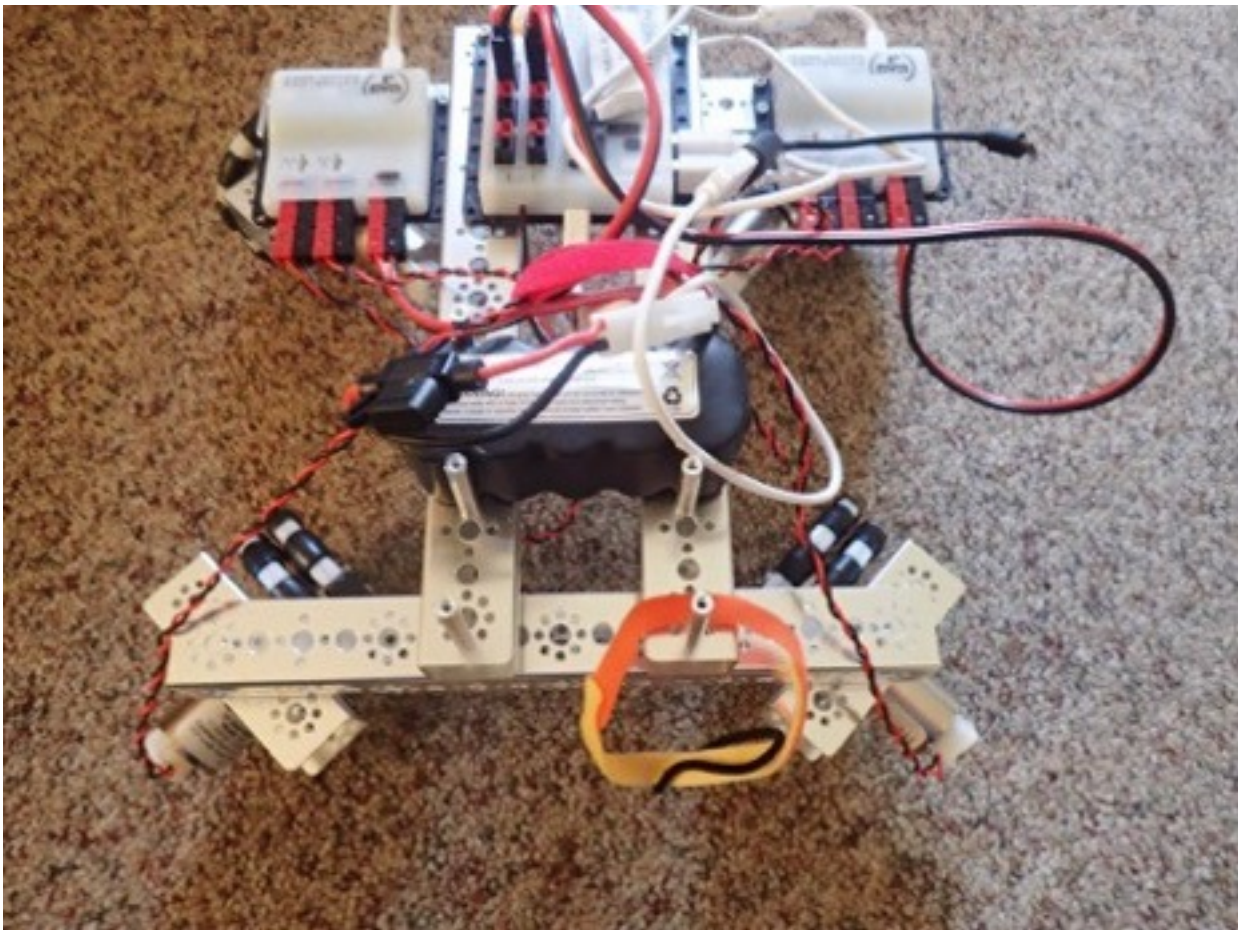
Dissassembly of Shooter
https://youtu.be/4QFogrofRgc

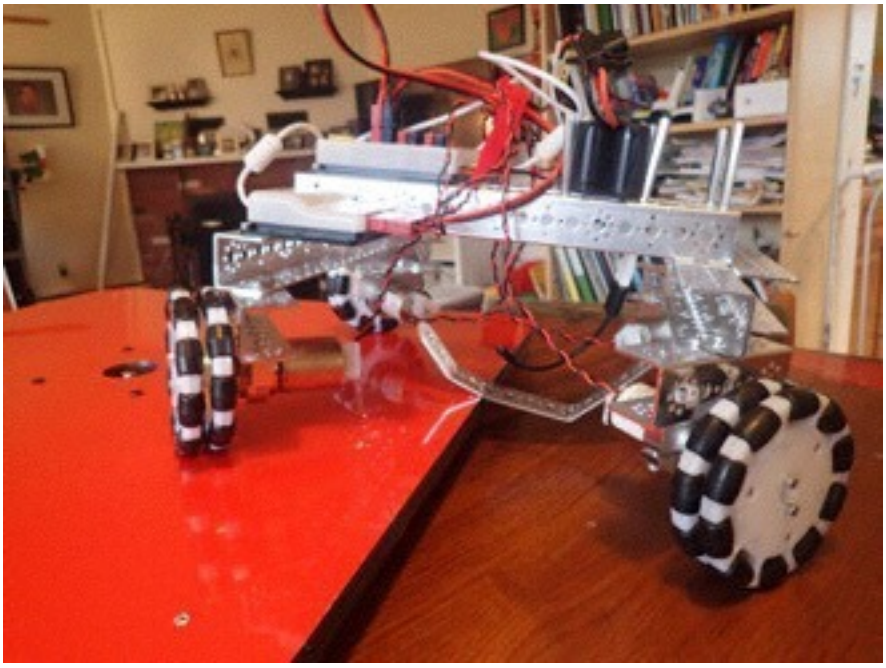
10/20/17-Andrew, Kenny, Luke, Myles
Team assembled the balance stone in order to test omni wheel designs. The team determined that the motors, when mounted for omni wheel driving, hit the balance board and prevented the robot from riding up the stone. The team discovered, during testing, that the motors hit the balance board on the rear of the drive. Several efforts were made to power the robot through the obstacle by increasing power through the program but this only resulted in damaging the balance board.

Demonstration of Driving Algorithm with Omni Wheels
This worked once by chance but it took dozens of trials to determine the steps, which took about 40 seconds to operate. The algorithm was to drive the front tires on the balance board.

https://youtu.be/1q8mN-XvhZU

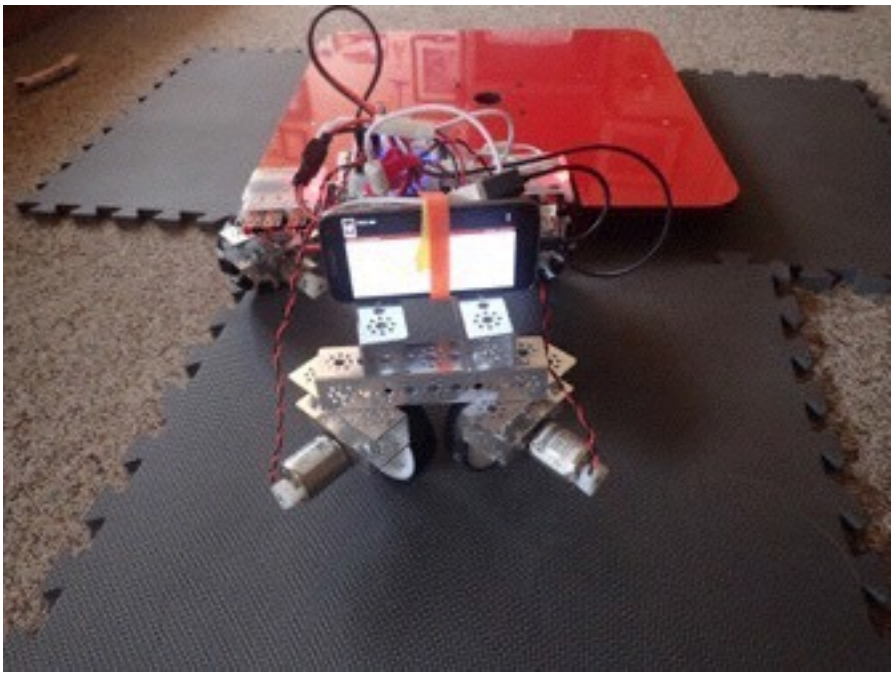
Several different robot designs were also attempted that placed the motors in different positions to prevent hitting the balance board before the wheels hit the balance board. In this design, the back wheels were turned inward. This enabled the wheels to hit the balance board first but they did not have enough traction/power to climb the board.



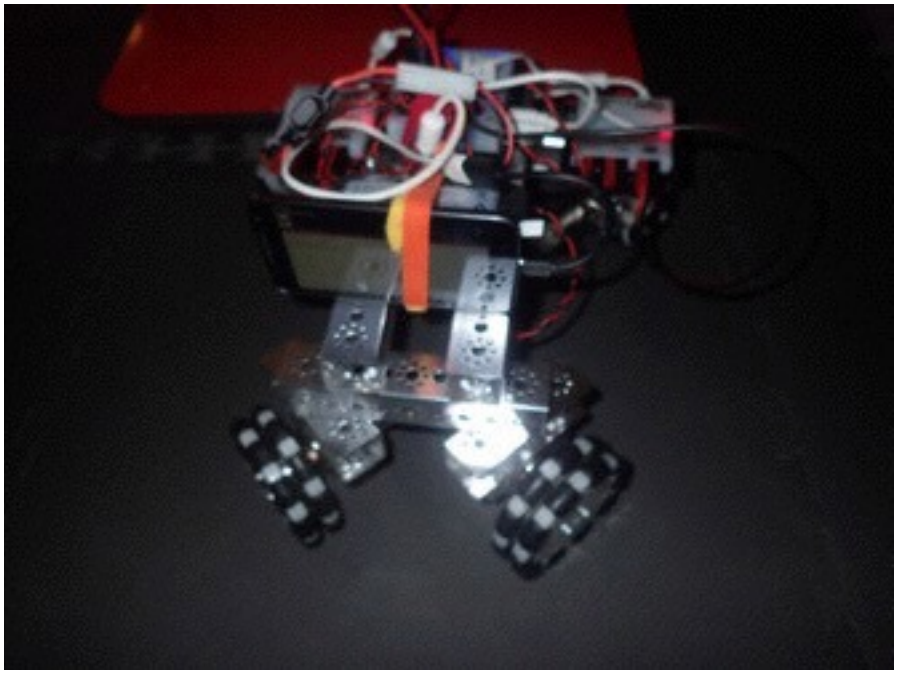


A bumper was added to the belly of the original design to keep the balance board down so that the rear wheels could drive onto the board.
However, this design also did not work.

A sprocket design was explored but it was rejected because the wheel mounts were roughly the same size as the motor and it was too complicated of a design.

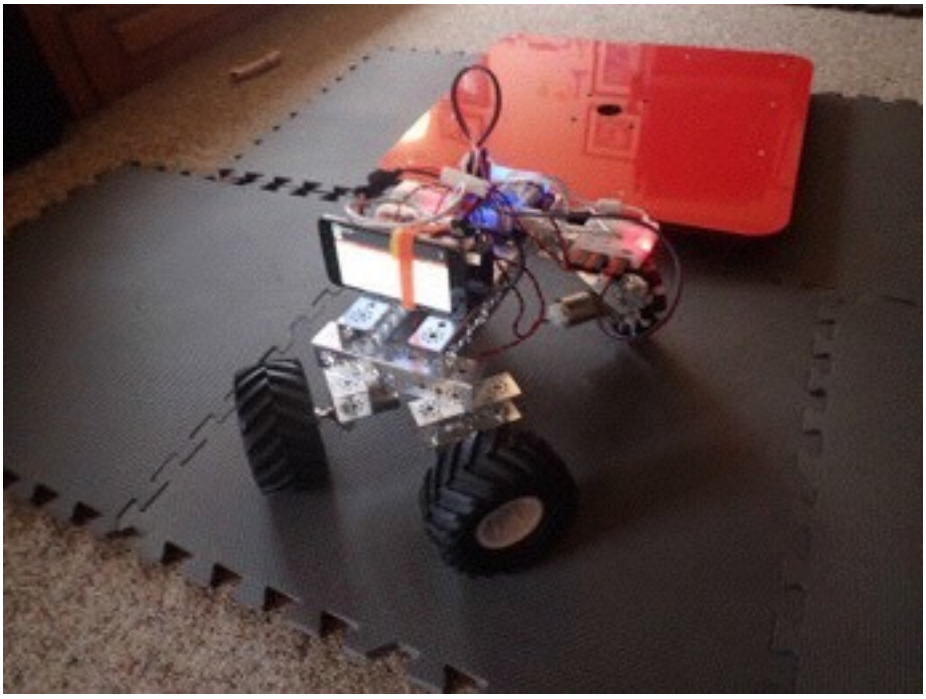


Another effort to get the robot up the balance board was to reduce the size of the rear of the robot. This would allow the robot to pivot onto the balance board with less space and it would be easier to keep it on the balance board.



This initial design was tested with the standard wheels to see if the increased rubber would help with the traction going up the balance board. It did not work.

The design was tested with the omni wheels to see if they would work better. They did not.



The design was testing using All Terrain Tires in the back of the robot. The robot easily drove up the ramp.

A verision of the robot was constructed using four All Terrain Tires and the original frame. this design worked great for going up and down the ramp. However, it did not drive in straight lines easily and there was loads of pressure on the mounts because of the forces acting on the robot.

Demonstration of the Omni Wheel Design with All Terrain Tires
https://youtu.be/8RKiHNsZYuI

10/28/17
Team tested some omni wheel designs, determined that mechanim wheels were needed.

11/3/17
Team testing of robot using All Terrain Tires to Drive Up Balancing Stone
https://youtu.be/UiHGIeLgkBI

Demonstration of robot using All Terrain Tires to drive-produced a wiggle
https://youtu.be/8RKiHNsZYuI

Demonstration of 80/20 Aluminum Extrusion using lego pulleys and tetrix for frame and motors. https://youtu.be/Nz3SdZ0Zssc

Team tested mechanim wheels and determined that wheels would work well for problems. https://youtu.be/X-5RuLmbiXk

11/3/17

Team testing of robot using All Terrain Tires to Drive Up Balancing Stone

https://youtu.be/UiHGleLgkBI

Demonstration of robot using All Terrain Tires to drive-produced a wiggle

https://youtu.be/8RKiHNsZYuI

Demonstration of 80/20 Aluminum Extrusion using lego pulleys and tetrix for frame and motors. https://youtu.be/Nz3SdZ0Zssc

Team tested mechanim wheels and determined that wheels would work well for problems. https://youtu.be/X-5RuLmbiXk

11/11/17

Practice 5

The team tested the gripper/lifter prototypes today.

The first tests were of the linear lift using the 80/20 extrusion using the smallest gear on the motor and the largest gear on the axle that was pulling the rope.

The team started testing with a single block. The team determined that the screws that were used to attach the long flats hit the blocks so that the entire surface was not gripping the blocks. The team discussed that they could solve this problem in serveral ways.

The team thought they coudl braze the flat to the outside connector. This was tabled at the moment.

The team also thought that adding more screws would increase the number of contact points. The team continued with this idea and added three more screws to each side. This enabled the grabber to hold onto (2) blocks.

The team determined that at .2 for power, it took nearly 32 sseconds to lift the blocks to the highest position. This position was higher than the two block goal so this was considered a viable option.

The team then determine that maximum number of trips that could be accomplished with this arrangement, which was as follows:

Step 1. grab block, place it on top of another block. 10-15 seconds of lift and driving time.
Step 2. 5-10 seconds grab and lift both blocks and drive them into the scoring station.
Step 3. repeat steps 1 and 2 for next two blocks.
Step 4. 10-20 seconds-bring both blocks back to glyph station and lift them ontoop of the next two blocks.
Total time for a column-45 seconds or more

This could be done twice, potentially, in the tele-op time period.

The team also decided to test the gripper with the relic. The team predicted that the plastic relic was grabbed well but was not held when the grabber was lifted.

https://youtu.be/Andyc7JR4ko

https://youtu.be/0GhSTZkt4Lk

The team discussed adding foam or other materials to the grabber. The team started by adding two laytex swim caps so the gripper arms using elastics. This enabled the team to grab the relix and lift it.

https://youtu.be/ZF4Ky5I7hrE

The team tested multiple lifts and drops to determine whether or not the Relic would fly outside of the scoring zone. These trials enabled the team to conclude that the relic should stay in the target zone.

https://youtu.be/V-Vfg-akWR0

https://youtu.be/ppV5GbT8tt8

The team discussed how they might use a rack and pinion slide to move the relic further away from the target but the team decided to table that for the moment as they needed to address other issues with the robot.

The team decided to try to decrease the time of the linear lift. The team discussed changing the gear ratio of the lifting gears. The team also discussed changing the power to the motor. The team decided to start with the power to the motor because that was easiest thing to change.

Original Test
https://youtu.be/-AuKTKxQe9E

The team increased the power and determined that the lift time was 26 sreconds at a power of 0.4. This was an improvement and the motor did not appear distressed.

https://youtu.be/hctthkGctcg

The team conducted a proportional analysis of the two data points and determined that for each .2 change in power, the team would save 6 seconds of lift time.

The team then tried .8 of power and determined that the lift time was 18 seconds. This was a large improvement. Since the motor was not hot, the team decided to try full power. At full power, the lift time was closer to 12 seconds.

https://youtu.be/21n8ypKVqp4

The team decided to give the rack and pinion a test as well. Testing was more difficult because the team had already used every channel they owned.

When the team started testing the rack and pinion lifter, they noticed that the gripper had difficulty holding the blocks.

The team modified the placement of the gripper on the lift so that the team could grab the block at a lower position.

This did not correct the problem, however, and the team discovered that one of the servos had some lose. When it was replaced, the team tested the lifter and determined that the time to lift two blocks to a standard height was less than 12 seconds, which meant that it was faster than the linear slide.

https://youtu.be/qxuwh-Hq35Y

During this process, however, the team discovered that the slide got stuck because of the position of the fastners relative to the motor hub. This was corrected by moving the gears and adding spacers so that the moving parts were not hitting each other.

The team then tweaked the gear ratio of the lifter, which enabled it to lift the two blocks to the proper height in around 6 seconds.

https://youtu.be/BjqZF475EAc

Several times the slide was stuck during use. This is because the sliders rotated slightly.

The rack lifter appears to take up less space than the linear lift. The rack is also mechanically more simple so that there are fewer failure points. The rack also appears to weight less because it has few parts, meaning the motor can be used at a lower gear ratio for higher speeds.

If the team chooses the rack set-up,the motor is positioned relatively high on the channel so it might make sense attach sprockets to the assembly and use chains to transmit the power. This will lower the center of mass for driving the robot up and down the balance board.

Ethan and Luke also assembled the glyph holder using rivets.

Ethan and Andrew drove the robot chassis with the machanim wheels.

The team discussed the autonomous phase and how the team might be able to write a java class that will allow a team member to drive the robot in teleOp mode while recording the encoder values and the x,y,z coordinates into a data file. Another class would be written, in autonomous mode, that loaded the data and used it to drive the robot. There would be branch in the programming loop for the left, center and right positions based on reading the image target.

The team also disucssed members working on more specific physics knowledge and programming knowledge to make their presentation stronger.

Next Time:
Additional Channels, motors with encoders and rack/pinion set-ups were ordered and should arrive on Monday, 11/13.

The team will want to do some design work to determine the chasis design and whether it will be composed of channels, 80/20 extrusions or another design. The team will want to replace the motors with the motors with encoders and the team will want to do some preliminary testing of the autonomous using the telemetry (writing data to files) in teleOp to get the robot to safety.

The team is likely to change the lifter to a sprocket and chain system is they select the rack and pinion. This will reduce the center or gravity. Likewise, they will adjust the placement of the base rack in order to optimize the lift height. The team will also want to optimize the placement of the grabber, relative to the blocks. (there was some discussion of extended the grabber arms to connect closter to the center of mass of the glyphs and to braze the final connect to optimize the surface contact. team also discussed using a rubber spray to coat the aluminum flat- something like https://www.lowes.com/pd/Plasti-Dip-11-fl-oz-White-Aerosol-Spray-Coating/3851549)

The team will want to choose the linear slide or the rack and pinion for the lifter. The team will want to do some teleOp testing of the composite system. The team will likely want to use the expansion Hub for the servos. The team will need to check if the robot can use the expansion hub and piggy back the expansion hub to the robot controller (with motor controller) to controll the motor on the lifter and the motor on the slider (later).

11/17/17

The team met today and focused on an integrated prototype and compared the prototype to a number of YouTube videos of working robots for Relic Recovery.

I would like for the team to have a practice on Saturday from 9-12, give or take. It would be great if the kids could watch the videos in the links before practice. There are loads of ideas in these videos, some of which are used in the prototypes.

https://youtu.be/ceDaZeTU5Y0

https://www.youtube.com/watch?v=w384EpOfDL4
Team Machanical Paradox
The team really liked this design. They were impressed by how the arm kept the glyph stationary. They also were impressed that the slide and lower functions for the relic were operated using a serperate assembly from the glyph arm. This robot also used a drive system that was similar to the chain drive tank system the team used two years ago. This robot usde a U frame.

https://www.youtube.com/watch?v=89S3aRfL7mY

The robot in this video appeared way over built. The noticed that both this robot and the previous robot used 6 wheel drive systems with a U frame.

https://www.youtube.com/watch?v=Hg35dr6h8x4&t=64s
Team Purple Fire
This team demonstrated three different robots. All of the robots used a U frame. These robots also combined a basic arm with a linear rack for lifting within the arm. The team was impressed that the arm could lift the glyph without the glyph rotating. This appeared possible because the arm seemed to be mounted at four points that were allowed to rotate. However, the four points of contact also held the orientation fixed. This team also used an “H” drive system with a wheel that was placed in the direct center of the robot to allow for strafing. The team was very impressed by the team’s approach to the relic at the end game. The team appeared to ram the wall and released the relic at the last moment. This gave the relic momentum so that it could land further along the landing strip.

https://www.youtube.com/watch?v=Trc1TeTBjkI

The team seemed very impressed by the cable housing that was used to protect the servo wires. The team was also impressed by the simple design and the placement of the control systems in a very small space. The team also really liked how this robot placed two glyphs at a time, which reduced the number of trips the robot needed to take to complete the glyph cipher.

At that time, I think the team should evaluate a prototype lift, grab and slide mechanism. I think this will take 10 minutes. A video of the prototype is in this link:

https://youtu.be/8enbV7vi9EY

The team is likely to want to do some work on the design and programming of the jewel detecting part of the autonomous programming. There will be a prototype for the team to evaluate. This will included a configuration file and a class that will assume the team is red. The class will also include instructions to move the robot to remove one of the jewels. This is likely to take 30 minutes to an hour. This is necessary to include the frame design.

https://www.youtube.com/watch?v=fXUxPL5s8QU

The team may also want to do some basic explorations of programming the autonomous using the Vuforia Localizer using a combination of sleep (programming for time and power) and encoders (programming for rotations). This would be preliminary work in order to establish protocols for programming the autonomous and to work as a proof of concept. I think this will take 30 minutes to an hour. This is also necessary bceause it influences the frame design.

https://www.youtube.com/watch?v=d0liBxZCtrA

The team is likely to want to explore how to integrate the Rev Expansion Hub with the Modern Robotics Robot Controllers. As of right now, we are going to have to run the Rev Expansion Hub for two motors, the sensors and three servos. The remaining (4) motors will need to be run through the modern robotics controller for two reasons: we cannot order an additional expansion hub because they are out-of-stock and we only have two adaptors for the encoders and we need (4). Rev thinks these materials may be in stock in late November. This process is likely to take 10-30 minutes.

https://www.youtube.com/watch?v=gBHW4kVQsiM

The plan for the team was to evaluate how the prototype functioned, how it could be improved and whether certain sub-systems were worth persuing.

In the last practice, the team had a functioning lifter/grabber and a slider. Since then, the lifter/grabber/slider was integrated into a single unit. This link has a demonstration video of the three systems, which were integrated together.

https://www.youtube.com/edit?o=U&video_id=8enbV7vi9EY

The OpMode that was used to control these systems focused on gamepad2 so that it woud not interfere with the robot drive systems, wihch were mapped to gamepad1.

The tetrix max motors were replaced with TorqueNado motors in order to use the built-in encoders for navigation during the autonomous period. The team made this decision after learning that the encoders of the tetrix max motors were no longer being produced.

In order to mount the three integrated systems to the robot chassis, the frame had to be modified in several ways.

First, the front and back of the frame was modified to allow for placement of the control systems and for the hardware to connect together.

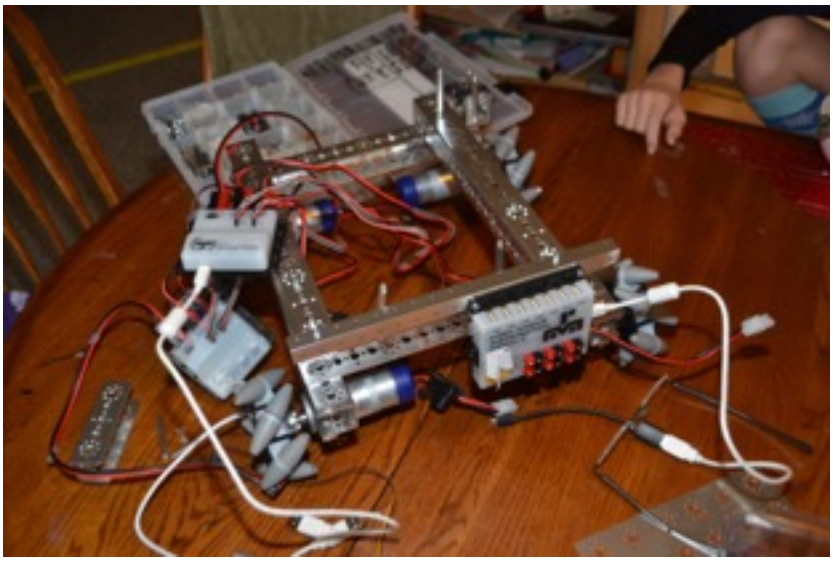
The three integrated systems were mounted to the chassis by conecting the fixed post of the lifter to the chassis at a level that was roughly equal to the motors. This was done to allow the robot to continue to climb the balance board.

The placement of the fixed lifter post required the grabber mount to be changed. It was too high, so it was lowered by attacking a flat tetrix piece to the original mount and then re-attaching the grabber at a more appropriate position, which was also lower on the robot.

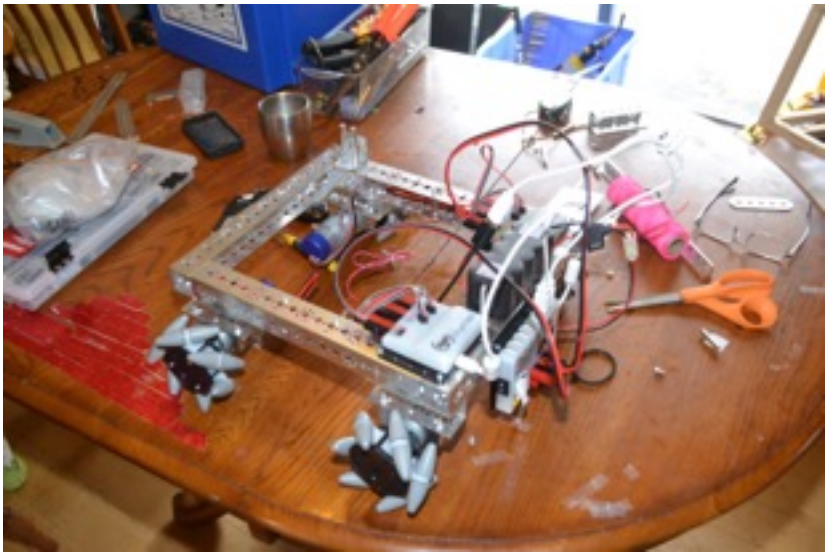
The rear of the robot was modified to allow for the placement of the controllers. With the integrated systems added to the robot, there was a total of 6 motors and three servos. This required room for two robot controllers, the servo controller, the modern robotics power module, the battery and the rev expansion hub (for the other two motors).

A number of different configurations were attemped, including the following:

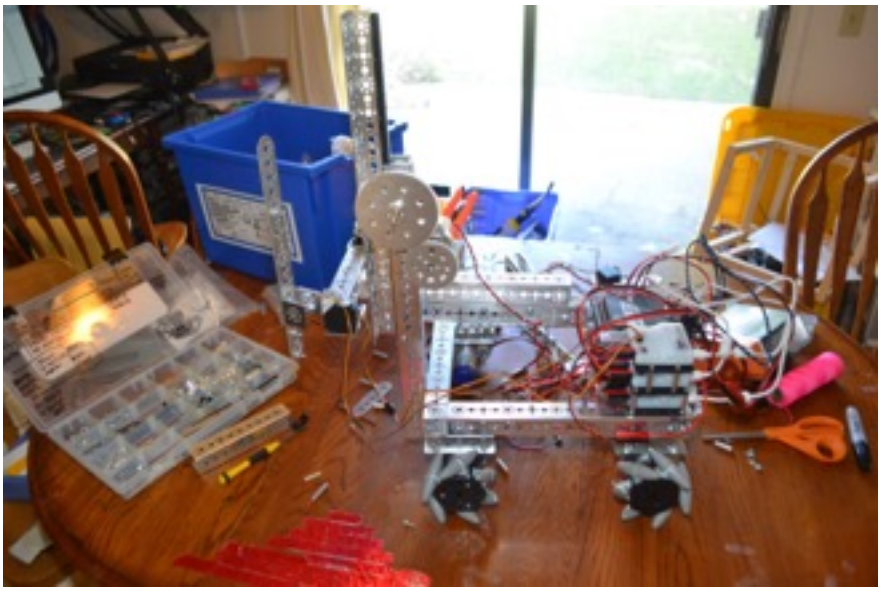
In this arrangement, the two drive motor controllers are spread on the wings of the robot with the wires facing outward. The channels were open toward the outside of the robot to make it easier to atttach things to the channels.The channels were flipped so that the wires could be tucked inside the robot.



In this arrangement, the power control module was placed on the back of the robot. This allowed room for placement of the other control modules on the back as well. It was hoped that this would allow for the entire control assembly to be removed without chaning the robot. However, the other controllers did not mount easily to this connection and their wires were oriented in ways that were difficult to control.



This arrangement blended the ideas of the previous two arrangements where the control module was placed on the back of the robot and the motor controllers were placed on opposite sides. The wires all tucked nicely into the channels and the battery fit nicely onto the back channel.



The final arrangement used posts to make a stack of modern robotics controllers on the left of the robot. The power module remained on the back bumper of the robot and the Rev Expansion Hub was mounted on the right of the robot. This allowed for the batter to be mounted parallel to the wings and supported by a tetrix flat that was attached to the back bumper.

Once this arrangement was completed, the team set-out to start testing the robot. Before testing could begin, the team needed a wire that would connect the Anderson Power Pole to the XT30 connector on the of the Rev Adaptor.

The team could not order this wire because it wa on back-order. The team tried to make a new wire by buying the connectors and making the wire. However, the team could not find either the Anderson Coupler or the XT30 connector at Lowe's or at Aubuchon. All of our local radio shacks have closed. The team did not want wait until the wire has back in stock, so the team set-out to make a connection wire from two wires the team already had.

This was accomplished by cutting an XT30 Connector from the Rev. Hub. power supply cord (connected direcly to the battery) and splicing it to a motor wire that had an Anderson coupler on it. The motor wire was selected because it was damaged during a competition last year. The two wires were then soldered together with a solder gun and covered with both shrink wrap and electrical tape.

Once everything was mounted and connected, the team power-up the system. To their surprise, nothing caught on fire and everything seemed to be in working order.

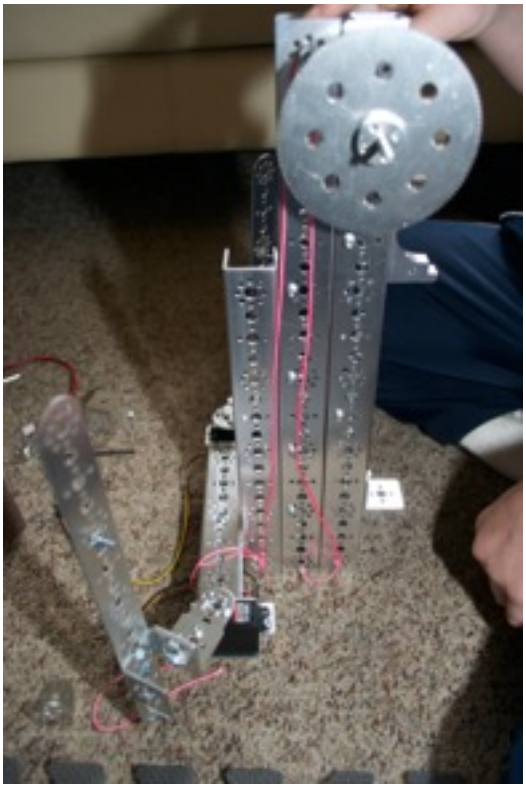
11/20/17 Andrew, Luke, Kenny and Myles

The team started practice today by reviewing the notes from the last session. The team re-watched a number of YouTube videos and discovered some new things.

The team noticed that one of the other teams used a base that was cut from steel and had been drilled into many different times. This sheet had cut-outs for the wheels and a large front gap for the lifter. This was the robot that had the controllers in plane with the motors. The team also noticed that this was the same robot that did really well in another autonomous video.

The team also noticed that several teams combined drawer slides with other mechanisms to create linear slides. The team observed that these slides were compact and appeared lift.

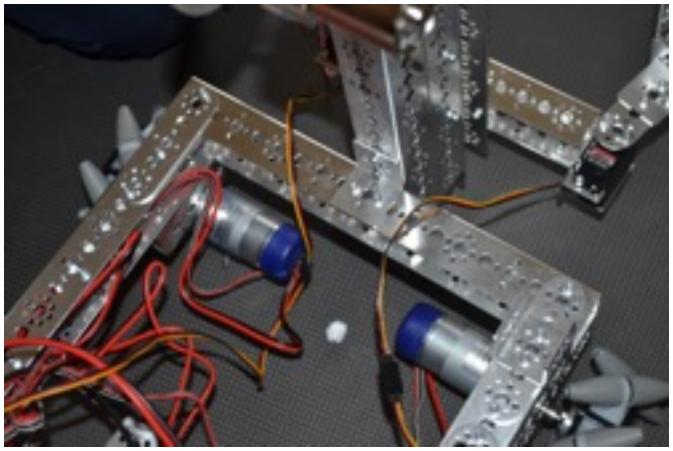
The team met to evaluate the full protype and recommended some incremental changes.

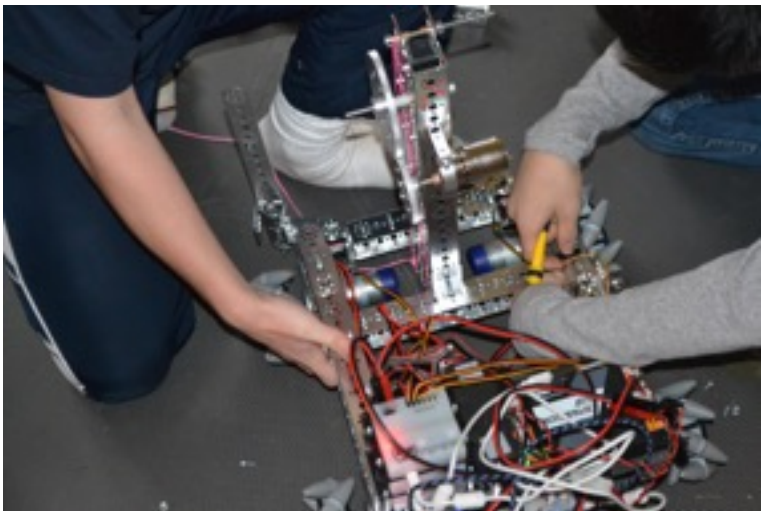


Kenny and Andrew started working on the lifter. They decided to first remove the slider. Then, they tested whether or not the rack and pinion set-up would be able to lift the two blocks high enough. They determined that it would not.

Kenny and Andrew then used the rack and pinion from the slider and re-arranged the lifter to make a pack and pinion lift that integrated another stage with a pulley lift. This improved design enabled the team to be able to lift the blocks high enough to complete a column, which was their goal.

Later, Andrew and Kenny mounted the lifer to the robot and tested it. It worked great and it was much more stable because the slider was removed and because they added two exteranal “L” connectors to create three points of contact. This corrected the lift capacity but the tower was still too high to be legal.





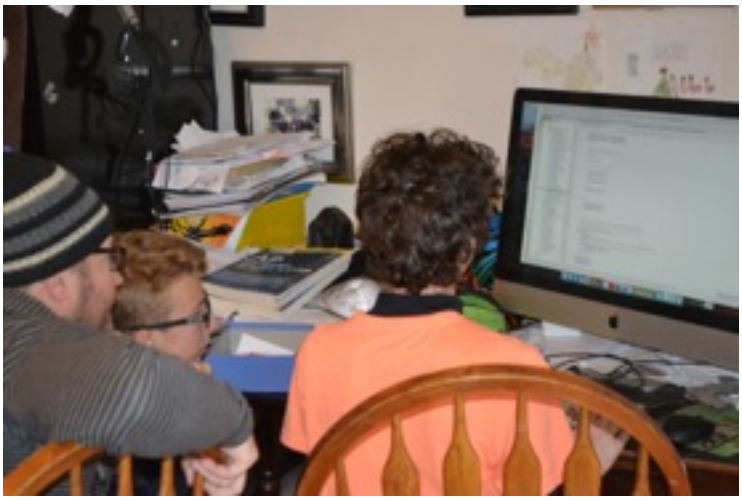
The team explored adding a chanel on the underside of the chassis and moving the mount for the lifter inside the frame. This enabled the lifter to be mounted lower, which got it under 18”. This also enabled the total distance to be shorter, which was good becaues the robot legnth was 22” with the lifter attached on the outside of the front bumper..

Andrew and Kenny tested the robot design in teleOp mode to determine if the changes were improvements and to determine if further changes were necessary. Mr. Kim helped them making some videos. This is a link to video of the initial test.

https://youtu.be/xXIK49ZYw7o

During the first test, Kenny and Andrew learned that the “human” factors had a big impact on the performance of the robot. They learned that when the glyphs were stacked, they should be kept as low as possible when driving and they should be lifted only once the robot was at the cipher location. They also learned that they needed to communicate with each other better to help them coordinate their actions. This video shows their progress.

https://youtu.be/RaLtiQ-XSvg



Luke and Myles worked on the autoOpMode using encoders with Mr. Peterson. They used code taken from the template for encoder and modified it using the values from the OmniDrive V3 program in order to map the signs to the encoder values and motor powers.

The team decided to refer to the forward and backward driving as Axial in their control function beacuse this was used in some demonstration videos of autonomous using Vuforia with omni drive. This demonstration robot used three motors.

The team referred to the right/lift straight driving as strafeing beacuse of the same video.

During the first test, severla problems were uncovered. First, the robot strafed to the right and did not go forward, as expected. Second, the robot appeared to move much further than 5".The drive command was placed inside the vuforia loop.

https://youtu.be/JWLeUGKcMRI

This shows the telemetry data from the first run.
https://youtu.be/kgyIRbRrf78

During the first test of the autoOp, the team discovered that the conventions of order the motors in the TeleOp was different than the AutoOp. In the teleOp the front motors were assigned then the back motors. In the autoOp it was the left motors and then the right motors. This was corrected to make both modes work correctly. The team picked the convention of the teleOp mode because that was already working correctly.

https://youtu.be/mPvL1OM9-o0\

This is the telemetry from the second run.
https://youtu.be/YsvkssXE5UY

The video shows that the motors were now operating in the correct direction but they were operating for longer than expected. The team believed this was because the call to the AxialDrive fuction was placed inside the vuforia telemetry loop.

The command was placed outside of the vuforia loop and it worked correctly.

https://youtu.be/5FSompYX5vo

A second drive function for strafing was then added. This created lots of problems because both the axial and the strafeing functions/methods were placed inside another method. They were both pulled out and edited so that they would work.

Andrew and Kenny tested the second full prototpe and learned about the “human” factor of using the robot in teleOp mode. They discovered that the shoudl carry the glyphs when they are lower rather than highter. They also discovered that they should lift the glyphs right before they are placed and that they needed to communicate with each other.

https://youtu.be/xXIK49ZYw7o

After the initial test, Andrew and Kenny completed a more advanced test where they completed a colum and drove up the balance board, which was a piece of cake.

https://youtu.be/RaLtiQ-XSvg

Next Practice

The team discussed adding a second channel to the inside of the back motors, which would allow the electronics to be moved further inside the frame. This would allow for the robot to have a case to enclose all of the wires, which was an issue last year.

The team also planned to work on the AutoApp and work to make some specific measurments of the distances to the left, right and center of the cipher box from the starting position.

The team also discussed moving the phone on the robot to allow for the robot to strafe and then drive forward to drop-off the glyph during autonomous. This would eliminate the need for building a turning method for the robot during autonomous. This would require the phone to be moved and it might not have a clear view of the relix image from a forward facing position.

The robot discussed taking measurements to program a turning method so that the robot could turn and place the glyph in the cipher box. This would allow the robot to keep the phone in its current position.

The team might consider how to used the vuforia images to select the routes for the left, center and right sections of the glyph box.

The color sensor will arrive on Wednesday along with the gyro-sensor and the distance sensor. The team may be able to integrate these sensors in the autonomous to make it the navigation more accurate. The team will definitely use the color sensor for the jewel detection part of autonomous.

Practice 8 11/21/17

Practice started today with a review of yesterday's practice and setting goals for today's practice. The team decided to spend time learning more about programming using FTC Robot Controller and to then apply that understanding to programming the autonomous.

Coach Fitz spend time explained to the team the basics of a java class with the package, the imports, the class name, declaring variables, mapping variables to hardware and running the Op Loop.

He explained about how each class is like a box of cake mix that includes the objects (ingredients) and methods (recipe) to make a cake. He showed the team the FTC Key document, which shows the methods with each class and he explained that when we load the important classes,the team can make new classes that are compositions of classes and methods from the imported classes.

The team also spent some time working on new programming conventions that required the team to add some comments at the end of each curly bracket so show where its scope ended. Mr. Peterson helped with that concept yesterday when one of the methods was pasted inside another method, which produced an error.

The scope also caused a problem when a method was called during a loop, which was not the intention. This was fixed by called the method oustide the loop.

The team reviewed how the four motor drive system worked by creating off-setting forces so that the robot would move in the middle of the two forces. They rewiewed which motor needed to turn clockwise or counter clockwise to create the forces needed to move the robot in the direction that was desired.

The team discussed working on a new method for the rotateDrive, which would allow the robot to turn to a specified amount. The team wanted this method to be able to control the robot.

In order to make this method, the team copied and pasted the strafeDrive method. Luke then renamed the method and modified several variables to make them clear to a reader. The strafeInches was changed to rotateDegrees. Luke then used the FullProtoRelicV1 to get the signs for the motors to get the robot to turn clockwise.

The team was unsure how many clicks the robot would need to run 90 degress so they started by creating a clicksPerDegree variable. they used this to convert 90 degrees to clicks, hoping that this would solve their problems. During their first run, the robot turned about 20 degrees.

https://youtu.be/cxVxIb4kOdU

The team felt that they need more turning so they multipled thier encoder value by 4. When they tested it, it was better but not quote enough.

https://youtu.be/fMLmkcH8vTs

They then multipled their input by 7.5, which was too large.

https://youtu.be/P_I8w7VR5Xo

Then they tried 6.25, which was still a little too much.

They tried 6.125, whcih was still a little too much but it as close so they tried it several times to determine whether it was too much.

They tried 6.0625, which was not quite enough. They were not positive, so they ran it several times.

They tried 6.09375, which worked perfectly. They tested this several times and it worked great, so they left it.

The team then took a break. When they returned, the team decided to try to program the path of an autonomous run.

In order to do this, the team evaluate the overhead view of the field. The team discussed that the field was not in fact a mirror image but that there were (4) starting positions.

In the red position, the phone would be on the right of the robot.

Red 1-(easier of the two red positions)
The team would drive forward, strafe left, drive forward.

Red 2- (harder beacuse it invovled turning or strafing off the balance board)
The team would drive forward, turn 90 degrees, strafe, and then drive forward.

In the blue position, the phone would be on the left of the robot.
Blue 1.

Blue 2.

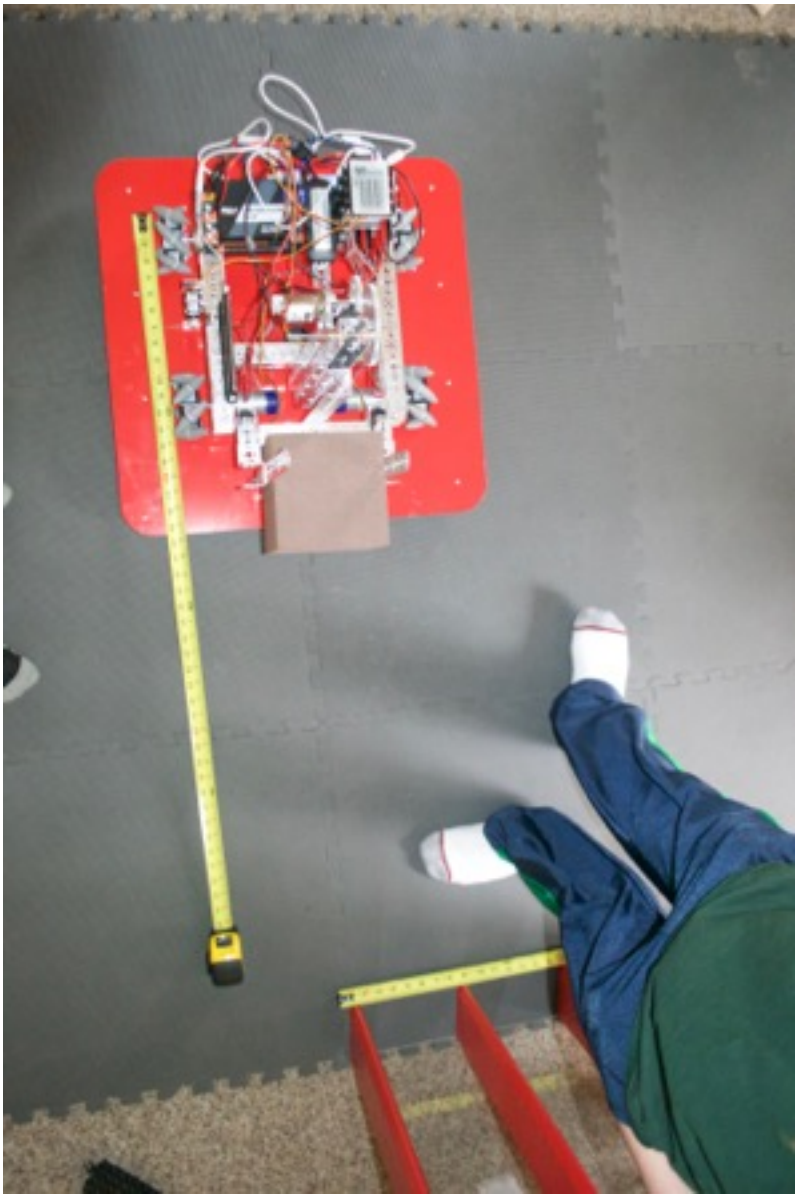
This realization increased the value of the methods for navigating the robot. The team also started to disucss the posibility of building two different phone mounts, based on whether the team was red or blue.

The team also discussed how important it would be to meet with the other teams to discuss their plans during the autonomous becaues the starting positions were not of equal difficulty. It might be important to discuss with other teams our team;s confidence in scoring points from each team and where their partners were going to participate at all in autonomous and whether or not their initial position mattered to them. The team would need to develop some negotations plans in the event of a conflict in order to maximize the points of each team.

The team then set-out to program a path to the center of the cipher box from the easiest red position. The team discussed whether they should add the glyph before their picked the path or after they did the path. The risk of programming with the glyph was that the team might lose valuable time programmign the servos. The risk of programming without hte glyph was that they might program a route that worked great without the glyph only to learn that they had to re-do the prath with the glyph.

The majority of the team wanted to program the path first but Myles felt that is more likely to be a waste of time. He argued that the team would eventually have to program with the glyph, so it would probably save time to do it from the get-go.

The team then copied and pasted code from the FullProtoRelicRecoveryV1 teleOp to determine how to integrate the servos. Luke important the sensor class and he borrowed the code to make the sensors run. He used the end position from the teleOp in the initialize section of the autoOp in order to place the glyph into the grabber. He then added the code to release the servos after the path was navigated.



The team then used their measurements to program the robot to drive forward using the axial drive for 24”.They initiall wanted 32” but discovered that the glyph would hit the back wall. (good thing Myles persuaded them to get it right the first time!)

The team then wanted the robot to strafe for 9 inches to the left, so they used a negative value for the strafing.

When the team tested the opMode, everything worked great but the robot strafed to the right, not the left.

https://youtu.be/dSOYSVv-d1M

The team then modified the autoOp to verify that it compiled correctly and that it was in the right part of the logic check.

The test re-tested and had the same result. Since the team had sign mapping errors in the past, they check this first. They did not discover any errors.

https://youtu.be/Ry8a1CE5bjU

Kenny noticed that they were subtracting a negative, which made it a positive. He recommended that they change the signs on their left strafe logic loop.

The ream recompiled the code, tested it and discovered that this solved the problem.

https://youtu.be/CAORjSKUUIs

The team also learned that the robot needed to strafe just a little bit more, so they increased it from -9 to -15 inches. With this change, the robot demonstrated that it could drive from the balance board and place a glyph into the center column of the cipher box.

https://youtu.be/gJw0CZySxpQ

This also revealed a problem with the driving methods because the axial drive and the rotate drive each used the same coding pattern as the strafe drive. Kenny recommended that those also be changed and the Axial drive was successfully tested and drove backwards.

https://youtu.be/4lJxmPJOdjU

The team began some prelminary work on creating handles to carry the robot and the team mounted the senor controller to the robot in anticipation of the sensors arriving on Wednesday.

Next Meeting

The team discussed meeting again on Friday or Saturday to continue work on the autonomous phase. The team deciced that they would first program the automous app to read the Vuforia Image and then choose a path to follow to the glyph for the red 1position.

The team also planned to begin programming the jewel task for the autonomous using the color sensor mounted to a servo. This should be relatively simple. The team may have to mount a sero on the right side of the robot as well when they work on the field from the blue side. The team has an additional servo but it may need to order an additional color senor fort the blue side.

If the team has time, the team may also choose to explore moving the electronics closer to the center of the robot to allow for an enclosure to keep all of the wires inside the robit.

Once the team has a completed autoOp from the red1 position, the team would want to develop similar opModes from the other positions.

11/28/17 Luke, Myles and Kenny

We watched some lessons on the difference between basic OpMode and Linear OpMode.

http://stemrobotics.cs.pdx.edu/node/4698

We worked on a new approach to programming where we focused on using more precise language and we tried to express ideas in terms of mindstorms too.

We talked alot about classes and methods and the FTC Key helped a lot with this because it showed the methods that go with each class.

http://ftckey.com/apis/ftc/index.html?com/

We also watched the videos from FTC on Programming OpModes, which also helped. We learned that the basic opMode has two required methods, start() and loop() whereas the linear opMode has the required method runOpMode(). We learned that linear opMode is supposed to be easier to program.

We talke alot about inheritance and how the “extends” modifier allows to use all of the methods of a class and then allows us to override any methods we want and it allows us to add new methods.

We talked about a class of a soccer player that had two methods: shoot and pass. We talked about how we could extend that class of soccer player with a new class, advanced soccer player, that could shoot and pass but this player could also tackle to get the ball back.

We talked about how we might have an andrew kim class of volley ball player and that we could extend that class and override the height to make the player 7 feet tall.

We talked about the void component in naming a class or a method and that it tells the computer that nothing gets returned. We contracted this with a method that would return something, like a number.

We talked about a class that multipled two integerts. Such a class migh look like this:

```
public int result Myles{int Number1, intNumber2){
        result = Number1*Number2;
        return(result);
}
```

This part of practice took almost two hours, but it seemed to help with the basic understanding of Java and how it works with the FTC Robot Controller App.

We then set-out to program the color sensor. We had a plan to first mount the servo, then tune the servo (begin programming the op mode), build an arm, mount the sensor to the arm, collect sensor data and then use the sensor data to make a decision that drives the robot.

Tuning the servo was tricky. We wrote a simple program to move the servo. Once the servo was being tested, the front left servo also operated. We attemped to resolve the issue by checking all of the connections. This did not correct the problem.

We restarted the phones and this did not correct the problem.

We moved the ports that were used, change the configuration file, and this did not solve the problem.

We decided that we should not continue to try to solve the problem as a group. The team divided into parts and set-out to continue programming the servo and to try to solve the probelm.

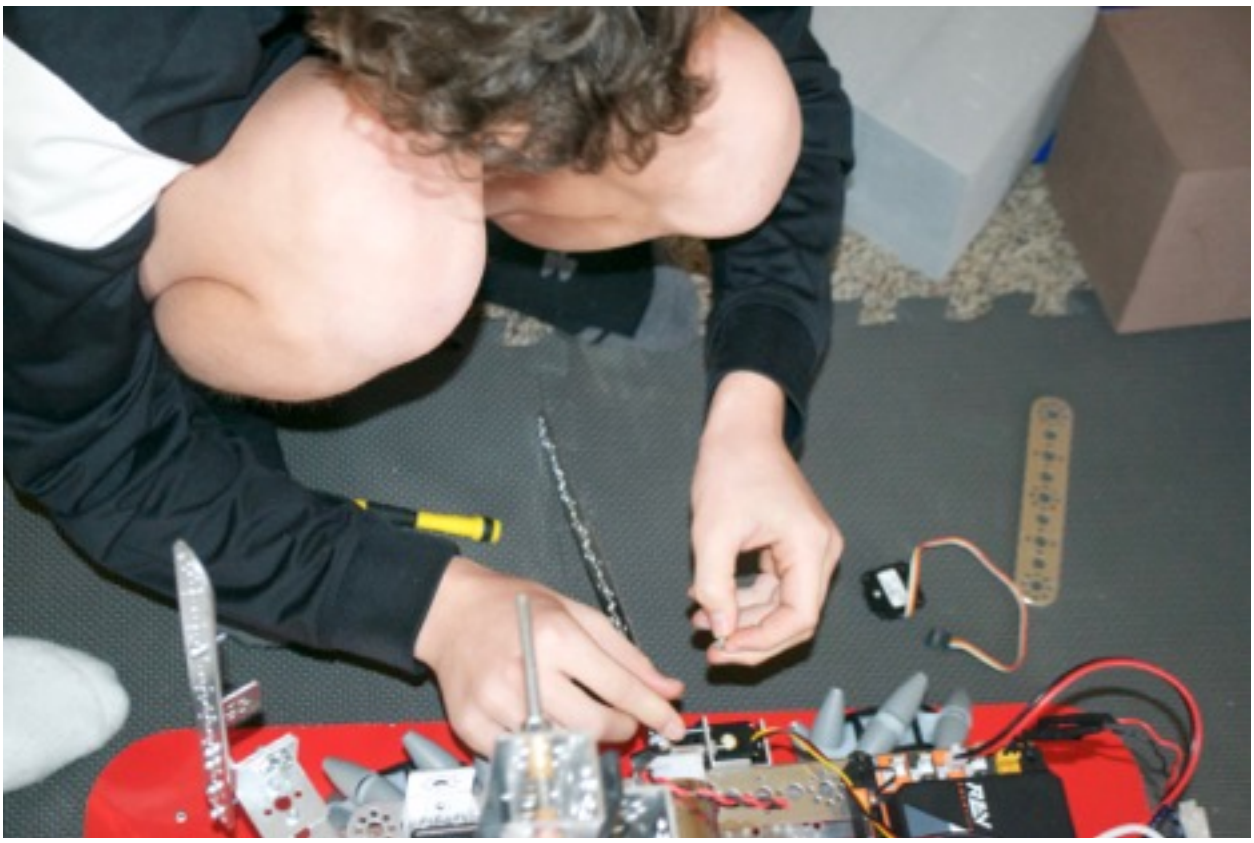
A web search revealed that the Techno-Chix fixed the problem by giving all of the servos a position. This corrreted the problem. We probably spent 20 minutes trouble shooting this probem.

Then, we set-out to tune the servo. When this was completed, we attempted to program multiple servo movments. The servo seemed to get stuck until we added some sleeps.

We are constantly struggling with time as we consider building more understanding of programming and having running programs and moving onto the next task for the robot design. This is difficult because we know that we will work faster if we have more understanding but we also know that we have a limited amout of time. These types of decisions are a gamble based on how likely we think we are to be successful.

```
    lservo.setPosition(lposition);
rservo.setPosition(rposition);
JewelArm.setPosition(JewelArmPosition);
sleep(1000);

waitForStart();
//these things happen after the start button is pressed

JewelArm.setPosition(0.25);
sleep(1000);
```

We discovered that if we added waits it seemed to help the servo operate. Without the weights, it seemed that the servo did not have enough time to execute it's task. In the drive by encoder program, there is section that says to wait while the motors and busy. Perhaps there is a similar think we could do to wait while the servos are busy?



Once we had the servo turned properly, so that it would rotate in the direction we wanted and move the amount we wanted, we set-out to build an arm that woudl contain the color sensor.

We discovered that the arm would require three different flats to be long enough to reach the balls from the balance board. We also discovered that we would have to move our robots to the side of the balance board. The sensor would not mount in a parallel fashion to the flat so we mounted it in a diagonal.

We we mounted the arm and sensor to the servo, there was a problem. The servo now turned the strong direction. Somehow, the team got turned around. The sensor initially started at the 1 position and then moved to .5 position.

However, someone changed the code to start the arm at the 0 position. This caused the servo to turn in the wrong direction. It took us 10-15 minutes to correct this problem.

Then, we added the color sensor. We discovered that the balls read 3 when a red or blue sensor was directly in front of the sensor. We determined this by sending telemetry data to the drive station. We originally forgot to incldue the telemetry.update() command. We put the add data into the While(OpModeIsActive()).

We struggled to get thsi opMode working because we forgot to include the telemetry.update() function. We disovered this when we looked through other examples of telemetry. We also discovered a logging funtion that will allow us to write data to a log file. This was something we wanted when we were considering how to best program the navigation of the robot in autonomous. This may have been a 10 minute delay.

Then, we added the axialDrive method and attempted to write an opMode that would drive the robot based on the sensor data.

This probved difficult, because the sensor was not close enough to the balls to register.

At about this time, the robot battery indicated that it was too low to operate properly. It took us some time to recognize that this was the problem and we must have restarted the robot, and the phones, 2-3 times before we can to this realization. This may have been a 15-20 minute delay.

The telemetry data was used to determine that the sensor reads 255 when it does not see something. so, we wrote a loop that drives the robot backwards if it does not get a color signal.

```
telemetry.addData("Red  ", colorSensor.red());
        telemetry.addData("Blue ", colorSensor.blue());
        telemetry.update();
        sleep(1000);

        while(colorSensor.red() == 255 && colorSensor.blue()==255) //This is looking at
red...meaning blue is in front
        {
            axialDrive(.05, -.5, 5);

        }

        if(colorSensor.red() > colorSensor.blue()) //This is looking at red...meaning blue is in front
        {
            axialDrive(.25, 5, 5);

        }
        else
        {
            axialDrive(.25, -5, 5);

        }

        sleep(1000);

        JewelArm.setPosition(1.0);
        sleep(1000);
```

Once this is done, the robot uses the axial drive. This worked better when there were multiple sleeps added. It would be nice to import the robot driving methods for our robot because they are complicated. The team may speak to some of the parents that are programmers to help with this task.

This video shows the detection of the Red jewel in the back, so the blue jewel is moved off of the board by driving forward.

https://youtu.be/Tt9wc_ucq_c

Once the program worked for the red ball, it was stitched to be able to go in reverse. This worked fine but the first few attempts were thwarted because the arm was too loose.

https://youtu.be/z4Li-UoTUiI

The practice ran much longer that we would had planned. Part of the problem was the ongoing issues that the phones kept crashing and downloading new opmodes took a very long time.

Next Practice:

1) work with the Vuforia image ID to determine where to put the glyph in order to make it run.
2) develop an axialDive method that uses data from the distance sensor to determine when the robot should stop.
3) develop a strafeDrive method that uses data from a second distance sensor to determine when the robot should stop.
4) if this is time, explore the gyro sensor to work on a rotateDrive method that uses data to determine when the robot should stop turning to complete a specific turn.

Programming goals

a) including more comments, especially end curly brackets
b) finds ways to use telemetry to communicate which part of a program is running and whether or not it is working as expected.
c) explore writing data to the log files to review after an app has been written.

11/29/17-Luke and Kenny

Today we met with the goal of working on integrating the Vuforia reading into the autonomous OpMode.

We started by making a stripped down version of the demonstration app. Remove all of the comments and things we were not planning to use. When we tested it, the image was not identified.

When we looked through the robot controlled to see the demonstration image, we could observed that the frame was very small. We assumed that the problem was the viewing size, resulting from the phone placement.

We then spent some time running the app and moving the phone in different places to find a good spot to place the phone. The image appear much larger when the phone was in portrait orientation, so we decided to change the position of the phone.

We then noticed that the phone would pick-up more data if it was placed further from the edge of the robot, so we considered mounting the phone on the non-moving post of the lifter. Since we would have to dissassembly the lifter, we decided to build proof-of-concept mounts before we took apart the assembly.

Kenny proposed using posts to hold the phone to the post. Andrew explored using channels and flats and Luke explored using the channels like an envelope.

The team worked on prototypes for about 30 minutes until they had a working prototype from Luke's design that did not require the re-build of the lifter.

When we tested the opMode, the initial problem persisted. This meant that the problem was with the OpMode. We confirmed this using the demonstration opMode. In hind sight, we probably should have tested that before we started working on the phone mount.

We went back to re-design the opMode using the template opMode. When we did this, we left in all of the original code and then tried to pull-out specific code we were not planning to use.

We then inserted some code to demonstrate how we wanted the robot to operate when it detected a particular image. We simulated the commands using the telemetry.

Again, this did not work.

So, we re-build the app from the demonstration app and left everything in place and just inserted our new code. Our new code is pasted here:

```
if(vuMark == RelicRecoveryVuMark.RIGHT) //found the right view mark
            {
               telemetry.addData("path", "The robot should be moving to the right");
               telemetry.update();
               sleep(2000);

               telemetry.addData("path", "The robot drive forward this amount");
               telemetry.update();
               sleep(2000);


               telemetry.addData("path", "The robot drive strafe left this amount");
               telemetry.update();
               sleep(2000);


               telemetry.addData("path", "The robot should drop the glyph");
               telemetry.update();
               sleep(2000);

               telemetry.addData("path", "add should end");
               telemetry.update();
               sleep(2000);

            }//ends the right viewmark statements
            else
            {
              if(vuMark != RelicRecoveryVuMark.CENTER)//found the center view mark
              {
                 telemetry.addData("path", "The robot should be moving to the center");
                 telemetry.update();
                 sleep(2000);

                 telemetry.addData("path", "The robot drive forward this amount");
                 telemetry.update();
                 sleep(2000);


                 telemetry.addData("path", "The robot drive strafe left this amount");
                 telemetry.update();
                 sleep(2000);


                 telemetry.addData("path", "The robot should drop the glyph");
                 telemetry.update();
                 sleep(2000);

                 telemetry.addData("path", "add should end");
```

```
            telemetry.update();
            sleep(2000);

        }//ends the center viewmark statements
        else //must have found the left view mark
        {
            telemetry.addData("path", "The robot should be moving to the left");
            telemetry.update();
            sleep(2000);

            telemetry.addData("path", "The robot drive forward this amount");
            telemetry.update();
            sleep(2000);


            telemetry.addData("path", "The robot drive strafe left this amount");
            telemetry.update();
            sleep(2000);


            telemetry.addData("path", "The robot should drop the glyph");
            telemetry.update();
            sleep(2000);

            telemetry.addData("path", "add should end");
            telemetry.update();
            sleep(2000);
        }//ends the left view mark statements


    }//ends the else for the IfRight statement

}
```

This code operated as hoped and when the image target was changed, the change to the path was also changed.

This code shows the telemetry for the left target.
https://youtu.be/nitJXedFEmY

This video those the telemetry for the right target.
https://youtu.be/O5D6uhQ4TDI

Next Practice:

1) The next step would be to add a break() into the loop to end the opMode once the robot has delivered the glyph to the station.
2) The team might also consider working on a distance drive mode for axial and strafe driving modes so that the robot can make more precise decisions about its location on the field.
3) the team should look into importing the drive modes for the robot so that the team does not have to copy/paste these commands every time they want to use them.
4) the team might want to look at deveoping a turb by gyro

12/27/17 (Luke/Kenny)
We began working on programming in Eclipse in order to better understand how to organize a class and to develop methods to organize, clarify and reuse code. We focused mostly on how to type variables and what happens when the types do not match. We paid special attention to casting and to concantenation with strings.

12/28/17(Luke/Andrew)
We continued working on eclipse with a focus on calling methods from different classes. We introduced the concepts of for loops and while loops. We also used the scanner class to collect information from users.

We conducted a series of teleOp tests and determined that we only got (1) glyph into the cipher box.

12/29/17 (Luke/Myles)

We continued working on programming and completed the eclipse portion for the season. We decided not to pursue working with arrays this year or with file management. This means that we are no longer planning on recording the teleOp encoder data to repeat the teleOp during autonomous.

We conducted a length discussion of the problems with the previous week's teleOp runs, where only (1) glyph was consistently placed in the cipher box.

We determined that there were a variety of human problems with the teleOp. First, we were not communicating. We decided to start discussing which glyphs we were going to select. Second, we discussed the selection process and making good choices that would make it easier to grab and deliver certain glyphs. Third, we discussed our strategy of stacking blocks and decided this would be the better way to go because we were only likely to make (1) trip to the cipher box. Finally, we discussed being efficient with our patterns for approach blocks to try to minimize movements doing things such as rotating.

We tested the teleOp using these human improvements and saw a dramatic improvement. The team of Luke driving and Andrew grabbing managed to place (4) glyphs virtually every time.

The team also discussed a variety of mechanical improvements, including adding some texture to the grippers. The team explore using flat foam, thin foam and fuzzy foam. The team decided to add fuzzy foam with elastics and the team discovered that they could easily place (4) blocks and they were close to(6) blocks.

The team attempted to grab the relic with the grippers but it did not work. The team discussed using more foam and using a different setting on the servos to hug the relic.

The team also got Android studio on all three laptops.

12/30 (Luke, Kenny, Ethan)

The team worked extensively on understanding how to convert OpModes to methods and how to organize those methods together. The team specifically spent time working on the TeleOp programs, using methods to control the lifter, gripper and teleDrive.

12/31/18 (Luke, Ethan and Kenny)

The team worked on using the range finder and the color/distance sensor to control the robot during autonomous.

The first test of the range finder was to install the range finder onto the robot and then to run the demonstration program. Then, the demonstration program was used to learn about how the sensor worked using both the default telemetry data and logging the data to LogCat.

During the first test, it was obvious that the best measure for the team was the distanceCM. The program was then converted into a method and added to the teleOp program so that the distance values would be reported while the robot was driving. This led to some confusing results, because the initial placement of the range finder was inside the robot. The distance did not update as the robot drove. The team theorized that the robot was being detected and not external objects.

The range finder was then moved further forward, which did not solve the problem.

The team then explored using the rev color/distance sensor. The team started by using the demonstration program. The extra code was removed from the program so that the team could focus on the distance readings.

In a few minutes, the team was able to determine that the optical distance sensor emitted a light beam that was very wide and not at all like a laser, which the team had hoped for. The wide beam faded too far to be detected at further than about 6 inches, or 25 cm. This limited the usefulness of the sensor when placed on the robot.

Nevertheless, the team attempted to use it several times.

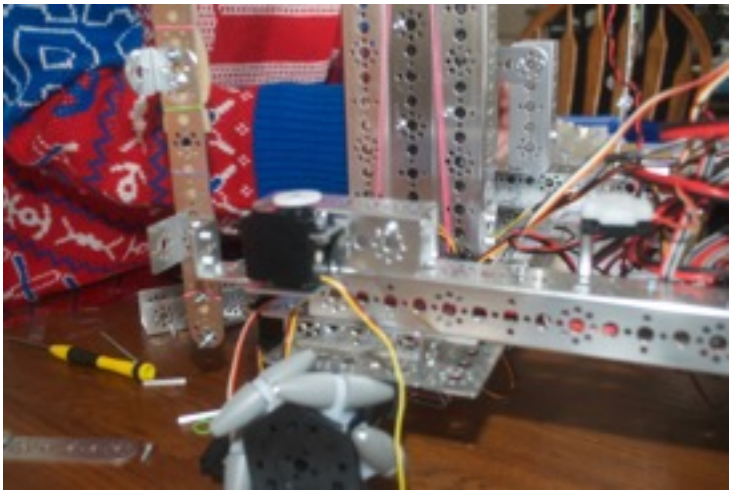
The team then set out to re-try the range sensor. This time, the team mounted the sensor on the outside of the robot. This allowed the team to test that the sensor would work from a different position. The new placement worked great and allowed the range finder to find distances in excess of a meter.

Finally, the range sensor was placed on the front grippers. This worked great and it allowed the team to test a method that would drive the robot until it was within a specific distance of an object.

```
public void driveRange (double speed, double distance){
    while (range.Distance.CM>distance){
        QuadDriveForward(speed);
     }
    QuadBrake();
```

01/02/18-Luke, Kenny and Ethan

The team worked on several different sub-projects today. The team reflected on whether it would be better to order (3) more range finders or to order a single range finder and two more servos. The team considered how they might mount a single range finder that could be re-positioned in order to allow the team to make measurements when in axial driving and in strafe driving.



Ethan set-out to work on mounting such a servo. He worked hard on the problem but did not make much progress. His initial mount placed the sensor too far inside the robot for it to work, based on our testing yesterday.

Ethan and Kenny then began working on the project of moving all of the electronics because the wires were sticking outside the robot, which would create problems during the competition. The wires were outside of the robot in order to make it easier to add/remove parts during development.

First, Kenny and Ethan mounted a channel inside the robot, symmetrical to the cross channel on the front of the robot.

Second, they added a second and smaller channel on top of the lower channel in order to mount the modern robotics power controller at the same height as it was during the preliminary design phases.

Third, Kenny and Ethan began the process of labeling every wire and all of the controllers so that the robot would be disconnected and reconnected quickly and under pressure.

Fourth, Kenny and Ethan made a floor using three tetrix flat rectangles. This assembly fit inside the two channels that spanned the interior or the robot. The flats were fastened to the back channel because the front channel was blocked by screws for the center post mount.

Once the floor was installed, the modern robotics power controller was installed on the inside of the robot at the same height as during the preliminary designs.

Then, the stack of controllers was added. The stack could not be fastened to the floor on 4 points of contact, but the controllers were moved to the inside of the robot. The micros usb cords were placed to the outside of the robot and the power cords were placed to the inside of the robot. The posts were installed in an alternating fashion.

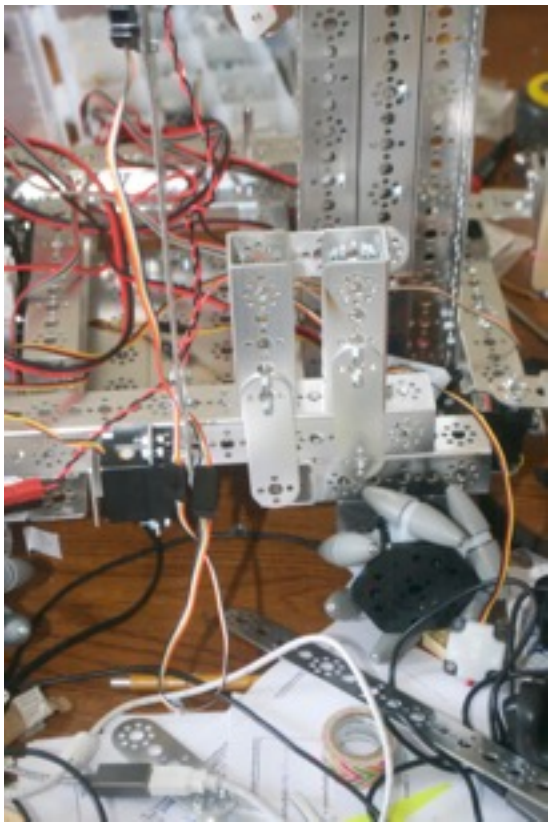
At this time, Luke mounted the battery using a single post to the center channel. The battery was held in place by the controller stack and the higher channel.

Luke then mounted the Rev Expansion Hub using wire ties. It was mounted on top of the back two channels, where they formed an L shape. This was the only place for the Hub where it could fit and where the wires would remain inside the robot.

Finally, elastics were added to the wires to hold them inside the robot.

While Ethan and Kenny were working on the electronics, Luke was working on the phone mount.

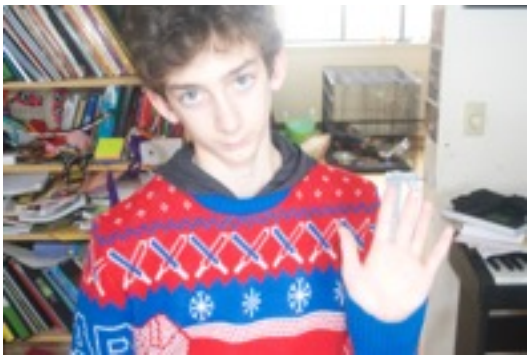
Luke worked to improve the phone mount so that it could be used on both sides of the robot. Luke designed the mount to slide on top of posts. This created an initial problem because the wide holes in the center of the bottom on the phone did not align with holes on the channel. Luke mounted a flat on the underside of the channel so that there would be places to mount the posts.



When Luke tried to place the phone inside the mount, he discovered some new problems with the phone placement because there were screws and nuts on both sides of the robot. This blocked the phone from resting safely. The nuts would have to be removed in order to allow the phone to rest properly.

Luke decided to braze the metals together and used a yellow torch to do this. The team learned to braze in the late summer. However, Luke quickly discovered that the thin aluminum from tetrix heats and then deforms very quickly. This was not something that happened over the summer because the aluminum was not drilled. This aluminum between the holes heated very quickly and became damaged. Brazing tetrix flats did not seem like a viable option.

Luke continued to work on the phone mount with the plan to glue/epoxy the pieces together. When Luke had a completed phone mount, he discovered that the left side geometry was not symmetrical with the right. This meant that the mount needed to be modified to work for both sides.



Luke redesigned the mount again and mounted it to both sides. It fit onto the channel perfectly. However, it no longer allowed the phone cable to rest. This problem was not resolved when practice ended.

Luke also used tape on his finger to hold nuts in hard-to-reach places.

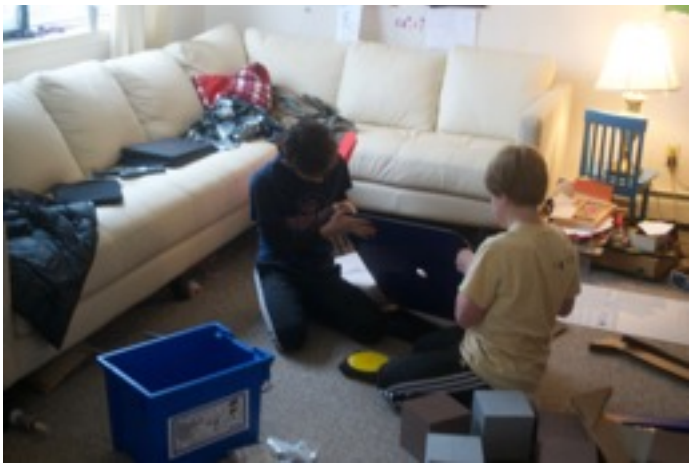
Next Steps
The range sensors and extension cords were ordered. They will be installed next session. The team will need to develop methods to drive the robot in strafe mode to the right or left using the range sensors. Likewise, the team will need to integrate a gyro turning method.

The team will need to build the rest of the "red" side in order to complete the work on the red autonomous.

The team may also need to get an additional 9 floor tiles to build the entire red side. The team can then re-use these for the right side in order to complete the autonomous using the range sensors.

The team members should continue to work on their programming activities.

1/7/18 (Luke, Kenny, Myles, Ethan)

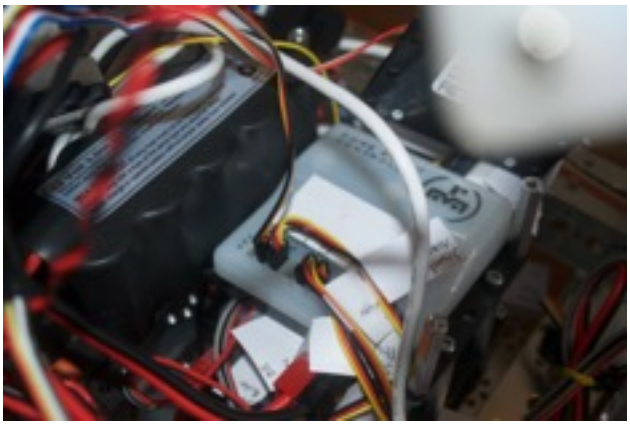


Luke and Kenny worked largely on assembling the blue section of the 1/2 field set-up. They built the second balance board and the second cipher station. They also identified a need to get more floor tiles.

Ethan and Myles worked on assembling the left arm for the blue autonomous. This decision was made because the range finding sensors had not arrived yet.

The team also reviewed some basic math to better understand how to use the encoders with the mecanum wheels. The team learned that he mechanim wheels work by placing the forces at 45 degree angle. If the wheels run a full rotation, 1440 clicks, then the wheels should move, roughly, the distance of the circumference of the tires. However, they do this at a 45 degree angle and both sides do this. This means that the net forward movement is determined by the altitude of the triangle formed by the wheels. This can be determined using the pythagorean theorm, which makes if 1440/sqrt(2). The team has not verified this yet.

Ethan and Myles worked to build-up some methods to control the left arm for the Jewel lift. This process started by making a second arm and then tuning the servo.

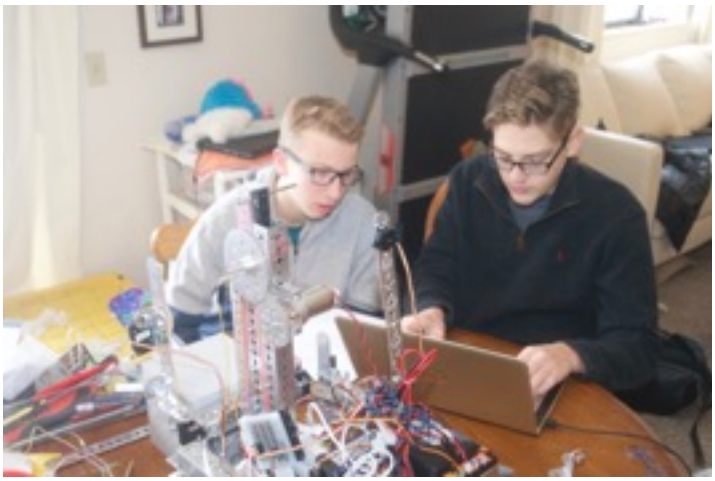


In order to tune the servo, the team ran the demonstration opMode. Before they could do this, the team had to rename the OpMode and then re-name the servo inside the opmode. The team also had to change the configuration file, to make it match the opmode. The team also needed to connect the servo to the servo controller. Ethan and Myles chose to place the servo in the 5rth port, leaving space between the two gripper servos and the two jewel arm servos. They made this decision to make it easier to distinguish the two groups of servos if an emergency repair needed to be made.

The team attemped to run the opMode from Ethan's PC but it did not work, despite numerous efforts to get it to work.

One the team ran the opMode, they detmined that all of the ervos activated when they tried to operate the single servo. This was managed by placing each servo in a specific position, based on the previous programming.

Once the opMode tan, the team determined that the left servo started at 0, and then moved toward 1 to lower the arm and the reverse was true to raise the arm.



Then, the team selected elements from the demonstration to make a brand new Opmode, which would begin first by setting the servo in the correct position.

The second version of the program dropped the servo arm and raised the servo arm.

The third version placed these commands into methods:

```
// Scan servo till stop pressed.
      while (opModeIsActive()) {

          JewelArmLeftDrop(.75);
          sleep(2000);
          JewelArmLeftLift(0);
          sleep(2000);


      }

  }

  public void JewelArmLeftDrop(double ljaTarget) {
      //LJA should start at 0
      //stop at .75

      telemetry.addLine("Should be lowering now");
      telemetry.addData("Servo Position", "%5.2f", JewelArmLeft.getPosition());
      telemetry.addData(">", "Press Stop to end test." );
      telemetry.update();
      while (JewelArmLeft.getPosition() <= ljaTarget) {
          JewelArmLeftPosition += INCREMENT;
          JewelArmLeft.setPosition(JewelArmLeftPosition);
          sleep(CYCLE_MS);
          idle();
      }

  }

  public void JewelArmLeftLift(double ljaTarget) {
      //.75
      //stop at 0
      telemetry.addLine("Should be rising now");

      telemetry.addData("Servo Position", "%5.2f", JewelArmLeft.getPosition());
      telemetry.addData(">", "Press Stop to end test." );
      telemetry.update();
      while (JewelArmLeft.getPosition() >= ljaTarget) {
          JewelArmLeftPosition -= INCREMENT;
          JewelArmLeft.setPosition(JewelArmLeftPosition);
          sleep(CYCLE_MS);
          idle();
      }

  }
```

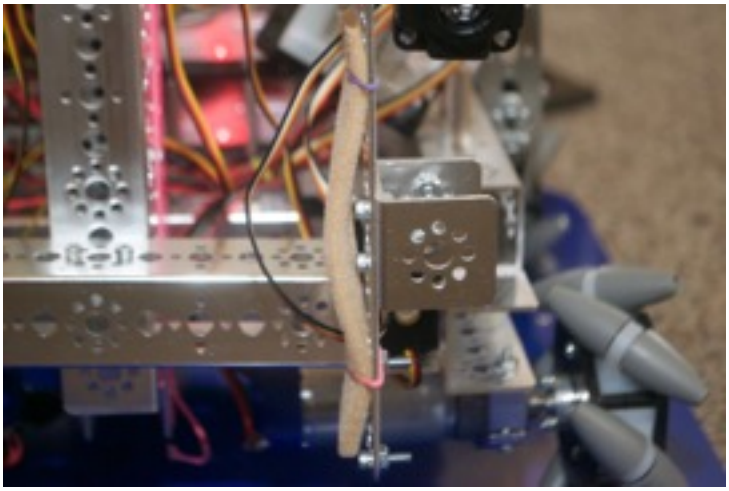
Once the servo arm was properly tuned and mounted, the color sensor was used to write some methods to support it.

Ethan and Myles started by adding a color sensor to the arm with the initial Opmode from the external samples. The team did not have the necessary extensions cords, because they did had not arrived yet. The team, therefore, placed the new color sensor into the extension cord of the old sensor and ran the opMode. Ethan and Myles were working a new opMode, from scratch, in order to see if they could improve on the old OpMode.

When they did this, they determined that the red values are greated than 0 when the sensor sees red and the blue are greater than 0 when the sensor sees blue. They determined that when the only object in front of the sensor is the ball, there is no mixed values of blue or red.

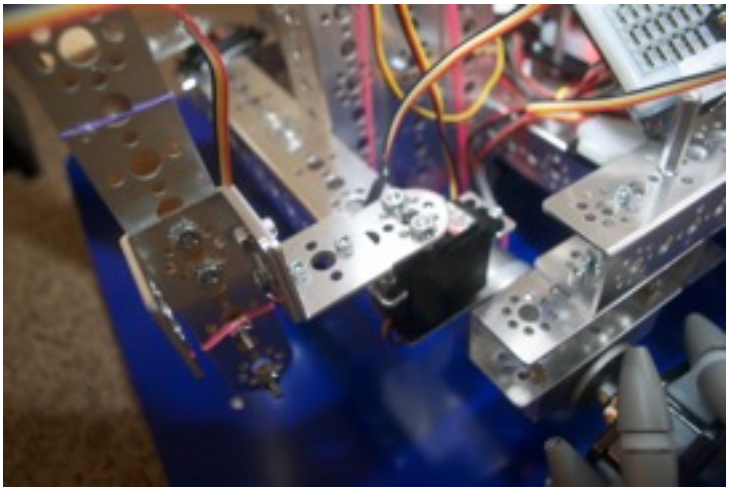
This was the method they developed to return whether the sensor saw a blue or red ball.

```
public String GetBallColor(){
        telemetry.addData("Red  ", colorSensor.red());
        telemetry.addData("Blue  ", colorSensor.blue());
        telemetry.update();
        String ballColor;

        if(colorSensor.red()==0 && colorSensor.blue()==0){
            ballColor="nothing";
            return  ballColor;

        }
        else{
            if(colorSensor.red()>0){
                ballColor="red";
                return  ballColor;
            }
            else
            {
                if(colorSensor.blue()>0){
                    ballColor="blue";
                    return  ballColor;
                }
                else{
                    ballColor="error";
                    return  ballColor;
                }
            }
        }
    }
```



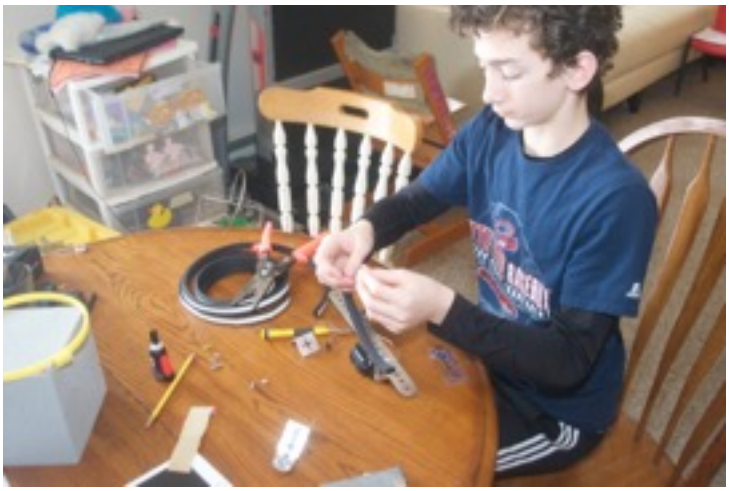
The team also discussed the fues placement problem. They need to develop some protocols to deal with replacing a fuse should it break. They also need to pick-up a 20 amp fuse form the auto parts store before the competition.

The team also considered re-doing the gripper arms. The U bracket that was being used was excessive. Likewise, the range senesors would need a U brack to be mounted.



Similarly, the team explored windown insulation materials to replace the fuzzy foam that was on the grippers.

Luke worked on both of these projects and he re-build both gripper arms and added the rubber insulation material. He noticed that the interference from the fasteners on the grippers was no longer a problem with the insulation material because it was as wide or wider than the fasteners.



The team considered several difference designs for the phone mount, that included using super-glue to connect a bracked to a sleeve and duck tape. The team did not make any final decisions.

For next practice:

1) install range sensers, develop methods to navigating during autonomous

2) complete color sensor install, test blue side jewel during autonomous

3) complete vuforia method to return left, center or right for cipher

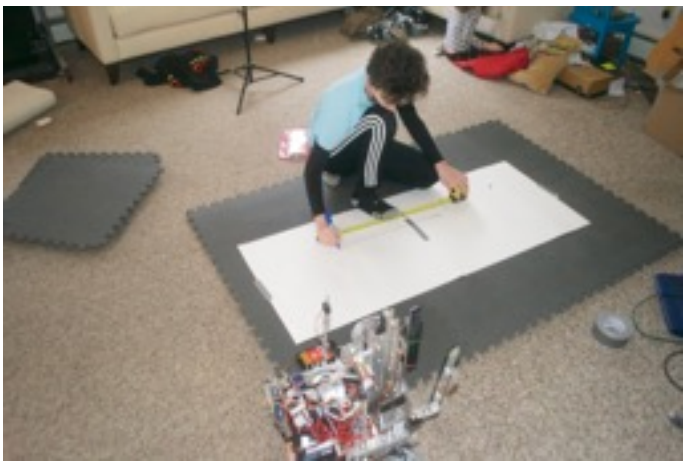
4) rehearse teleOp with new hardward

5) test end-game to see if runner grabs relic better

6) protocol for fuse replacement

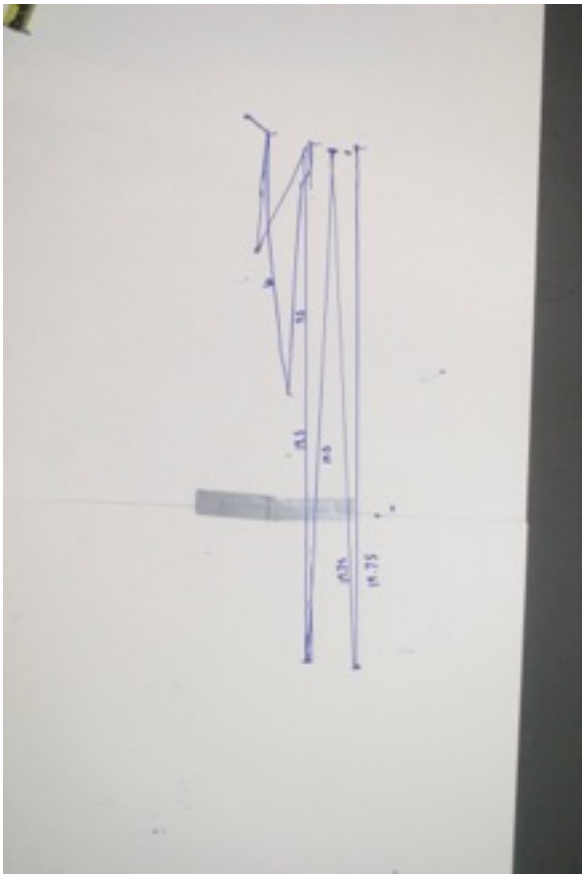
1/8/18

Luke calibrated the encoder methods and determined that the axial drive had .5 to .25 inches error, regardless of distance traveled. He also determined that in the axial direction, the mecnum wheels operate like normal wheels. This also confirmed that the encoders, along, might not be precise enough to control the robot in autonomous.



He determined this by making an OpMode that ran the robot 20 inches forward and then it would strafe for an inch and go 20 inches backward. The robot then repeated this for 10 inches and 5 inches. Luke marked each end point using a marker. He also placed a pencil inside the frame to trace the path.

This procedure was frustrating because the wheels worked irregularly. It seemed like it was partly due to the phone touch screen being hit by the case. This means that another phone case might be explored.



Luke then tested the strafe mode using the same techniques. Strafe mode did not operate as well as axial and the robot appeared to drift, which made the calculations harder to make.

In strafe mode, the robot appeared to operate more like omin wheels, meaning that when the two wheels driving the strafemotion were run 100 units, the robot would move 100/sqrt(2) units. The measurements were close to this but not exact.

This conformed that the strafe mode distance calcualtions would need to be updated to include multiplication by the sqrt(2) in order to get the robot to advance to the correct position.

Next time…
1)the (3) new range sensors need to be installed
2)the second color sensors needs to be installed
3)the configuration files need to be updated
4)a half field needs to be set-up to use the range sensors to complete

1/9/18

Luke installed three range sensors onto the grippers. He noticed that they seemed to make the grippers heavier. The grippers would need to be tested to ensure that the work properly.

After the range sensors were installed into the sensor controller, Luke realized that all of the sensors slots were filled. This was a problem because the second color sensor was not connected to the sensor board.

The color sensor could not be attached to the open IC2 slots on the expansion hub, because the hub was too far away from the sensor and the sensor also needed an extension cord to connect everything together.

The gyro was located closer to the expansion hub and it did not move. Therefore, it made sense to move the gyro from the MR Sensor controller to the expansion hub.

The configuration files need to be updated to add the range sensors, color sensor and to change the position of the gyro sensor.

1/11/18
The initial testing of the color and range sensors was a distaster. None of the sensors reported meaninful data. Many changes were made at once because all of the code was tested before use so the sensors were not added (1) at a time to verify in a sequence.

The most logical cause of the error was problems in the code where the sensors were not properly distinguished from one another in either the configuration file or the opMode. This was checked and tested and retested and was determined not to be the problem.

The next most logical problem was that the mapping of the sensors was not correct so this was checked by tracing each of the sensors back to the core deice controller.

Then, the potential for interference from the sensors themselves was tested by remove some of the sensors.

It was not obvious which port was 0 and which port was 5, which meant that there might have been color sensors placed where the software was looking for range sensors and vice versa.

Thus, all of the sensors were removed and then replaced one at a time in order to determine the problem. This started with the color sensors.

When the first color sensor was placed into the controller the OpMode worked fine. When the second color sensor was placed into the controller, the opMode did not work. This was incredibly frustrating because it was never easy to tell if the robot needed to be restarted to correct the problem, if the apps needed to be restarted, if the phones needed to be restarted or if there was a logitimate problem. So, each of these steps was repeated to see if it correct the problem or gave more meaningufl error data. When testing, there was no error. The data just did not make sense because it read 0 whether the ball was in front of the sensor or not.
In order to verfiy that the OpMode was being updated to the phones, additional telemetry was added to the output to show that a new version has been installed. This confimed that the phones were getting the updated opModes.

Then, a counter was added to the loop on the OpMode to make sure that the opMode was actually runnning through the loop and not stopping at a value of 0.

Finally, the left color sensor was tesed and confirmed to work. Then it was removed. The right color sensor was tested and confirmed to work. Then they were both attached we the error returned.

1/12/18
An internet search confirmed that the problem was having multiple sensors of the same type. Each of the types of sensor, range, color etc.,., has a specific address. However, every individual sensor of the same type has the same address. This means that the robot can not distinguish which sensor is which.

The FTC forum contained several threads on the problem and each referred to a single document, but the link to the document was broke. After several more internet searchs, the document was found on another site.

http://www.patronumbots.com/uploads/1/3/1/6/13161577/multiple_color_sensors_-_setup_document.pdf

The document explained that the team used the Core Device Discovery app to change the i2c address of each sensor. The Core Devic eDiscovery page had a move that explained how to use the app to change the address of the sensors. The progam is only available for PC's.

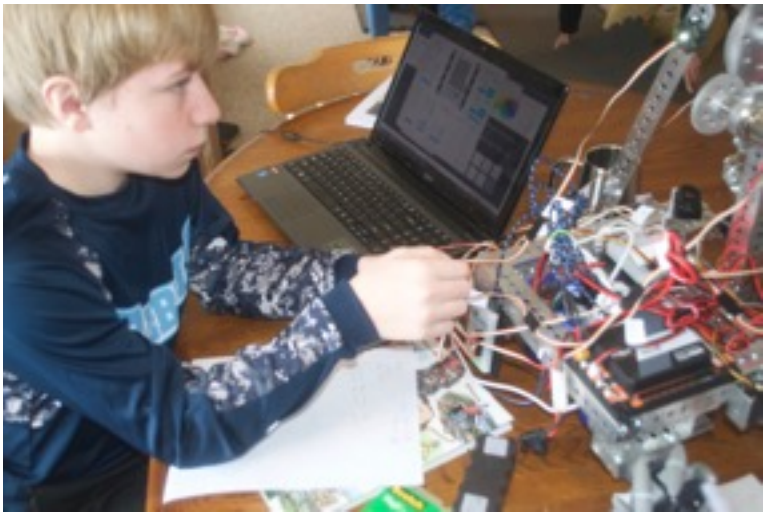
http://modernroboticsinc.com/coredevicediscovery

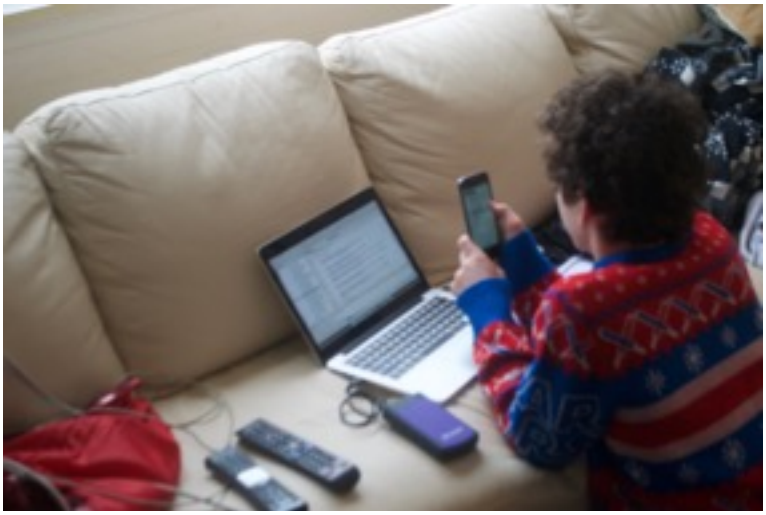
The video was then used to change the address of one of the color sensors. The video did not explain how to determine an appropriate change in the address so the example change was used.

However, it did not work when tested with the opMode. The problem persisted despite changing the i2C address.

The original document explained how to change the mapping of the address inside an opMode. Then, the document explained how to convert the 8 bit address to a 7 bit number for the opMode. This was an essential help beacuse the device discovery app did not explain how to modify the program to get everything to run. However, in the process of typing in the commands to change the address, an 8 bit method appear in the auto fill. This method was seleted, instead of the 7 biit method, and then everything worked. fine.

1/13/18





The following day, the team re-addressed the range sensors using the process of the previous day. This process started by installing a range sensor. Then, the address was changed. Neither the video, nor the document, descirbed how to select an appropriate address for the sensor. Some numbers were rejected. The discovery app showed the text field as red when this happened. The team then entered numbers, slightly larger or smaller, in the field until it turned green. Then, the address was changed and the sensor cable was labeled with the new i2c address.

This process was repeated until three of the sensors had their i2c numbers changed.

```
colorLeft = hardwareMap.get(ColorSensor.class, "leftColor");
colorLeft.setI2cAddress(I2cAddr.create8bit(0x3c));


colorRight = hardwareMap.get(ColorSensor.class, "rightColor");
colorRight.setI2cAddress(I2cAddr.create8bit(0x3a));

rangeFrontLeft = hardwareMap.get(ModernRoboticsI2cRangeSensor.class, "frontLeftRange");
rangeFrontLeft.setI2cAddress(I2cAddr.create8bit(0x26));

rangeFrontRight = hardwareMap.get(ModernRoboticsI2cRangeSensor.class,
"frontRightRange");
rangeFrontRight.setI2cAddress(I2cAddr.create8bit(0x28));
```

Then, the values were displayed using this method:

```
 public void showSensorData(){
        telemetry.addData("Left Color=", robot.colorLeft.red());
        telemetry.addData("Right Color=", robot.colorRight.red());

        telemetry.addData("Left Front Range=",
robot.rangeFrontLeft.getDistance(DistanceUnit.INCH));
        telemetry.addData("Right Front Range=",
robot.rangeFrontRight.getDistance(DistanceUnit.INCH));
        telemetry.addData("Left Range=", robot.rangeLeft.getDistance(DistanceUnit.INCH));
        telemetry.addData("Right Range=", robot.rangeRight.getDistance(DistanceUnit.INCH));

        int heading = robot.modernRoboticsI2cGyro.getHeading();
        telemetry.addData("heading", "%3d deg", heading);

        telemetry.update();

    }
```

Once the address were changed, they were all tested against in a stationary position to make sure that the data was reasonable. At that time, the team decided to change the measurement unit to inches instead of CM so that they could better understand the output.

Once it was determined that the sensors were all sending meangful data, the team placed the statements to made instances and then map them into the RelicRobot class.

Then, they set-out to check the range of motion of the lifter and gripper with the additional sensors. There was concerned that the sensors might be too heavy for the gripper or the cords might limit movement.

This was tested, first, without power. The cords were scotch taped into place in specific areas to allow them to move without getting stuck.

Based on the initial testing, the robot was powered on and testing continued. The left side range sensor cords got stuck on the side of the robot, so they were taped different.

Once all of the sensors were attached, the team wrote some methods to raise and lower the JewelArms during teleOp. This was considered important because there was a possibiility that one of the arms might fall, accidentally, during the teleOp and the team would need a way to lift that arm to get it out of the way.

The team used Ethan's methods to control the left jewel arm by copy/pasting the methods into the opMode and then writing another method to call those methods when the left bumper and trigger were pressed. After this code was determined to be functional, Luke modified the methods to work for the right arm.

```
public void JewelArmLeftDrop(double ljaTarget) {
        while (robot.JewelArmLeft.getPosition() <= ljaTarget) {
        robot.JewelArmLeftPosition += robot.INCREMENT;
        robot.JewelArmLeft.setPosition(robot.JewelArmLeftPosition);
        sleep(robot.CYCLE_MS);

        }
    }
    public void JewelArmLeftLift(double ljaTarget) {
        while (robot.JewelArmLeft.getPosition() >= ljaTarget) {
            robot.JewelArmLeftPosition -= robot.INCREMENT;
            robot.JewelArmLeft.setPosition(robot.JewelArmLeftPosition);
            sleep(robot.CYCLE_MS);

        }
    }


    public void JewelArmDrop(double ljaTarget) {
        while (robot.JewelArm.getPosition() <= ljaTarget) {
            robot.JewelArmPosition -= robot.INCREMENT;
            robot.JewelArm.setPosition(robot.JewelArmPosition);
            sleep(robot.CYCLE_MS);

        }
    }
    public void JewelArmLift(double ljaTarget) {
        while (robot.JewelArm.getPosition() >= ljaTarget) {
            robot.JewelArmPosition += robot.INCREMENT;
            robot.JewelArm.setPosition(robot.JewelArmPosition);
            sleep(robot.CYCLE_MS);

        }
    }
    public void JewelArms(){
        if(gamepad2.left_bumper){
            JewelArmLeftLift(0);
        }
        else {

            if (gamepad2.left_trigger>0){

                JewelArmLeftDrop(.75);
            }
            else{

            }
        }
```

```
    if(gamepad2.right_bumper){
        JewelArmLift(1.0);
    }
    else {

        if (gamepad2.right_trigger>0){

            JewelArmDrop(.275);
        }
        else{

        }
    }
}
```

With all of the methods tested, it was determined that the opMode appeared to get stuck after the arms were raised and lowered. The idle() command was removed from all of these methods but more testing is needed.

Next session

1) the team needs to develop methods to navigate with the range sensors
2) the team needs to develop methods to complete the methods to work with vuforia for image for target identification
3) the team needs to develop an OpMode that integrates all of the methods together to solve the autonomous

1/17/18-Luke, Kenny and Andrew

The whole team worked together to make some methods to drive the robot using the range sensors.

The team reviewed some basic programming ideas and then set-out to make the methods.

The first method was to drive the robot forward using the QuadDriveForward from the teleOp.

The method was developed and it seemed to work on the first run. However, one of the range sensors was 2" from the target and the other was 3". The team set the goal of moving to 2" from a cardboard box. This made the team question if the range sensors were correctly listed in the configuration file.

The team attempted to re-run the opMode, to verify the first run, but it took a long time because it never seemed to run two times in a row. Instead, the team would have to shut down the robot, re-start the robot and exit the driver station and then restart the driver station and the robot. After far too long, the team decided to try another opMode to see if it ran better.

When the team finally got the showSensors() method running, the team became very confused. The sensor drift persisted but now the sensor data made no sense. The range sensors, multiple range sensors, appeared to make dramatic changes from 50 to 1.79....The sensors seemed to report some of the data accurately because when someone moved their hand in front of the sensor, the phone would display meaningful data. However, the meaningful data still had big shifts in the size of the number it showed.

The team first tested whether something had changed when the sensors were added to the relic robot class. This was considered a possibility because the gyro drift appeared after the

sensors were moved to the robot class. This was tested by making a showSensors() method that did not use the robot class. This did not correct the problem.

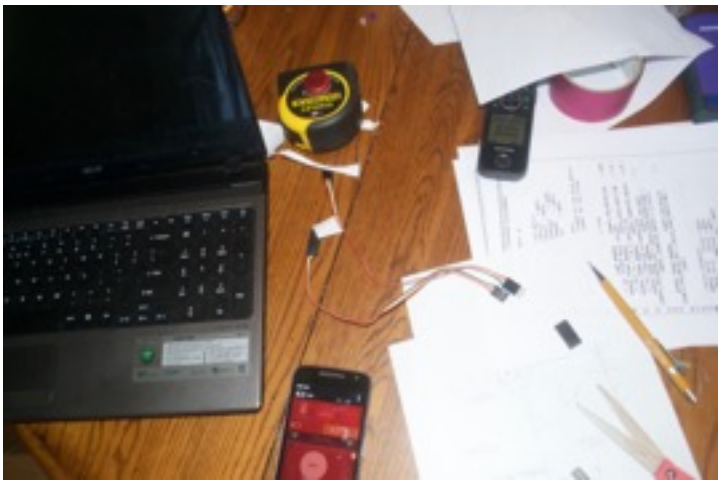
The team considered the possibility of interference from the sensors being mounted so close together so the team tested this possibility by removing the right range sensor from the gripper and leaving the left two sensors alone. This did not solve the problem.

The team considered the possibility that the front two sensors were emitted sounds waves and detecting sound waves from the sensors on the opposite side of the robot, which was also facing forward. To test this, the left front range sensor was unplugged at the connection to the sensor extension cord. This appeared to correct the problem, so the front left facing sensor was left unplugged.

The team discovered, however, that the problem persisted. The team did a quick internet search and discovered that Modern Robotics has a two sensor demonstration program that treats the sensors as i2c objects and not as range sensors. The documentation indicated that the sensor cache might not update properly. The team tested this, but there was drift in this demonstration program as well. Two team members left at this time.

Then, the team tested decided to test the ic3 address to see if there was an error in the assignments. When the discovery app was run, only (2) range sensors appeared, and not the three that were expected. This happened when the sensors shared an address, which indicated that one of the sensors may not have been assigned correctly.

Each sensor was tested, one at a time, to determine which of the three sensors was not appearing properly. The final sensor that was tested had a strange connection. The wires were not enclosed, like the others. This sensor was not appeared properly, which suggested that there was a problem with the wire. This sensor appeared when it was directly connected to the core controller, which indicated that the problem was a wire.



The wire was replaced and all (4) range sensors appeared in the discovery app.

This allow the team to continue testing the original method with good data. The team determined that the initial program was not working properly because the inequality was originally a less than but it should have been a greater than.

The method was then modified to strafe to the left and right. The method was tested and it worked properly but not all of the motors operated as expected.

1/20/18-Luke, Andrew, Ethan and Kenny

The team developed an integrated method to complete the red jewel. The team used Ethan's servo methods to lift and raise the JewelArm, which is on the right side of the robot. The team wrote a method to handle three cases, the sensor detected red, blue or nothing. When the sensor dectects red, the robot drives forward to remove the blue ball. When the sensor detects blue, the robot drives backwards to remove the blue ball. The team noticed that the sensor had a tendency to lose connection when it moved, which is why the nothing was important. The team added a "filter" to handle weird data. The team noticed that sometimes the sensors read 255 when they are not seeing anything. The team wrote the method to ignore values of 0 (sees nothing) and 255 (electrical burp). The team also added the ability for the robot to go backwards, .25 inches, if the robot does not see the ball. The team added this because the sensor has to be quite close to the robot in order to detect it. The team attempts to detect the ball (4) times, for a total backward travel distance of 1 inch. If the team did not detect the ball, the robot stops checking for it. The team added this component so that the team did not hit the wrong ball. They did not want to give their opponents a bonus if they hit the wrong ball. This is all contained inside a single method, RedJewel().

The team developed a new method for axial driving that simplified the previous method by using only a single encoder. The method also used the RUN_USING_ENCODERS as the motor mode, which is supposed to be better for situations where a tire might get stuck from friction. This method was developed because the previous method had poor consistently and some times the motors would not fire at all but others would. This method receives a distance in inches and then converts the distance to clicks for the motor. The method then calls another method, either QuadForward or QuadBackward. These other methods set the sign and speed for the motors. The PID algorithm is supposed to adjust the power to maintain the speed, which is assigned to all (4) drive motors.

The team created new methods to resetEncoders() which will change the mode of the motors to stop and reset encoders. The team created another new method to place the robot into RUN_USING_ENCODERS mode as opposed to RUN_TO_POSITION or RUN_WITHOUT_ENCODERS modes. These methods save space and make the other methods clearer.

The team next worked on the vuforia method. The Vuforia method was the hardest to use because it is not well documented and there are not many different types of examples. In order to determine the minimum code needed to make the vuforia work, the team using the example program and continued to remove code from it until only the basics were left.

Then, the team attempted to move this code into a method and call the method. This did not work because the VuforiaTrackables, and the relicTemplate were objects that did not exist in the scope of the method run outside of the opMode. In order to fix this, the objects had to be passed to the method. This is the only method that is used by the team that requires objects to be passed into the method.

Another issue with the Vuforia method is that the values:CENTER, LEFT,RIGHT are not easy to access. They are part of an ENUMERATOR, which is only used in Vuforia. As a work around, the team used a return of a string to determine what was seen by the camera.

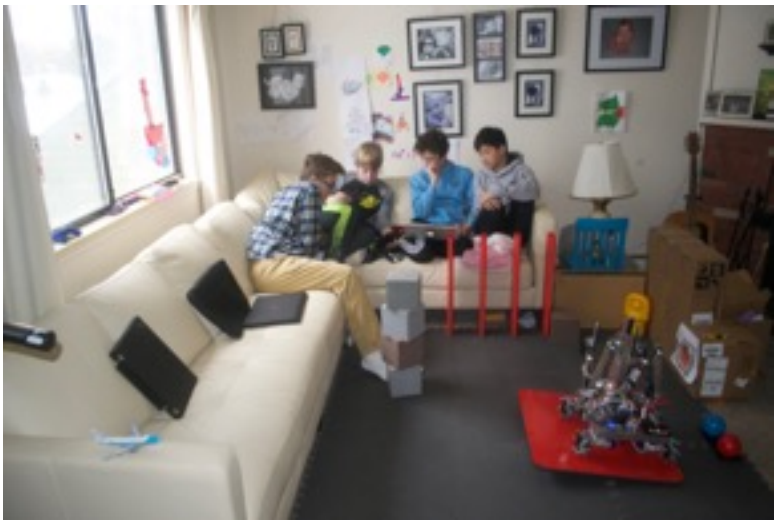
The team also used a loop, similar to the RedJewel method, which advanced the robot if it did not see anything. The robot would continue to do this 10 times. If the robot did not see an image target at the end of 10 times, the robot would place the glyph in the right bay of the cipher station. This was selected as the default because it was the closest bay and it should have the highest change of success.

After the Vuforia method was completed, the team worked on a method to drive the robot using the front right range sensor. Before beginning the method, the team mapped the zone where the sensor could detect objects. In order to do this, one of the cipher stations was placed at a distance of 65 inches from the robot. Then, stacks of three glyphs were moved toward the center line of the sensor. The driver station was placed on top of the stack so that the person moving the stack could determine when the stack was detected. The stacks were placed at different distances in order to determine how much the sound waves spread as they traveled. The team determined that at a distance of 48", the sound waves covered about 8 inches. At 24", the sound waves covered a space of about 7 inches. At inches, the sound waves covered about 4".

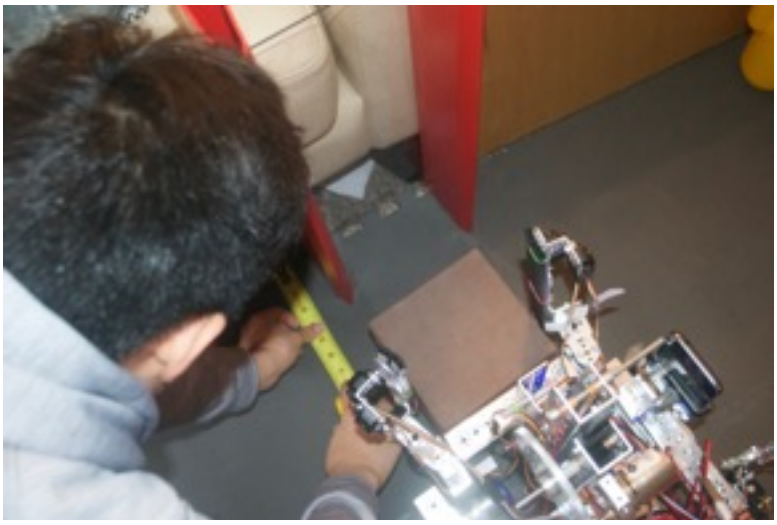
The team also noticed that the sensor makes an reading of 1.78E8 fairly regularly and that the readings have a tendency to jump around.

The team developed a method that took an input for a stop distance and then drove the robot forward until that distance was met. The

robot drove with the encoders and used several of the methods from the AxialEncoders method. The team also wrote a filter for bad data so that if the robot saw a reading of 1.78E8, the robot would stop but continue to make measurements. The team used a timer to determine how long the robot should attempt this method, 10 seconds. The team determined the stop distance by placing the robot, with a glyph, in a position they liked and then they ran the MapUltra OpMode, which allowed them to take continuous measurements with the sensor. The team selected a value of 10 inches.



The team then wrote another method to strafe the robot to the left at a specific distance from the right wall. This was done by copying and pasting the axialRange method and changing the sensor that was being used. This was a bit of a problem because it was hard to remember that the name in the configuration file is the same name as in the OpMode, which the words reversed in order. The team discovered that the robot threw errors when the sensors had the same names everywhere in the program.



Once the sensor was mapped, the team wrote another MapUlta method in order to take some measurements on where the robot should be when it drops the glyph. When the method was used, the team determined they needed to change the sign of the inequality because the robot now needed to have a value that was growing instead of shrinking.

The method worked as planned in 2/3 trials. https://youtu.be/vHHUdetgRjs

Next time...

1) the team needs to finish the red jewel autonomous by determining the distance to the other two bays in the Cipher Station and by connecting the data from the vuforia to the distances to the other bays. This will result in another method.

2) the team meeds to develop a turning by gyro method so that they can do the second side of the red jewel. The sensor bleed on the gyro has been posing some issues that need to be resolved.

1/22/18-Luke, Kenny, Andrew

The team continued working on the methods needed to complete the red autonomous phase. The team reviewed all of the methods that had been developed and then set-out to tune the OpMode.

The team started by adding some code to use the target, from the Vuforia method, to determine how far the robot should strafe to the left.

The team used data from the last session and added a series of if statements to modify the specific distances the robot would need to travel. This was problematic because the lighting in the practice space was not ideal at the end of the day. The team used the screen video capture from the logCat to determine whether the robot did or did not properly see the relic figure. This could be determined because a review of the video would show whether or not the axis appeared that indicated that the image target was detected.

The team learned that the robot typically missed the image target when the blue ball was in the rear position. The team determined this was because the robot did not advance far enough forward to see the target. The team modified the RedJewel method in order to correct the problem .The team added a lift of the jewel arm and then forward push for 4 inches. Lifting the arm was important in order to prevent an accidental release of the red jewel.

The robot did the right section of the cipher station perfectly, where it saw the image target or not. The right was the default so it no image was detected, the robot would deliver the glyph to the right cipher bay,.

Revoery after missing image target for right bay of cipher station
https://youtu.be/eDuNVnUTn6k (iphone)

Driver Station Telemetry for Same Run
https://youtu.be/_5x9YNG8JZI (DU Recorder Android)

Red Ball Picking Up Center Image Target
https://youtu.be/ewiUh2l27hU (from logCat Screen Cast-Mac)

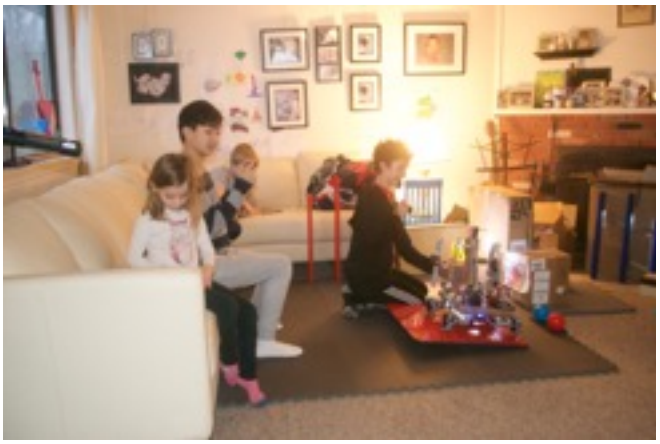
Driver Station Telemetry for Same Run

Red Ball Left Target
https://youtu.be/9-8322_Qyrg (logCat Screen Cast-Mac)

Blue Ball Missed
https://youtu.be/w_GM9TjH-os (logCat Screen Cast-Mac)

Blue Ball Got Left Target
https://youtu.be/qcMl5W7M0E8 (logCat Screen Cast-Mac)

The robot completed the center section, but there was noticeable drift where the front end of the robot rotated away from the cipher station. This was not a problem, however, because both the robot and the glyph were still able to score.
https://youtu.be/zI6MNqMHpwE (iphone view)



However, the far left was a problem. The robot drifted too far to push the glyph iinto position.

https://youtu.be/UtXk_cKwnvM (iphone view)
Driver Station Telemetry of the Run to the Far Left
https://youtu.be/tju2tHPMkSU (DU Recorder on Android)

The team used a variety of documentation to determine the problem. First, the team made a screen cast of the telemetry from the Driver Station using an App from the Play store.

The team also made a screen cast, using the logCat, from the RobotController Station.

The team also made a video recording, using an iphone, to determine how the robot was moving. The video on the iphone was particularly important because it allowed the team to watch the wheels to determine if they all appeared to be moving in the same way.
https://youtu.be/UtXk_cKwnvM (iphone view)
Driver Station Screen Recording of the Same Run
https://youtu.be/l2SwQ_EXT54 (DU Recorder)

The team then decided to modify the power to the wheels of the robot. This led to a surprise that the right and left wheels where not mapped correctly. When the left wheels were set to 0, there was too much friction to move the robot.

https://youtu.be/OJnHuHqKmHc (iphone View)
Driver Station Video of Same Run
https://youtu.be/WUsLJBWxl9M (DU Recorder)

The team then set-out to made the relative power between the right and left sides of the robot different.This did not correct the problem.

The team then made the back section have more power, by multiplying those values by 1.3 in order to over-come the push of the front two wheels. This did not correct the problem.

The team then decreased the power on the front two wheels, by .8, and this made things significantly better.

The team then tuned the ratios using trial and error with values of 1.25, 1.125 and finally 1.1 (on the positive side) and (.75, .8, .85 on the weaker side) to get the wheels balanced. The final ratio worked well on the tests but it was not tested using the whole autonomous section.

During compilation time, the team did some team building.
https://youtu.be/uyJpiRJKtCw (iphone view)

Next Time

1) An additional phone mount needs to be added to the left side of the robot to enable the phone to be mounted there.

2) The updated strafe mode needs to be tested. The team might consider changing all of the driving methods so that the motors match their descriptions.

3) In order to complete the second starting position for red, the team will need to make some changes to the positioning of the range sensor. It seems that the robot will need left and right side sensors to strafe. However, a rear facing range sensor might make the second positions, for both red and blue, easier to manage.

4) The second position will also need a gyro to help control the 90 degree turn so that the glyph can be delivered to the correct location.

5) The team might consider eliminating all of the opModes that are not in current use. This might reduce the time to push the app to the phones. This team might first try to remove the comments from the disable. If that does not work, the team might want to make a new project and then copy and paste only the classes that are being used.

1/25/18-Ethan, Kenny and Luke

The team worked on the autonomous from the easy red position. The team hoped to correct the issue where the robot would drift during long strafe runs.

During the previous practice, the team attempted to "tune" the power to the motors during the strafe run. The team tested the previous practices final values and continued the process of tweaking them. The team determined that they kept the heading relatively constant but they had a backwards drift. They measured this drift as 2" per 39", and wrote a method to correct the drift. The team tested the methods in isolation from the autonomous and the methods appear to work fine. This entire process took nearly three hours. Right at the end of practice, the team tested the autonomous with the new methods and the robot still drifted.

1/27/18 Luke and Andrew

The team met today. The team took stock of their time and the remaining tasks and decided to not worry about the far left cipher station. The team discussed the probability of actually needing to run the far left position and determined that there are 12 possible assignments, 6 for the blue and 6 for the red. Of those 12 assignments, the team would expect to have to run to the far station in two cases. Since the team will participate in 5 runs, there is a reasonable chance that they will not have to make the far run a single time and at most, they could make it once and no more than twice. The team felt that they were far more likely to need to turn the robot during autonomous, because that is required in 6 of the 12 assignments. Therefore, the team decided to focus on developing turning methods and return to correcting the long strafe if they have time in the future.

The team developed the turning methods by taking some code from the demonstration program. The team then made a new opMode using only the commands they felt were necessary to make the opMode function. The team placed these commands into the opMode and made a method to display the gyro heading data.

Demonstration of Turning Method
https://youtu.be/dksH33__amI

Once the team determined that the gyro was not drifting was watching the display to determine that the values remained constant when the robot was not moving. The team determined that the display was working correctly by moving the robot, manually, to determine that the changes in the sensor display were consistent with how the team moved the robot.

The team determined that the initial heading of the gyro is 0. When the team tried to turn to the right, the sensor values got smaller. This meant that the 0 and 360 position were basically the same.

The team then determined that a 90 degree turn to the right would show-up as 270 degrees on the gyro.

The team then imported the relic robot class so that they could access methods that would drive the robot, such as QuadRotateRight() and QuadBrake().

The team then developed a method to turn to the right. When the team tested this method, they learned that the robot would not move because the sensor heading was 0, which was lower than the target value of 270.

The team corrected this problem by starting the robot turning right before they entered the while loop. This technique enabled to robot to turn 90 to the right and stop at a heading of 270.

Demonstration of working method to turn 90 degrees to the right(clockwise)
https://youtu.be/iTTSkMTlJZM

The team then created a new OpMode, RedAutoP2. They added all of the autonomous methods from their previous OpMode for the first red position. The team then had to remove all of the references to the gyro that came from the relic robot class which included several demonstration programs of the gyro.

Once everything was together, the team set-out to drive the robot on the second red position. The team considered going backwards and using the front sensors and going forwards and moving one of the front range sensors to the back. Ultimately, the team decided to move one of the front sensors to the back so that they did not also have to deal with a different JewelArm.

The team then copied/pasted the axialRangeRight program and replaced all of the references to the rangeFrontRight to the rangeFrontLeft(which was now in the back). Initially, the inequality was set-up incorrectly in the while loop and it needed to be corrected to work properly.

During initial testing, the range finder worked fine but it started bouncing between 11 and 39. Luke thought this was because the robot started to detect the balance board or the cipher station. It was unclear which was the problem, so Luke used a value of 38 as a stop point because the robot detected this value consistently.

The team would then add slightly different values to the axialDriveEncoderInches method. Luke thought that using the range sensor would correct problems with slight differences in the starting position of the event or in the end position of the vuforia and jewel methods.

https://youtu.be/wYkWQyKmzDY

Next time, Tuesday evening?:
1) The team needs tol fine-tune both red positions and develop a new OpMode for the RedAutoP1.

2) the team should consider using a method to create instances of the sensors and or place the sensor instances and methods back in the RelicRobot class.

3) the team needs to develop methods for the right position, chiefly modifications of the JewelArm methods so that the robot can use the left JewelArm.

4) the team needs to get the robot design into PTC creo

5) the team needs to continue working on the engineering notebook

1/30/18 Luke, Andrew, Ethan

The team started practice by mapping all of the methods in the RedAutoP2 op mode. This was done because very few team members were involved in every method that was developed for the team. The team takes time to make sure that everyone understands all of the code and what code is already developed in order to keep everyone involved in programming.

The team had originally planned to delegate the programming to working groups, but attendance was too sporadic to wait for individual working groups to finish because other groups needed their work products in order to do theirs.Since the team meets, generally once a week, a delay of a single session can put the team off two weeks or more.

After reviewing and mapping the methods, which took almost 45 minutes, the team began working on tuning the methods to complete the opMode.

The team started by reviewing the notes from the last session. During that session, Luke, Andrew and Kenny developed methods to turn the robot and to do the axial drive using a rear facing range sensor. The sensor is still called frontLeftRange because it was moved from the front left to the back.

The team watched the videos of the final run from last practice, where the robot nearly completed the blue position. Luke measured the distance from the current position of the robot to the desired position as 2". The team used the axislEncoderInches method to drive the robot 2" more on the drive before the turn and the robot executed the turn perfectly.

The team then removed the commends from the final push and the robot placed the glyph perfectly.
https://youtu.be/k74_aHcDgdw

The team then added this code into the if loop under the left section. The team moved onto the center section. They changed the target manually because it was too dark for Vuforia to work correctly. The team then added a distance of 9" instead of 2" to the axialEncoderInches method. Again, the robot completed the program perfectly.
https://youtu.be/cAoZy5_cr9w

The team was shocked, however, by the final run to the left. The team made manual changes and then realized that the previous code should have been placed under the "RIGHT" section of the code and not the left.

The team made this change and proceeded to test the final distance. The team was disappointed that the run did not seem to operate correctly. When the team checked the telemetry, they learned that the robot reporting that it drove with the range senesor until correct distance was measureed. However, a visual review of the video indicated that this was not the case.

The team repeated the run several times to determine if the error was a fluke or whether it was persistent. The team discovered that the error appeared to re-appear.

The team decided to do some more trouble shooting so the team added more telemetry to the program so that it would report where in the program the robot was operating. This would enable the team to determine if the robot was using the wrong input due to a logic error in the code. The additional telemetry indicated that the robot was exectuing the code correcly and that the sensor readings from the senor were the problem.

Luke adjusted the sensor to be slighly higher, hoping that the error was due to the detection of lower objects. Additional testing indicated that this did not correct the problem.

Telemetry Data from Final Run
https://youtu.be/Hd5XUFOptwo

Video of Final Run
https://youtu.be/L_oKip9bDmU

It was late and Luke was the last member working so he decided to stop working for the day.

Next time

1) Explore the problems with the far left run. Develop some alternatives in case the range sensor appears not to work properly

2) work on the blue starting positions

3) convert the DrivingWithEncodersTelemetry program into RedAutoP1

2/2/18-Luke and Ethan

They started witha review of the last session and discussed the pros/cons of completing the red position versus starting the blue position. The advantge working on the red was that the team was nearly done. The advantage to working with the blue was that the team had no points yet on the blue side and the team was running out of time.

Luke and Ethan decided to work on the red. They repeated the OpMode from last time and repeated the same error, with the robot stopping too soon.

Luke and Ethan attempted to determine if the problem was with the range sensor method or the encoder method. In order to make a decision, they re-ran the OpMode and took a video with rules laid out on the floor. They also marked the expected positions using duck tape on the floor.

When they ran the OpMode, they determined that the range sensor and the encoder appear to work fine. They decided that the problem had to be the parameters.

This raised a problelm, however, because the previous session had the center and right positons work pefectly. This suggests that something changed form the center and right when compared to the right.

One possibility was that the battery power was low. This could explain why the robot did not run as well. The battery was swapped out and replaced with a fresh battery.

Then, the robot was directed back to the center. When this was done, the robot fell short again. This suggests that the problem was with the parameters passed to the methods.

Luke and Ethan took some new measurements and got the robot to the far left position.

This shows the demonstration result.
https://youtu.be/4i0rbu5lBZI

Then, they decided to work on the blue position.

They decided to do this by tweaking their existing methods.

First, they copied and pasted the RedAutoP2 class into a BlueAutoP2 class. Then, they identified all of the methods that would be needed to complete the Jewel on the blue side, which included: RedJewel(), JewelArmDrop(), JewelArmLift().

They started with the JewelArmDrop and JeweLArmLift. They changed all of the references from the JewelArm to JewelArmLeft. They also changed the JewelArmPosition variable to a local variable, LeftJewelArmPosition instead of the robot.JewelArmPosition. They used used this value inside the while loop to update the position of the servo. They also had to change the internal logic of the methods because the servo positions where changing opposite, relative to the initial method. In other words, the drop program was not changing the servo

position from 0 to .775 instead of 1 to .275. This also meant that the decrement was changed to an increment and the inequality changed from greater than to less than.

This process was repeated for the JewelArmLift method, which was changed to LeftJewelArmLift.

Inside the BlueJewel method, the color.Right was changed to color.Left. Likewise, the motor direction was changed in the axialDriveInches inside the method so that the robot now backed-up when it saw red, to remove the red jewel from the board.

This shows that the BlueJewel() method was working.
https://youtu.be/pqzSbku2-HU

2/7/18 Luke and Andrew

Luke and Andrew worked on the Blue Jewel method and tested it with bothboth the blue and red ball in the forward position.

This demonstration is with the Blue Ball in the rear position so the robot drives forward to remove the red ball.
https://youtu.be/Zd7r0k52ByQ

This was an attempt at a double demonstration with each ball in the forward position. The robot was unable to remove the red ball when it was in the rear position because it did not go far enough backward.

https://youtu.be/ANYjlW3S_hY

This shows the successful completion of the Jewel with the Red Ball in the rear position an adjustment to the starting position and an increase in the distance per backward movement to scan.
https://youtu.be/QObRe3X7VtY

Luke and Andrew also worked on completing the BlueAutoP2 opMode

Next Practice

1) team needs to work on BlueAutoP1-makde decsion about fd turn fd turn fd verus fd strafe
2) team needs to get started on PTC Creo Drawings

2/10/18 Luke, Ethan, Kenny and Andrew

The team made several key decisions today. The team decided to not use the vuforia method and run to a specific location on the cipher board. The team made this decision in light of two important pieces of information. First, the autonomous was taking nearly the full 30 seconds without the Vuforia. Second, the robot seems to struggle with completing the various positions. On some days, a certain set of parameters work. On other days, different parameters work. The team had demonstrated they can get it to work but they would need to practice on the actual field to feel more confident.

After deciding to not use the vuforia, the team decided to make the center position on the glyph station the primary target. This over-rode a previous decision that focused on going to the nearest target. The team made this change because the center gave the team more ways to get points if something went wrong. If the nearer target was left, and the team under-shoot the target, the robot might not make it to safety. Likewise, if the team over-shot the far target, the team robot might not make it into the safety.

The team also made a decision to replace the algorithm for the autonomous on the Blue1 position. The team decided to use fd, turn, fd, turn, fd instead of strafing. The team made this decision because they felt that the turning, with the gyro, was more consistent that the strafing. The previous algorithm was FD range sensor, strafe range sensor.

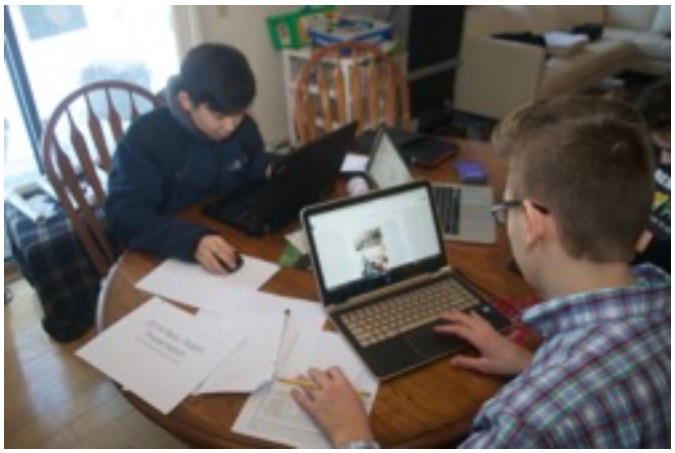
Luke completed the final position and made updates to the OpMode to reflect the team's decisions on the autonomous.

Run to the Center from the Red 1 Position
https://youtu.be/7uQVnnN85kY

IMG 9460-Red P1-Failed run that still works
https://youtu.be/AJV8EH_xl2Y

Final Run Red P1
https://youtu.be/G48cONOTt6g

2/14/18 Luke, Ethan and Andrew



Ethan went over the engineering notebook and made some edits.

Andrew worked on the lifter assebmly in PTC Creo. He was almost finished when his battery died.

Luke worked on complete the programs for autonomous in order to do full testing of the robot. When Luke was working on the programs for the RedP1, he decided to use the multi-turn approach insteda of the strafe. He made this decision because he felt it worked more consistently.

Luke copied the data from the BlueP1 into the red and then modified the turns. This required Luke to make the method, gyroRightTurnParameter . This was because the second turn required the robot to turn back from 90 toward 0. This created some problems but they were soon corrected.

The programming evolution went from R1 to R2, thinking that the R1 position would be easiest. Then, the R2 to the B2, thinking this would be easier than B1. B1 was devloped using new methods to take two turns instead of strafeing. Finally, the R1 position was based on the B1. This made a full circle of OpModes evoloving from the basic assumptions to field tested robot.

The team then started testing the robot. They made a decision to not integrate the teleOp methods into the autoOp modes.

The team did several full tests, which produced a fw problems.

First, the wheels had a tendency to pop-off during teleOp.

Second, the robot missed the jewel several times. Possibly because of starting position problem. If might be good to know if the jewels are set before the robot is placed, or after.

This is the first working full run that earned all of the expected points.
https://youtu.be/h4kxgwn8FE8

Third, the battery on the phone died. This led the team to have to stop testing.

2/15/18

The team met yesterday. Andrew completed work on the lifter cad assembly. He used a vareity of solid part files, which created problems when the assembly was integrated into the main computer. It took a long time to identify each file that was different on Andrew;s computer versus Luke's computer. In addition, the constraints posed some problems.

Ethan worked with Luke on the Red 1 position using two turns. This caused problems in the previous practice and was Luke's top priority. Luke began testing the OpMode and collected loads of video telemetry. The gyro was producing weird errors. Luke added a number of calls to the displayHeading() method to verify that the senor was collecting data correctly.

In several instances, Luke noted that the value being displayed was incorrect, at something like 3636 degrees. This didn't make any sense.

At the same time, the robot did not appear to consistently run the axialRangeRight() method and it stopped before getting to the proper distance.

Luke recorded several runs trying to pin-point the distance problem and he noticed that the blue light on the REV Expansion Hub flashed before the robot seemed unable to collect new data. Luke thought this might indicate a loss of signal.

Luke inspected the wires and noticed pressure between the mini usb cord that connected ot the expansion hub and several other wires, including the wires that connect the controllers to the power module. The battery was also pressing on these wires and Luke felt strongly that the wires would beed to be position differently in order to reduce pressue and not cause a loss of signal.

Luke continued to test the robot trying to get it to work with two turns. He was able to get it working with the first turn, but, then it would not work two times in a row.

In frustration, Luke questioned whether or not the two turn algorithm was in fact better than strafeing. The team decided to abandon strafeing when they were planning to deliver the glyph to the appropriate bay at the ciper station. The problem with the strafe was that it could not get to the far position. The team developed the two turn algorithm to get to the far position, but it was not working on the red side. However, the robot could strafe into the middle position, which became the primary goal. Thus, Luke decided to make a change and re-insert the strafe. He took data from the original OpMode and used to quickly complete the run.

2/16/18
AM

Luke considered some different ways to move the wires to reduce tension. One approach was to elevate the expansion hub above the connectors for the power module. A second approach was to run the motor wires under the expansion hub and through the gap that was created by elevating the expansion hub. Third, the battery was moved form the center of the robot to the back, where it placed no pressure on any other wires.

During testing, Luke became concered about the vuforia initializintion that caused the robot to get stuck. This was corected by restarting the robot controller app, which would be a problem if it happend during the competition.

Similarly, sometimes the gyro got stuck when it was calibrating. Luke thought it would be best to remove those elements if they were not using them. So, in position Red1 and Blue1, the team would use strafe and not double turn. The team would eliminate the vuforia and gyro calibration from these OpModes. In position red2 and blue2, the team would keep the gyro calibration but elminate the vuforia.

Luke also wanted to add some error detection to see if there was a way capture when the robot might have a catastrophic faliure.

PM

The team worked on the Red1 position and introduced a new algorithm for navigating to the cipher station in autonomous. The new algorithm collects a single valid distance to the wall, then calculates the distance needed to travel to get within 10” of the wall. This value is used in by the axialEncoderInches() method to drive the robot into position. This is considered more stable because it uses only (1) sensor value.

The hardest thing about this method is ensuring that there is a valid distance. The team spent nearly an hour trouble shooting the problem of valid data before the team realized that they were using the wrong sensor to collect distance data.

Once the team had the right sensor, the program worked nearly perfectly. The only problem was to filter out the values of 1.79E8

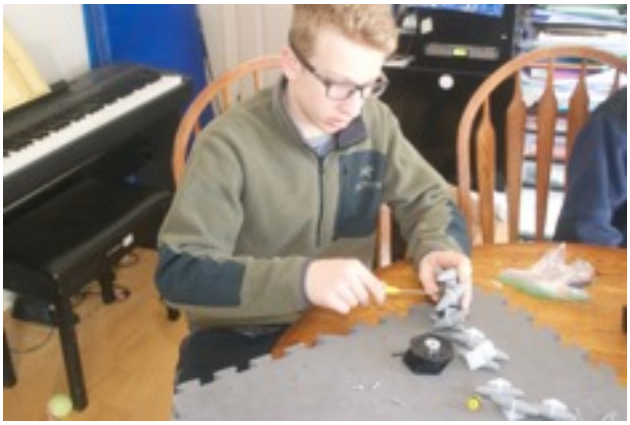
Next, the robot strafes to left using the strafeRange() method.

The team also worked on the red2 position. It worked fine but had to be modified to work on the center target. However, the team discovered that the relic was quite hard to get at because of the angle between the robot and the corner. In the process of testing this part of the game, the robot lost is wheels each time.

The team researched possible solutions, including order VEX wheels, order wheel converters and so on. The problem with ordered new parts is that the team made this realization at 5:30 EST on a Friday. This mean that the parts could not be ordered until Monday. Even with over night deliver, the wheels might not arrive until Wednesday. This would also cost nearly $500 for a full replacment when the shipping was added.

2/17/18

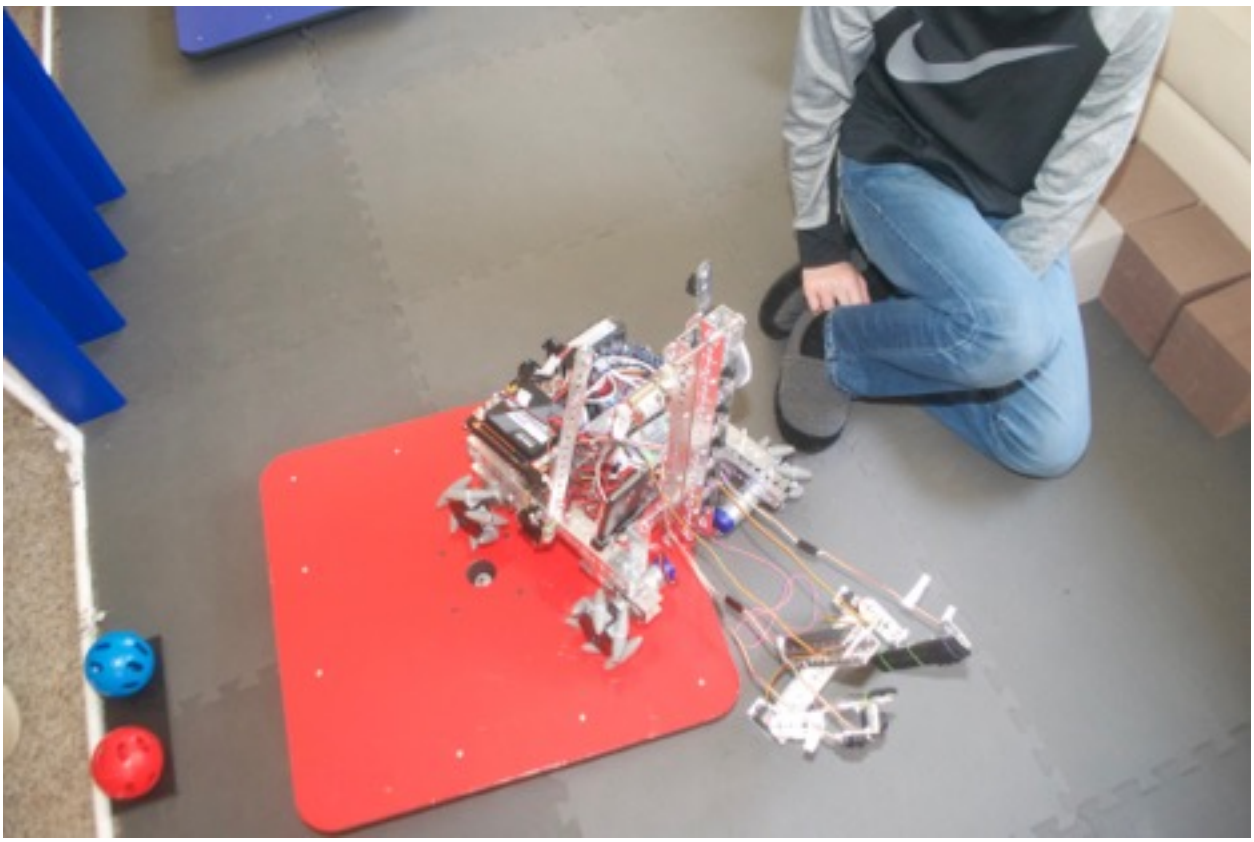
The team worked on the wheels today and got them working successfully. The team discovered that the wheels could be aligned in a way that blocked the bolts from passing through the wheels. By chance, some of the wheels were assemble such that the bolts could pass through and others were assembled in such as a way that the bolts could not pass through.



After the wheels were mounted, the team set-out to test the entire run-through. On the first run, the team discovered that the lifter had a design flaw. It got stuck on a glyph during teleOp when Adnrew tired to lower the lifter. The second stage moved downward, which created slack in the pulley line. When the secodn stage hit a low point, the third stage became dislodged. This was a problem because the robot could not drive back onto the ramp.

The team realized that they could correct the problem by limiting the range of motion

of the second and third stage. The pulley rop limits the lower limit of the third stage, but it cannot limit the range of the third stage if it is relased from the pulley. To correct this, the team placed a safety line around the pulley that would block if from falling out of place.



The team also realized that they should limit the range of motion on the second stage because it became dislodge by going too high on the relic a few nights ago. The simplest way to manage this problem would be to use motor encoders. The problem was that the team was using tetrix motors and they did not have a third tetrix motor controller. This motor was powered through the REV expansion hub, which needed an adapator to receive the tetrix encoders. The team did not think it had such an adaptor.

The team tied a string onto the second stage to limit its upward motion.

The team then tried to test the run-through again but faced a series of set-backs, focused on a combination of weird electrical failures such the left motors not operating depsite not throwing an error. The team decided that in such an instance, they should just drive the robot back onto the ramp for the points at the end. As long as the robot does not go too far, it can get back on the board.

The robot also had issues with the range sensor. It several instances, the robot got stuck. In other instances, the robot when backwards unexpectedly.

The team continued to rehearse and discussed various strategies and how to interact with other teams.

Red 1-
Team would do Jewel (30 pts) worked almost always
Team would attmpet Glyph to Cipher Station (15 pts)
Robot to Safety (10 pts)

During TeleOp
Stack (2) sets of (2) glyphs for column bonus
5 glyphs for 10 pts
i column of (4) for 20 pts

End Game
Back onto balance board 20 pts.

The team discussed how to approach other teams to discuss how to handle the robot. During this practice, the team discovered that their current robot design could not reach the relic in the corner. The team explored a couple of possibilities for making sligh modifications to the gripper to make this possible.

Next Time
1) modify gripper for relic
2) Test Red 2
3) Test Blue 1
4) Test Blue 2

Practice next week-
Monday 6-7:30
Tuesday 3:30-5:30
Wednesday 3:30-5:30
Thursday 3:30-5:30 (Van Cliburne at UVM Yekwon Sunwoo)
Friday 3:30-5:30

2/20/18 Kenny, Ethan

Kenny deisgned and cut plastic housing to mount the team number for the robot.

Kenny also worked on modifications to the gripper that would allow for the robot to lift the relic when the relic was in the corner.

Kenny tried a variety of approaches to solve this problem, including adding heavy guage copper wire, bent into hooks, to grab the relic from underneath. He also tried adding flats.

In each case, the additions to the gripper made Kenny concerned that the additions might make it harder for the robot to place glyphs in the cipher station.

2/21/18 Luke, Ethan, Myles and Andrew
Yesterday, the team worked on a script to approach other teams to learn their capabilities. The team also evaluated some robot executive summaries and ran significant amounts of testing of the teleOp.

This testing revealed several important notes.

1) The robot lifted cannot place a fourth glyph on top of three glyphs. This is because the wires from the controllers are not long lifted to lift that high. This means that when the team plans to make two runs, with 1 block and then with two blocks, the team should place the 1 block first and follow it with the two blocks.

2) The team had difficulty accessing the glyphs when they were placed in a pile because the gripper closes on the outside of the glyph. The team spent lots of time during the teleOp trying to select specific glyphs that they could easily grab. In many runs, no glyphs were placed. In order to create some room for the gripper, the team explored driving through the glyphs and plowing them into a better position. This had some success from the back but was not effective from the front or sides because the wheels had a tendency to get stuck.

Next time
1) consider bumpers for the sides and back to allow the robot to plow through the glyphs to seperate them making them easier to grab.

2) communicate with other teams to see if they plow the glyphs to make things easier.

2/22/18 Luke and Andrew

Luke and Andrew reflected on the probelms from yesterday during the teleOp. They decided that the main problem in teleOp was that the glyphs were too close together and that they needed a simple way to plow through the glyphs to create space. The decided to make bumpers to keep the glyphs from getting stuch on the wheels.

The fist bumper was placed directly into space between the right side tires. The team could not make an identical bumper for the other side because the team did not have the necessary parts and there was not enough time to order new parts.

The team tested the initial bumper and determined that it gave them a significant advantage compared to not having the bumper. Luke and Andrew decided to make bumpers for the other two sides, the back and the left.

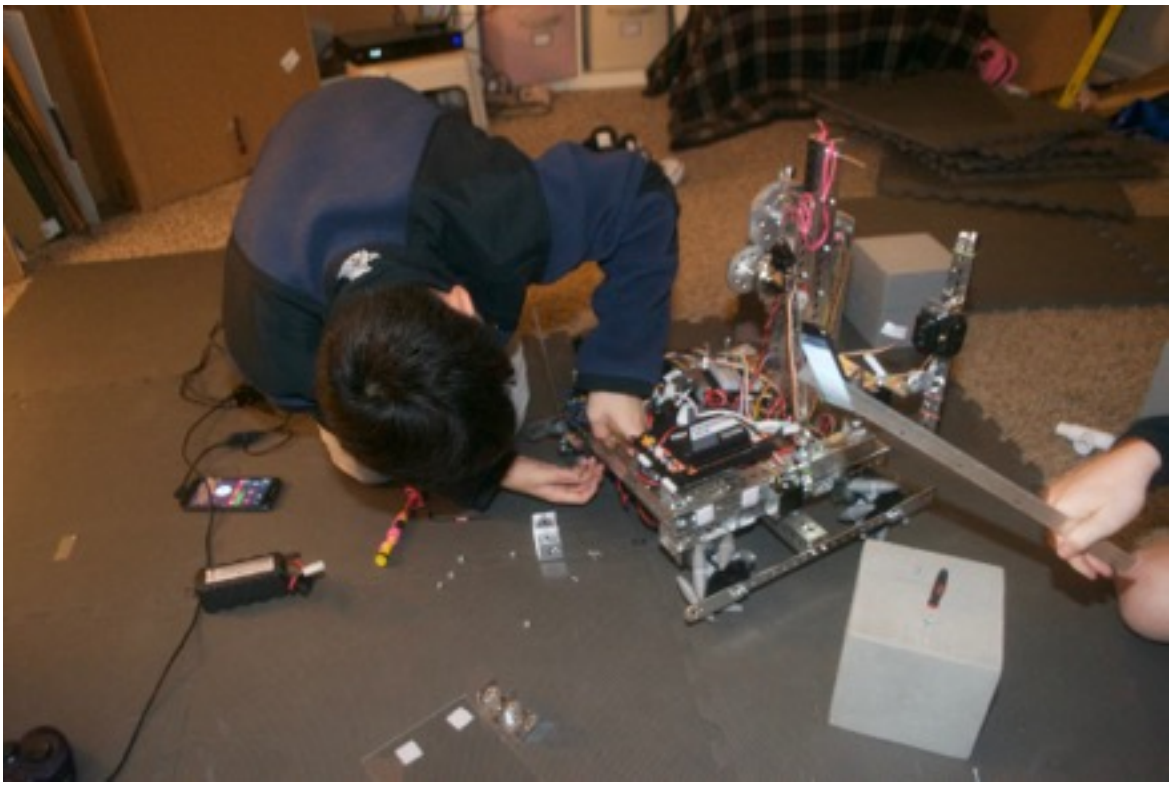
The left side bumper was fabricated from a piece of flat aluminum purchased at Lowes. Luke cut it to size using a mitre saw and he and Andrew drilled holes into the aluminum using a drill press with a 9/16” bit.

Andrew then attached two small tetrix channels to the underside of the bakc of the robot. He had to remove the range sensor to do this and he was not able to attach a second set of channels because it was too difficult. He thought he would have had to disassembled the rear end of the robot to make the change.

Luke cut a 17” peice of flat alumunum and used the drill press to make holes to fasten it to the rear end of the robot.

During the initial testing of the full bumper system, the robot was able to easily plow through the glyphs. This opened some new possibilities for the team, including plowing glyphs close to

the cipher station and opening space to make it easier to grab the glyphs.



Video of Initial Testing
https://youtu.be/wH7FFoKwcLY

During the first telOp similation, the team easily placed anough glyphs for a full column with a trip with a single glyph and then a second trip with two glyphs.

During the second teleOp simlation, the team placed two sets of two glyphs.

During the third trial, the team placed a single glyph, then two glyps, then a single glyph and another single glyph for (4) total runs. This enabled the team to complete both a row and a column.

First Position Composition Video
https://youtu.be/kj2N9a4l2Xo

Second Position Video

https://youtu.be/ws0qdOBqS2c\

During testing, it was clear that the first position is preferrable to the second starting position beause of the angle between the driver and the glyph station.

Next time

1) rehease more teleOp
2) Discuss how to manage other teams now that we have a clear perference for starting position
3) review robot presentations
4) review ambassador interactions
5) attach number and straw
6) update PTC Creo Drawings to include bumpers
7) Integrate PTC Creo Drawings into Slide Show

2/23/18 Kenny and Luke

The team worked on the final touches of their robot. Kenny and Luke used the drill press to cut holes into the plastic housing. The housing was difficult to place because the screws and nuts were almost impossible to reach when the plastic was placed on. For this reason, one of the screws was attched after using the drill press to connect a series of holes together to make a slot. This allowed the fasteners to be connected before the plastic was added and allowed the plastic to slide into position and then be fastened tighter to hold it in place.