Voyager

Engineering Notebook

FTC Team # 7923

Table of Contents

FTC Team # 7923 | Summary

Hello. We are FTC team # 7923, also known as Voyager. This is a quick summary of our team.
Voyager started when Jasper was very interested in doing FTC robotics during his high school years. He found a program at the University of Alaska Fairbanks, and asked if he could start a team. This team built up three members, and worked at the university three days a week. We went to the local competition, and ended up finishing as the winning alliance, also taking home an inspire award and an innovate award. We moved on to the state competition, but fell short of super regionals.

This year, Voyager will again be working up at the university and plans to make a competitive robot while practicing gracious professionalism and connecting with other teams and the public via outreach and mentoring.

Please consider the following pages in our notebook:

- Pages 15-18
- Pages 28-29
- Pages 31-32

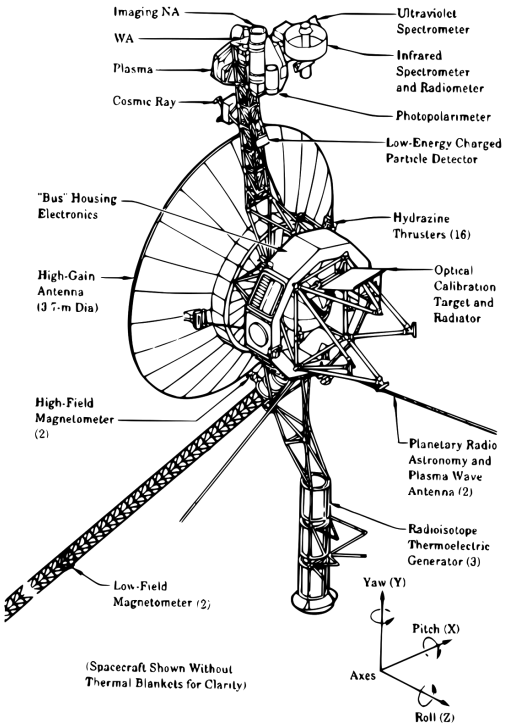
Team and Outreach

This section provides information about our team and our outreach during this season.

About Our Team

Our team started in 2013 when Jasper Holton decided to start a robotics team at West Valley High School. He did this with the help Clay Allen, an FTC veteran and an intern at the Alaska Satellite Facility. Clay managed to find space, materials, and funding to support our team and several other teams working at the University. None of our team members had any experience with FTC, though some of us had past experience with robotics and programming. Despite this, we managed to do quite well in both the local and the state competition.

In 2014, our team started up again, this time slightly later in the season. Despite this, we gained a new dedicated member, and built a robot using the knowledge we had gained from our previous year competing.



Our team name, Voyager comes from both the Voyager 1 Probe, and the Star Trek TV show.

The funding for our team comes through the Alaska Satellite Facility at the University of Alaska Fairbanks. This funding is for outreach, and comes from a grant from NASA.

Meet the Team

Supervisor: Tim Illguth

Background
Tim was raised in Fairbanks Alaska and began working for his grandfather full time at the age of 12. He took correspondence courses while working full time to allow him to graduate with his peers in the class of 1999. Afterwhich he went on to work in many different trades positions, before attending UAF full time in 2010 as an Electrical Engineering student.

Coming to FTC
Tim was hired on with the Alaska Satellite Facility (ASF) as a student and became a full time employee in 2014. He began volunteering in robotics and moved into a position leading the efforts of the NASA robotics outreach grant for UAF. He now is mentoring and coaching 1 veteran FTC team and 2 Rookie teams.

Team Captain: Jasper Holton (Grade 10)

Jasper Holton was born and raised in Fairbanks and is a student at West Valley High School. He has been involved with robotics since he was seven years old. He keeps himself busy with skiing, running, science, math and engineering. Jasper is great at programming, and likes to build autonomous robots to do complex tasks. Jasper has been involved with FIRST since he was in 7th grade, and he hopes to have another excellent FTC season. He is the founder and Team Captain for Voyager.

Team Member: Louis Bastille (Grade 10)

Louis was born in Fairbanks, and is currently a sophomore at West Valley with a specialty in math and science. He is the main driver, which has a funny story behind it with him being the one who “plays the most video games,” yet he doesn’t even have a console, being with Voyager from the start. Besides that, his views in life have always been directed towards robotics, having participated in FIRST since 2nd grade. Aside from these activities, he is a hard competitive swimmer with a right-oh attitude. Louis may tend to procrastinate, but when he needs to set his mind to it, he can help in all situations.

Team Member: Nathan Sanches (Grade 9)

Nathan was born and raised in Fairbanks Alaska. He is a 9th grader at Lathrop High School, and has been doing FTC since 8th grade. He enjoys running, eating, sleeping, and robotics. This is his second year with Voyager, and he is looking forward to another great season with them.

Team Member: Alex Cater (Grade 11)

Alex a junior at West Valley High School. Alex favorite classes are AP Chemistry, computer programming, and physics. This is his first year with Voyager and already have had a lot of fun. Alex plans doing FTC next year and having even more fun building awesome robots.

Outreach

Our team did several community outreach events in order to try to get more people interested in FIRST, and to promote robotics in our community.

Outreach #1 | Science Potpourri

The Science Potpourri is an event held by the College of Natural Science and Mathematics invites kids come and learn about science. We participated in an event there where we taught kids about robotics. We set up a field and let the kids drive our large FTC robots, and we set up a small arena for smaller, lego sumo robots. This was a fun outreach, as it is great to be teaching young kids about science and mathematics, and getting them interested in robotics.

Outreach #2 | Tanana Valley State Fair

// Need to add information about outreach, and pictures (if at all possible)

We haven't done much outreach so far, but we plan to do more. We plan to do outreach to middle schools such as Ryan and Randy Smith, and participate in other community events.

Strategy

This section gives a description of our strategy for the robot game.

Autonomous:

Though our autonomous program will always be a work in progress, we have a fairly good one currently. Our autonomous program drives from any one of the two starting positions and picks up the smallest of the three tubes. It then brings the tube to the scoring area, and dumps the balls that it is carrying into the tube. Eventually we plan to have more features in our autonomous program, such as the ability to knock down the stand holding up the balls in the center goal.

Teleop:

Our strategy for teleop is also fairly simple. We plan to simply collect as many balls as possible, and put them into the larger two tubes. Our robot is quite good at picking up balls, and can score them into the tubes without much difficulty.

Endgame:

Our plan for endgame is to move the goal tubes up onto the ramp, and also park on the ramp ourselves. Our robot has a mechanism designed to hold the goal tubes and move around with them, so getting them onto the ramp should not be very difficult. However, we tend to have some difficulty parking on the ramp. Also, our robot is capable of scoring in the center goal. However, we do not know if we will have enough time to do this. We will certainly be able to do it if our alliance partner is not able to.

Phoenix:

Phoenix is what we call our prototype board system. This system has several main functions. The first is as a failsafe. When Phoenix detects that the NXT on our robot has become unresponsive, it restarts the NXT and thus fixes the problem. This means that if our robot dies halfway

through a match, we will be able to recover and continue scoring. The second function of Phoenix is as a monitor. It shows the status of both the NXT and Tetrix battery mounted on the robot. Using this, we are able to see if we need to change our batteries in order to have a more successful match. The third function of Phoenix is to give us a way to select our autonomous program. On the bottom of Phoenix is a switchboard with several switches that we can use to alter the state of our robot. We use these to select our autonomous program, among other things. Here are some pictures of Phoenix:

Engineering

This section gives a day by day description of the process of brainstorming, designing, building, programming, and testing our robot to prepare it for competition.

Example entry:

~~~~~~

Day X: Month day, year | Start Time to End Time
By Author's Name, Additional Author's Name

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Stuff is written here.	Stuff is written here also.

Additional notes/ideas/drawings/whatever is on your mind.

~~~~~~

Some opening notes:

We tried to meet as three times a week, but that didn't always work out perfectly. You will notice that some days are missing from our regular schedule. These days are days where we were unable to meet due to various other schedule conflicts.

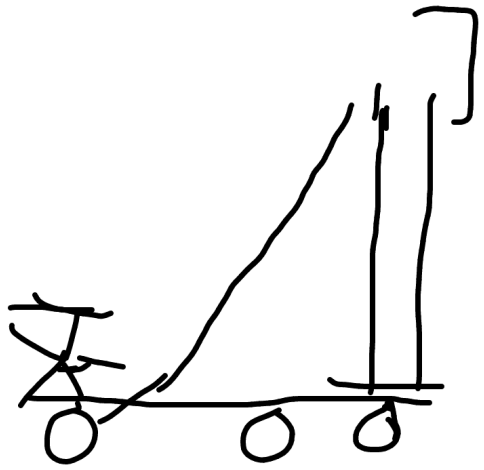
We wrote an entry for our engineering notebook every day we were in the lab, with few exceptions. There were one or two days where we met but forgot to make a notebook entry, but otherwise what you see in the notebook is an accurate portrayal of the process of building our robot.

That's all there is, so enjoy reading!

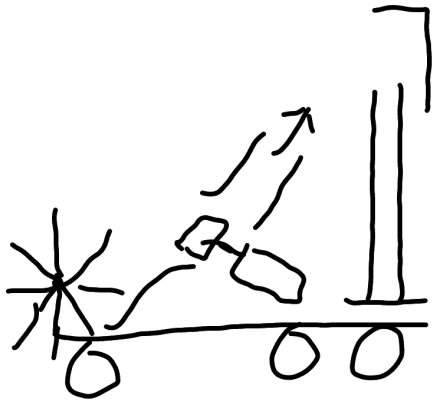
Day 1: September 6, 2014 | 11:00 AM to 2:00 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was the FTC kickoff. We learned what the challenge was, read through the rules, and talked with our team members. We did a lot of discussion of our plans for the next season. We also did a read through of the rules with several other teams, including Schrodinger's Hat and Denali Nerd Association.	This went pretty well. We came up with a few good ideas and did a lot of discussion. We now understand the rules of the challenge a lot better, but we have some questions about ambiguities in the game manual. Can you carry around scoring tubes full of balls in the robot without getting penalties? What about dragging them?

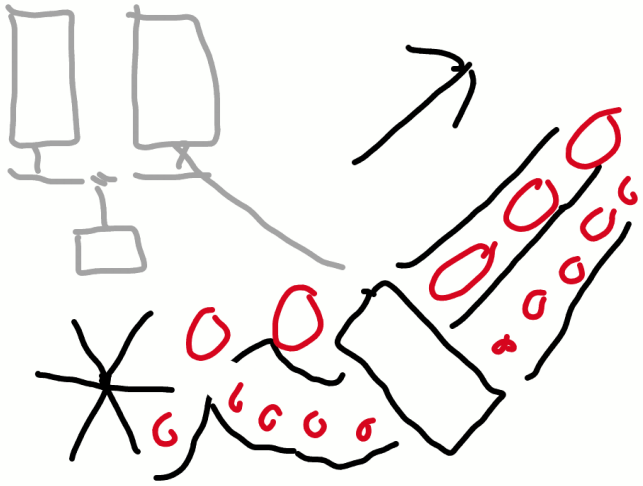
Here are a few sketches of robot design ideas:



This is a rough sketch of a design idea involving the robot taking the tube inside itself and holding it, while a sweeper sweeps balls up into the tube. Behind the tube is a catcher on a linear slide to keep the balls from going out the back. This idea is a good one because it means that the robot doesn't really have to worry about how many balls it has in it, since it is constantly putting the balls directly in the tube. However, this idea won't realistically work so we refined it:



In this refined version, the balls go into a motorized shooter that shoots them up to the top of the catcher, where they fall down into the tube, no matter it's height. This means that we can just drive around picking up balls and filling the tube. Then, we can drive to the top and park the tube up there before going and getting another one. However, since we have different sizes of balls, we needed to develop a simple sorter to account for that:



Small balls go through the bottom, and large ones go through the top. The shooter wheel is smaller on top to account for different sizes, and the balls go part way up a tube before being released, for additional accuracy.

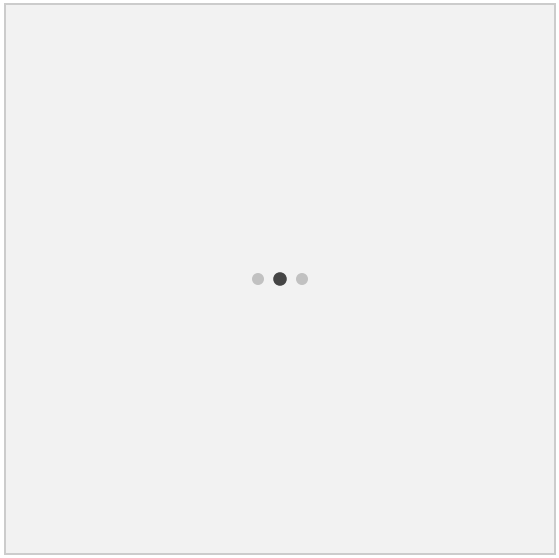
Also note that in all of these drawings the front wheels are omni wheels. The front is where the sweeper wheel is.

Day 2: September 9, 2014 | 11:00 AM to 12:00 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
This was a short meeting that was mainly for brainstorming. We also looked over the game manual a bit more and found out that the ideas we had before are indeed legal. The main thing we did was come up with an idea to resolve the NXT brick freezing problem. We had the idea of using a Hitechnic protoboard to break the connection to the NXT brick when it detected that the brick had become unresponsive. After the connection break, it would resume the connection so as to restart the brick.	This went pretty well. We got some good new ideas, and we pretty much made a decision about using the protoboard restart idea. It seems like such a good idea and it is very low risk, and very high gain. Overall, we are happy with how this meeting went.

Here is a pretty basic description of the protoboard restart idea:

We attach a protoboard sensor with a 9v power supply to our robot. This protoboard will monitor signals sent from the NXT brick during the game. It will be able to detect when the NXT has frozen up and is unable to receive commands. Once this happens, it will break the battery connection using a normally closed relay, and then resume the connection, thus restarting the NXT brick. If this works, within 20 seconds of an NXT brick freeze we could be up and running and in the game again. We have codenamed this idea the “Phoenix.” The idea behind this is that the NXT brick will be able to reborn from the fire and ashes of a freeze up.



Because the relay is normally closed, there is really no risk in doing this. If the battery fails, the device simply won’t work, but the robot will continue functioning as normal. If we get the code right, it should never restart unless it needs to. Anyway, we think it’s a really good idea and are excited to see it implemented.

We haven't had access to the lab yet, but once we do, we will start meetings where we are actually working on robots. Right now, we are simply brainstorming ideas.

I dont really know what to write anymore so yeah.

Day 3: October 2, 2014 | 2:30 PM to 5 PM
By Nathan Sanches

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
Today was our first meeting at the lab. We worked on two main things today; building the robot chassis and beginning the process of designing our robot for the challenge.	Overall, this meeting went well. All of the members who showed up helped with the construction of a stable chassis, and everyone contributed their ideas for implementing our solution to this years challenge to the robot.

Currently the chassis is very simple. It is basically a square of metal with four wheels and two motors. We tried to leave it pretty simple today so that modifying the base will be easy, and it will be relatively simple to begin creating mechanisms for our game strategy.

Day 4: October 5, 2014 | 2:30 PM to 5 PM
By Jasper Holton

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
Today we worked on trying to find a way to get the balls into the air for our design. We pretty much decided that using two spinning wheels is going to be impossible, so we created a fast spinning gear train to flick the balls up into the air. We built a single gear train and added it to the robot, and have plans to build another.	This went pretty well. We managed to develop a fairly good gear train that can flick the balls into the air. This worked sufficiently well in initial testing, but will need to be tested much more extensively. We don't have a large size wiffle ball, so we can currently only test with the smaller ones.

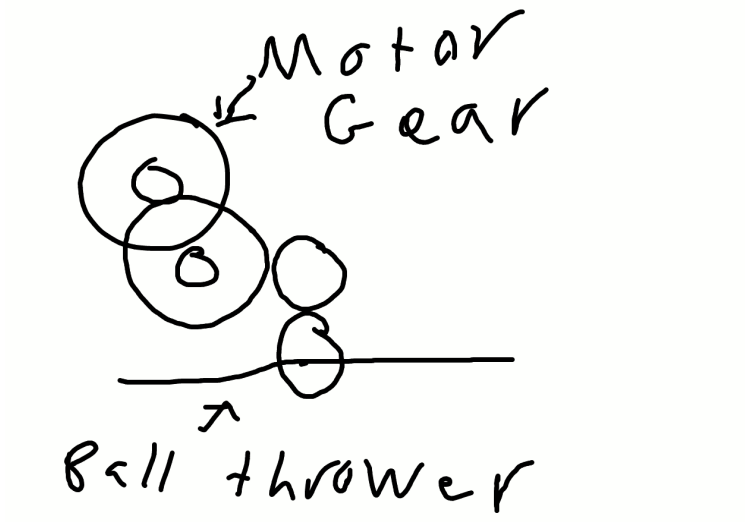
The chassis is a bit more complex. It has one gear train running a fast spinning axle with flaps on it. It also has a ramp on the front so the balls can be lifted into the air. It is able to take in balls and flick them up with enough velocity for them to well clear the five foot limit, so we should be good in terms of height.

Day 5: October 7, 2014 | 2:30 PM to 5 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we decided that our previous gearing system would not suffice. It wasn't secure, didn't hold the gears well, was off centered, and generally wasn't going to work. We spent the day building two much better gear trains that are a lot sturdier. We had some minor setbacks in mounting them to the robot, but after a little while we managed to figure out a way to make it work. During testing, it was able to flick a small wiffle ball pretty high into the air. It also makes a nice fan.	This went pretty well. We managed to develop a solid gear system with two motors that can easily flick the balls high into the air. We tested this system and it works well with the smaller balls, but we have yet to test it with the larger, as we have none available. At the end of the meeting, we roughly added a tube to direct the balls. This worked sufficiently well in initial testing, but will need to be tested much more extensively.

There has been a minor setback in the protoboard idea. The problem with this is that we do not currently have funding available to make this happen. In order to solve this problem, we may simply buy the board with our own money. Alternatively we may be able to wait until funding is available, but it is unlikely that we would have it in time for the competition if we did that.

Here is a drawing of our gear train:



This gear train has two motors and is big:small:big:small so it is very fast but still fairly powerful, as there are two motors.

Day 6: October 9, 2014 | 2:30 PM to 5 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we made a lot of adjustments to the robot. First of all, we added a cage-like structure around the ball flicking area, and made a better flicker. Now this part of the robot is much more stable and works a lot better. We also added fancy new metal ramps to the front of the robot, and this makes sucking up the balls and throwing them much easier for the robot. Also, we solved the problem with the ramp scraping against the ground with this new ramp. We also added a new mount for the tube which is much more stable. During testing, we managed to launch a ball with enough velocity that it could have easily cleared 10 feet, so we consider this a success. During this meeting, Nathan mostly worked on getting some linear slides to work. None of us really know how they work, but Nathan is experimenting with them to try to get them to work for our purpose. His current project is filing down bolts so the linear slides will be able to slide together, but he also promises to get help with using them from his robotics class at lathrop. The rest of the team worked on disassembling the old robot and sorting the parts. We are already beginning to have some shortages of a few parts, so we will need to recycle them from the old robot. Also, near the end of the meeting, most of the team worked on mounting controllers and the NXT. We hope to have the robot wired up and driving around by the end of next meeting. Since it can pick up and throw balls, it should be a lot of fun to drive!	This meeting went very well. Overall, we got a lot done. Our robot looks more professional and all around generally better. We are pretty confident in our system of flicking the balls so far, and everything is really starting to come together. We are close to having a driving robot.

We got a lot done in this meeting. Our robot looks quite good, and is really coming along. We should have a driving robot really soon. Hopefully we will get the field soon, which is supposed to be coming in some time next week. We really need to get a large wiffle ball for testing, it would help a lot.

Day 7: October 10, 2014 | 2:30 PM to 6 PM
By Nathan Sanches

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was a little different from our normal labs. While we were working today, we also had an open lab. Anyone who was interested in starting a team or who just wanted to come check it out could come and do so. As far as I know, we only had one person come and look around, but he seemed very interested, which is a very good thing. We also had a very productive day working on the robot today. The linear slides I was working on can now fit within the 18 inch size restrictions when compacted. I'm still trying to find someone at Lathrop who knows how they work, but I'm sure I'll find someone soon. We also got the robot driving. Our ball launching system is still a work in progress, but our new gear train should serve our purposes.	The open lab was fun, even though we only had one person come check it out. Hopefully he will be getting some friends together and start a second team at the lab, which would be great for competition practice and for idea exchanges. The robot also is working very well. Unfortunately, we can't yet be sure that our game strategy will work without all of the field elements, but we should have those soon as well.

Today we also got that large whiffle ball that Jasper wanted for testing purposes. We were not able to get it out of the robot at all, but that we think that's just a problem of having a flimsy material for our ball launcher. By finding a stiffer material we should be able to fix this problem very easily.

Day 8: October 14, 2014 | 2:30 PM to 5:00 PM
By: Jasper Holton

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
We didn't really get too much done today. We mostly just did a lot of prototyping of the ball paddle. We have tried many different materials including cardboard, lexan, reinforced cardboard, weighted cardboard, duct taped cardboard, and various others. However, we can't seem to be able to get a mechanism that works well with both the large and the small balls. We can only seem to get one that works with one or the other.	We really just need to find something that will work for both the small and the large balls. We haven't been able to get anything to work with both so far.

It seems like it's find a material that is sufficiently balanced. We need something that is firm enough to hit the balls and get them to go high, but also flexible enough to prevent jamming. This has proven to be something that is very difficult.

Day 9: October 16, 2014 | 1:00 PM to 5:00 PM
By: Alex Cater, ed. Jasper Holton

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
 We discovered that the design of the cardboard was creating blowing the bigger balls out away from the robot making it difficult to get the big balls through the tube. To fix this we put holes in the tube to create a vacuum to suck the balls in and make it easier to hit them out of the tube. Also we saw that the cardboard that was spinning to hit the balls was not very effective. We had to make the cardboard stiff enough to be able to hit the balls with	The spinning cardboard and band saw contraption still does not work well enough for us. This is because it still get jams a lot and it is really unreliable. We've been thinking about adding something a buffer. How ever we are going to stick with this design for at least a little bit longer before we accept failure and switch to a buffer.

enough amount of force to get them out of the tube. Also the cardboard could not be too stiff or else it would get jammed a lot. To fix this we cut up the blade for a band saw (which makes a really cool sound when dropped on the floor. So we dropped it on the floor a lot) put it inside the cardboard. The band saw was stiff enough to be able to hit the balls hard enough but still flexible to not get jammed that much.	

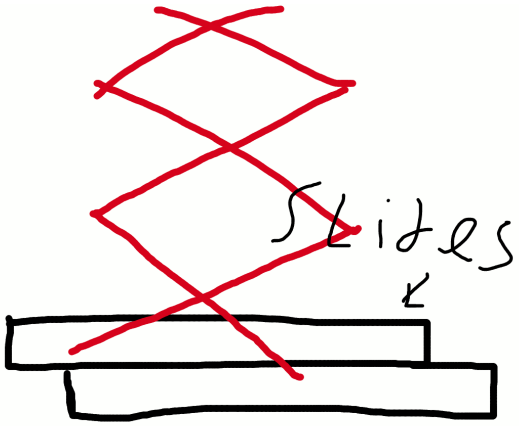
INJURY: Stephano burned his finger with a lighter, and Nathan heated up a screw driver to melt random stuff.

Day 10: October 17, 2014 | 1:00 PM to 5:00 PM
By: Jasper Holton

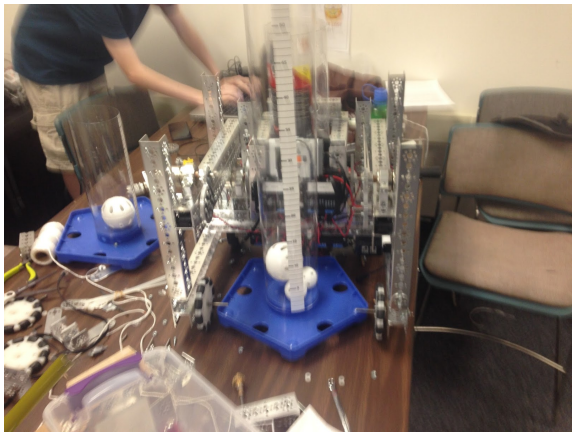
Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
 Most of the field came in today! Yay, finally! We spent the first hour or so putting together some of the game elements. We now have all of the rolling goals assembled. We also drove the robot around and tried to get it to pick up balls. This was kind of depressing for us, as it didn't really go as we would have hoped. We would have liked the paddle to be able to pick up the balls and throw them into the air, but it didn't really work like this. Instead, it got jammed frequently and was barely able to do	We got a lot done today. We are having some setbacks with our paddle mechanism, but otherwise we are still doing really well. Even if the paddle doesn't work, we will still be able to use a similar mechanism to take in balls and raise them up to the height of the tube. Testing should be a lot easier now that we have most of the field here and assembled. It's also nice to have plenty of wiffle balls.

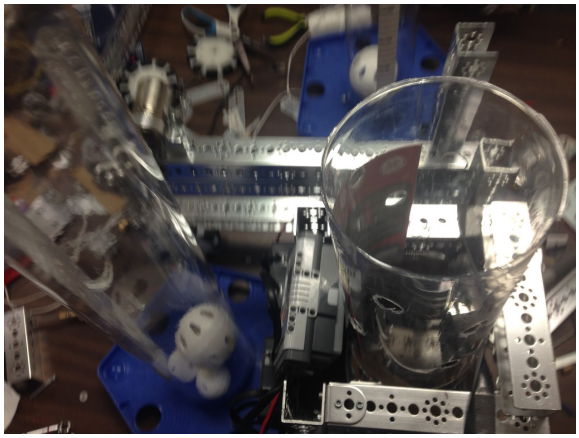
anything. This led us to some serious discussion and debate about whether this idea really would work. We didn't quite come to a decision, but we moved on from working on the paddle and went to working on a mechanism for raising the catcher, being as this could also be used to bring balls to the top of the tube in a slower but more accurate way. Speaking of this mechanism, we found that our linear slides will be pretty much useless for this purpose. We decided instead to build a double scissor lift (one lift on each side), which is working very well so far. We mounted two rack and pinion slides on either side of the robot and assembled two scissor lifts, which we have not yet mounted.	

Being as today was an early out, we had quite a lot of extra time to work on our robot and assembly of the field.
Here is a drawing of the rack and pinion/scissor lift system:



Here are some pictures of the robot near the end of the day:





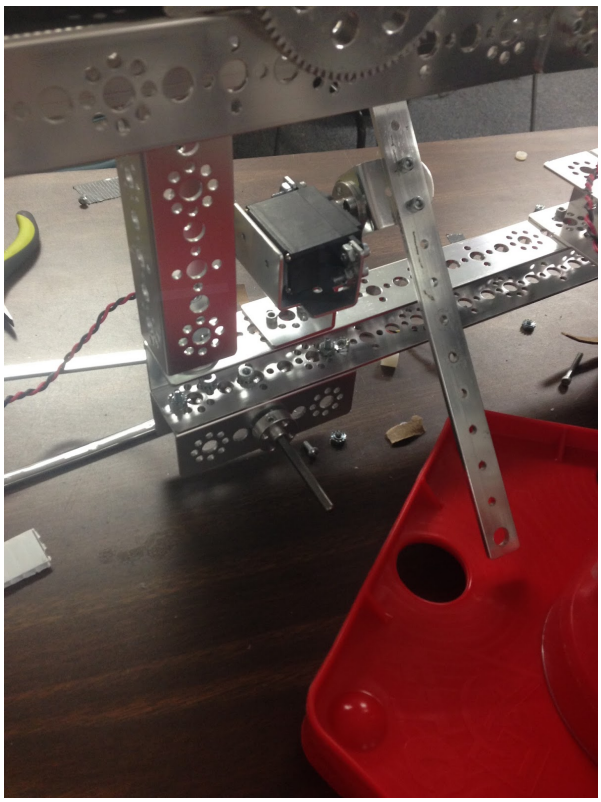
Note the rack and pinion systems on the side.

Day 11: October 21, 2014 | 2:30 PM to 5:00 PM
By: Nathan Sanches

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was an interesting day. We had a new team come in to the lab, this one from Effie Kokrine Charter School. We spent a little bit of time swapping ideas with them. We also completed our scissor lifts. There is now a scissor lift mounted on either side of our ball launching mechanism. They will eventually hold a catcher, to catch the balls flung from the launcher, or a tray will be mounted on the lifts, which will be used with the lifts to bring the balls into the goal tube. This choice will depend on which system we decide to use in our final design. We also attempted to put the field ramps together, only to find out that we needed a rivet gun, which we didn't have in the lab.	Today went very well. Ideas were swapped with the new team, an important mechanism was added to our robot, and we have new parts coming in, and hopefully a rivet gun too.

Day 12: October 22, 2014 | 2:30 PM to 5:00 PM
By: Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today, Jasper finished the scissor lift and started on the tray to lift the balls up to the goal.  Now Jasper is going to add a funnel of some type to direct the balls into the tube. Also he is going to add a servo to start and stop the balls from falling. Nathan spent a **very** long time putting the ramp together. He put too many rivets into one ramp and had to take them out. I started making something to hold the ramp in our robot.	We made a lot of progress today. We actually worked for the majority of the time. A lot new people came and was looking to start a new team. There are three new teams. One of them seems like they are going to keep coming and try.



The problem with the servos, is that they are flimsy. I may have to make something more stable to trap the goals.

Day 13: October 28, 2014 | 2:30 PM to 5:00 PM
By: Nathan Sanches

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was a moderately productive day. We worked on improving our scissor lift dump bucket. As we were working on this, we discovered that we had an issue with the scissor lift: the two sides of the scissor lift were raising at different speeds. One side was moving much slower than the other, which would be a problem since the purpose of that system is to collect balls and put them in the tubes, not back into our robot. Fortunately, I believe we have a solution, however temporary it might be.	Although the scissor lift was a bit of a setback, and was our main focus all day, we seem to have a solution, which should allow us to continue progressing our robot.

Day 13: October 28, 2014 | 2:30 PM to 5:00 PM
By: Nathan Sanches

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was a moderately productive day. We worked on improving our scissor lift dump bucket. As we were working on this, we discovered that we had an issue with the scissor lift: the two sides of the scissor lift were raising at different speeds. One side was moving much slower than the other, which would be a problem since the purpose of that system is to collect balls and put them in the tubes, not back into our robot. Fortunately, I believe we have a solution, however temporary it might be.	Although the scissor lift was a bit of a setback, and was our main focus all day, we seem to have a solution, which should allow us to continue progressing our robot.

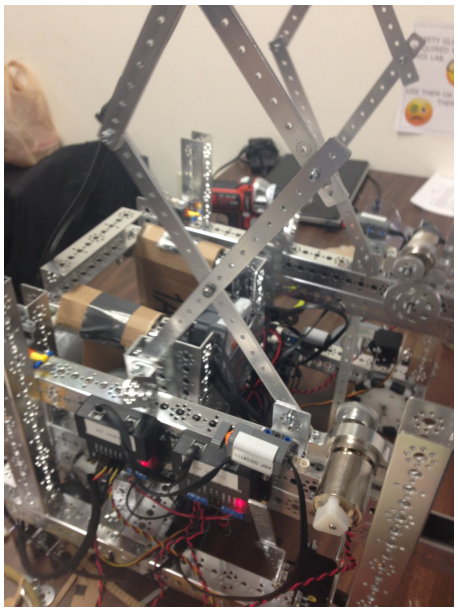
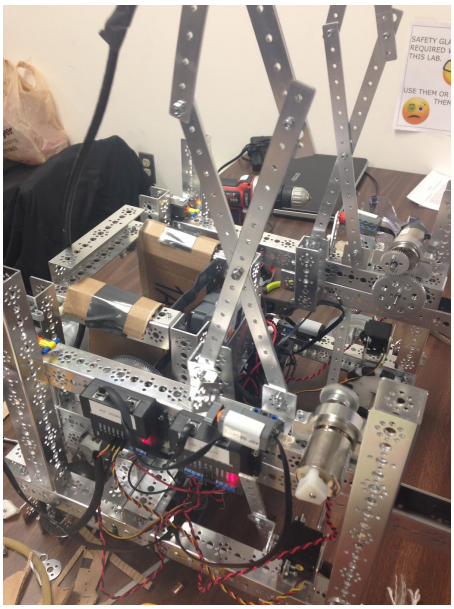
Day 14: October 28, 2014 | 2:30 PM to 5:00 PM
By: Nathan Sanches

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we worked on a few things. First of all, we got the center part of the goal finished. We played around with this and experimented with getting it to fall over. We determined that because of the large slot in the back of our robot, we should be able to fairly easily knock over the post. We did, however discover that the balls tend to get jammed. However, we suspect that the refs will fix this jamming problem at the end of each autonomous period. One of the other major things we did is we (mostly) finished programming our scissor lift. We added adjustments to the code and also added more touch sensors so you can simply press a button on the controller and the scissor lift will go to a predefined height. We accomplished this using a lot of touch sensors and two touch sensor multiplexers. By the time this is finished, we should have 8 or more touch sensors. Wow! This system isn't completely done. We have found that the powers of the motors on either side tend to vary, and also the level of friction tends to vary. We are probably going to need to either lubricate our robot, or find good modifiers for the motor power. I was also thinkinking that we could possibly replace the linear slide racks on the side that is slow (that is lift2 in the code, or the right side of the robot). Another thing to try would be replacing the motor with a newer one, as the motor that is slow is an older motor. Once we fix this probelm and add the rest of the sensors, the idea is that we will be able to press a button on the controller and the	This went pretty well. We are certainly getting a lot done, but we still have a few parts of our robot that haven't been tested and definitely need some work. These are: 1. The front paddle for the ball intake. This hasn't been tested nearly enough and will certainly need some work. 2. The two servos in the front that hold the goal tube inside the robot. This includes driving with the goal tube, keeping it from knocking over, etc. 3. Getting the goal tube positioned correctly inside the robot (so balls can be deposited into it) 4. Our samantha module 5. The sensor multiplexer (I haven't looked into programming this yet) 6. Our gyro sensor (I know how to program this, and already have code, but I haven't tested it yet. 7. The ball gate (this is the gate that keeps the balls from dumping out of the tray on top of the robot. This consists of a servo which can either let the balls out or keep them in. We haven't even tested this yet. There are probably a few other things here that I am still missing, but oh well. Otherwise, we are making great progress but we still have a ways to go.

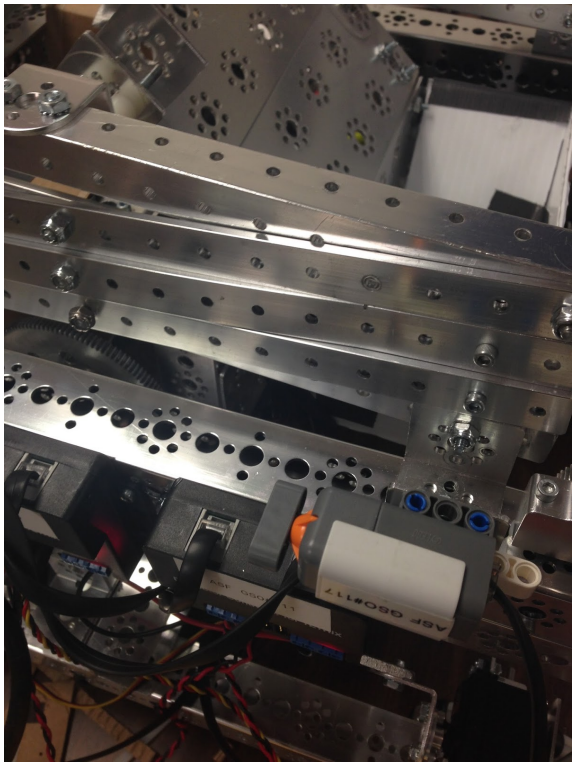
scissor lift will go to a certain height (according to what button is pressed). This way, we could automate the process of moving the lift.	

All of the sensor ports on our NXT are now taken up. Here is what the ports look like currently:
Port 1: Controllers
Port 2: Sensor multiplexer (general, with external battery)
Port 3: Touch sensor multiplexer for right side
Port 4: Touch sensor multiplexer for left side

Here are some pictures of our robot raising it's scissor lift:

Here is a picture of our touch sensor limit switch:



Day 15: November 4, 2014 | 2:30 PM to 5:00 PM
By: Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we mostly worked on our ball intake method. We came up with a new method for getting the balls into the robot. This method involves using tread with inserts on it to push the balls up to the desired height.   We started working on this, and have discovered one minor flaw in the design. This is that the balls can sometimes still get jammed behind the tread in this design. However, one solution we have to this looks promising. This is to slant each tread insert so the large balls move to an area where they will not jam. Cici finished creating the holder for the balls, and it works quite well. We are able to hold plenty of balls with it easily. 	Even though we didn't get a whole lot done today, overall this was a fairly productive day. We are well on o ur way to a functioning robot.

We are still waiting for a few more parts to come in, but then we should have our scissor lift perfected and finished.

Day 17: November 11, 2014 | 2:30 PM to 5:00 PM
By: Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
We a few new things today that we were missing. Our group specifically got parts for the scissor lift but we did not get the self locking bolts we needed. Now our scissor lift is even taller which makes it even more unstable because the bolts we're using now come unscrewed easily. I added another X to the scissor left and then Jasper finished it. While Jasper finished that I got started on getting the chains put together. I spent a **very excruciating long time** working on this. First we thought we had to take our one link from the first chain and then put one link in the second. but we actually only had to take away one from the first chain. So I had to undo what I just did and this frustrated me immensely because I had to make sure the peg was lined up exactly and everything else be exact which was very hard. However I did feel very accomplished when I finally got everything right.	There should be some machine to do the chain modifications for you. There is a simple mechanism to take the pegs out of the chain but this is not very useful because the hard part is putting them back in.

Day 18: November 8, 2014 | 2:30 PM to 5:00 PM
By: Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
I spent the day trying to make something to grab the goals so we can hold them to score and take them places. This is difficult because there is not a lot of space left so I have to make it very space efficient. I also used hot glue to lock the bolts for the scissor lift in place. We finally gave up waiting for the actual self locking bolts and just locked them ourselves. Louis was trying to make something to put on the chains to carry the balls up into the tray. He had problems with it because the zip ties was making the chain go off the tracks. Cici didn't believe that taking apart the chains and putting them back together was so difficult so she tried to put two chain pieces together. She gave up from frustration ten minutes later. CC also fixed the tray that holds the balls because it wasn't strong enough to hold five of the bigger balls. Everyone was fascinated with the newfangled label maker. So of course everything got a label. We will never forget that our robot is a robot and the wall is actually a wall.	Today went well. We did get various things done, but wasted most of our time labeling things.

Day 19: November 26, 2014 | 2:30 PM to 5:00 PM
By Jasper Holton

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
Today we mostly worked on perfecting our mechanism for picking up balls and finding a place to put the NXT on our robot. Before now, the NXT has really just been floating around near the robot, simply attached by a few wires, Now, we have found a permanent place for the NXT and secured it to the robot. Alex worked on using hot glue to improve the basket and prevent it from jamming while balls are in it. Additionally, we mostly got Samantha mounted towards the end of the day. Hopefully we will be able to test our robot with the FCS soon.	Today went quite well. We are well on our way to having a functioning robot that can score balls. Within the next few weeks, we should be ready for competition if nothing else goes wrong. The only major problem we are looking at with our robot currently is the ball intake mechanism. This mechanism is having problems currently and really needs to be fixed. However, we have several ideas for improvements and we are cautiously optimistic about the upcoming competition. We are also looking forward to our outreach opportunity.

Today we were offered an outreach opportunity for something that the University is doing. This will happen the Saturday after next, and we are told it will involve robots. We plan to bring our robot and teach young children about designing and building robots. It should be a lot of fun.

Day 20: December 4, 2014 | 2:30 PM to 5:00 PM
By Louis Bastille

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
This is my first journal entry as part of the team, due to being involved in swimming and not coming in to robotics until a few weeks ago. So far, I have been involved with the ball intake and getting the balls to be able to go in the lifting mechanism. Today, I reattached the flaps that push the balls up the chamber while waiting on Jasper so that	Fairly productive day, now that the flaps are realigned, there should no longer be any problems with ball intake. Once the synchronization is fully functional, there will likely be very few problems aside from tightening and cleaning up the robot.

he could work on the synchronization of the scissor lift (the zip ties on the intake flaps had to be realigned). Also, I added side blockers onto the chamber at the base so that balls wouldn't roll off the side. The Effie teams had a mini rubber band war, but that is beside the point. Once Jasper arrived, we were able to work on testing and tweaking the synch for the scissor lift, I had to remove samantha, since it got in the way. We also added spacers on the motors to allow greater control, and I had to hammer off a gear on one of the motors to be able to attach one of the modules.	

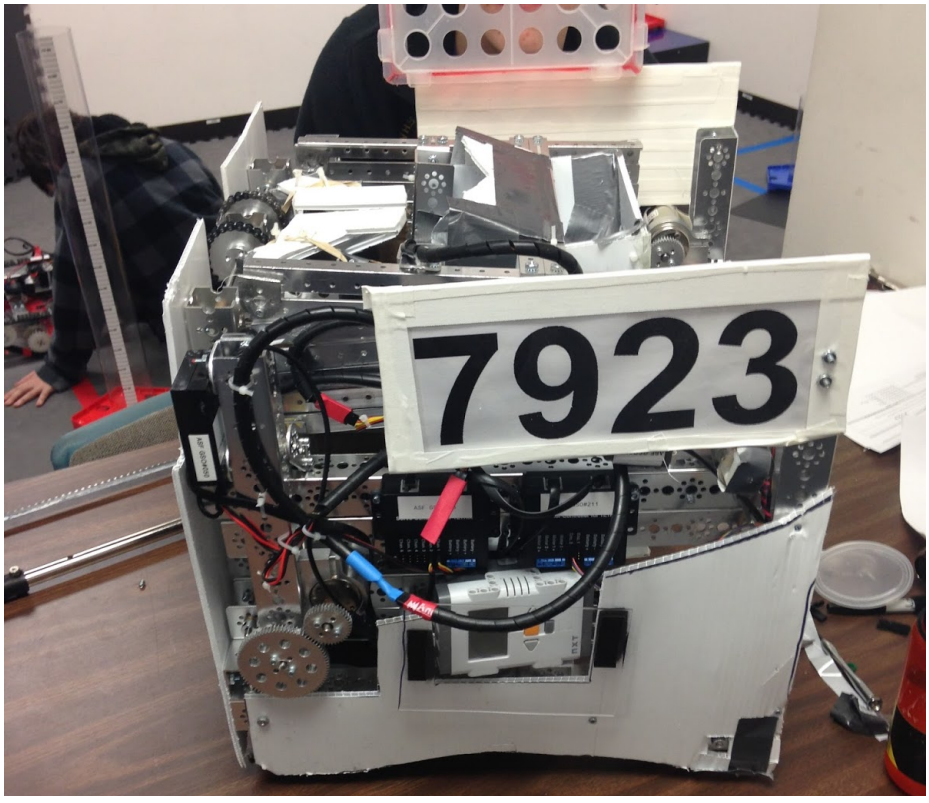
Day 21: December 5, 2014 | 2:30 PM to 5:00 PM
By Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today, I worked on making it so the balls don't get stuck in the scissor lift bucket. Even now, two of the big balls sometimes jam it. I am thinking making it a tube instead of a box to prevent this from happening. Louis had to replace some of the normal bolts with button heads because the normal ones were interfering with the scissor lift going up and down. Jasper worked on programming and trying to get the motors to raise and lower with the same speed. This is difficult because all motors are made unequal the motors have different friction and so operate at different speeds. We were planning on doing a scrimmage between the different teams today but no one was quite ready today, so we rescheduled it for next Friday.	I am excited that our robot is almost done and we're going to be able to finally test it out. However I am also, sad that we're also finishing it. We'll probably build another robot or make this one better but it won't be a competition.

Day 22: December 12, 2014 | 2:30 PM to 5:00 PM
By Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we put a lot of finishing touches on the robot. We added the team numbers by drilling holes in them and mounting them on the right and left side, near the top of the robot. We also made some small modifications and improvements to the ball basket. We then did some testing, and we were able to quickly score points with the robot. Unfortunately, our robot's batteries ran out while we were testing, and we didn't have any charged batteries. Since our robot was pretty much ready, we decided to take a break. We experimented with game elements, and also built boxes out of the floor tiles.	We are very satisfied with our progress so far. We have come a long ways, and we are about ready to compete.

Our first scrimmage is tomorrow! We are excited to see how our robot will work on the field.

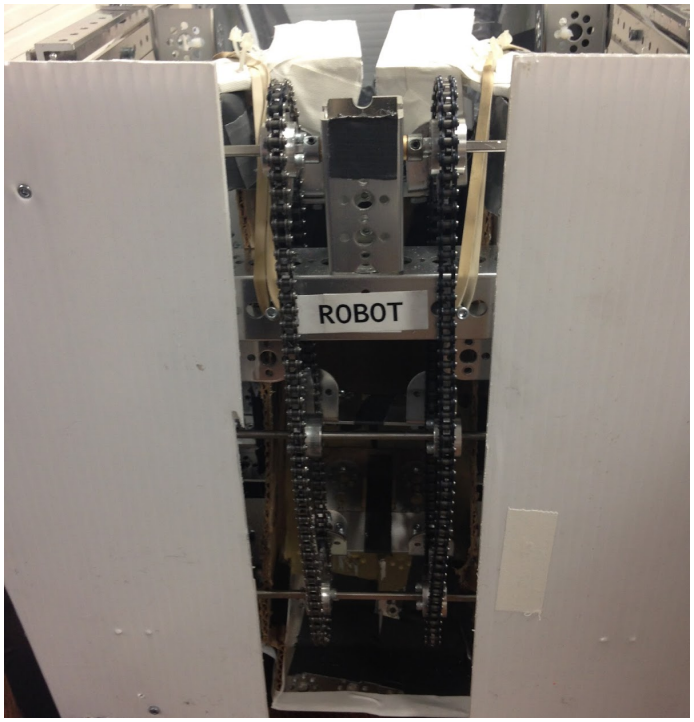


Day 23: December 11, 2014 | 2:30 PM to 5:00 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we mostly worked on fixing, testing, and improving our robot. We learned of several problems with our robot from the scrimmage we did several days ago. One of these problems was that all of our servos did not work. However, when using the robot today, we didn't have this problem which is somewhat strange. Anyways, it's a good thing. The other problem was that our robot had some gears clicking. We tightened these gears and this seemed to fix the problem. We also had some problems with balls getting jammed or stuck in various places around our robot. We worked a lot on improving this, and it seems to work much better now. We also did some modifications to the programming. We added a "direction switch" button which causes the robot to reverse the front/back directions so we can drive with the back being the front or the front being the back. This allows for more intuitive driving of the robot. Lastly, we improved and better tested our tube gripping apparatus. We added rubber to the end so it can carry the tubes around more efficiently, and we also changed the servo values so the tube grabbers don't get jammed on the ground. This seems to work well	We have done a lot of fixes and optimizations to our robot. We are very satisfied with our progress. The only major problem so far is that balls can get stuck in various places in our robot. Fortunately, this is a very simple problem to fix with some of the plastic/cardboard/lexan that we have available.

Day 24: January 6, 2015 | 2:30 PM to 5:00 PM
By Louis Bastille

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today it was only myself and Jasper who showed up. At first we installed new plastic plating on the front of the robot, so now almost the entire thing looks white. However, after setting up the miniature field in the lab we discovered that many sections of the shielding prevents us from going up the ramp. After trimming edges and testing some more, the robot eventually fell over, and the ball intake axle had slipped out of its mount. So, we spent the rest of our time working on fixing this problem, even when we had to move to the upstairs room, where we had some trouble getting a gear into the right place upon the axle, I believe it to be something chipped inside the gear mount, causing an abundance of friction on the axle.	The intake system is likely not the only area with errors, on Thursday I will try to find and fix any other problematic mechanisms. Other than this, there are no new devices that need to be made, and up until the competition will be bug fixes, working on autonomous, and driving practice.



Day 25: January 8, 2015 | 2:30 PM to 5:00 PM
By Jasper Holton

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
Today we mostly worked on small improvements and fixes on our robot. We started out doing some driving practice, and discovered some problems with the plating on our robot. This was fairly easy to fix, and we got to the point where our robot was able to go up the ramp easily and had a lot better maneuverability. We also discovered that one of the essential parts for our lift mechanism had broken. This ended up being a fairly simple fix, but took a while to do. Additionally, we discovered that small balls were not getting picked up properly. Adding some zip ties to the ball intake mechanism fixed this problem quite nicely. Jasper mainly worked on the autonomous program. He wrote code for the gyro sensor so the robot can make precise turns. For example, now the robot can turn exactly 90 degrees using the gyro sensor. This was a little bit difficult, because the sensor readings from the gyro sensor have to be processed before they become useable, helpful information.	Today went well. We would have liked to see a few more people come, but otherwise we got a lot done. Once we have our autonomous program working well, we will be ready to compete.

Day 26: January 9, 2015 | 2:30 PM to 5:00 PM
By Alex Cater

Task - What are we doing and discovering?	Reflections - What are our thoughts on this?
Today we worked on autonomous mode. To do this we had to put the robot on the ramp, run the program, have the robot do its thing, change something a little bit in the code and reset. Jasper was doing the programming, while me and Louis tried to find an effective	Will get done soon.

way of picking up the robot. We were unsuccessful because the protective siding we put on to protect from getting stuck on balls, gets in the way	

Day 27: January 12, 2015 | 2:30 PM to 5:00 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today Jasper was the only person in the lab, which was really quite tragic. He mostly worked on the protoboard recovery and status display. He managed to get it successfully mounted to the robot, and put a piece of lexan in front of it to prevent it from getting damaged. The battery monitors on the board are working well, and the protoboard is able to reset the NXT when it becomes unresponsive. There was a slight problem with one of the battery monitor lights, but this problem was easily fixed with a little bit of solder. Jasper also worked on autonomous today, which is coming along quite nicely. We are getting fairly reliable behavior from our robot.	Today went well. We are very glad that our protoboard is working well, as we believe that this will give us a slightly competitive edge by allowing us to regain control of our robot if it becomes unresponsive during a match.

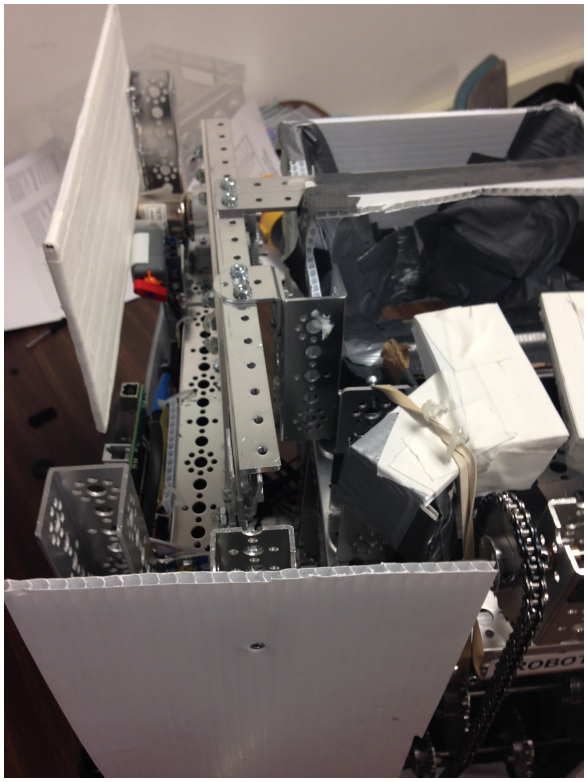
Day 28: January 13, 2015 | 2:30 PM to 5:00 PM
By Alex Cater

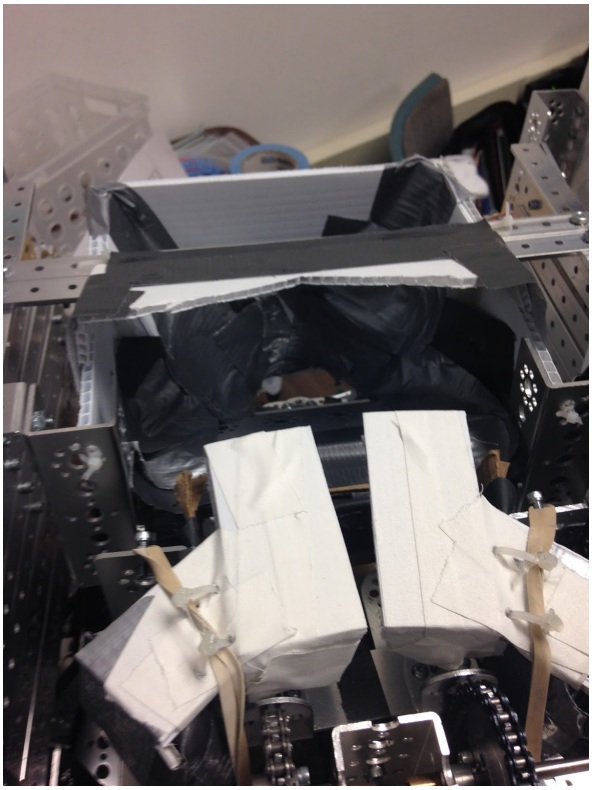
Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Jasper finished working on the first autonomous mode for our robot. It starts from the ramp, gets the shortest goal tube, moves it to the goal zone. He has started on a second autonomous mode which starts from the goal zone and the does the same thing. This way we have flexibility and we can work with our alliance partner more effectively.	It was a very productive day on Jasper's part. However I can not say the same about everyone else. There is not much we can do except help Jasper reset the robot and give him ideas.

Louis was bored so he made a cube out of the field material. Nathan got back from skiing and was also bored so sat in the red NASA box. It was a boring day because Jasper was doing all the coding while we moved the robot back to its start position. We also helped Jasper come up with ideas on what to do in autonomous mode. However overall we pretty much fooled around. Since we were bored and Nathan's box had wheels, we pushed Nathan out of the room and into the elevator and sent him up some floors. We thought he would come back right away but he decided to sit in his box in the elevator.	

Day 29: January 15, 2015 | 2:30 PM to 5:00 PM
By Louis Bastille

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today was very productive, during the first half-an-hour we were able to finish the secondary autonomous program, then tested the ramp autonomous to ensure it was still alright. Then we set to work on fixing ball intake problems. A previous problem was that the small balls would evade the zip ties, and the zip ties got so bent that they became ineffective. Jasper thought up a new method to fix this problem, including some of our shield material and rubber bands in which the balls would be shoved off from an angle into the holder (it required trimming for rare cases). However, the small balls and eventually the large balls began to be shot out the wrong way, so we applied some more material on the underside to slide them in the correct direction. Once this was complete, we set up the medium goal tube on a slant upon the top of the ramp, and with the support poles, we played GOLF!	Twas an extremely productive day, I am proud of how much was accomplished by our team. We are basically ready for the competition next week.

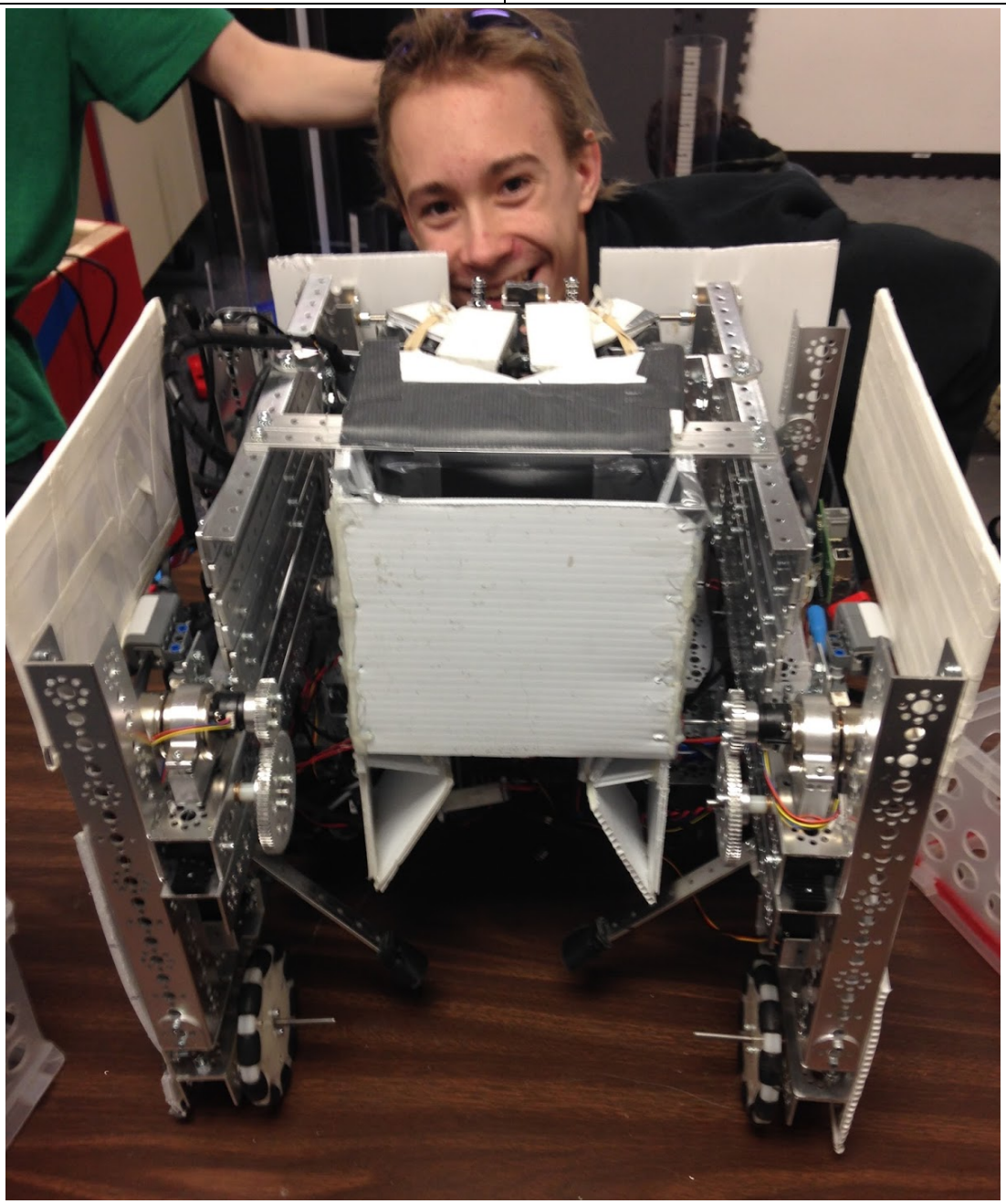




Day 30: January 21, 2015 | 2:30 PM to 5:00 PM
By Jasper Holton

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Today we made some major improvements to our robot, and also did some fixes. We made a major improvement to the ball intake mechanism, particularly the system which takes the balls from the chain link track and gets them into the scissor lift basket. By using slanted pieces of signboard put at a 45 degree angle to the back of the robot, we were able to create an effective way of getting the balls into the basket. Below you can see some pictures of this system. Additionally, we ran into a problem involving part of the scissor lift bending, which happened twice despite us trying to fix it. However, we are fairly sure that we have a solution to this problem so it shouldn't	We are ready for the competition, despite some minor hiccups along the way. All we need to do now is get our notebook together and we will be good. We also need to add more pictures to the notebook.

happen in the future. We believe the cause was due to both an under powered motor and the fact that the scissor lift was mounted improperly. We should have this fixed by the next meeting.	



Day 30: January 22, 2015 | 2:30 PM to 5:00 PM
By Alex Cater

Task - What are we doing and discovering?	**Reflections - What are our thoughts on this?**
Alex: signs Louis: Jasper: autonomous, flaps Tshirts	

Robot Code

Here you can see the code that we use for our robot. Some of the processes have comments indicating what they do.

Autonomous Start

```
#pragma config(Hubs,  S1, HTServo,  HTMotor,  HTMotor,  HTMotor)
#pragma config(Sensor, S1,     controllers,    sensorNone)
#pragma config(Sensor, S2,     HTSMUX,         sensorI2CCustom9V)
#pragma config(Sensor, S3,     touchmux1,      sensorHiTechnicTouchMux)
#pragma config(Sensor, S4,     HTSPB,          sensorI2CCustom9V)
#pragma config(Motor,  mtr_S1_C2_1,     flicker1,      tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C2_2,     lift1,         tmotorTetrix, PIDControl, encoder)
#pragma config(Motor,  mtr_S1_C3_1,     right,         tmotorTetrix, openLoop, reversed)
#pragma config(Motor,  mtr_S1_C3_2,     motorG,        tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C4_1,     left,          tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C4_2,     lift2,         tmotorTetrix, PIDControl, encoder)
#pragma config(Servo,  srvo_S1_C1_1,    holder1,              tServoStandard)
#pragma config(Servo,  srvo_S1_C1_2,    holder2,              tServoStandard)
#pragma config(Servo,  srvo_S1_C1_3,    gate,                 tServoStandard)
#pragma config(Servo,  srvo_S1_C1_4,    servo4,               tServoNone)
#pragma config(Servo,  srvo_S1_C1_5,    servo5,               tServoNone)
#pragma config(Servo,  srvo_S1_C1_6,    servo6,               tServoNone)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

// The protoboard delay
#define protoDelay 10

#include "JoystickDriver.c"
#include "drivers/hitechnic-superpro.h"
#include "drivers/hitechnic-sensormux.h"

#include "drivers/hitechnic-gyro.h"    //for gyro sensor
#define gyroSensor              msensor_S2_2

//Bit map definitions
#define mux_button1 0x01
#define mux_button2 0x02
#define mux_button3 0x04
#define mux_button4 0x08


#define BIT1O 1
#define BIT2O 2
#define BIT3O 4
#define BIT4O 8
#define BIT5O 16
#define BIT6O 32
#define BIT7O 64
#define BIT8O 128

#define BIT1A 254
#define BIT2A 253
#define BIT3A 251
#define BIT4A 247
#define BIT5A 239
#define BIT6A 223
#define BIT7A 191
#define BIT8A 127
```

```
#define HIGH true
#define LOW false
#define INPUT false
#define OUTPUT true

byte mode = 0;
byte output = 0;

void pinMode(byte pin, bool newMode)
{
	switch (pin)
	{
	case 0:
		if(newMode)
			mode = mode | BIT1O;
		else
			mode = mode & BIT1A;
		break;

	case 1:
		if(newMode)
			mode = mode | BIT2O;
		else
			mode = mode & BIT2A;
		break;

	case 2:
		if(newMode)
			mode = mode | BIT3O;
		else
			mode = mode & BIT3A;
		break;

	case 3:
		if(newMode)
			mode = mode | BIT4O;
		else
			mode = mode & BIT4A;
		break;

	case 4:
		if(newMode)
			mode = mode | BIT5O;
		else
			mode = mode & BIT5A;
		break;

	case 5:
		if(newMode)
			mode = mode | BIT6O;
		else
			mode = mode & BIT6A;
		break;

	case 6:
		if(newMode)
			mode = mode | BIT7O;
```

```
                else
                        mode = mode & BIT7A;
                break;

        case 7:
                if(newMode)
                        mode = mode | BIT8O;
                else
                        mode = mode & BIT8A;
                break;

        }
        // UNCOMMENT THIS NGPE[O;ALLIHJK.SDZHJGLASIDJKLGSA
        HTSPBsetupIO(HTSPB, mode);
        wait1Msec(protoDelay); // Wait for a bit
}

void digitalWrite(byte pin, bool newMode)
{
        switch (pin)
        {
        case 0:
                if(newMode)
                        output = output | BIT1O;
                else
                        output = output & BIT1A;
                break;

        case 1:
                if(newMode)
                        output = output | BIT2O;
                else
                        output = output & BIT2A;
                break;

        case 2:
                if(newMode)
                        output = output | BIT3O;
                else
                        output = output & BIT3A;
                break;

        case 3:
                if(newMode)
                        output = output | BIT4O;
                else
                        output = output & BIT4A;
                break;

        case 4:
                if(newMode)
                        output = output | BIT5O;
                else
                        output = output & BIT5A;
                break;

        case 5:
```

```
                if(newMode)
                        output = output | BIT6O;
                else
                        output = output & BIT6A;
                break;

        case 6:
                if(newMode)
                        output = output | BIT7O;
                else
                        output = output & BIT7A;
                break;

        case 7:
                if(newMode)
                        output = output | BIT8O;
                else
                        output = output & BIT8A;
                break;

        }
        // UNCOMMENT THIS NOW OR BA DTHINGS WITKSD.GKNAJSDNGKASJDIOP;GASNJ.DGHOERJLAKSD
        HTSPBwriteIO(HTSPB, output);
        wait1Msec(protoDelay); // Wait for a bit
}



task phoenix()
{
        // Setup pins
        pinMode(0, OUTPUT);
        pinMode(1, OUTPUT);
        // Switch the functionality off for a bit, just to let the microcontroller know that we are ready for
        // action, in the event that this happened directly after a restart.
        digitalWrite(0, LOW);
        wait1Msec(200);
        // Tell the microcontroller that we want the restart functionality to be on again
        digitalWrite(0, HIGH);
        bool last = false;
        while(nNxtButtonPressed != kExitButton)
        {
                if(last)
                {
                        digitalWrite(1, HIGH);
                }
                else
                {
                        digitalWrite(1, LOW);
                }
                last = !last;
                wait1Msec(100);
        }
        // Tell the microcontroller that we want the restart functionality to be off
        digitalWrite(0, LOW);
        digitalWrite(1, LOW);
        StopAllTasks();
```

```
}
```

```
// Code for shift registers ////////////////////////////////////////////////////////////////////////////
int SER_Pin = 2;   //pin 14 on the 75HC595
int RCLK_Pin = 3;  //pin 12 on the 75HC595
int SRCLK_Pin = 4; //pin 11 on the 75HC595

//How many of the shift registers - change this
#define number_of_74hc595s 1

//do not touch
#define numOfRegisterPins number_of_74hc595s * 8

bool registers[numOfRegisterPins];


//set all register pins to LOW
void clearRegisters(){
	for(int i = numOfRegisterPins - 1; i >=  0; i--){
		registers[i] = LOW;
	}
}


//Set and display registers
//Only call AFTER all values are set how you would like (slow otherwise)
void writeRegisters(){

	digitalWrite(RCLK_Pin, LOW);

	for(int i = numOfRegisterPins - 1; i >=  0; i--){
		digitalWrite(SRCLK_Pin, LOW);

		int val = registers[i];

		digitalWrite(SER_Pin, val);
		digitalWrite(SRCLK_Pin, HIGH);

	}
	digitalWrite(RCLK_Pin, HIGH);

}

//set an individual pin HIGH or LOW
void setRegisterPin(int index, int value){
	registers[index] = value;
}

// End shift register code ////////////////////////////////////////////////////////////////////////////

float currHeading = 0;
```

```
int offset = 0;

// Task to keep track of the current heading using the HT Gyro
task getHeading () {
	float delTime = 0;
	float prevHeading = 0;
	float curRate = 0;


	//PlaySound(soundBeepBeep);
	while (true) {
		time1[T1] = 0;
		curRate = HTGYROreadRot(gyroSensor);
		if (abs(curRate) > 3) {
			prevHeading = currHeading;
			currHeading = prevHeading + curRate * delTime;
			if (currHeading > 360) currHeading -= 360;
			else if (currHeading < 0) currHeading += 360;
		}
		wait1Msec(5);
		delTime = ((float)time1[T1]) / 1000;
		//delTime /= 1000;
	}
}

int power1 = 50;// Needs to be a bit more
int power2 = 60;

void lift(int position)
{
	// Down
	if(position == 0)
	{
		if(!(SensorValue(touchmux1) & mux_button1))
			motor(lift1) = power1;
		else nMotorEncoder[lift1] = 0;
		if(!(SensorValue(touchmux1) & mux_button3))
			motor(lift2) = -1 * power2;
		else nMotorEncoder[lift2] = 0;
		//place = 1;
	}
	// Up
	else if(position == 1)
	{
		if(!(SensorValue(touchmux1) & mux_button2))
			motor(lift1) = -1 * power1;
		///else motor(lift1) = 0;
		if(!(SensorValue(touchmux1) & mux_button4))
			motor(lift2) = power2;
		//else motor(lift2) = 0;
	}
	// neither
	else
	{
		motor(lift1) = 0;
		motor(lift2) = 0;
	}
```

```
}


void stopRobot()
{
	motor[left] = 0;
	motor[right] = 0;
	wait1Msec(1000);
}



// Experimental, precise 90 degree turn
void turn902(bool direction)
{
	currHeading = 0;
	wait1Msec(10);
	currHeading = 0;
	if(direction)
	{
		motor[left] = -40;
		motor[right] = 40;
		wait1Msec(80);
		while(currHeading < 70)
		{
			nxtDisplayBigStringAt(25, 25,"%d",currHeading);
			wait1Msec(10);
		}
		stopRobot();
	}
	else
	{
		motor[left] = 40;
		motor[right] = -40;
		wait1Msec(80);
		while(currHeading > 290)
		{
			nxtDisplayBigStringAt(25, 25,"%d",currHeading);
			wait1Msec(10);
		}
		stopRobot();
	}
}
// Do not use this!
```

```
// false = left
// true = right
// NUMBER 2 FOR AUTONOMOUS2
void turn902(bool direction)
{
	currHeading = 0;
	wait1Msec(10);
	currHeading = 0;
	if(direction)
	{
		motor[left] = -40;
		motor[right] = 40;
		wait1Msec(80);
		while(currHeading < 70)
		{
			nxtDisplayBigStringAt(25, 25,"%d",currHeading);
			wait1Msec(10);
		}
		stopRobot();
	}
	else
	{
		motor[left] = 40;
		motor[right] = -40;
		wait1Msec(80);
		while(currHeading > 290)
		{
			nxtDisplayBigStringAt(25, 25,"%d",currHeading);
			wait1Msec(10);
		}
		stopRobot();
	}
}
// NUMBER 2 FOR AUTONOMOUS22//////////////////
void turn9022(bool direction)
{
	currHeading = 0;
	wait1Msec(10);
	currHeading = 0;
	if(direction)
	{
		motor[left] = -40;
		motor[right] = 40;
		wait1Msec(80);
		while(currHeading < 70)////////////
		{
			nxtDisplayBigStringAt(25, 25,"%d",currHeading);
			wait1Msec(10);
		}
		stopRobot();
	}
	else
	{
		motor[left] = 40;
		motor[right] = -40;
		wait1Msec(80);//////////
		while(currHeading > 284)
```

```
                {
                        nxtDisplayBigStringAt(25, 25,"%d",currHeading);
                        wait1Msec(10);
                }
                stopRobot();
        }
}

// NUMBER 1 FOR AUTONOMOUS 1
void turn901(bool direction)
{
        currHeading = 0;
        wait1Msec(10);
        currHeading = 0;
        if(direction)
        {
                motor[left] = -40;
                motor[right] = 40;
                wait1Msec(80);
                while(currHeading < 71)
                {
                        nxtDisplayBigStringAt(25, 25,"%d",currHeading);
                        wait1Msec(10);
                }
                stopRobot();
        }
        else
        {
                motor[left] = 40;
                motor[right] = -40;
                wait1Msec(80);
                while(currHeading > 295)
                {
                        nxtDisplayBigStringAt(25, 25,"%d",currHeading);
                        wait1Msec(10);
                }
                stopRobot();
        }
}
void forward(long millis)
{
        motor[left] = -40;
        motor[right] = -40;
        wait1Msec(millis);
        motor[left] = 0;
        motor[right] = 0;
}
void forward(long millis, int offset)
{
        motor[left] = -40 + offset;
        motor[right] = -40 - offset;
        wait1Msec(millis);
        motor[left] = 0;
        motor[right] = 0;
}
void backward(long millis)
{
```

```
	motor[left] = 40;
	motor[right] = 40;
	wait1Msec(millis);
	motor[left] = 0;
	motor[right] = 0;
}

void backward(long millis, int offset)
{
	motor[left] = 40 + offset;
	motor[right] = 40 - offset;
	wait1Msec(millis);
	motor[left] = 0;
	motor[right] = 0;
}


void setHolder(bool holder)
{
	if(holder)
	{// down position
		servo(holder1) = 200;
		servo(holder2) = 255 - 233;

	}
	else
	{// up position position
// down position
		servo(holder1) = 225;
		servo(holder2) = 255 - 245;
	}
}

// Like sort of half down position ish
void setHolderCenter()
{

servo(holder1) = 215;
		servo(holder2) = 255 - 240;

}

void setGate(bool position)
{
	// For gate
	if(position)
	{
		servo(gate) = 140;
	}
	else
	{
		servo[gate] = 55;
	}
}


// Simple mapping function
```

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
	return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}




int getExtBatLevel()
{
	if(externalBatteryAvg > 9999)
		return map(externalBatteryAvg,10000,13000,0,100);
	return 0; // should be 0
}
int getIntBatLevel()
{
	return map(nAvgBatteryLevel,3000,9000,0,100); // min 3v
}


task displayBattery()
{
	/*for(int i = 0; i <numOfRegisterPins - 1; i++)
	{
	registers[i] = LOW;
	}
	writeRegisters();
	digitalWrite(5, LOW);
	digitalWrite(6, LOW);
	digitalWrite(7, LOW);*/

	while(true)
	{
		if(getExtBatLevel() > 20)
			registers[4] = HIGH;
		else
			registers[4] = LOW;
		if(getExtBatLevel() > 40)
			registers[3] = HIGH;
		else
			registers[3] = LOW;
		if(getExtBatLevel() > 60)
			registers[2] = HIGH;
		else
			registers[2] = LOW;
		if(getExtBatLevel() > 80)
			registers[0] = HIGH;
		else
			registers[0] = LOW;
		if(getExtBatLevel() > 90)
			registers[1] = HIGH;
		else
			registers[1] = LOW;
		writeRegisters();
```

```
                if(getIntBatLevel() > 20)
                        digitalWrite(7, HIGH);
                else
                        digitalWrite(7, LOW);
                if(getIntBatLevel() > 40)
                        digitalWrite(6, HIGH);
                else
                        digitalWrite(6, LOW);
                if(getIntBatLevel() > 60)
                        registers[6] = HIGH;
                else
                        registers[6] = LOW;
                if(getIntBatLevel() > 80)
                        digitalWrite(5, HIGH);
                else
                        digitalWrite(5, LOW);
                if(getIntBatLevel() > 90)
                        registers[5] = HIGH;
                else
                        registers[5] = LOW;
                writeRegisters();

                wait1Msec(4000);
        }
}

// Code for LEDs
bool readSwitch(int switchToRead)
{
        int succeses = 0;

        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);


        return succeses > 4;
}

void strobe(byte data)
{
        memset(HTSPB_I2CRequest, 0, sizeof(tByteArray));
        HTSPB_I2CRequest[0] = 3;                        // Message size
        HTSPB_I2CRequest[1] = HTSPB_I2C_ADDR;           // I2C Address
        HTSPB_I2CRequest[2] = HTSPB_OFFSET + HTSPB_STROBE; // Start digital output read address
        HTSPB_I2CRequest[3] = data;                     // The specified digital ports
        writeI2C(HTSPB, HTSPB_I2CRequest);
        wait1Msec(protoDelay); // Wait for a bit
```

```
}
byte strobeData = 0;
void setLed(int led, bool state)
{
	// TOP RED
	if(led == 1)
	{
		if(state)
			strobeData = strobeData | BIT4O;
		else
			strobeData = strobeData & BIT4A;
		strobe(strobeData);
	}
	// TOP GREEN
	if(led == 2)
	{
		if(state)
			strobeData = strobeData | BIT2O;
		else
			strobeData = strobeData & BIT2A;
		strobe(strobeData);
	}
	// TOP BLUE
	if(led == 3)
	{
		if(state)
			strobeData = strobeData | BIT3O;
		else
			strobeData = strobeData & BIT3A;
		strobe(strobeData);
	}

	// BOTTOM RED
	if(led == 4)
	{
	HTSPBwriteAnalog(HTSPB, HTSPB_DACO1,DAC_MODE_DCOUT,1, state ? 1023:0);
	}
	// BOTTOM GREEN
	if(led == 5)
	{
	HTSPBwriteAnalog(HTSPB, HTSPB_DACO0,DAC_MODE_DCOUT,1, state ? 1023:0);
	}

	// BOTTOM BLUE
	if(led == 6)
	{
		if(state)
			strobeData = strobeData | BIT1O;
		else
			strobeData = strobeData & BIT1A;
		strobe(strobeData);
	}


}
```

```
void setup()
{
	// Start of program
	writeDebugStream("Starting...");



	// Preform gyro setup
	HTGYROstartCal(gyroSensor);
	startTask(getHeading);
	// Gyro setup done


	// Preform protoboard setup
	StartTask(phoenix);
	nNxtExitClicks = 5;
	pinMode(SER_Pin, OUTPUT);
	pinMode(RCLK_Pin, OUTPUT);
	pinMode(SRCLK_Pin, OUTPUT);
	pinMode(5, OUTPUT);
	pinMode(6, OUTPUT);
	pinMode(7, OUTPUT);
	//clearRegisters();
	//writeRegisters();
	startTask(displayBattery);
	// Protoboard setup done

	/*while(true)
	{
		nxtDisplayBigStringAt(25, 25,"%d",HTSPBreadADC(HTSPB, 0, 10));
		wait1Msec(100);
	}*/



}




void autonomous1()
{

	/*for(int i = 1; i <= 6; i++)
	{
	setLed(i, HIGH);
	wait1Msec(1000);
	setLed(i, LOW);

	}*/

	// NOT THIS ONE THE OTHER ONE
	setHolder(false);
	forward(2000);
	stopRobot();
```

```
turn901(false);


forward(860);

stopRobot();


turn901(true);

stopRobot();

forward(2300);




stopRobot();

setHolder(true);

backward(5700, -2);

stopRobot();

turn9022(false);
stopRobot();


forward(600);

stopRobot();

setHolder(false);

lift(1);
wait1Msec(700);
lift(2); // stop lift

stopRobot();

setGate(false);

setHolderCenter();

//while(true){}
wait1Msec(1000);

forward(600);

stopRobot();
wait1Msec(500);

setGate(true);
```

```
		wait1Msec(1500);
		setGate(false);
		wait1Msec(1000);


		backward(300,0);
		stopRobot();
		setHolder(false);
		backward(500,0);
		stopRobot();

		lift(0);
		wait1Msec(420);
		lift(2); // stop lift




		while(true)
		{
			//nxtDisplayBigStringAt(25, 25,"%d",heading);



			wait1Msec(100);
			eraseDisplay();
		}
}



void autonomous2()
{

		/*for(int i = 1; i <= 6; i++)
		{
		setLed(i, HIGH);
		wait1Msec(1000);
		setLed(i, LOW);

		}*/

setGate(false);
		setHolder(false);
		forward(1000);
		stopRobot();

		turn902(false);

// go forward to tube
		forward(4000,-2);
		forward(1200,-4);

		stopRobot();

		setHolder(true);
```

```
backward(5700, -1);

stopRobot();

turn9022(false);
stopRobot();



forward(600);

stopRobot();

setHolder(false);

lift(1);
wait1Msec(700);
lift(2); // stop lift

stopRobot();

//setGate(false);

setHolderCenter();

//while(true){}
wait1Msec(1000);

forward(600);

stopRobot();
wait1Msec(500);

setGate(true);
wait1Msec(1500);
setGate(false);
wait1Msec(1000);


backward(300,0);
stopRobot();
setHolder(false);
backward(500,0);
stopRobot();

lift(0);
wait1Msec(420);
lift(2); // stop lift




while(true)
{
	//nxtDisplayBigStringAt(25, 25,"%d",heading);
```

```
                wait1Msec(100);
                eraseDisplay();
        }
}


task main()
{
        setup();

        wait1Msec(2000);

        if(readSwitch(0))
        {
                setLed(5, true);
                wait1Msec(2000);
                setLed(5, false);
                wait1Msec(100);
                //setLed(0, LOW);
                autonomous2();
        }
        else
        {
                setLed(4, true);
                wait1Msec(2000);
                setLed(4, false);
                autonomous1();
        }



}
```

Autonomous End

Teleop Start

```
#pragma config(Hubs,  S1, HTServo,  HTMotor,  HTMotor,  HTMotor)
#pragma config(Sensor, S1,     controllers,    sensorNone)
#pragma config(Sensor, S2,     HTSMUX,         sensorI2CCustom9V)
#pragma config(Sensor, S3,     touchmux1,      sensorHiTechnicTouchMux)
#pragma config(Sensor, S4,     HTSPB,          sensorI2CCustom9V)
#pragma config(Motor,  mtr_S1_C2_1,     flicker1,      tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C2_2,     lift1,         tmotorTetrix, PIDControl, encoder)
#pragma config(Motor,  mtr_S1_C3_1,     right,         tmotorTetrix, openLoop, reversed)
#pragma config(Motor,  mtr_S1_C3_2,     motorG,        tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C4_1,     left,          tmotorTetrix, openLoop)
#pragma config(Motor,  mtr_S1_C4_2,     lift2,         tmotorTetrix, PIDControl, encoder)
#pragma config(Servo,  srvo_S1_C1_1,    holder1,              tServoStandard)
#pragma config(Servo,  srvo_S1_C1_2,    holder2,              tServoStandard)
#pragma config(Servo,  srvo_S1_C1_3,    gate,                 tServoStandard)
#pragma config(Servo,  srvo_S1_C1_4,    servo4,               tServoNone)
#pragma config(Servo,  srvo_S1_C1_5,    servo5,               tServoNone)
#pragma config(Servo,  srvo_S1_C1_6,    servo6,               tServoNone)
```

```
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//


// The protoboard delay
#define protoDelay 10

#include "drivers/hitechnic-superpro.h"
#include "drivers/hitechnic-sensormux.h"

// THIS!!!! https://github.com/botbench/rdpartyrobotcdr/blob/master/hitechnic-gyro-SMUX-test1.c
// Assuming the Sensor MUX is connected to NXT sensor port 4 (S4)
// Assuming the following sensors are connected to the Sensor MUX ports:
// Port 1: Gyro
// Port 2: Sonar
// Port 3: IRSeeker
// Port 4: Touch
#include "drivers/hitechnic-gyro.h"    //for gyro sensor
#include "JoystickDriver.c"
#define gyroSensor              msensor_S2_2


//Bit map definitions
#define mux_button1 0x01
#define mux_button2 0x02
#define mux_button3 0x04
#define mux_button4 0x08


#define BIT1O 1
#define BIT2O 2
#define BIT3O 4
#define BIT4O 8
#define BIT5O 16
#define BIT6O 32
#define BIT7O 64
#define BIT8O 128

#define BIT1A 254
#define BIT2A 253
#define BIT3A 251
#define BIT4A 247
#define BIT5A 239
#define BIT6A 223
#define BIT7A 191
#define BIT8A 127

#define HIGH true
#define LOW false
#define INPUT false
#define OUTPUT true

byte mode = 0;
byte output = 0;

void pinMode(byte pin, bool newMode)
{
        switch (pin)
```

```
{
case 0:
        if(newMode)
                mode = mode | BIT1O;
        else
                mode = mode & BIT1A;
        break;

case 1:
        if(newMode)
                mode = mode | BIT2O;
        else
                mode = mode & BIT2A;
        break;

case 2:
        if(newMode)
                mode = mode | BIT3O;
        else
                mode = mode & BIT3A;
        break;

case 3:
        if(newMode)
                mode = mode | BIT4O;
        else
                mode = mode & BIT4A;
        break;

case 4:
        if(newMode)
                mode = mode | BIT5O;
        else
                mode = mode & BIT5A;
        break;

case 5:
        if(newMode)
                mode = mode | BIT6O;
        else
                mode = mode & BIT6A;
        break;

case 6:
        if(newMode)
                mode = mode | BIT7O;
        else
                mode = mode & BIT7A;
        break;

case 7:
        if(newMode)
                mode = mode | BIT8O;
        else
                mode = mode & BIT8A;
        break;
```

```
		}
		// UNCOMMENT THIS NGPE[O;ALLIHJK.SDZHJGLASIDJKLGSA
		HTSPBsetupIO(HTSPB, mode);
		wait1Msec(protoDelay); // Wait for a bit
}

void digitalWrite(byte pin, bool newMode)
{
		switch (pin)
		{
		case 0:
			if(newMode)
				output = output | BIT1O;
			else
				output = output & BIT1A;
			break;

		case 1:
			if(newMode)
				output = output | BIT2O;
			else
				output = output & BIT2A;
			break;

		case 2:
			if(newMode)
				output = output | BIT3O;
			else
				output = output & BIT3A;
			break;

		case 3:
			if(newMode)
				output = output | BIT4O;
			else
				output = output & BIT4A;
			break;

		case 4:
			if(newMode)
				output = output | BIT5O;
			else
				output = output & BIT5A;
			break;

		case 5:
			if(newMode)
				output = output | BIT6O;
			else
				output = output & BIT6A;
			break;

		case 6:
			if(newMode)
				output = output | BIT7O;
			else
				output = output & BIT7A;
```

```
                break;

        case 7:
                if(newMode)
                        output = output | BIT8O;
                else
                        output = output & BIT8A;
                break;

        }
        // UNCOMMENT THIS NOW OR BA DTHINGS WITKSD.GKNAJSDNGKASJDIOP;GASNJ.DGHOERJLAKSD
        HTSPBwriteIO(HTSPB, output);
        wait1Msec(protoDelay);// Wait for a bit
}



task phoenix()
{
        // Setup pins
        pinMode(0, OUTPUT);
        pinMode(1, OUTPUT);
        // Switch the functionality off for a bit, just to let the microcontroller know that we are ready for
        // action, in the event that this happened directly after a restart.
        digitalWrite(0, LOW);
        wait1Msec(200);
        // Tell the microcontroller that we want the restart functionality to be on again
        digitalWrite(0, HIGH);
        bool last = false;
        while(nNxtButtonPressed != kExitButton)
        {
                if(last)
                {
                        digitalWrite(1, HIGH);
                }
                else
                {
                        digitalWrite(1, LOW);
                }
                last = !last;
                wait1Msec(100);
        }
        // Tell the microcontroller that we want the restart functionality to be off
        digitalWrite(0, LOW);
        digitalWrite(1, LOW);
        StopAllTasks();
}




// Simple mapping function
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
        return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

```
int getExtBatLevel()
{
	if(externalBatteryAvg > 9999)
		return map(externalBatteryAvg,10000,13000,0,100);
	return 0;
}
int getIntBatLevel()
{
	return map(nAvgBatteryLevel,3000,9000,0,100); // min 3v
}


// Code for shift registers ////////////////////////////////////////////////////////////////////////////
int SER_Pin = 2;   //pin 14 on the 75HC595
int RCLK_Pin = 3;  //pin 12 on the 75HC595
int SRCLK_Pin = 4; //pin 11 on the 75HC595

//How many of the shift registers - change this
#define number_of_74hc595s 1

//do not touch
#define numOfRegisterPins number_of_74hc595s * 8

bool registers[numOfRegisterPins];


//set all register pins to LOW
void clearRegisters(){
	for(int i = numOfRegisterPins - 1; i >=  0; i--){
		registers[i] = LOW;
	}
}


//Set and display registers
//Only call AFTER all values are set how you would like (slow otherwise)
void writeRegisters(){

	digitalWrite(RCLK_Pin, LOW);

	for(int i = numOfRegisterPins - 1; i >=  0; i--){
		digitalWrite(SRCLK_Pin, LOW);

		int val = registers[i];

		digitalWrite(SER_Pin, val);
		digitalWrite(SRCLK_Pin, HIGH);

	}
	digitalWrite(RCLK_Pin, HIGH);

}
```

```
//set an individual pin HIGH or LOW
void setRegisterPin(int index, int value){
	registers[index] = value;
}

// End shift register code ////////////////////////////////////////////////////////////////////////////////

float currHeading = 0;
int offset = 0;

// Task to keep track of the current heading using the HT Gyro
task getHeading () {
	float delTime = 0;
	float prevHeading = 0;
	float curRate = 0;


	//PlaySound(soundBeepBeep);
	while (true) {
		time1[T1] = 0;
		curRate = HTGYROreadRot(gyroSensor);
		if (abs(curRate) > 3) {
			prevHeading = currHeading;
			currHeading = prevHeading + curRate * delTime;
			if (currHeading > 360) currHeading -= 360;
			else if (currHeading < 0) currHeading += 360;
		}
		wait1Msec(5);
		delTime = ((float)time1[T1]) / 1000;
		//delTime /= 1000;
	}
}

int power1 = 50;// Needs to be a bit more
int power2 = 60;

void lift(int position)
{
	// Down
	if(position == 0)
	{
		if(!(SensorValue(touchmux1) & mux_button1))
			motor(lift1) = power1;
		else nMotorEncoder[lift1] = 0;
		if(!(SensorValue(touchmux1) & mux_button3))
			motor(lift2) = -1 * power2;
		else nMotorEncoder[lift2] = 0;
		//place = 1;
	}
	// Up
	else if(position == 1)
	{
		if(!(SensorValue(touchmux1) & mux_button2))
			motor(lift1) = -1 * power1;
		///else motor(lift1) = 0;
		if(!(SensorValue(touchmux1) & mux_button4))
```

```
                        motor(lift2) = power2;
                //else motor(lift2) = 0;
        }
        // neither
        else
        {
                motor(lift1) = 0;
                motor(lift2) = 0;
        }
}

task displayBattery()
{
        /*for(int i = 0; i <numOfRegisterPins - 1; i++)
        {
        registers[i] = LOW;
        }
        writeRegisters();
        digitalWrite(5, LOW);
        digitalWrite(6, LOW);
        digitalWrite(7, LOW);*/

        while(true)
        {
                if(getExtBatLevel() > 20)
                        registers[4] = HIGH;
                else
                        registers[4] = LOW;
                if(getExtBatLevel() > 40)
                        registers[3] = HIGH;
                else
                        registers[3] = LOW;
                if(getExtBatLevel() > 60)
                        registers[2] = HIGH;
                else
                        registers[2] = LOW;
                if(getExtBatLevel() > 80)
                        registers[0] = HIGH;
                else
                        registers[0] = LOW;
                if(getExtBatLevel() > 90)
                        registers[1] = HIGH;
                else
                        registers[1] = LOW;
                writeRegisters();


                if(getIntBatLevel() > 20)
                        digitalWrite(7, HIGH);
                else
                        digitalWrite(7, LOW);
                if(getIntBatLevel() > 40)
                        digitalWrite(6, HIGH);
                else
                        digitalWrite(6, LOW);
                if(getIntBatLevel() > 60)
                        registers[6] = HIGH;
```

```
                else
                        registers[6] = LOW;
                if(getIntBatLevel() > 80)
                        digitalWrite(5, HIGH);
                else
                        digitalWrite(5, LOW);
                if(getIntBatLevel() > 90)
                        registers[5] = HIGH;
                else
                        registers[5] = LOW;
                writeRegisters();

                wait1Msec(10000);
        }
}

// Code for LEDs
bool readSwitch(int switchToRead)
{
        int succeses = 0;

        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);
        if(HTSPBreadADC(HTSPB, switchToRead, 10) > 1000)     succeses++;
        wait1Msec(100);


        return succeses > 4;
}

void strobe(byte data)
{
        memset(HTSPB_I2CRequest, 0, sizeof(tByteArray));
        HTSPB_I2CRequest[0] = 3;                    // Message size
        HTSPB_I2CRequest[1] = HTSPB_I2C_ADDR;           // I2C Address
        HTSPB_I2CRequest[2] = HTSPB_OFFSET + HTSPB_STROBE; // Start digital output read address
        HTSPB_I2CRequest[3] = data;                 // The specified digital ports
        writeI2C(HTSPB, HTSPB_I2CRequest);
        wait1Msec(protoDelay); // Wait for a bit
}
byte strobeData = 0;
void setLed(int led, bool state)
{
        // TOP RED
        if(led == 1)
        {
                if(state)
                        strobeData = strobeData | BIT4O;
                else
```

```
                        strobeData = strobeData & BIT4A;
                strobe(strobeData);
        }
        // TOP GREEN
        if(led == 2)
        {
                if(state)
                        strobeData = strobeData | BIT2O;
                else
                        strobeData = strobeData & BIT2A;
                strobe(strobeData);
        }
        // TOP BLUE
        if(led == 3)
        {
                if(state)
                        strobeData = strobeData | BIT3O;
                else
                        strobeData = strobeData & BIT3A;
                strobe(strobeData);
        }

        // BOTTOM RED
        if(led == 4)
        {
        HTSPBwriteAnalog(HTSPB, HTSPB_DACO1,DAC_MODE_DCOUT,1, state ? 1023:0);
        }
        // BOTTOM GREEN
        if(led == 5)
        {
        HTSPBwriteAnalog(HTSPB, HTSPB_DACO0,DAC_MODE_DCOUT,1, state ? 1023:0);
        }

        // BOTTOM BLUE
        if(led == 6)
        {
                if(state)
                        strobeData = strobeData | BIT1O;
                else
                        strobeData = strobeData & BIT1A;
                strobe(strobeData);
        }


}


void setup()
{
        setLed(2, false);
        HTGYROstartCal(gyroSensor);
        writeDebugStream("Starting...");
        // Calculate offset
        startTask(getHeading);


        // Preform protoboard setup
```

```
	StartTask(phoenix);
	nNxtExitClicks = 5;
	pinMode(SER_Pin, OUTPUT);
	pinMode(RCLK_Pin, OUTPUT);
	pinMode(SRCLK_Pin, OUTPUT);
	pinMode(5, OUTPUT);
	pinMode(6, OUTPUT);
	pinMode(7, OUTPUT);
	//clearRegisters();
	//writeRegisters();
	startTask(displayBattery);
	// Protoboard setup done



	offset = currHeading;

	setLed(2, true);
	// Wait for the start of the match
	waitForStart();
}


int place = 1;
// 0 = fully down
// 1 = fully up



task main()
{
	int mod = 1;
	bool holder = false;
	setup();

	nMotorEncoder[lift1] = 0;
	nMotorEncoder[lift2] = 0;
 //waitForStart();

	while(true)
	{
		getJoystickSettings(joystick);

		if(joy1Btn(1))
		{
			mod = mod * -1;
			wait1Msec(600);
		}

		if(mod == 1)
		{
			motor[left] = abs(joystick.joy1_y2) < 7 ? 0 : joystick.joy1_y2 * mod;
			motor[right] = abs(joystick.joy1_y1) < 7 ? 0 : joystick.joy1_y1 * mod;
		}
		else
```

```
		{
			motor[right] = abs(joystick.joy1_y2) < 7 ? 0 : joystick.joy1_y2 * mod;
			motor[left] = abs(joystick.joy1_y1) < 7 ? 0 : joystick.joy1_y1 * mod;
		}

		int power = 40;
		// For paddle
		if(joy1Btn(4))
		{

			motor(flicker1) = power;
			//motor(flicker2) = -1 * power;
		}
		else
		{
			motor(flicker1) = 0;
			//motor(flicker2) = 0;
		}

		// For gate
		if(joy1Btn(3))
		{

			servo(gate) = 140;
		}
		else
		{
			servo[gate] = 55;
		}

		if(joy1Btn(2))
		{
			holder = !holder;
			while(joy1Btn(2)){}
		}

		if(holder)
		{// up position
		servo(holder1) = 200;
		servo(holder2) = 255 - 233;

	}
	else
	{// down position

			servo(holder1) = 225;
			servo(holder2) = 255 - 245;
	}



			int power1 = 50;// Needs to be a bit more
			int power2 = 60;
			//writeDebugStream("%d %d \n", nMotorEncoder[lift1], nMotorEncoder[lift2];
			/*wait1Msec(100);
			if(nMotorEncoder[lift1]  < nMotorEncoder[lift2])
			{
```

```
                power1 = 50;
                power2 = 100;
        }
        else
        {
                power1 = 100;
                power2 = 50;
        }*/


        if(joystick.joy1_TopHat == 2)
        {
        motor(lift1) = 50;
        }
        else if(joystick.joy1_TopHat == 6)
        {
        motor(lift2) = -50;
        }
        else
        {
        // Down
        if(joy1Btn(5) == true)
        {
                if(!(SensorValue(touchmux1) & mux_button1))
                        motor(lift1) = power1;
                else nMotorEncoder[lift1] = 0;
                if(!(SensorValue(touchmux1) & mux_button3))
                        motor(lift2) = -1 * power2;
                else nMotorEncoder[lift2] = 0;
                //place = 1;
        }
        else if(joy1Btn(6) == true)
        {
                if(!(SensorValue(touchmux1) & mux_button2))
                        motor(lift1) = -1 * power1;
                ///else motor(lift1) = 0;
                if(!(SensorValue(touchmux1) & mux_button4))
                        motor(lift2) = power2;
                //else motor(lift2) = 0;
        }
        else
        {
                motor(lift1) = 0;
                motor(lift2) = 0;
        }
}

        // Place is desired position
        // Lift2 is slower
        // For lift up
        /*if(place == 1 && !(SensorValue(touchmux1) & mux_button1))
                motor(lift1) = 50;
        else if(place == 0 && )
                motor(lift1) = -50;
        else
                motor(lift1) = 0;
```

```
                if(place == 1 && !(SensorValue(touchmux1) & mux_button3))
                        motor(lift2) = -60;
                else if(place == 0 && )
                        motor(lift2) = 60;
                else
                        motor(lift2) = 0;*/


        }
}
```

Teleop End

Robot Control

This section explains how we use sensors to improve our robot's ability to function, both in Teleop and Autonomous.

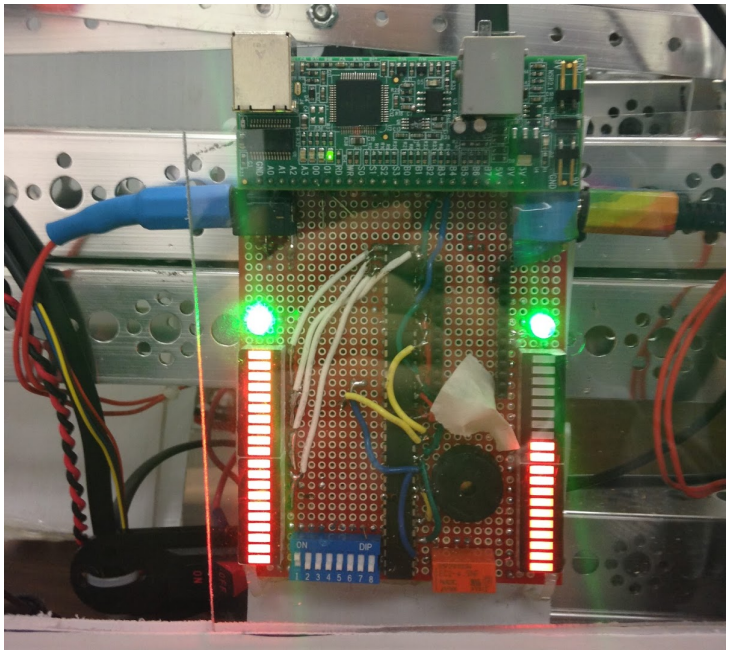
Phoenix

First of all, the phoenix system helps us identify problems with our robot (such as low battery) and helps us recover from static shock episodes by restarting the NXT.

Phoenix was originally conceived at our second meeting, right after the kickoff. The idea was that we could use the hightechnic prototype board to give us a failsafe system for our robot. After doing some planning for this, we purchased the components and began assembling an apparatus for the prototype board. The final product looks like this:

The prototype board works by connecting to the battery inside the NXT and providing a bridge between two of these terminals that is closed by a relay located near the prototype board. Because of this relay, we are able to power cycle the NXT and recover it from static shock episodes. Additionally, the prototype board has several switches on it which allow us to control how the robot behaves in autonomous before the match. It also has several LEDs which help us keep track of what the robot is doing.

Here is a picture:



The Gyro Sensor

We use a gyro sensor to do precise turns during the autonomous period. Because the gyro sensor is able to detect how fast the robot is turning, we can have it precisely turn a certain number of degrees within a very small margin of error.

Motor Encoders

We use motor encoders on our scissor lift system. The encoders sync the left and right side of the lift, meaning that the lift ideally rises with both sides being the same. This means that we save a lot of time, as we don't need to constantly correct the lift system manually.

Limit Switches

We also use limit switches for our scissor lift system. These limit switches prevent the robot from going past it's boundaries and burning out motors. Even if the driver is pressing a button on the controller, for example telling the lift to go up, the lift will not continue to rise if it is not already at it's maximum height. Thus, we are able to prevent ourselves from burning out motors.

Autonomous:

- Use battery monitors and status LEDs to check if robot is ready
- Use switch to select robot location
- Start on the ramp, or in the goal area.
- Drive to the smallest tube, using the gyro sensor as a guide
- Pick up the tube and drag it to the goal area
- Put the balls in the tube
- Push tube further into goal area
- Stop and wait

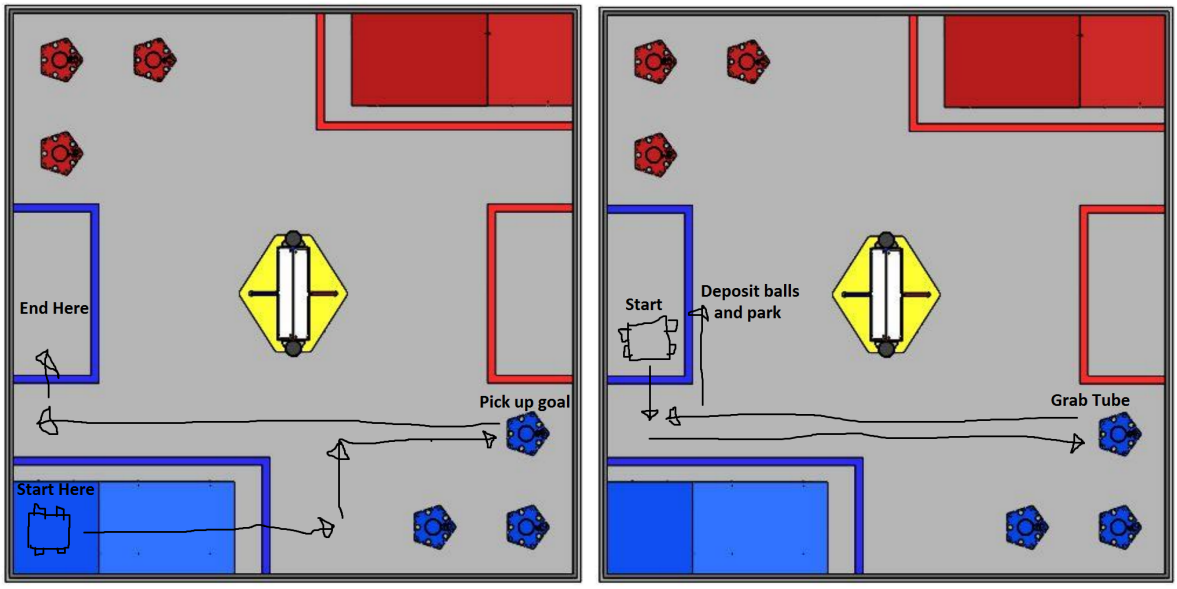
Driver Controlled:

- Use motor encoders to precisely move the scissor lift
- Use limit switches to prevent the scissor lift from burning out
- Use buttons on controller to change how controls work (e.g. a button to reverse the direction of the robot)
- Use the Phoenix system to restart the robot if anything goes wrong.
- Use status LEDs to check whether certain functionality of the robot is active

Autonomous

Here you can visually see how our autonomous program works.

Autonomous 1: (Starting on the ramp) | Autonomous 2: (Starting on the ground)



Conclusion

We hope that you enjoyed looking through our notebook and learning about our team and robot. It has been a good year so far, and we look forward to what is to come in this competition. We also look forward to some fun, competitive gameplay and the opportunity to practice gracious professionalism and learn from each other. Thank you for considering our team for the awards that are available.

Signed,