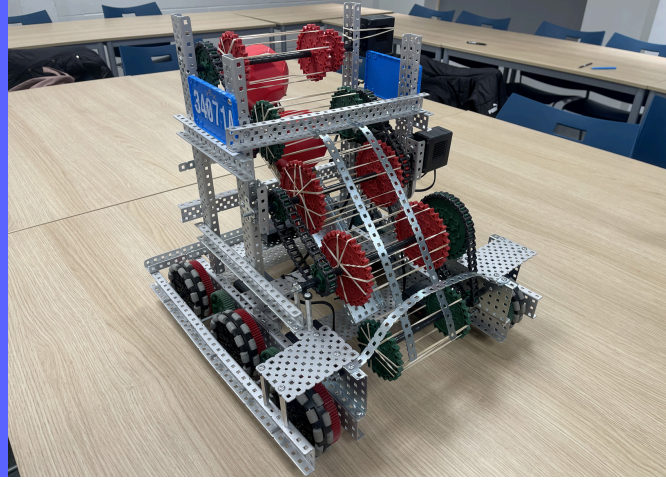


VEX V5 34071A HG5_ELECTR!C



Robotics Notebook 2025-2026 Season

School: Haberdashers' Girls' School

Our Team



Engineers: Tia Odedra, Isabel Barnwell, Sanjana Vijay, Sophie Foster

Programmers: Anika Das, Sophie Foster, Anika Gupte

Introduction

We are HG5_ELECTR!C - a team of six Year 12 students: Anika D, Anika G, Isabel, Sanjana, Sophie and Tia. Two of our members (Tia and Isabel) took part in VEX two seasons ago, whilst the rest of our team began Push Back without any previous experience with VEX. As a whole, our team blends prior knowledge with new perspectives, but we can agree that we've all developed our skills throughout this season.

We have spent the past 6 months building, coding and making amends to our robot. This is our first year competing as a part of this team, and we have found the experience extremely rewarding. We believe teamwork is a vital part of VEX Robotics and so we have all worked together as a team to combine our creativity, innovate skills and ideas together, and as a result, produce a high-performing robot.

Time management and organisation have been crucial for our team. Since most of the team study four A Levels, we had to find a system to allow us to progress as a team despite our hectic schedules. We meet every Thursday after school and during period 5 on Fridays - this is where we touch base, review progress and plan our next steps. Between these sessions, we kept a shared to-do list, allowing team members to work on the robot/programming when they have time. Although this led to less continuous build periods, it encouraged clear communication and planning from the team.

At the start of the season, we built the Hero Bot so new members could familiarise themselves with VEX, including assembling of parts, programming and game strategy. From this, everyone could develop a solid technical foundation as well as practical experience. In practice, we saw

the Hero Bot's conveyor intake to be inefficient and we soon found limitations in its design. However, we liked the concept of the conveyor and decided to do our own build based on this, with the main aim of improving reliability.

Through collaboration, iterative testing, and strict organisation, we have grown both as engineers and as a team. We are very proud of our progress and for qualifying for VEX UK Nationals 2026!

Our Goals

Before the season started, we set ourselves clear goals on what we wanted to achieve this season:

1) Be organised and communicate well with each other so that we have a clear plan for the robot and our entire team can get involved in helping out with different aspects of the robot

2) Ensure we log everything we do to allow us to reflect and learn from our mistakes, as well as to allow us to keep a clear record of our progress

3) Manage our time well as a team and take shifts on working on the robot to ensure plenty of time is spent on it



VEX V5 ROBOTICS COMPETITION PUSH BACK

Overview

Here's a quick guide overview of our notebook and season, just scroll down...



Conclusion

Our Biggest Obstacles

- Managing our time effectively
- Working around things not going to plan
- Learning new code

Key Skills We've Gained

- Innovative thinking
- Strong communication skills
- Teamwork
- Organisation
- Programming skills
- Problem solving

Contents

Introduction

[Page 1 - Summary](#)

[Page 2 - Contents](#)

[Page 3 - Introducing the Team](#)

[Page 4 - Reflecting on Past Seasons](#)

[Page 5 - Time Management](#)

[Page 6 - Planning and Design](#)

[Page 7 - Design Cycle](#)

[Page 8 - Game Analysis](#)

[Page 9 - Game Strategy](#)

Hero Bot

[Page 10 - Hero Bot Building Process](#)

[Page 11 - Programming the Hero Bot](#)

[Page 12 - Amendments on Hero Bot](#)

[Page 13 - Key Takeaways from Hero Bot](#)

Rebuild

[Page 14 - Brainstorming our New Design](#)

[Page 15 - Rebuilding the Drivetrain](#)

Programming

[Page 16 - Programming Introduction](#)

[Page 17 - How to Work the Controller](#)

[Page 18 - Python Code](#)

[Page 19 - Autonomous Programming in Python](#)

[Page 20 - C++ Code](#)

[Page 21 - Autonomous Programming in C++](#)

[Page 22 - Quick Turns Feature](#)

[Page 23 - Coding Best Practices](#)

[Page 24 - Collaborative Coding](#)

Regional Competitions

[Page 25 - Herts League of Legends & Radley Robot Rumble](#)

[Page 26 - Cranleigh Christmas Regional & Stowe New Year Event](#)

[Page 27 - Garden City Gambit](#)

[Page 28 - Radley Robot Ruckus](#)

[Page 29 - Overall Reflection](#)

Nationals Preparations

[Page 30 - Pre-Nationals Meeting](#)

[Page 31 - Preparing for Nationals](#)

[Page 32 - Narrowing the Robot](#)

[Page 33 - Adjustments](#)

[Page 34 - Drivetrain Edits](#)

[Page 35 - Adding Pneumatics](#)

[Page 36 - Coding Pneumatics](#)

[Page 37 - New Loader Mechanism](#)

[Page 38 - Updated Driver Code \(Right\)](#)

[Page 39 - Updated Driver Code \(Left\)](#)

Introducing the Team

Tia Odedra



Prior Experience:

- Competed in the 2023/2024 season
- Earned a place at nationals
- Won the Energy Award
- Mentor for the 2024/2025 season

Role in Team:

- Team Leader
- Builder
- Driver Coach

Tia has been a vital part of our building process and has taken up the role of being our team leader. She picked up many skills from DT GCSE, as well as being a member of Student Robotics which gave her the creativity and innovative thinking skills that has allowed our team to thrive. She has spent a lot of time on the building process, coming up with innovative ways to overcome any issues we have regarding the robot. Tia would like to study engineering in the future and therefore has a passion for robotics. She is also very organised, which has allowed our team to always have clear goals on what our next steps are.

Isabel Barnwell



Prior Experience:

- Competed in 2023/2024 season
- Mentor in 2024/2025 season

Role in Team:

- Builder
- Logbook Writer

Isabel has taken up the role of the logbook, and has been recording all of our progress and hard work. She has also helped out with the building process and familiarised herself with the game manual to allow our team to have an effective game strategy. Having attended many of the regional competitions, Isabel has been able to observe which areas our robot needs improvement and provide constructive feedback. Her two years of experience with VEX has been extremely useful, especially to the newer members of the team. Isabel would like to study maths in the future, with those skills being transferable to VEX when calculating ideal dimensions of the robot, as well as optimised paths for the robot to take during the autonomous period.

Sanjana Vijay



Prior Experience:

- Mentor in 2024/2025 season

Role in Team:

- Builder
- Display Creator

Sanjana has invested lots of time in building the robot, helping us come up with effective solutions when faced with problems. She has also been helpful in the scouting process, being strategic when looking for the best alliance for our robot. She devoted a lot of her time to building the Hero Bot, allowing her to get to grips with the building process. Sanjana did Computer Science GCSE, which has supplied her with strong computational thinking skills, transferable to our robot. Additionally, Sanjana has helped out lots with game strategy, helping our team brainstorm ideas to optimise our points in the game. Sanjana would like to study Economics at university and so her mathematics skills have been extremely useful in the building of our robot.

Anika Das



Prior Experience:

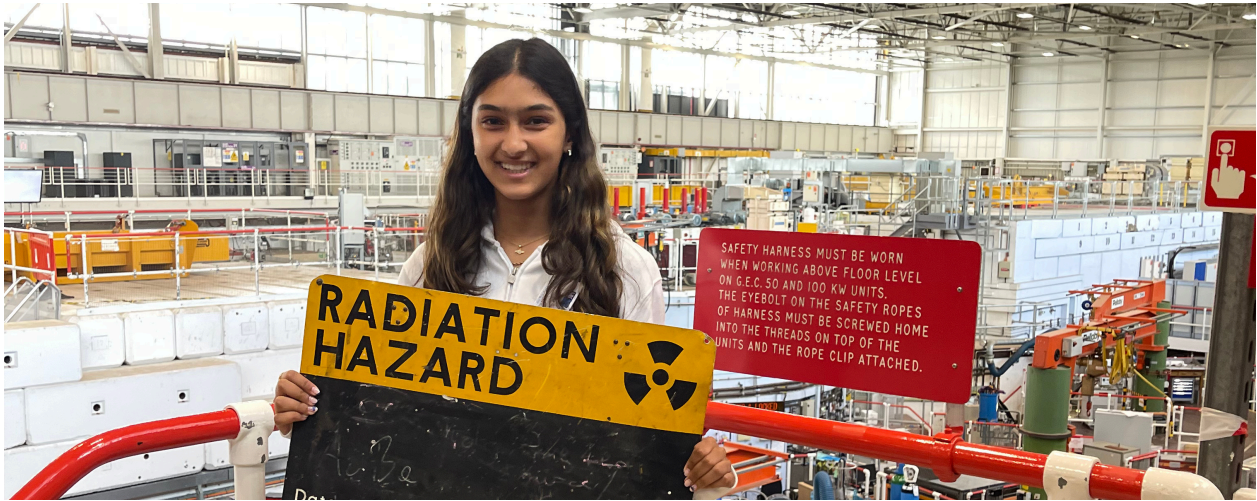
- New to VEX Robotics

Role in Team:

- Programmer
- Driver

Anika is our most recent member of HG5_ELECTR!C and has made a fundamental difference to our team. Despite having no prior experience with VEX Robotics, she has learnt very quickly and has devoted a lot of her time to VEX. Her prior knowledge of Python and C# has been extremely beneficial to our programming. Anika studies A-Level Computer Science, which allows her coding skills to be put into use. She is also a great driver which allowed us to allocate her the role of driving the robot. She has also helped out on building, giving us very effective solutions when we have come across problems. Anika would like to study Computer Science in the future, making her perfect for the role of being one of our programmers.

Anika Gupte



Prior Experience:

- New to VEX Robotics
- Currently competing in a Student Robotics team, a different robotics competition with a similar game of strategy, technique and teamwork

Role in Team:

- Programmer

Anika completed Harvard University's CS50x when she was 13 years old, teaching herself how to code primarily in C++. Since then, she has developed and refined her skills through MOOCs, professional placements and practical projects, using languages including Python, JavaScript, HTML and CSS. These skills have been highly transferable to VEX, where a combined knowledge of hardware and software is incredibly useful. She is also teaching herself Computer Science A Level to sit the exam one year early and is aspiring for a career in software engineering, aligning perfectly with her role as a programmer on our team.

Sophie Foster



Prior Experience:

- Mentor for the 2024/2025 season

Role in Team:

- Programmer
- Builder

Sophie has been very important in coding for the robot, already knowing how to use Python which helped her program the controller for driving the robot and the intake mechanism, as well as some of the autonomous period. She has great problem solving skills which she has applied to the issues faced in both programming and building, and has learned more about building the robot and the different mechanisms involved, as well as how to come up with effective solutions in order to quickly improve the robot under pressure at competitions. She has also helped come up with new ideas for what to include on the robot, helping to develop the robot's design. Sophie would like to study engineering or computing in the future, and both skills have made her a very capable member of our team.

Reflecting on Past Seasons

Having competed in the 2023/2024 season in separate teams, both Tia and Isabel have had prior experience with VEX Robotics. Therefore we were able to use this to our advantage as they were able to reflect on the 2023/2024 season and learn from their experience. It also meant our entire team could learn about the building process a lot faster, and we weren't all starting from scratch.

Tia: I competed in Over Under two seasons ago and that was my first year participating in VEX. There were four other girls in HG5_ELECTR!C who had all competed in IQ prior to this, however I had no previous experience with anything robotics related. In a way, this season was new to us all, since V5 introduced new parts and concepts. Nevertheless, I got to grips with the basics in just a few weeks and grew to be a strong designer and builder of our robot.

Similar to this season, we started by building the Hero Bot, but after the first competition decided to completely rebuild it. Whilst the Push Back Hero Bot features strong concepts used by most teams today, the Over Under Hero Bot was very different to the common approach. In the end, we decided to build a Push Bot with a flywheel and pneumatic wings. We were successful enough to qualify for nationals through skills and won the Energy Award for our enthusiasm and teamwork. Being one of only two all-girls teams, this was a very proud achievement and we were delighted to represent women in STEM at such a prestigious competition.



Upon reflection, I can say that season was an amazing experience and I learnt a lot from it. As one of the youngest teams and one with little experience, everyone was so helpful at competitions, which allowed us to progress and build on our skill set throughout the season. My experiences showed me how supportive and nurturing the VEX community is and as an older student, I aim to show that same kindness and hope to encourage younger students to get involved in VEX. The previous HG5_ELECTR!C girls have since left the school, but I look forward to working with my new team members and hope they will love VEX Robotics just as much as I do!

Isabel: Similar to Tia, I also competed in Over Under, however I was in a different team. As it was my first year of VEX Robotics, I found it extremely useful in understanding how VEX works and taking part in the regional competitions. We initially started off by building the

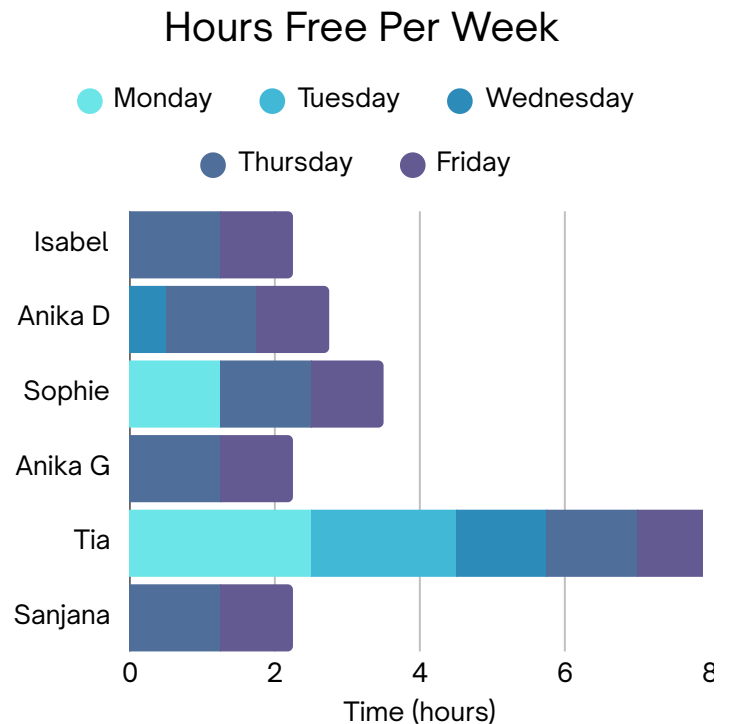
Hero Bot, which allowed us to work together as a team, all of which were new to VEX, and get to know the terminology of the different components. It definitely strengthened my teamwork skills and allowed me to build up my knowledge of VEX. The competitions were also a huge learning experience for us as it allowed us to observe other robot designs and reflect on our own. I thoroughly enjoyed coming up with solutions to any problems we encountered while building and brainstorming ideas with the rest of my team.

The Over Under season was the building blocks for all of my current knowledge regarding robotics and I am extremely grateful that I had the opportunity to start robotics in year 10 as it has helped me so much in this season and has allowed me to pass on my knowledge to the newer members of our team.

Despite not qualifying for nationals in that season, having that prior experience has been hugely beneficial to me this season and has allowed me to gain a strong passion for VEX Robotics.

Time Management

Starting this season, we are very aware of the importance of time management. As most of our team members do 4 (or 5!) A Levels, it was initially difficult for us to find the time for robotics. However, after a team meeting, we collectively agreed we would devote a lot more time to our robot. After a lot of discussion and a survey on when we are all free (see statistics below), we were able to plan effectively when tasks could be achieved.



We found this to be extremely beneficial to our team and allowed us to achieve a lot more in a single week. At the end of our first full week following this schedule, we agreed it was both manageable and effective. We concluded that balance was key and by dividing our time effectively, we found ourselves under a lot less pressure and stress. As our entire team would meet at the end of the day on a Friday, we were able to plan for the week ahead as an entire team as well as give any feedback on how the previous week went. This communication with the entire team was so helpful for all of us to be aware of any problems we are having and allow us to brainstorm as a team. We managed to stick to this schedule throughout the season and allowed our team to be much more efficient.

Planning and Design

Our team understood early on that the key to success in VEX Robotics was thorough planning and strong communication. We believe organisation has been vital for us and we have found that by communicating our plans effectively, we were able to develop each others ideas and work as a team to make our ideas a reality. Communication allowed the entirety of the team to be on the same page and meant we had a clear list of priorities, allowing the building process to run smoothly.

Benefits of Planning

Improve Communication

“Improved communication leads to better results”

Help Identify Weaknesses and Evaluate Performance

“When you plan ahead, you get the chance to detect any upcoming problems. When you are already expecting problems, you’ll be able to find solutions faster and with less hustle.”

Improve Resource Allocation and Efficiency

“Understand your biggest needs and act according to them”

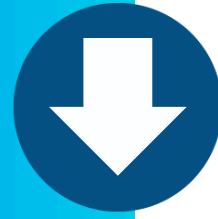
Quotes from the Engineering Management Institute

Design Cycle

When it comes to solving any problems we come across, we have decided to lay out our steps on how we go about coming up with solutions. After lots of debate, we settled upon this design cycle. Iterative design has been extremely useful to us as it means we have a logical approach to overcoming any obstacles and allows us to go about the problem in an organised manner. This cycle repeats itself if we find that the fifth stage (testing the solution) still does not work, allowing us to amend our design/program until it is doing exactly what we need it to do. The third step (brainstorming solutions) also allowed us to come together as a team and expand on each others ideas until we have an effective solution.

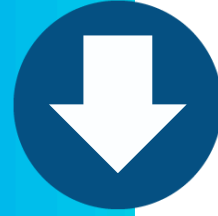
1

Identifying the Problem



2

Research & Analysis



3

Brainstorming Solutions



4

Selecting & Executing



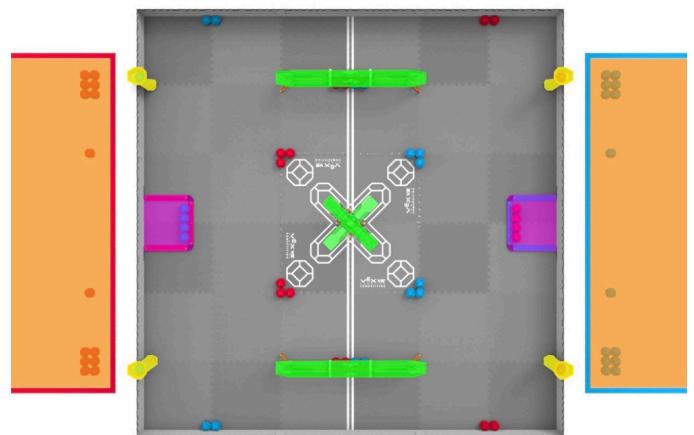
5

Testing the Solution



Game Analysis

One of the very first things we did entering this season was ensuring we knew the game inside out. From previous experience with VEX we understood the importance of understanding the game and the knowing the point system so that we could design our robot such that we would maximise our points. Therefore we set the entire team the task of reading the game manual so that we all had a thorough understanding, allowing us to build the best possible robot. It also allowed us to use the appropriate terminology, which was especially useful to the newer members of VEX and gave them an insight into how the matched work with alliance selection etc.



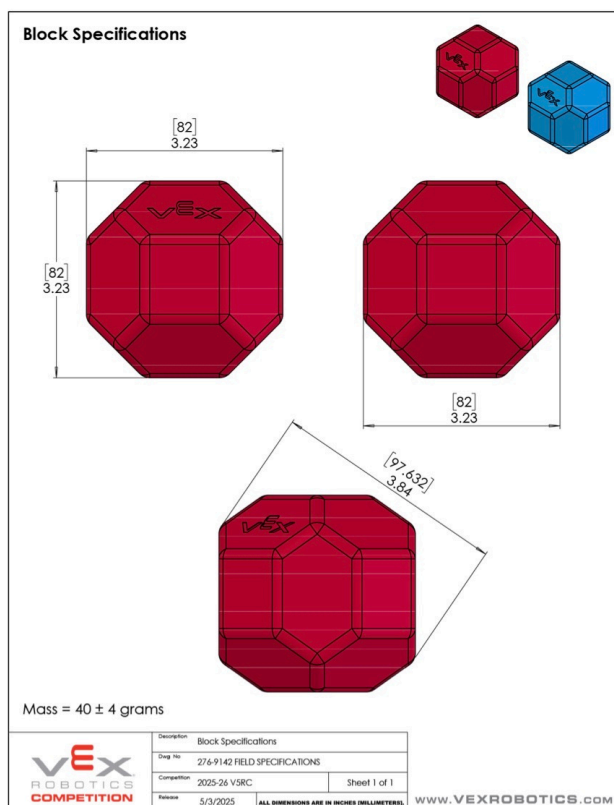
Audience View

Notes from the Game Manual

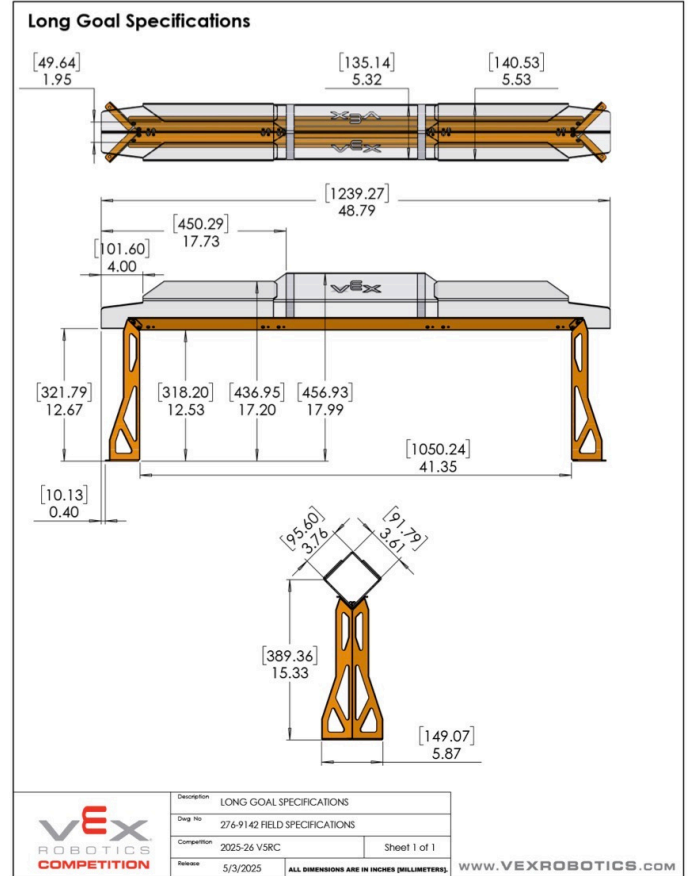
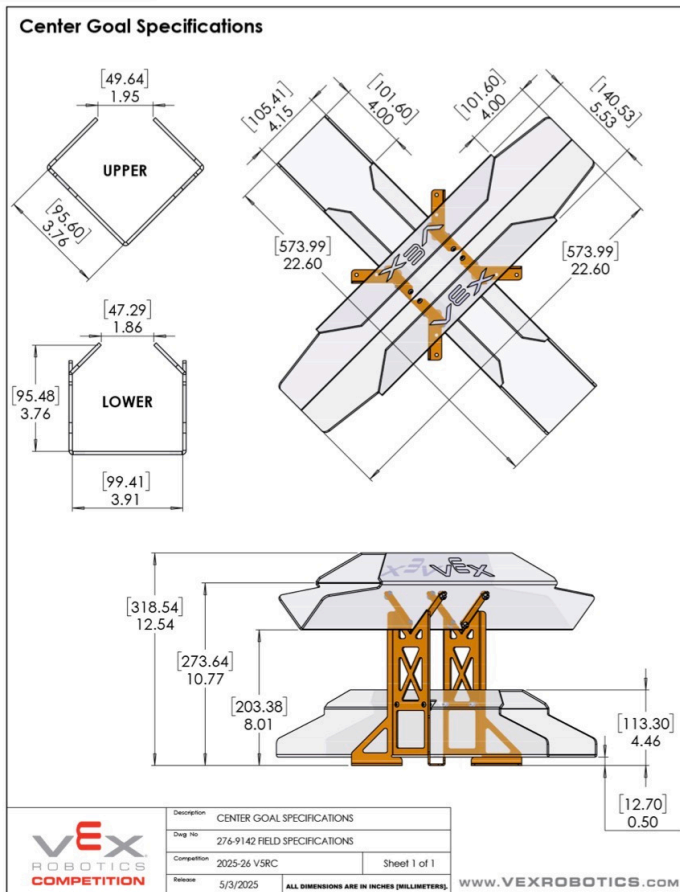
There are 44 blocks of each colour:

- 2 preloads (blocks that both allied robots start with)
- 12 match loads (placed in orange section on diagram) which can be added to loader whenever it has space
- 18 blocks on field
- 12 blocks starting in loader

There are also two centre goals (lower and upper) and two long goals, where there is a control zone (the section between the white tape); whichever colour block is in the majority of a control zone earns extra points



We made sure to study the dimensions of all the vital parts of the game and work off those dimensions when building our robot. We needed an intake mechanism that could intake both a width and height of 3.23 inches and be able to hold the block of these dimensions as it travelled up the robot. The game manual was extremely helpful in saving us time from measuring dimensions ourselves. A huge part of this process was trial and error; when building, we would centre our build around the dimensions, however when testing, we would often have to make adjustments. This allowed us to get the optimal design for our robot and ensured the block would be secure when travelling up the robot such that it wouldn't fall out but it also wouldn't get stuck. This part took a lot of trial and error, as there had to be a very specific amount of tension for the block to move up effectively. The block having the same width as height made it a lot easier as it meant that no matter which way the block was angled, the robot would have no problem picking it up as it was the same dimensions each time.



Analysing the heights of the goals were very important as we needed to make sure the robot would release the blocks at the right heights. We didn't need to worry about width for scoring because we knew that as long as the block could be carried by the robot, the only important thing was the height it was released at; width would come into play later in the game, especially when trying to park.

Lower Centre Goal: 0.5 inches from ground

Upper Centre Goal: 8.01 inches from ground

Long Goals: 12.67 inches from ground

Therefore we decided that it was important that our robot could both pick up the blocks and release them from the top but also release them back from the bottom. We needed to make a design that could score from both the long goals and the centre goals. Therefore we ensured that the intake mechanism was slightly raised such that it could drop the blocks back into the centre goals after picking them up and the release mechanism was approximately 12.6 inches from the ground.

All diagrams taken from the VEX Game Manual on 21/10/25

Game Strategy

<i>Autonomous Bonus</i>	10 Points
<i>Each Block Scored</i>	3 Points
<i>Each Controlled Zone in a Long Goal</i>	10 Points
<i>Controlled Center Goal - Upper</i>	8 Points
<i>Controlled Center Goal - Lower</i>	6 Points
<i>1 Parked Alliance Robot</i>	8 Points
<i>2 Parked Alliance Robots</i>	30 Points

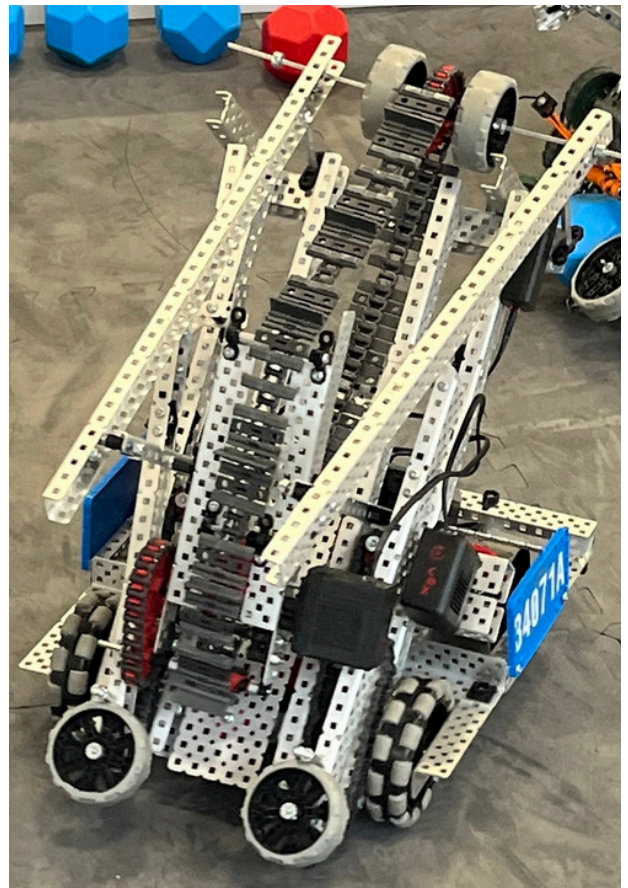
<i>Each Block Scored in a Goal</i>	1 Point
<i>Each filled Control Zone in a Long Goal</i>	5 Points
<i>Each filled Control Zone in a Center Goal</i>	10 Points
<i>Each Cleared Park Zone</i>	5 Points
<i>Each Cleared Loader</i>	5 Points
<i>Parked Robot</i>	15 Points

Allianded Competition

As the driver of our team, Anika D invested a lot of time thinking up a game strategy that would work effectively during matches using both the Hero Bot and our newer robot.

With the Hero Bot, it was more worthwhile to defend in alliances rather than to score as the conveyor system at the time was easily broken when being rammed by other robots and was also quite clunky when trying to score because of the slow conveyor speed. The advantage of the Hero Bot was its small size, allowing for it to move quickly across the field in order to block opponents from scoring in the goals, as well as to park when our allied robot was unable to. This would allow for us to make up for not being able to score blocks when focusing on the defence and would also give our partners the chance to score or descore more blocks.

This picture on the right shows the inconsistency with the conveyor as the block needed space to pop out at the top, leading to a misalignment after setting back down



In our last few matches before the retirement of the Hero Bot, we trialled a fixed descorer by attaching it onto part of our conveyor mechanism. The idea behind this was valid, but unfortunately, when trying to descore blocks, it would rotate and become unusable thereafter, so this was something to think about for our new robot design.

With our newer robot, it has been easier to defend goals against opponents due to its larger size and with a more consistent scoring mechanism, we have also had the opportunity to score more blocks in exchange for being unable to park. The newly added descorer has been crucial in gaining control zone points as not only can it remove opponent blocks, it can also push allied blocks into the control zone, gaining 10 points in the long goals.

Driver Skills

With the Hero Bot, we decided that it would be better to clear both parking zones to gain 5 points each side and then to park in an empty zone to gain a total of 25 points as scoring within the time in between was a struggle with a slow conveyor system.

With our newer robot, the focus was mostly on scoring as it was hard for our larger robot to clear or park in the parking zones and although we expected this to work well, it unfortunately didn't due to our slower moving drivetrain and inability to be able to pick up multiple blocks at a time as previous ones would unload themselves out of the top. This has been reconsidered in our run-up to nationals and will hopefully be more successful.

Pre-Match Strategizing

At competitions, we have found having a conversation with our upcoming alliances extremely important. It allows us to get to know the strengths and weaknesses of their robot and allows us to strategize with them. Some examples of things we may talk to them about are:

- Who will park
- Which robot is better for defence/attack
- Which robot has autonomous (potentially has to start on certain side of field)
- How effective their robot is at scoring (which goals do they score best in etc.)
- Give/receive any last minute advice on amendments that can be made

We believe this pre-match planning is important as it allows us to have an agreed strategy before every match as it can vary greatly from team to team.

Hero Bot Building Process

The VEX library is an invaluable resource for anyone looking to build a robot, providing a comprehensive collection of instructions and components that simplify the process. When constructing the Hero Bot, the VEX library played a crucial role, particularly in the design and modification of its intake and conveyor mechanisms.

Using the VEX Library

The VEX library offers detailed instructions and parts that allow builders to create complex robots with relative ease. For the Hero Bot, the library's step-by-step guides ensured that the basic structure was sturdy and functional. The modularity of VEX components meant that builders could easily swap parts in and out, offering a high degree of customisation.

The Hero Bot's intake and conveyor system is critical for its progress in competitions, as it allows the robot to collect and score blocks efficiently. However, the initial design often requires adjustments to optimize performance, so we decided to change the intake and conveyor systems to allow the robot to function better.

Impact of Adjustments

These adjustments, informed by the flexibility and guidance provided by the VEX library, significantly improved the Hero Bot's efficiency and reliability. The modifications allowed the robot to adapt better in match scenarios, showcasing the benefits of using a robust library and instructions in robotics building.

Programming the Hero Bot

The code for our Hero Bot was written in Python, as this was a common language that all three of our programmers (Anika G, Anika D and Sophie) were familiar with.

We were able to read through the VEX V5 Python library and implement an “army tank” style control where both joysticks need to be pushed forward to drive straight forwards, resulting in better control over the robot when turning at speed.

This controller mechanism has stayed with our robot the entire season, even with our new robot and new C++ code, and we have kept the dual joystick movement due to its enhanced control.

Using both joysticks for driving forward has been incredibly useful when driving the Hero Bot as this has allowed for better manoeuvrability with a 2-motor drivetrain and for quickly dashing over to defend against opponents.

Amendments on Hero Bot

After testing the original Hero Bot, we decided to make improvements before coming up with a completely new design. This would allow us to gain a better understanding of the most important features of the robot for the game, which would make our new robot design even better.

The Descorer

Identifying the Problem

We realised that our Hero Bot wasn't great at defence and therefore tried to find a solution to this. After observing other robots at competitions, we realised that the best robots at defending had a descorer mechanism which would knock out blocks from the long goals. Having witnessed how effective this was in competitions, we decided we would try and make one ourselves.

Research & Analysis

After a bit of research, we discovered that the descorer mechanism incorporated pneumatics, something we did not have. Therefore, after analysing the structure of other robots descorsers at one of our regional competitions, we had the idea of trying to make a last minute descorer without pneumatics.

Brainstorming Solutions

Our team came together to brainstorm ideas regarding how to make a descorer without the use of pneumatics. After much trial and error, we came up with a design that we thought might work.

Selecting & Executing

We managed to attach a temporary descorer onto our robot using a malleable piece of metal and a long screw at the end to knock the blocks out of the long goal. We could see that it wasn't very stable and agreed that this was a temporary solution.

Testing the Solution

After testing this, we realised that having it permanently on the side of the robot and unable to retract led to it often getting caught on other robots or the goals, hindering movement on the field. This made us realise the importance of pneumatics in the new design.

The Intake Mechanism

Identifying the Problem

We noticed that our intake mechanism needed amending as the chains connected to the wheels at the front often broke, which could lead to us being unable to pick up blocks during a match.

Research & Analysis

We did lots of research on intake mechanisms which we could incorporate into our hero bot without having to rebuild the entire thing from scratch. We came across the use of sprockets and elastic bands and we agreed as a team that it could work.

Brainstorming Solutions

It took a bit of time to come up with a way to actually attach this new intake mechanism without hindering any other parts of the robot. We debated how far along the metal shafts to put the axle holding the sprockets and elastic bands and decided to settle on somewhere that wouldn't get in the way of the rest of the intake mechanism.

Selecting & Executing

We found the perfect length of axle that would reach both metal shafts on either end and decided on a sprocket size that allowed the elastic bands to be taut enough to grip the blocks.

Testing the Solution

We concluded that this intake mechanism was much more reliable and less prone to breaking, and so we kept this intake mechanism for the new robot design. We also found that it made things slightly easier for the driver as less buttons were needed to control the intake. We found that this new intake mechanism made a profound impact during competitions, making our robot much more reliable.

Releasing The Blocks

Identifying the Problem

Another problem with the Hero Bot was that, in order for the robot to be within the height limit, when releasing blocks the conveyor mechanism expanded upwards in order to have enough room for the block to leave the robot and enter the goal. However, this didn't always work, and the block often got stuck at the top of the robot while it was being released.

Research & Analysis

After analysing our robot, we concluded that we had this problem partially because the motor system wasn't powerful enough. We also agreed that the fact that our robot needed

to expand in the first place to allow the block to be released was problematic and would likely cause us more problems in the future.

Brainstorming Solutions

We were thinking of improving our motor system through programming as well as potential amendments to the location of the motor. Another idea was to improve the design of the section of robot that released the blocks in such a way that allowed the blocks to be released without the top of the robot needing to lift up to give the block space to pass through.

Selecting & Executing

We attempted to change the design of the block release, however it was a difficult job we were very hesitant as we were unsure that any design we tried would be effective.

Testing the Solution

We discovered that our new design led to further problems with both the release and the conveyor. We concluded that the best thing to do was to start from scratch with a whole new robot.

Key Takeaways from Hero Bot

Conveyor System

The conveyor system faced significant challenges in competitions, with the chain attaching it to the motor being easily broken when knocked several times from the other team, so we decided that the whole intake system would have to be rethought in order to create a more reliable mechanism.

Unstable Descorer

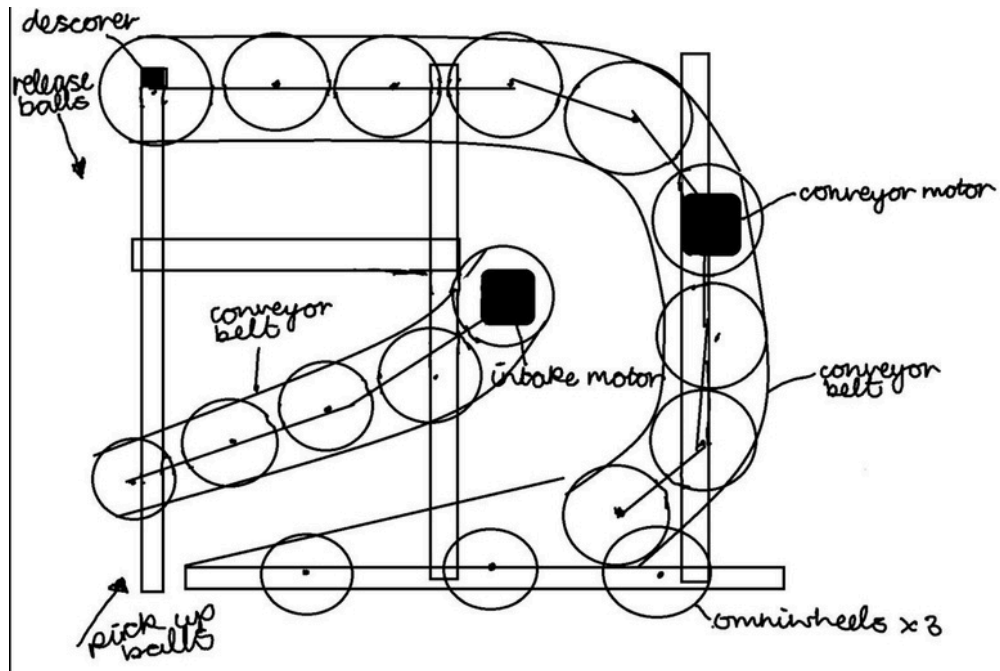
The fixed descorer we added had frequent misalignments with the long goals and would also twist after knocking against the walls of the goal. The concept was a good one, but our application of it needed work so this was something we thought about incorporating in our new design.

Drivetrain Improvement

After seeing how a 2-motor drivetrain performs in a match with the Hero Bot, we decided that our new design should use a 4-motor drivetrain which would allow for higher torque and acceleration, especially with a top-heavy design. Moreover, we wanted to add gearing in our drivetrain as it is more versatile with its output speeds, and would make it more stable overall, helping with defence against pushing. Furthermore, using the 8 motor allowance, we would still have two motors to spare as we were planning to use the other two in making the conveyor.

We believed that addressing these issues would be crucial in our new design and would drastically improve our performance at later competitions.

Brainstorming our New Design



Sophie came up with an idea for a design and drew out a rough sketch digitally (as seen above). We liked the conveyor belt mechanism and found it to be very effective on the Hero Bot.

After using and driving the Hero Bot and gaining an understanding of its strengths and weaknesses, a new design was created which would help us at competitions. A new intake mechanism which relied on sprockets and elastic bands was important, as the old mechanism of wheels that grabbed the balls from either side was not very powerful, and was often faulty with the chains frequently breaking. These sprockets were then connected by chains to make the complete conveyor system. More motors were also needed for more power and reliability in the conveyor system, allowing the blocks to be taken in and released quickly and with as few malfunctions as possible. The design also had both the intake and release mechanism on the front of the robot for ease of driving, leading to increased precision when releasing the balls into the long goals. A descorer relying on pneumatics was also vital to the new design as the Hero Bot had no way of removing the other alliance's blocks from the goals. The drivetrain also needed to be made more powerful, with more wheels and motors, as a two-motor, four-wheel drivetrain was relatively slow and unreliable.

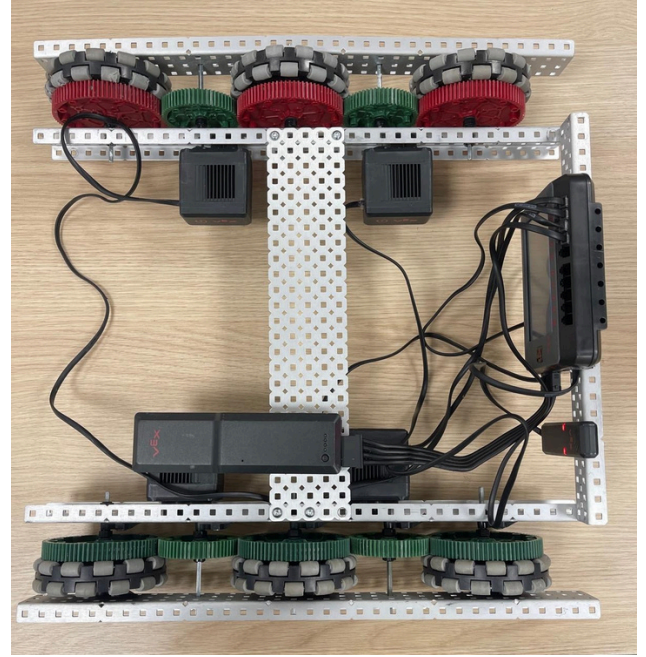
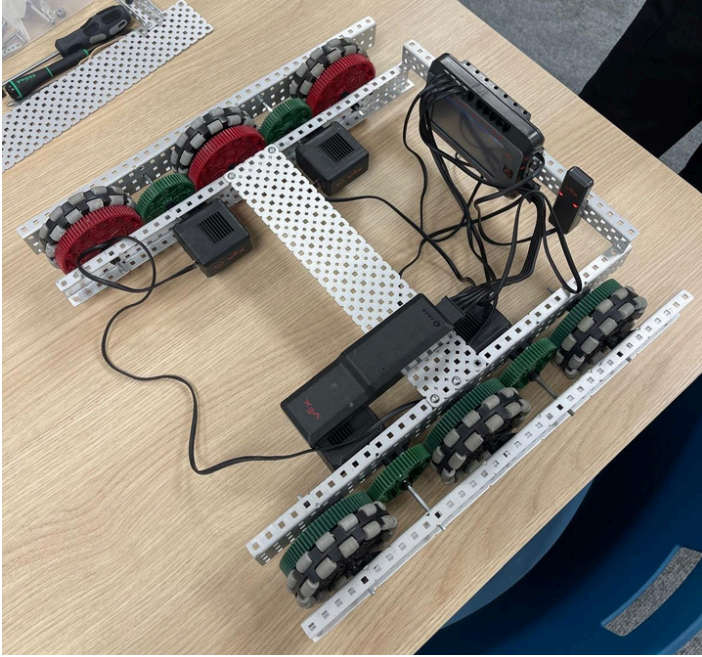
Verdict

For the final design, we found many of the individual parts of this design useful, and the improved drivetrain, intake system, conveyor system and descorer were all used in the final design. The conveyor system was also upgraded to make use of gear ratios for increased speed. However, we realised that having the both the intake and release mechanisms on the front of the robot was unrealistic as it could more easily cause problems with the robot and was more likely to lead to failures in the conveyor mechanism as it was longer. It would also

mean that it would take more time for the blocks to be released in the high goals after intaking, which could slow the robot down and potentially lead to less blocks scored overall. This thinking led to us adopting a design more similar to the Hero Bot with the intake at the front and release at the back.

Since we have some time until our next competition, building a new robot from scratch is finally feasible. We started with building the drive train, as this sets a solid foundation for the rest of the robot. We have designed a conveyor-style mechanism, however using an inclined row of rubber band rollers, as we believe this will be more efficient for scoring. This will be more reliable than the Hero Bot conveyor, which kept malfunctioning. During this build, extra care will be taken at every step to ensure every part is secured - this will prevent technical difficulties in the future.

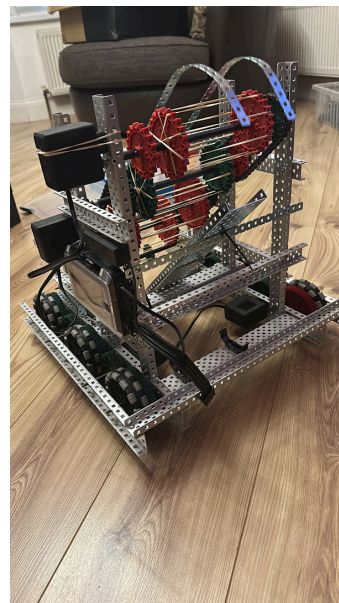
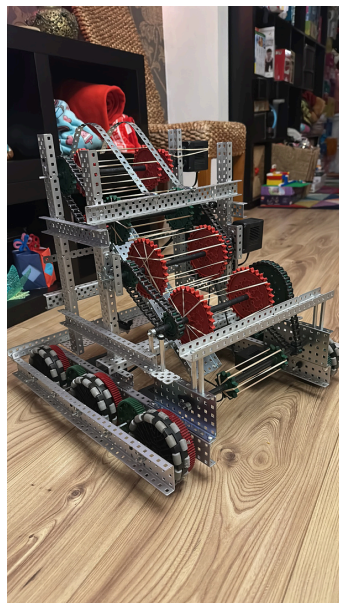
Rebuilding the Drivetrain



For our new drivetrain design, we have opted for a 6-wheel, 4 motor drivetrain. Why?

- We want to incorporate gearing as it's more versatile, since we can experiment with different output speeds. Our drivetrain is geared down, sacrificing some speed for torque, which is necessary for stability and pushing power
- 4 motors are being used for higher torque and acceleration; this is needed since the robot will be top heavy. It's also more stable than two motors, helping with defence against pushing

Building the Conveyor



To build the conveyor, we made vertical pillars using c-channels. This allowed us to experiment with different roller heights to find the optimal angle for the intake chain. The bar

at the front was added to ease carrying the robot around.

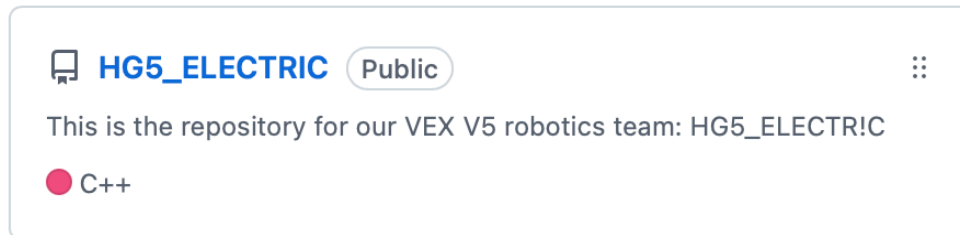
- There are two motors - one driving the top two rollers and another driving the bottom three
- They are connected by linked sprockets, but they all have the same speed apart for the first one. The intaking roller spins 2.5x faster than the others, due to a sprocket ratio implemented $\gg 12T/30T$. This allows the ball to enter the conveyor at a higher speed, giving it a higher momentum

Programming Introduction

Our programming team consists of Anika G, Anika D and Sophie.

Following coding best practices, the code for our robot was frequently uploaded to GitHub throughout the season, where we were able to collaborate, track version history and create pull requests.

The GitHub repository can be viewed here: [🌐 GitHub - anikagupte/HG5_ELECTRIC](#)



We began our season coding in Python, familiarising ourselves with the VEX library in a language that most of us were familiar with. This allowed us to get to grips with programming a robot, autonomous code and just having a general understanding of how it all works.

Before Nationals, we made the decision to use C++ instead of Python for more control over specific hardware and processor level actions. Out of the programmers, only Anika G had used C++ before and Anika D had experience in C#, so this took lots of research (and YouTube) to understand.

Hence, our GitHub repository is split into two folders, one for Python and the other for C++. While we used Python for all of our competitions until this point, our Nationals code is in C++ for the advantages previously mentioned.

▼ c++

-  left_side_comp.cpp
-  main.cpp
-  pneumatics.cpp
-  right_side_comp.cpp

▼ python

-  Controller.v5python
-  Left_Side_Comp.py
-  Right_Side_Comp.py
-  pneumatics.py
-  pneumatics_test.py

 README.md

How to Work the Controller



Photos taken from the VEX Library

We decided to have one joystick control either side of the drivetrain using the controller as this would allow for greater manoeuvrability of our robot, so by alternating the direction of the joysticks, we were able to rotate the robot to face the direction required.

With the L and R buttons, we decided to make the L buttons control the conveyor direction and the R buttons control the descorer release as this would make it easier for the driver to find what function they require. Both L2 and R2 control the startup of the descorer and intake

system, while L1 and R1 control the closing of the descender and reverse direction of the conveyor. The quick and easy access of these buttons when driving have allowed for quick moves in matches and have been very fundamental for our successes.

For our loader, we chose to use the arrow keys as the way to open and close it as it made sense to be able to put the loader “down” to open it and “up” to close it.

Python Code

As we started the season in Python, below is our Python code. The controller works as described on the previous page, but we've been sure to incorporate docstrings and comments to make it clear to the reader!

Our Python code can also be found here:

[HG5_ELECTRIC/python](#) at main · anikagupte/HG5_ELECTRIC

```
1  # -----
2  #
3  #   Project:      Controller Code
4  #   Author:       HG5_ELECTRIC
5  #   Description:  VEXcode V5 Python Test Code
6  #
7  # -----
8
9  """
10 This is the test code for our robot - it is used solely during training
11 and does not include a competition instance.
12 """
13
14 # library imports
15 from vex import *
16 import urandom
17
18 # brain should be defined by default
19 brain = Brain()
20
21 # robot configuration code
22 left_motor_a = Motor(Ports.PORT1, GearSetting.RATIO_36_1, False)
23 left_motor_b = Motor(Ports.PORT3, GearSetting.RATIO_36_1, False)
24 left_drive_smart = MotorGroup(left_motor_a, left_motor_b)
25 right_motor_a = Motor(Ports.PORT2, GearSetting.RATIO_36_1, True)
26 right_motor_b = Motor(Ports.PORT4, GearSetting.RATIO_36_1, True)
27 right_drive_smart = MotorGroup(right_motor_a, right_motor_b)
28 drivetrain = DriveTrain(left_drive_smart, right_drive_smart, 319.19, 295, 40, MM, 1)
29 conveyor_motor_a = Motor(Ports.PORT5, GearSetting.RATIO_18_1, False)
30 conveyor_motor_b = Motor(Ports.PORT6, GearSetting.RATIO_18_1, False)
31 conveyor = MotorGroup(conveyor_motor_a, conveyor_motor_b)
32 descorer = DigitalOut(brain.three_wire_port.a)
33 match_loader = DigitalOut(brain.three_wire_port.b)
34
```

```

35 # wait for rotation sensor to fully initialize
36 wait(30, MSEC)
37
38 ✓ def initializeRandomSeed():
39     """
40     make random actually random
41     """
42     wait(100, MSEC)
43     random = brain.battery.voltage(MV) + brain.battery.current(CurrentUnits.AMP) * 100 + brain.timer.system_high_res()
44     urandom.seed(int(random))
45
46 # set random seed
47 initializeRandomSeed()
48
49 ✓ def play_vexcode_sound(sound_name):
50     """
51     helper to make playing sounds from the V5 in VEXcode easier and
52     keeps the code cleaner by making it clear what is happening
53     :param sound_name: string
54     """
55     print("VEXPlaySound:" + sound_name)
56     wait(5, MSEC)
57
58 # add a small delay to make sure we don't print in the middle of the REPL header
59 wait(200, MSEC)
60 # clear the console to make sure we don't have the REPL in the console
61 print("\033[2J")
62

```

```

63 ✓ def user_control():
64     """
65     begins project code and defines controller buttons
66     """
67     brain.screen.clear_screen()
68     controller = Controller()
69     drivetrain.set_drive_velocity(100, PERCENT)
70     while True:
71         left_motor_a.set_velocity(controller.axis3.position(), PERCENT)
72         left_motor_a.spin(REVERSE)
73         left_motor_b.set_velocity(controller.axis3.position(), PERCENT)
74         left_motor_b.spin(REVERSE)
75
76         right_motor_a.set_velocity(controller.axis2.position(), PERCENT)
77         right_motor_a.spin(REVERSE)
78         right_motor_b.set_velocity(controller.axis2.position(), PERCENT)
79         right_motor_b.spin(REVERSE)
80
81         wait(5, MSEC)
82
83         # move conveyor belt up (pick up or place ball into high tube)
84         conveyor.set_velocity(100, PERCENT)
85         if controller.buttonR1.pressing():
86             conveyor.spin(FORWARD)
87         # place ball into lower tube
88         elif controller.buttonR2.pressing():
89             conveyor.spin(REVERSE)
90         else:
91             # stop conveyor belt
92             conveyor.stop()
93         wait(20, MSEC)
94

```

```
95         # moving the descorer
96         if controller.buttonL2.pressing():
97             # open descorer
98             descorer.set(True)
99         elif controller.buttonL1.pressing():
100             # close descorer
101             descorer.set(False)
102
103         # moving the loader
104         if controller.buttonDown.pressing():
105             # open loader
106             match_loader.set(True)
107         elif controller.buttonUp.pressing():
108             # close loader
109             match_loader.set(False)
110
111         """
112         while line 116 would conventionally be wrapped in if __name__ == "__main__"
113         we have not done this as it appeared to be unsupported in the VEX environment
114         """
115
116         user_control()
```

Autonomous Programming in Python

We created two autonomous routines depending on which side of the field we start on during a match.

Starting on the right:

```
69  ✓ def autonomous():
70      """
71      starts facing the right wall, loads 3 balls and then scores all 3
72      """
73      brain.screen.clear_screen()
74      brain.screen.print("Autonomous code running...")
75      drivetrain.drive_for(REVERSE, 125, MM)
76      drivetrain.turn_for(RIGHT, 80, DEGREES)
77      drivetrain.drive_for(REVERSE, 690, MM)
78      conveyor.set_velocity(100, PERCENT)
79      conveyor.spin_for(REVERSE, 1250, DEGREES) # load 3
80      drivetrain.turn_for(RIGHT, 45, DEGREES)
81      drivetrain.drive_for(REVERSE, 110, MM)
82      drivetrain.turn_for(RIGHT, 10, DEGREES)
83      conveyor.spin_for(FORWARD, 3240, DEGREES) # score 3 balls
84
```

Starting on the left:


```
69  ✓ def autonomous():
70      """
71      starts facing the left wall, loads 3 balls and then scores all 3
72      """
73      brain.screen.clear_screen()
74      brain.screen.print("Autonomous code running...")
75      drivetrain.drive_for(REVERSE, 125, MM)
76      drivetrain.turn_for(LEFT, 80, DEGREES)
77      drivetrain.drive_for(REVERSE, 690, MM)
78      conveyor.set_velocity(100,PERCENT)
79      conveyor.spin_for(REVERSE, 1250, DEGREES) # load 3
80      drivetrain.turn_for(RIGHT, 80, DEGREES)
81      drivetrain.drive_for(REVERSE, 100, MM)
82      drivetrain.turn_for(LEFT, 12, DEGREES)
83      conveyor.spin_for(REVERSE, 3240, DEGREES) # score 3 balls
84
```


C++ Code

Before Nationals, we decided to switch our code to C++. While for most robot tasks, the difference may be negligible, C++ gives us a huge advantage for time-critical operations such as fast autonomous routines because it is compiled directly to machine code.

C++ gives us lower-level access to the brain's hardware and fine tune motor skills which can give us the edge in a match where every millisecond counts.

It is also the dominant language in embedded systems, robotics and game development, so using this language simulates real-life technology.

Our C++ code can also be found here:

[HG5_ELECTRIC/c++ at main · anikagupte/HG5_ELECTRIC](#)

```
1  /*-----*/
2  /*                                           */
3  /*   Module:      main.cpp                 */
4  /*   Project:     Controller Code          */
5  /*   Author:      HG5_ELECTRIC            */
6  /*   Description: VEXcode V5 C++ Test Code */
7  /*                                           */
8  /*   This is the test code for our robot - it is used solely during */
9  /*   training and does not include a competition instance.         */
10 /*                                           */
11 /*-----*/
12
13 #include "vex.h"
14
15 using namespace vex;
16
17 /*-----*/
18 /*           Robot Configuration           */
19 /*-----*/
20
21 brain Brain;
22
23 motor left_motor_a  = motor(PORT1, ratio36_1, false);
24 motor left_motor_b  = motor(PORT3, ratio36_1, false);
25 motor right_motor_a = motor(PORT2, ratio36_1, true);
26 motor right_motor_b = motor(PORT4, ratio36_1, true);
27
28 motor_group left_drive_smart  = motor_group(left_motor_a, left_motor_b);
29 motor_group right_drive_smart = motor_group(right_motor_a, right_motor_b);
30
31 // Drivetrain: wheel circumference=319.19mm, track=295mm, wheelbase=40mm
32 drivetrain Drivetrain = drivetrain(left_drive_smart, right_drive_smart, 319.19, 295, 40, mm, 1);
```

```

33
34 motor conveyor_motor_a = motor(PORT5, ratio18_1, false);
35 motor conveyor_motor_b = motor(PORT6, ratio18_1, false);
36 motor_group conveyor = motor_group(conveyor_motor_a, conveyor_motor_b);
37
38 // Descorer on 3-wire port A, match loader on 3-wire port B
39 digital_out descorer = digital_out(Brain.ThreeWirePort.A);
40 digital_out match_loader = digital_out(Brain.ThreeWirePort.B);
41
42 controller Controller = controller(primary);
43
44 /*-----*/
45 /*                      User Control                      */
46 /*                      */
47 /*          Defines controller buttons and begins project code.          */
48 /*-----*/
49 void user_control() {
50     Brain.Screen.clearScreen();
51
52     Drivetrain.setDriveVelocity(100, percent);
53
54     while (true) {
55         // Tank drive - Axis3 = left stick vertical, Axis2 = right stick vertical
56         int leftSpeed = Controller.Axis3.position();
57         int rightSpeed = Controller.Axis2.position();
58
59         left_motor_a.setVelocity(leftSpeed, percent);
60         left_motor_a.spin(reverse);
61         left_motor_b.setVelocity(leftSpeed, percent);
62         left_motor_b.spin(reverse);

```

```

63
64         right_motor_a.setVelocity(rightSpeed, percent);
65         right_motor_a.spin(reverse);
66         right_motor_b.setVelocity(rightSpeed, percent);
67         right_motor_b.spin(reverse);
68
69         wait(5, msec);
70
71         // Conveyor belt control
72         conveyor.setVelocity(100, percent);
73         if (Controller.ButtonR1.pressing()) {
74             conveyor.spin(forward); // pick up / place into high tube
75         } else if (Controller.ButtonR2.pressing()) {
76             conveyor.spin(reverse); // place into lower tube
77         } else {
78             conveyor.stop();
79         }
80
81         wait(20, msec);
82
83         // Descorer control
84         if (Controller.ButtonL2.pressing()) {
85             descorer.set(true); // close
86         } else if (Controller.ButtonL1.pressing()) {
87             descorer.set(false); // open
88         }

```

```

89
90     // Match loader control
91     if (Controller.ButtonDown.pressing()) {
92         match_loader.set(true); // open
93     } else if (Controller.ButtonUp.pressing()) {
94         match_loader.set(false); // close
95     }
96 }
97 }
98
99 /*-----*/
100 /*                               Main                               */
101 /*-----*/
102 ✓ int main() {
103     // Initialize random seed
104     srand((int)(Brain.Battery.voltage(voltageUnits::mV)
105         + Brain.Battery.current(currentUnits::amp) * 100
106         + Brain.Timer.systemHighResolution()));
107
108     wait(200, msec);
109     Brain.Screen.clearScreen();
110
111     // Run driver control directly (no competition instance - training only)
112     user_control();
113
114     return 0;
115 }

```

Autonomous Programming in C++

Likewise, using C++, we created two autonomous routines depending on which side of the field we start on during a match.

Starting on the right:

```
50      /*-----*/
51      /*                      Autonomous                      */
52      /*-----*/
53  ✓ void autonomous() {
54      Brain.Screen.clearScreen();
55      Brain.Screen.print("Autonomous code running...");
56
57      // Auton code - start facing right wall
58      Drivetrain.driveFor(reverse, 125, mm);
59      Drivetrain.turnFor(right, 80, degrees);
60      Drivetrain.driveFor(reverse, 690, mm);
61
62      conveyor.setVelocity(100, percent);
63      conveyor.spinFor(reverse, 1250, degrees);    // load 3
64
65      Drivetrain.turnFor(right, 45, degrees);
66      Drivetrain.driveFor(reverse, 110, mm);
67      Drivetrain.turnFor(right, 10, degrees);
68
69      conveyor.spinFor(forward, 3240, degrees);    // score 3 balls
70  }
71
```

Starting on the left:

```

52  /*-----*/
53  /*              Autonomous              */
54  /*-----*/
55  ✓ void autonomous() {
56      Brain.Screen.clearScreen();
57      Brain.Screen.print("Autonomous code running...");
58
59      // Auton code - start facing left wall
60      // HAS NOT BEEN TESTED
61      Drivetrain.driveFor(reverse, 125, mm);
62      Drivetrain.turnFor(left, 80, degrees);
63      Drivetrain.driveFor(reverse, 690, mm);
64
65      conveyor.setVelocity(100, percent);
66      conveyor.spinFor(reverse, 1250, degrees);    // load 3
67
68      Drivetrain.turnFor(right, 80, degrees);
69      Drivetrain.driveFor(reverse, 100, mm);
70      Drivetrain.turnFor(left, 12, degrees);
71
72      conveyor.spinFor(reverse, 3240, degrees);    // score 3 balls
73  }
74

```

Quick Turns Feature

We wanted to add some features that would make it easier for our driver (Anika D) to control the robot, so we decided to add quick turns.

This feature allows us to turn exactly 90 degrees with one click, rather than manually turning the robot with the joysticks. It also means a double click can turn the robot 180 degrees, allowing it to quickly flip around and change direction.

This is particularly useful in a match because it allows for versatility when dealing with unpredictable opponents and alliances.

To execute this, we used the left and right buttons on the controller, as these made logical and intuitive sense.

Below is the code for our quick turns feature:

```
119         // Quick turns
120         if (Controller.ButtonRight.pressing()) {
121             Drivetrain.turnFor(right, 90, degrees); // turn 90 degrees right
122         } else if (Controller.ButtonLeft.pressing()) {
123             Drivetrain.turnFor(left, 90, degrees); // turn 90 degrees left
124         }
125
126         wait(20, msec);
127     }
128 }
```

Filters ▾

Q is:pr is:open

🔍 Labels 9

📌 Milestones 0

New pull request

☐ 1 Open ✓ 6 Closed

Author ▾

Label ▾

Projects ▾

Milestones ▾

Reviews ▾

Assignee ▾

Sort ▾

☐  added quick turns

#7 opened 30 minutes ago by anikagupte

Coding Best Practices

We recognise the importance of following programming conventions such as coding best practises. All of our code is neatly commented, but here are other ways we collaborated on our code.

These helped us maintain effective teamwork and communication, replicating how development teams work in the real world.

Version Control

When implementing large changes to the code, we created a new branch of code on the GitHub repository to keep a working copy of the code on the main branch at all times. We then created pull requests and peer reviewed the code before merging it back into the main branch.

The main branch always contains stable, competition-ready code throughout the entire season.

Docstrings

As we have multiple code files with different purposes, each file has a module-level docstring detailing a quick summary of the program. This has especially helped when working in a team, as everyone can take over from another person and continue to work on a piece of code.

Functions include their own docstrings to label what they do so that it is clear when they are called later in the code.

Labelling Parameters

In each subroutine, our parameters are labelled so that it is clear which data type each subroutine takes in to avoid any errors. It also makes the code very easy to follow for other people who haven't written that particular section of code.

Consistent Code Formatting

All of our Python code followed PEP 8 formatting so that it was consistent and remained within Python conventions. This includes meaningful variable names, comments for hardware component ports and reviewing comments before a branch is merged.

Similarly, our C++ code is formatted according to standard conventions, remaining consistent throughout.

Descriptive Commit Messages

To ensure that everyone knows about latest code updates, we made sure all commit measures were descriptive about the changes that were made. This helped us work in a team as it ensures good communication.

Commits on Feb 22, 2026

Update Controller.v5python An1Da5 authored 2 days ago	Verified3a18e08
cleaned code anikagupte authored 2 days ago	Verified3db9042
added coding best practices anikagupte authored 2 days ago	Verified75c5c34
swapped buttons over anikagupte authored 2 days ago	Verified9e0cfa9
Pneumatics for loader and dsecorer sf176 authored 2 days ago	Verified01a2e47
added smartdrive to drivetrain fix anikagupte authored 2 days ago	Verifiedcafd378
first commit anikagupte authored 2 days ago	Verifiedde7a415
added auton (right side) anikagupte authored 2 days ago	Verifieda1289ec
translated newer version anikagupte authored 2 days ago	Verifiedceb5922
moved to python folder anikagupte authored 2 days ago	Verified3f65b73

Collaborative Coding

Anika G: Over the summer, I did a two week placement at STFC, the UK's science and technology council, working on DevOps with the cloud team at the scientific computing department. During this time, I learnt a lot about how real programming teams work, using platforms such as GitHub and Slack to collaborate on code.



I was able to transfer these skills to our VEX team, particularly with my new knowledge of GitHub. Using commit messages, pull requests, peer reviews and more, we were able to revolutionise the efficiency of our team. Below is an outlined process of how we made large changes to the code:

1. Create a new branch to isolate the changes
2. Make code changes
3. test the changes on our robot
4. Create a pull request
5. Peer review the code with feedback on any errors, ways to make code more efficient or just a general check
6. Merge branch into main branch
7. Code is successively updated while maintaining version control



An1Da5 commented 1 hour ago

Collaborator ...

Thanks for adding the match loader controls into C++, looks good to me!





anikagupte commented 3 minutes ago

Owner ...

you've solved the issue - amazing!



Herts League of Legends -

14/10/25 - 19/12/25

 34071A - HG5_ELECTR!C 
@ V5RC | Push Back | Herts League of Legend...

Qualification Ranking

Current Rank

10

WP / AP / SP / W-L-T
4 / 50 / 206
2-10-0

Skills

Skills Rank

5

Score / Attempts
 20 / 1
<> 0 / 1

After spending part of the summer and start of the academic year building the Hero Bot, we attended our first competitions. A few schools in the area formed a VEX league, and we competed in multiple mini competitions after school, one of which we hosted ourselves. This was a valuable experience, since we gained an insight into gameplay early on - even after revising the manual and simulating matches on our field at school, we've realised that there is always more to be learnt during real competitions, as they allow us to interact with other teams to improve teamwork and strategy skills. This league also allowed us to get to know these few teams a lot better, and seeing their progress throughout the weeks was very encouraging.

The start of the school year has been very busy for us as we try to familiarise ourselves with sixth form life and learn to balance robotics progress with the demands of A Levels. Recognising this struggle already, we are going to be more organised and improve our time management.

Radley Robot Rumble - 16/11/25

This was our first full day regional competition, and we were all very excited to spend the day competing and working on our robot. This competition was great for the team to get used to the competition format, for example getting used to the interviews and alliance selection. It also allowed us all to practise driving and spend time on the competition field. We overcame many minor problems with the Hero Bot, and the process of coming up with solutions allowed us to get used to using our design cycle. We thoroughly enjoyed this competition and loved working as a team to overcome any problems we were faced with. We also had the opportunity to do skills (achieving a score of 20!) as well as spending lots of time on the practise field, allowing us to test our solutions (as part of our design cycle).

We used this competition to mostly get used to how the competitions run, allowing us to prepare better for the next one.

34071A - HG5_ELECTR!C

@ V5RC | Push Back | Radley Robot Rumble | (...)

★

Matches

P2	66618Z	23	5	34071A
10:41 AM	34071B			84899H
Q4	84899F	40	47	1408V
11:18 AM	1649E			34071A
Q8	34071B	5	58	17711C
11:42 AM	34071A			1408H
Q11	1408H	28	8	34071A
12:00 PM	11552A			84899A
Q13	84899A	75	20	1408C
12:12 PM	17711V			34071A
Q18	66618Z	21	70	84899F
1:41 PM	34071A			17711W
Q22	17711C	17	46	1649E
2:03 PM	34071A			84899D
Q28	1408G	18	48	17711V
2:36 PM	34071A			84899H
Q32	84899H	36	82	17711W
2:58 PM	11552A			34071A
QF 4-1	1408V	84	24	11552A
	1649E			34071A

Qualification Ranking

Current Rank

12

WP / AP / SP / W-L-T

6 / 30 / 165

3-5-0

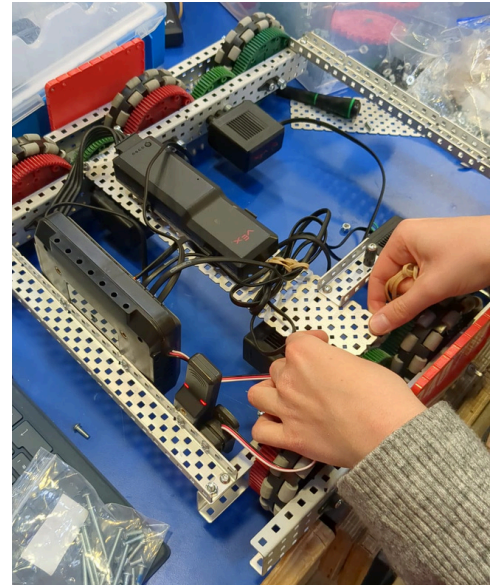
Cranleigh Christmas Regional -

06/12/25

Anika D: For this competition, many members of our team were unable to go due to varying circumstances, and it therefore fell to two of the coders of the team (me and Sophie) to attend the competition.

On the journey there, we had to make the decision to either use the Hero Bot or try and use the two sides of the drivetrain to create, in the most simplest terms, a moving piece of metal. We opted for the latter as we had brought the parts needed to create some sort of intake mechanism and believed that we could build enough of the robot for it to be functional at the competition... or so we thought.

It turned out that we had forgotten one side of the robot at school, and hadn't brought enough spare parts to build a new side to attach the sprockets to! We cobbled together the two parts of the drivetrain and went into the competition with a little less than a metal frame.



34071A - HG5_ELECTR!C					
@ V5RC Push Back Cranleigh Christmas Reg...					
Matches					
P2	18650E	39	13	18650D	34071A
9:36 AM	23218Y				
Q2	23218Y	73	25	34071A	18650E
10:06 AM	23218X				
Q4	1197Z	76	6	34071A	23218X
10:18 AM	46037D				
Q8	34071A	9	68	18650D	23218Y
10:42 AM	7975A				
Q11	1197X	53	0	10432A	34071A
11:00 AM	10432B				
Q14	34071A	34	52	10432B	71611A
11:18 AM	46037D				
Q19	34071B	29	71	18650D	71611A
11:48 AM	34071A				
Q21	7975A	5	76	46037D	1197Z
12:45 PM	34071A				
Q25	18650E	50	21	34071A	1197X
1:09 PM	34071B				
Qualification Ranking					
Current Rank		WP / AP / SP / W-L-T			
13		0 / 30 / 129			
		0-8-0			

We tried our hardest to keep up with the other robots in defending the goals and parking at the end of the match, but unfortunately, as the matches went on, the drivetrain started to come apart and disconnected from one of the motors, rendering one side useless. On the other side of the drivetrain, the rubber parts of the omniwheels were starting to fray due to the drivetrain incorrectly having the metal sides facing inwards and rubbing against the wheels when the motor spun.

In the end, not all of our alliances could make up for our robot and we ended up losing all of our matches as the drivetrain breaking down made up unable to even move as we rotated back and forth with one side of it working.

This competition was a learning experience for us that we shouldn't put ourselves in a vulnerable position if we aren't prepared for the consequences, and I think that knowing this was crucial for our decisions to come.

Stowe New Year Event - 16/01/26

For this competition, we planned to go as we wanted the chance to redeem ourselves from the one before, but due to our robot situation, we decided it was best not to go as, learning from last time, we weren't prepared and wanted to prioritise rebuilding our new robot, as the Hero Bot was now being disassembled for parts.

Garden City Gambit - 24/01/26

Tia: This was the first competition with our new robot - we had only finished building the day before, so we hadn't had time to test the conveyor mechanism on the field. I was the only team member able to attend, so this was also my first solo competition. As a builder, my expertise does not lie with programming, nor am I the driver/reserve driver for our team, so I was quite nervous to begin with. Despite this, I was keen to try out our robot and see how it performed, so I persevered and in the end had a great experience.

What I Learnt & Problems I Faced

- When I arrived at the competition, my first priority was to upload code onto the robot, as this had not been done yet. I have no experience with coding and was not familiar with the VEX software, but with remote help from Anika D, I managed to do this and realised it is very straightforward! I also edited parts of the code to adjust the intake and drivetrain speed, allowing me to gain an understanding of what each part of the code is responsible for
- Since I don't usually drive, I experienced some issues with running the code during matches. I feel like the first few matches were wasted due to technical difficulties, as I was unfamiliar with the controller layout and had no prior practice. In the first match I accidentally ran the wrong code, which was a slot used two seasons ago for our last robot! (this has since been deleted from the brain...)
- At this competition, I realised that the robot was too wide. Whilst it fit within the 18" restrictions, it was hard to manoeuvre it around the long goals and it kept tipping outside of the parking zone. When I tried to clear the parking zone during skills runs, the chassis kept getting stuck. However, I decided to use this sizing to my advantage; during matches, I took a defensive strategy and focused on preventing the other alliance from scoring. This proved to be very effective, as our robot is wide and heavy, making it hard to be pushed around and easy to block the opponents' path

←

34071A - HG5_ELECTR!C

@ V5RC | Push Back | Garden City Gambit | (M...

★

Matches

Q4

10:14 AM

20785Z

20785X

38

33

34071A

17711V

Q10

10:42 AM

34071A

3116G

8

75

2623K

17711C

Q19

11:30 AM

10478B

10478C

69

6

34071A

1408H

Q29

1:45 PM

46037X

34071A

0

103

3116E

1816X

Q32

1:59 PM

34071A

34294A

5

25

11667A

13765V

Q37

2:24 PM

1649E

17711W

93

13

34071A

34071B

R16 4-1

17711W

17711C

90

11

41908A

34071A

Qualification Ranking

Current Rank

26

WP / AP / SP / W-L-T

2 / 15 / 65

1-5-0

Skills

Skills Rank

23

Score / Attempts

5 / 3

0 / 0

Radley Robot Ruckus - 08/02/26

This was our last competition before nationals and this was the first time our newly built robot was put to proper use.

When testing our robot out on the practice field, we noticed that the blocks would occasionally shoot out of the robot between the first and second roller. Therefore we followed our design cycle:

Identifying the Problem

Block sometimes came out of robot when moving up.

Research & Analysis

We needed some sort of way to ensure the block wouldn't fall out when the motors turned. We considered that the gap between the first and second rollers were too big. However, we agreed that trying to amend this was too big of a job given our first match was approaching.

Brainstorming the Solution

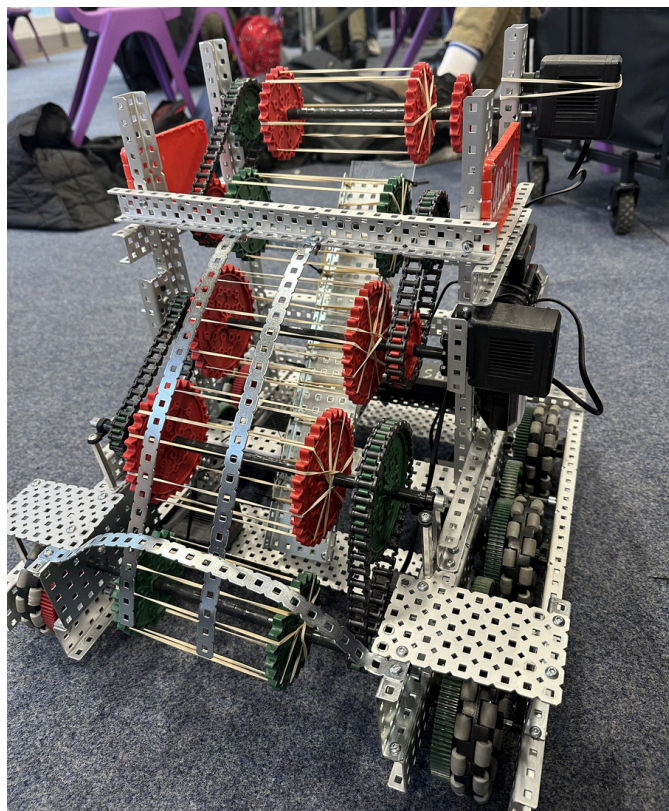
Tia came up with the idea of using pieces of malleable metal to somehow act as a barrier. We agreed that this was the best solution as it would be both quick and effective.

Selecting & Executing

We got two pieces of malleable metal (as seen in the picture) and bent them in alignment with the robot, screwing them in with nuts and bolts to the metal shaft.

Testing the Solution

We found these effective in holding the block in place, however sometimes they lifted up due to their malleable nature and we predicted that they would be easily bent/damaged in matches. Therefore we got



34071A - HG5_ELECTRIC					
@ V5RC Push Back Radley Robot Ruckus ...					
Matches					
P5	13765V	37	81	20785X	11850W
10:49 AM	34071A				
Q4	84899G	36	28	83697A	34071A
11:16 AM	10432A				
Q8	13765J	26	74	13765V	84899F
11:38 AM	34071A				
Q17	84899A	37	31	34071B	34071A
12:28 PM	55113B				
Q20	11850X	6	64	20785Z	34071A
1:41 PM	93568A				
Q26	34071A	3	38	13765Y	10478D
2:14 PM	33422U				
Q34	1408G	41	6	84899H	34071A
2:58 PM	13765Y				
Q40	34071A	52	36	11850X	55113A
3:31 PM	11850W				
R16 6-1	55113B	81	17	34071A	33422U
	84899A				
Qualification Ranking					
Current Rank			WP / AP / SP / W-L-T		
18			4 / 30 / 136		
			2.5.0		

another piece of metal and attached it horizontally to the barrier, which kept the pieces of metal in place. We found this temporary solution to be quick and efficient.

Overall Reflection

As a team, we decided to reflect on all of our regional competitions and consider what aspects needed improvement. One thing we decided on improving was our communication in the competition environment. We found that, despite having great communication when working on the robot at school, the competition environment often made us have to come up with solutions rapidly, due to having upcoming matches. We recognised that this had the potential to cause miscommunications and also made keep the logbook up to date during competition more difficult. Therefore we agreed that this would be something we worked on and ensure we were all in the loop of what was happening or what needed to be done with the robot. This means that more of the small details can be recorded in the logbook and also means that our brainstorming solutions part of the design cycle can be much more effective as everyone is on the same page of what the problem we are trying to solve is. We have put this into practice and have found our communication skills have improved hugely, especially between the programmers and builders as a lot of the mechanisms are interlinked.

We also decided that we should try autonomous skills rather than only driver control skills, especially considering half of our team is made up of programmers. Therefore, our programmers have been working hard on developing a code for autonomous skills, which we believe with a bit of practice we have the potential to excel in.

Another thing is that we have decided to stick to one driver for the entirety of the season. We believe that this is much better than sharing the role as it means our driver can get a lot more practice in and learn from any mistakes. We believe this was a very sensible decision and has allowed our driver to perfect their driving skills.

We believe we have worked really well as a team so far this season and have all been very involved in working on the robot, splitting jobs strategically between the team and giving each other lots of support.

Our team consists of 3 experienced coders, which we believe has given us an advantage over other teams in the programming aspect of VEX Robotics. We also believe our incorporation of C++ has given us an edge, and so we plan to use it lots more throughout our future programming.

Starting out with the Hero Bot this season was extremely useful for getting to grips with VEX Robotics, allowing us to flourish when building our new robot. Our team has picked up the skills needed for VEX Robotics so quickly and we are so proud to have qualified for nationals.

We cannot wait to put all of our plans into action at nationals and use all of our experience from regional competitions to do our absolute best.

Pre-Nationals Meeting - 12/02/26

We qualified for nationals with two weeks to go, so with the competition around the corner, our entire team got together for a meeting. We went through everything we needed to get done during half term and divided the tasks between us. Throughout this meeting, Sophie put together a table, separating all our tasks into four categories, as seen below. This was extremely helpful as it allowed us to have everything laid out in front of us and gave us clear goals on what needed to get done as well as what our main priorities were.

BUILD ↳ narrow robot ↳ pneumatics ↳ descorer + match loader ↳ add aligner ↳ drivetrain stability + speed ↳ intake speed	CODE ↳ auton ↳ right + left sides ↳ pneumatics ↳ descorer + match loader ↳ C++ ↳ motor speeds ↳ configure buttons ↳ drivetrain
NOTEBOOK ↳ move notebook to canva ↳ display board ↳ improve formatting ↳ team member roles ↳ setbacks + improvements ↳ new designs ↳ code evolution ↳ collate photos	OTHER ↳ driving practice ↳ main + backup ↳ read manual ↳ get pneumatics Rit ↳ task list in more detail ↳ team meeting + discussion

Our Jobs Leading Up to Nationals > team meeting notes:

Tia

- Narrowing the robot
- Pneumatics
- Adding aligner

Sophie

- Correcting motor speeds
- Researching pneumatics code
- Auton for both sides
- Helping add to logbook

Sanjana

- Display board

- Helping add to logbook

Isabel

- Working on the logbook
- Reread game manual and come up with game strategy

Anika G

- Translate Python code for both the driver-controlled and autonomous sections into C++
- Reformat logbook into Canva and continue adding to it

Anika D

- Getting in driving practice
- Recoding auton and adding in pneumatics controls
- Working on the logbook

Preparing for Nationals

Now having decided what our tasks were, we began to make a list of tasks and allocated each team member to one to ensure everyone understood their part in contributing to our success.

Reading the Game Manual

Understanding the VEX Push Back game manual was necessary if we were to do well at nationals. We recognized the importance of everyone being familiar with its contents to bring up rules that could have been broken during a match.

First, we shared a link to the game manual and made sure that everyone got to grips with the main game, scoring and skills rules before nationals.

After this, we checked that everyone had installed the app V5RC Hub as this would give us all access to a manual on the go as well as a point calculator to ensure that the team was in agreement with the referees when calculating the final scores of a match. This also had the other benefit of growing our knowledge of the scoring rules as the confirmation given from the referees would show that we had scored ourselves accurately.

Notebook (Everyone)

- Before we come back from half term, have notebook up to date with everything we been doing so far
- After half term, update on latest modifications

Display (Sanjana)

- Physical board
- Photo collage of build progress, annotations etc.

Build (Tia/Isabel/Sanjana)

1. Narrow robot
2. Match loading mechanism
3. Descorer
4. Goal alignment mechanism
5. Change drivetrain motor cartridges to blue
6. Pack drivetrain with spacers and tighten
7. Top flap for scoring
8. Tighten all screws and health check before

Coding (Anika D/Sophie/Anika G)

- Write code for pneumatics

- Increase drivetrain speed
- Ensure auton code is working as expected
- Translate to C++

Narrowing the Robot

Identifying the Problem

Throughout the regional competitions, we observed that our rebuilt robot was too wide, causing the robot to be much slower on the field. However, it passed inspection, so we concluded as a team that it was too big of a job to do during one of the competitions. We acknowledged it wouldn't be a quick fix so Tia took the robot home to narrow it, which allowed her to take the robot apart more carefully and be able to put it back together without any problems occurring (lost parts etc.). One of the biggest problems was the struggle for our robot to park. We hoped that narrowing the robot would make parking, as well as just manoeuvring around the field a lot easier. We especially struggled moving past the long goals as the dimensions of the robot meant that the path was too narrow for the robot to pass through smoothly, wasting time that could have been spent scoring/defending. Narrowing the robot also allows us to incorporate a descoring mechanism, which we have been wanting to do for some time. Before narrowing, we wouldn't have had the space on the robot to do so.

Research/Analysis

At regional competitions, we observed that the robots best at parking were the narrower robots. We also observed how effective the descorer mechanism was on other robots, however we were unable to incorporate this mechanism due to the width of our robot exceeding the dimensions limit. Therefore, it was pretty clear that the solution was to narrow the robot. The difficult part was figuring out how to do so, without changing other parts of the robot.

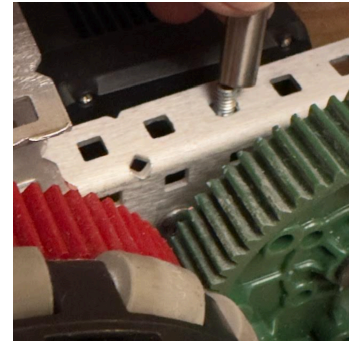
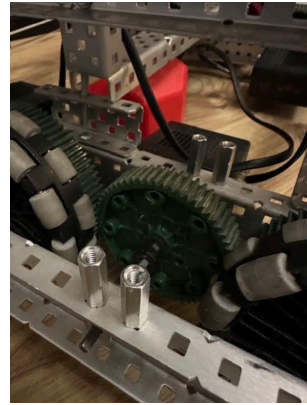
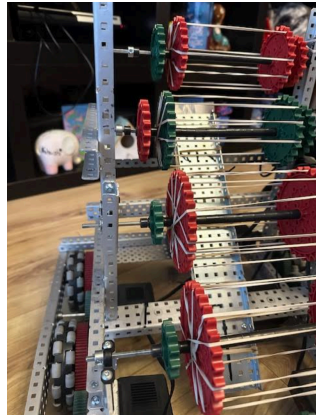
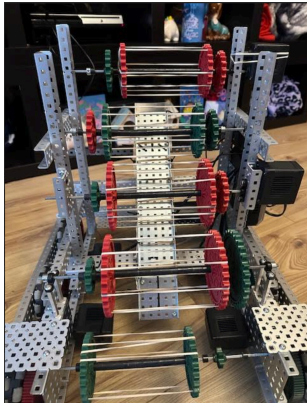
Brainstorming Solutions

After spending a lot of time analysing the robot as a team, we concluded the most effective thing to do was to remove spacers. Sophie also had the idea of moving the vertical shafts attached to the drivetrain from the inner side of the wheels to the outer side, as there was empty space between the wheels of the drivetrain.

Selecting & Executing

- Removed most spacers from conveyor
- Removed channels connecting both sides of the drivetrain, which have been moved further forward (gears were in the way so this was attached by being raised on standoffs); this reduced the space between the wheels and the channels
 - Attaching the standoffs was tricky as a motor was in the way, so a magnet was used to pull small screws through hard-to-reach holes
- Narrowed robot by nearly 2"

- This was because it did not need to be so wide and due to the difficulties we had parking and manoeuvring around the field (especially around long goals). Also, we were just within the official size restrictions, so narrowing it allowed us more space to add a descoring mechanism



Testing the Solution

- Now, a gap has been created at the back of the drivetrain, where we can later add a triangular shaped component for easy alignment with long goals
- The original bar was limiting the conveyors proximity with the goal, making scoring less efficient
- We managed to effectively narrow the robot from 18 inches to 15 inches
- We practised parking and found it a lot easier with a narrow robot

Adjustments

Adjusting Back of Robot

We removed the c-channel at the back which held drivetrain together, and moved this further forward (as gears were in the way so we attached this raised on standoffs). Attaching these standoffs was tricky as the motor was in the way so we used a magnet to pull small screws through hard-to-reach holes).

Now there's a gap at the back of the drivetrain where we can later add a triangular shaped component for easy alignment with long goals. The original bar was limiting the conveyors proximity with the goal, making scoring less efficient.

Adjusting the Conveyor

After careful consideration of some of the problems with the conveyor and intake system, we brainstormed ideas like shifting rollers around, possibly adding more to reduce gaps between them, and began to execute this idea as follows:

We moved the top three rollers further forward to reduce gaps in conveyor (these gaps made intaking slower and balls were more prone to flying out between them).

This also made the conveyor slightly steeper and generated more space at the back of the bot; we intend to add an extra roller at the top to help with scoring in the long goals, so now there is horizontal space for this.

We hope adding this extra roller, as well as an alignment component, will help us score quicker and more efficiently, during matches, as it had when we tested the system at school.

Finishing the Conveyor

- We cut 1 inch off the axles using a hacksaw, and filed them using a metal file for safety
 - For quality control, we marked this out using a ruler and sharpie
- Then we put spacers back into the rollers and tightened the shaft collars

Before

After

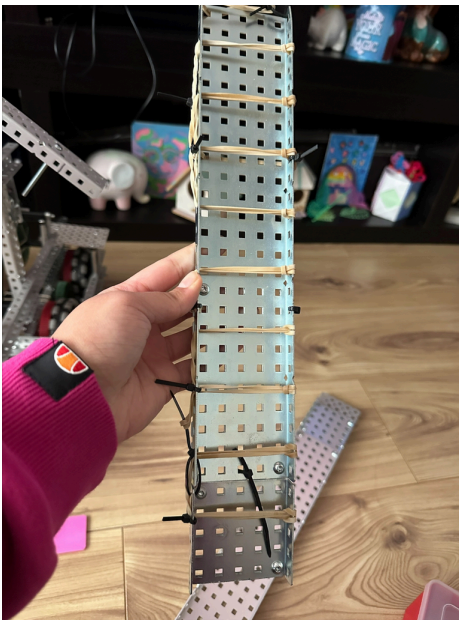


Centre Ball Channel

We remade the channel the balls travel up - it was originally made out of steel, but this was very very heavy so we made one out of aluminium instead (there's a small piece of steel since we didn't have any more of the right size).

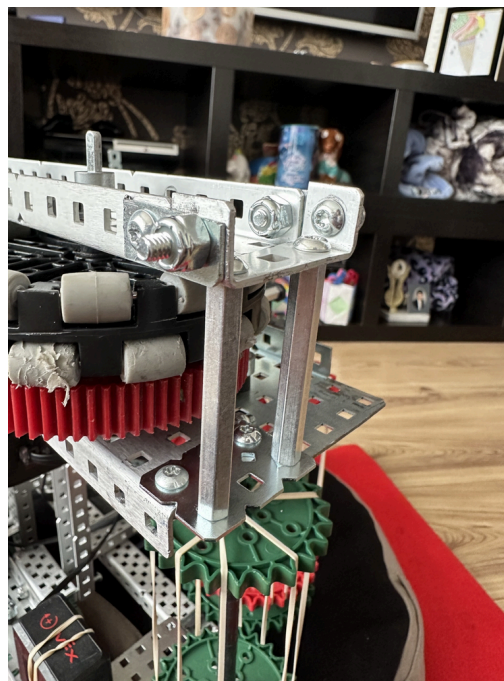
This needs to be tested again without the elastic bands added to it to see how much of a difference it would make. The reduced friction could potentially allow more balls to travel up the channel without gaps, preventing balls from being released when more are intaken.

We also removed the temporary polycarbonate attached to extend it and extended it with metal instead to make it more cohesive.



Extending the Chassis

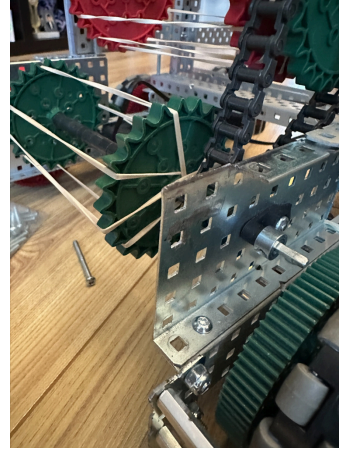
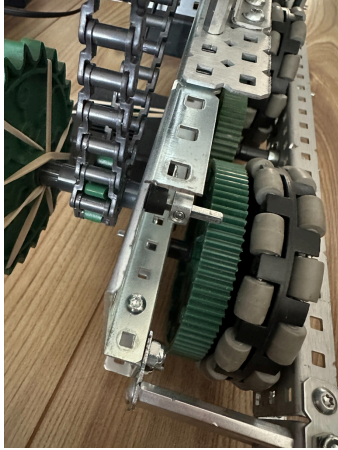
- Currently, the rear end of the chassis where the wheels are sandwiched between two c-channels are held together by horizontal standoffs to prevent flexing. The front end is held together by a flat piece of metal on top of two vertical standoffs
- We wanted to change this, as this structure is going to obstruct the match loader we plan to add. We cannot put horizontal standoffs in place right now as there is no space.
- Ideally, we would move wheels, motors and gears one space back, however this is time consuming and would require taking the drivetrain apart again
- Instead, we cut 1 x 3 pieces off the rear end and attached it to the front, then held it together with standoffs as desired. This was done using a multitool as it was quicker and more efficient than using the hacksaw



Cutting Other Pieces

- Whilst the multitool was out, we cut some other pieces to save time later

- 2 x 3 flat pieces x4 (used in extending of the chassis)
- 2 x 12 c-channels x2 : this is for the alignment triangle for the back of the robot to help with scoring in long goals
- 2 x 11 c-channels x2 : these are the arms for the match loader
- When we initially narrowed the robot, the channel holding the drivetrain together was too long and was significantly overhanging so cut this
- There is a wide channel at the front of the robot which holds the first intake roller so we cut 3 holes off the overhanging part : this creates space for the match loader to be retracted and rested at 90°



Health & Safety



- Goggles and dust mask to prevent metal filings entering eyes and lungs
- Gloves to prevent skin irritation from metal filings
- Laid down sheet to prevent filings getting on floor and causing harm later
- For the pieces cut whilst on the robot, we covered the motors with tape >> stop filings entering motors, which could cause jamming or overheating due to extra abrasion internally
- Used sandpaper to file everything down to avoid getting injured on sharp edges

Quality Control

- Marked out pieces to be cut using a steel rule and masking tape

Drivetrain Edits

Modifying the Drivetrain Step by Step

How We Changed the Old Drivetrain

After brainstorming ideas among the engineers of our group we decided it was best to start on the drivetrain one side at a time to eliminate need for taking whole robot apart, saving time and increasing efficiency through quick planning.

We could then begin the execution:

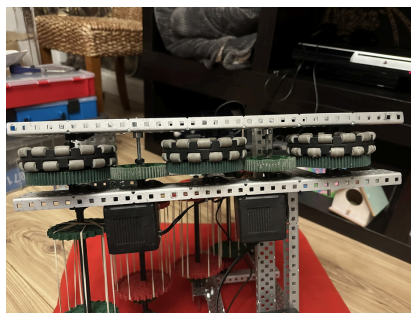
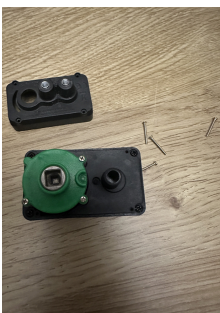
- We first removed the gear and wheels, then moved bearings to outside of the c-channels, and this was crucial to allow for further narrowing, as the channels can be closer to the wheels
- We changed the green motor cartridges to blue, this increased the speed of the drivetrain
- We further secured motors with rubber bands for stability and to prevent flexing
- Next, we added a rubber shaft collar between wheels and gears, this was due to the fact that they were too close together, so the gears were shredding the rubber of the omni-wheels
- We packed each shaft with spacers to prevent movement (we also added shaft collars inside channel of gears to stop them coming out the motors)
- Additionally, we added standoffs to the rear to hold both channels in place; this was difficult to implement in the front as there is was no space, but the channels are held together by raised platform on standoffs

Drivetrain Calculations

- Driving gears: 60T
- Driven gears: 84T
- Gear ratio: $84/60 = 1.4 \gg 7:5$

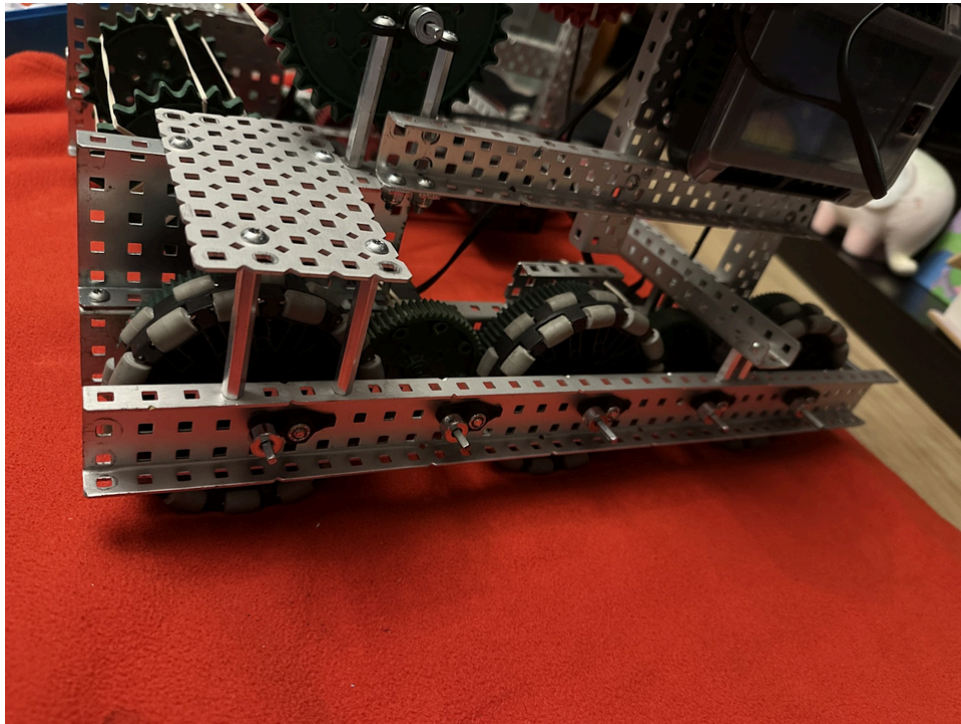
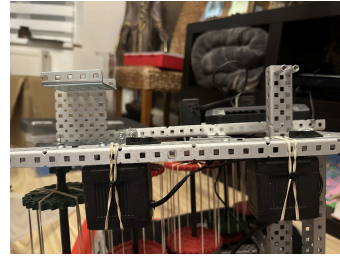
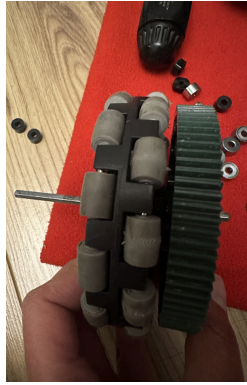
Previous

- Motor RPM: 200
- Output RPM: $200/1.4 = 143$



New

- Motor RPM: 600
- Output RPM: 429



These metrics could be too fast, but can be adjusted via the code. This change gives us more room for experimentation.

Adding Pneumatics - Descorer

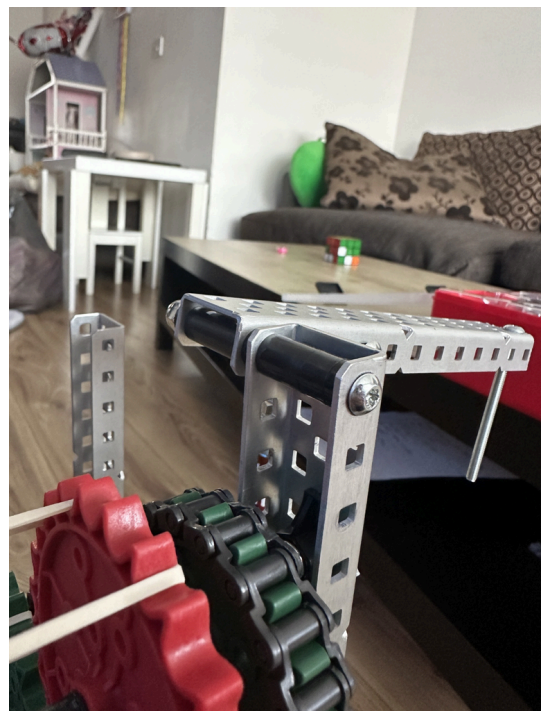
At previous competitions we found that matches between the best robots often ended in quite low scores and win margins - this is because most of the balls got removed from the goals. Learning from this, we decided to add a descorer as it is a great strategic component. It can be also be used for pushing our balls into the goal and into the control zone. This analysis led us onto brainstorming solutions:

We started by experimenting with the three pneumatic cylinder sizes, ultimately deciding on the smallest one. This is because the descorer mechanism is quite short and requires little movement. The smallest cylinder will also use less air, which is beneficial as we will be using a single air tank.

The Execution

- To make the mechanism, we attached a short c-channel with a long screw at the end to the rear top of the robot. The rest position will be at 90°, and it will shoot upwards to position it above goal before bringing it back down to the rest position again
- It has been built packed with spacers for stability; the mini wing had to be strong in case of contact with another robot (with an important consideration that if engaged when descoring, a lot of stress could be put on this joint)
- We then attached the cylinder with a shaft collar, and secured it to the robot using a long screw and nyloc nut; this was attached loosely to allow movement of the cylinder

Field element dimensions were kept in mind when making this mechanism, and later, additional testing was carried out on the actual field when we got back to school.



Testing the Solution

Upon return to school after half term, we encountered an issue with this design. Anika D tried using the descorer during driving practice.

Identifying the Problem

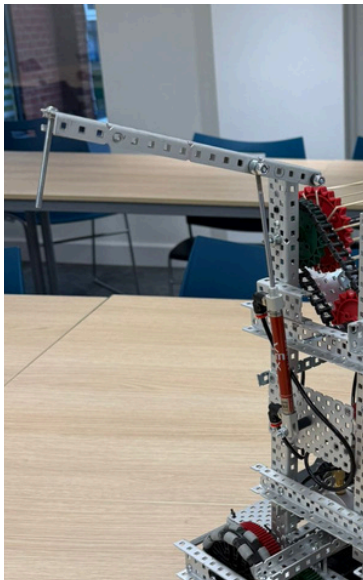
The descorer was too short; when driving forward with the descorer in the long goal, the drive train gets stuck on the base of the goal.

Analysis & Brainstorming Solutions

One solution could be to replace the c-channel with a longer one. However, this would result in the width of the robot being out of the 18" starting size. We also considered beginning the match with the descorer in the 'up' position, however this poses the same issue. Changing the position of the cylinder to begin matches in a 'down' position could also be an option, however this would mean the channel only extends to 90 degrees, making it harder to get the screw into the descorer.

Selecting and Executing

The solution we have decided on, is lengthening the c-channel, but also replacing the smallest cylinder with the largest. This allows the descorer to start matches in a 'down' position, but extend upwards at an obtuse angle to be positioned in the goal.



Coding Pneumatics

To figure out how to code pneumatics, research was important so that we could easily find the best way to write the code in both Python and C++. The VEX website was used for instructions on how best to code pneumatics using Python, and a base code was created which could be edited when the ports and buttons were finalised. This meant that both the new intake mechanism for the match loader and the descorer would have set buttons which were easy to access and well understood for driving.

Python:

```
1      # configure pneumatics
2      descorer = DigitalOut(brain.three_wire_port.a)
3      match_loader = DigitalOut(brain.three_wire_port.b)
4
5      if controller.buttonL2.pressing():
6          # cylinder up for descorer
7          descorer.set(True)
8      elif controller.buttonL1.pressing():
9          # cylinder down for descorer
10         descorer.set(False)
11
12     if controller.buttonDown.pressing():
13         # cylinder up for loader
14         match_loader.set(True)
15     elif controller.buttonUp.pressing():
16         # cylinder down for loader
17         match_loader.set(False)
```

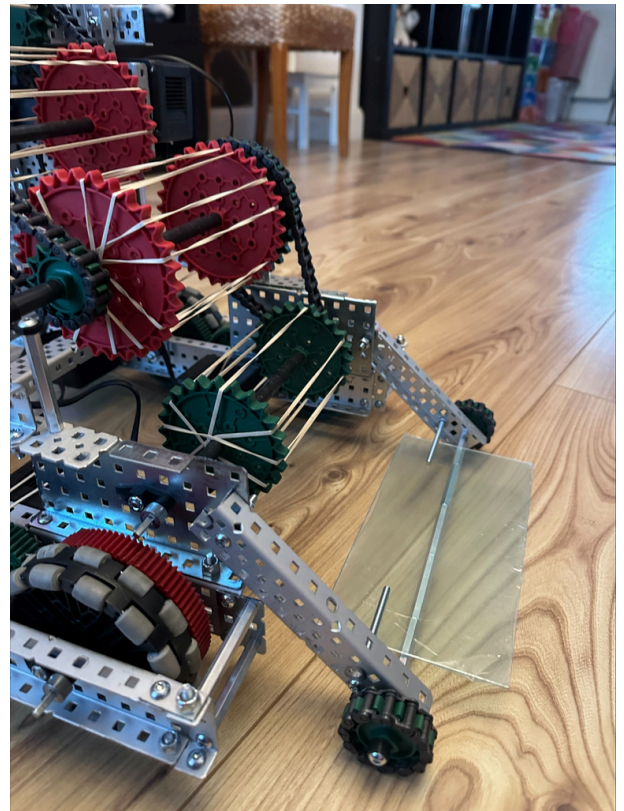
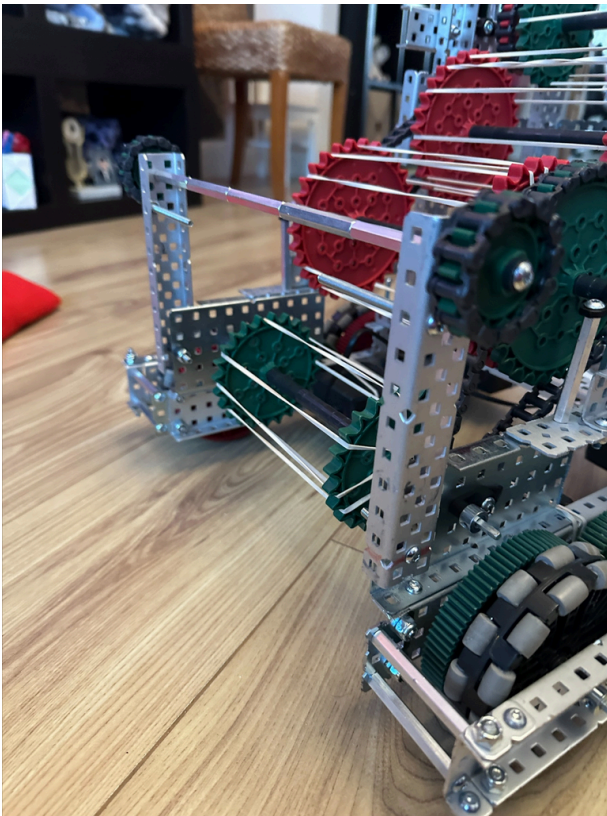
C++:

```
1 // Configure pneumatics
2 digital_out descorer = digital_out(Brain.ThreewirePort.A);
3 digital_out match_loader = digital_out(Brain.ThreewirePort.B);
4
5 // Moving the descorer
6 if (Controller1.ButtonL2.pressing()) {
7     // Descorer up
8     descorer.set(true);
9 }
10 else if (Controller1.ButtonL1.pressing()) {
11     // Descorer down
12     descorer.set(false);
13 }
14
15 // Moving the loader
16 if (Controller1.ButtonDown.pressing()) {
17     // Loader down
18     match_loader.set(true);
19 }
20 else if (Controller1.ButtonUp.pressing()) {
21     // Loader up
22     match_loader.set(false);
23 }
```


New Loader Mechanism

We wanted to add a match loading mechanism since taking balls from tubes to score in goals is more efficient and gives us more balls to work with rather than just ones on the field

- Two c-channels cut earlier are the arms
- Made a shaft out of standoffs attached end on with couplers >> more rigid and strong than using a regular shaft
- Attached to end of arms by screws >> added small sprockets with links on them at either end >> less friction so can glide over floor easier and also to prevent damage to the field
- Need to add polycarbonate sheet. Screws restrict polycarb to a specific resting angle >> just above lip of match loader. When pushed against the bottom of the ball, it will rotate along the shape of the match loader and make a mini ramp for the balls to fall out from
- A midsize pneumatic cylinder will be attached to the left arm, with its resting position being 90°



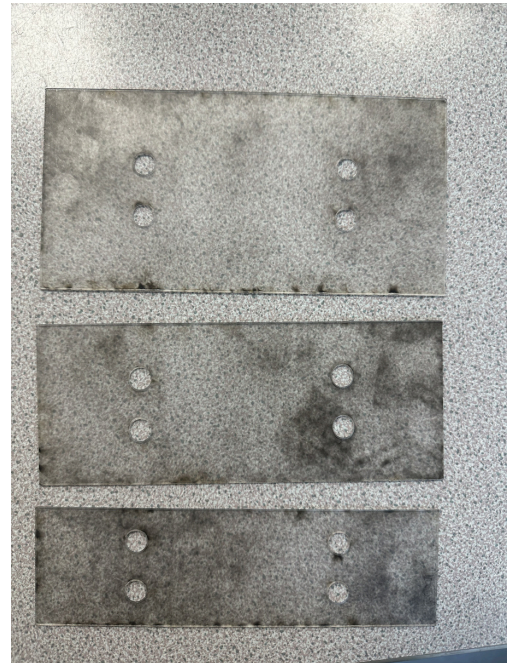
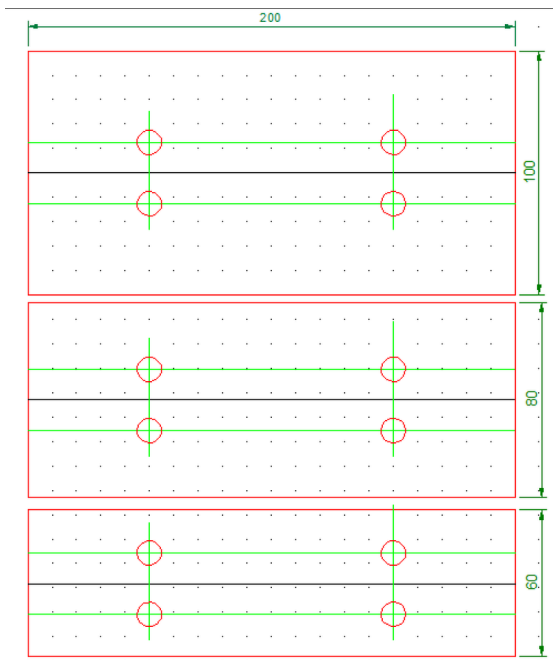
Testing the Solution & Identifying the Problem - Plastic

Our current polycarb sheets measures 200x100mm, however when testing by hand it seems this may be too wide, since the balls sometimes get stuck in the loader.

Brainstorming Solutions

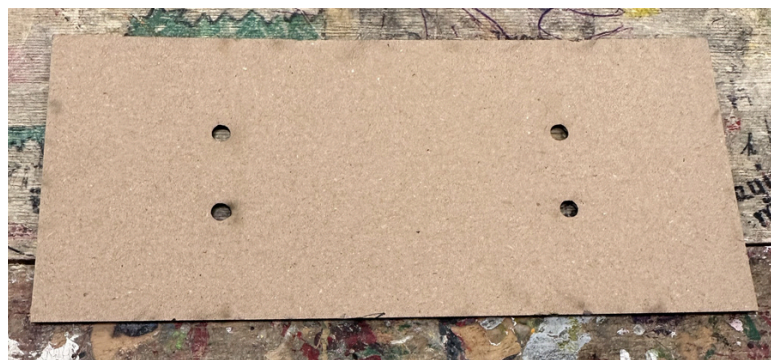
To allow the balls to fall out the loader smoothly and without getting stuck, we wanted to experiment with different sizes to find the optimal dimensions. In terms of resources, we only have two 200x100mm polycarbonate sheets, but we have a lot of acrylic plastic. Therefore, we decided to experiment with these acrylic sheets so not to waste the polycarb which we will be competing with.

To save time and make our measurements more precise, Tia designed three iterations of plastic sheets on TechSoft 2D design and cut them using a CNC laser cutter. The CAD is shown below:



Selecting and Executing

We tested the different widths of acrylic by hand at the mouth of the loading tube. 60mm was too small and didn't create enough of a 'ramp' for the balls to fall down. The 100mm one was still causing issues after multiple tests, but the 80mm worked best. However, there were a few instances where the balls didn't fall out, suggesting 80mm was still slightly narrow. Therefore, we decided to test again using 200x90mm cardboard (cardboard is cheaper and quicker to laser cut - the clear acrylic was melting slightly and burning).



We tested this and found it to be the perfect size. Finally, Tia cut the polycarbonate sheet to the right dimension.

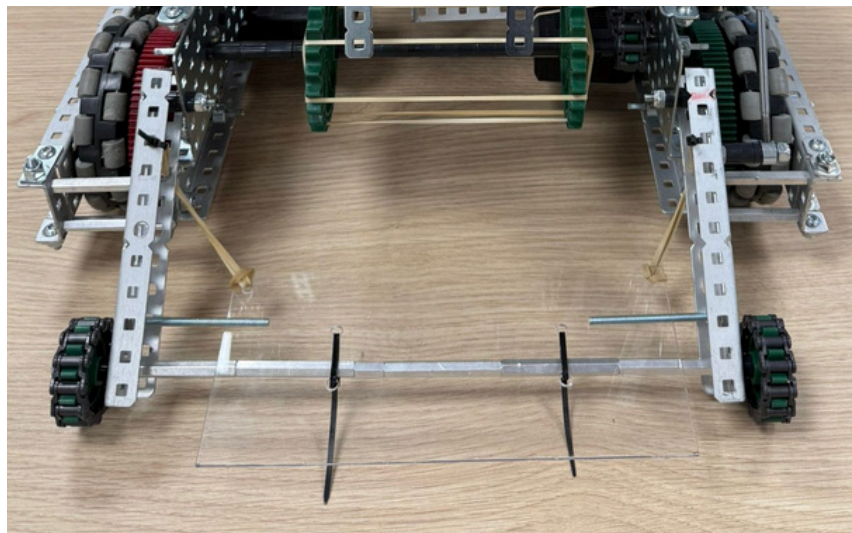
Quality Control

She used the cardboard as a stencil, since the dimensions were very accurate as it was cut using the CNC laser cutter.

Health and Safety

The sheet was cut using a band saw and a hand drill, as putting polycarbonate in a laser cutter can release toxic fumes.

Lastly, we attached the plastic sheet to the end of the match loader using zip ties. Furthermore, we secured the back of the plastic to the arms using rubber band to ensure it always returns to the optimal resting angle when removed from the loader.



Updated Driver Code (Right)

Below is the C++ code beginning on the right side of the field. This is important for the autonomous section because we have different strategies depending on which side of the field we start on.

```
1  /*-----*/
2  /*                                          */
3  /*   Module:      main.cpp                */
4  /*   Project:     VEX Competition Code    */
5  /*   Author:      HG5_ELECTRIC           */
6  /*   Created:     02/12/2025             */
7  /*   Configuration: Right                */
8  /*                                          */
9  /*-----*/
10
11  #include "vex.h"
12
13  using namespace vex;
14
15  /*-----*/
16  /*                      Robot Configuration      */
17  /*-----*/
18
19  brain Brain;
20
21  motor left_motor_a  = motor(PORT1, ratio36_1, false);
22  motor left_motor_b  = motor(PORT3, ratio36_1, false);
23  motor right_motor_a = motor(PORT2, ratio36_1, true);
24  motor right_motor_b = motor(PORT4, ratio36_1, true);
25
26  motor_group left_drive_smart  = motor_group(left_motor_a, left_motor_b);
27  motor_group right_drive_smart = motor_group(right_motor_a, right_motor_b);
28
29  // Drivetrain: wheel circumference=319.19mm, track=295mm, wheelbase=40mm
30  drivetrain Drivetrain = drivetrain(left_drive_smart, right_drive_smart, 319.19, 295, 40, mm, 1);
31
32  motor conveyor_motor_a = motor(PORT5, ratio18_1, false);
33  motor conveyor_motor_b = motor(PORT6, ratio18_1, false);
34  motor_group conveyor   = motor_group(conveyor_motor_a, conveyor_motor_b);
35
```

```

36     controller Controller = controller(primary);
37
38     // Competition instance
39     competition Competition;
40
41     /*-----*/
42     /*                      Pre-Autonomous                      */
43     /*-----*/
44  ✓ void pre_autonomous() {
45         Brain.Screen.clearScreen();
46         Brain.Screen.print("Waiting for autonomous to begin...");
47         wait(1, seconds);
48     }
49
50     /*-----*/
51     /*                      Autonomous                      */
52     /*-----*/
53  ✓ void autonomous() {
54         Brain.Screen.clearScreen();
55         Brain.Screen.print("Autonomous code running...");
56
57         // Auton code - start facing right wall
58         Drivetrain.driveFor(reverse, 125, mm);
59         Drivetrain.turnFor(right, 80, degrees);
60         Drivetrain.driveFor(reverse, 690, mm);
61
62         conveyor.setVelocity(100, percent);
63         conveyor.spinFor(reverse, 1250, degrees);    // load 3
64
65         Drivetrain.turnFor(right, 45, degrees);
66         Drivetrain.driveFor(reverse, 110, mm);
67         Drivetrain.turnFor(right, 10, degrees);
68
69         conveyor.spinFor(forward, 3240, degrees);    // score 3 balls
70     }
71

```

```

72  /*-----*/
73  /*                      User Control                      */
74  /*-----*/
75  ✓ void user_control() {
76      Brain.Screen.clearScreen();
77
78      while (true) {
79          // Tank drive - Axis3 = left stick vertical, Axis2 = right stick vertical
80          int leftSpeed  = Controller.Axis3.position();
81          int rightSpeed = Controller.Axis2.position();
82
83          left_motor_a.setVelocity(leftSpeed, percent);
84          left_motor_a.spin(reverse);
85          left_motor_b.setVelocity(leftSpeed, percent);
86          left_motor_b.spin(reverse);
87
88          right_motor_a.setVelocity(rightSpeed, percent);
89          right_motor_a.spin(reverse);
90          right_motor_b.setVelocity(rightSpeed, percent);
91          right_motor_b.spin(reverse);
92
93          wait(5, msec);
94
95          // Conveyor belt control
96          conveyor.setVelocity(100, percent);
97          if (Controller.ButtonR1.pressing()) {
98              conveyor.spin(forward);          // pick up / place into high tube
99          } else if (Controller.ButtonR2.pressing()) {
100              conveyor.spin(reverse);          // place into lower tube
101          } else {
102              conveyor.stop();
103          }
104

```



```

105         // Descorer control
106         if (Controller.ButtonL2.pressing()) {
107             descorer.set(true);    // close
108         } else if (Controller.ButtonL1.pressing()) {
109             descorer.set(false);   // open
110         }
111
112         // Match loader control
113         if (Controller.ButtonDown.pressing()) {
114             match_loader.set(true); // open
115         } else if (Controller.ButtonUp.pressing()) {
116             match_loader.set(false); // close
117         }
118
119         wait(20, msec);
120     }
121 }
122
123 /*-----*/
124 /*                      Main                      */
125 /*-----*/
126 ✓ int main() {
127     // Initialize random seed
128     srand((int)(Brain.Battery.voltage(voltageUnits::mV)
129         + Brain.Battery.current(currentUnits::amp) * 100
130         + Brain.Timer.systemHighResolution()));
131
132     wait(200, msec);
133     Brain.Screen.clearScreen();
134
135     // Register competition callbacks
136     Competition.autonomous(autonomous);
137     Competition.drivercontrol(user_control);
138
139     // Run pre-autonomous
140     pre_autonomous();
141
142     // Idle loop - competition switch drives autonomous/user_control callbacks
143     while (true) {
144         wait(100, msec);
145     }
146
147     return 0;
148 }

```

Updated Driver Code (Left)

Below is the C++ code beginning on the left side of the field. This is important for the autonomous section because we have different strategies depending on which side of the field we start on.

```
1  /*-----*/
2  /*                                          */
3  /*  Module:      main.cpp                  */
4  /*  Project:     VEX Competition Code      */
5  /*  Author:      HG5_ELECTRIC              */
6  /*  Created:     02/12/2025                */
7  /*  Configuration: Left (HAS NOT BEEN TESTED) */
8  /*                                          */
9  /*-----*/
10
11  #include "vex.h"
12
13  using namespace vex;
14
15  /*-----*/
16  /*                      Robot Configuration      */
17  /*-----*/
18
19  brain Brain;
20
21  motor left_motor_a  = motor(PORT1, ratio36_1, false);
22  motor left_motor_b  = motor(PORT3, ratio36_1, false);
23  motor right_motor_a = motor(PORT2, ratio36_1, true);
24  motor right_motor_b = motor(PORT4, ratio36_1, true);
25
26  motor_group left_drive_smart  = motor_group(left_motor_a, left_motor_b);
27  motor_group right_drive_smart = motor_group(right_motor_a, right_motor_b);
28
29  // Drivetrain: wheel circumference=319.19mm, track=295mm, wheelbase=40mm
30  drivetrain Drivetrain = drivetrain(left_drive_smart, right_drive_smart, 319.19, 295, 40, mm, 1);
31
32  motor conveyor_motor_a = motor(PORT5, ratio18_1, false);
33  motor conveyor_motor_b = motor(PORT6, ratio18_1, false);
34  motor_group conveyor   = motor_group(conveyor_motor_a, conveyor_motor_b);
35
```

```

36     controller Controller = controller(primary);
37
38     // Competition instance
39     competition Competition;
40
41     /*-----*/
42     /*                      Pre-Autonomous                      */
43     /*-----*/
44  ✓ void pre_autonomous() {
45         Brain.Screen.clearScreen();
46         Brain.Screen.print("Waiting for autonomous to begin...");
47         wait(1, seconds);
48     }
49
50     /*-----*/
51     /*                      Autonomous                      */
52     /*-----*/
53  ✓ void autonomous() {
54         Brain.Screen.clearScreen();
55         Brain.Screen.print("Autonomous code running...");
56
57         // Auton code - start facing left wall
58         // HAS NOT BEEN TESTED
59         Drivetrain.driveFor(reverse, 125, mm);
60         Drivetrain.turnFor(left, 80, degrees);
61         Drivetrain.driveFor(reverse, 690, mm);
62
63         conveyor.setVelocity(100, percent);
64         conveyor.spinFor(reverse, 1250, degrees);    // load 3
65
66         Drivetrain.turnFor(right, 80, degrees);
67         Drivetrain.driveFor(reverse, 100, mm);
68         Drivetrain.turnFor(left, 12, degrees);
69
70         conveyor.spinFor(reverse, 3240, degrees);    // score 3 balls
71     }
72

```

```

73  /*-----*/
74  /*                      User Control                      */
75  /*-----*/
76  ✓ void user_control() {
77      Brain.Screen.clearScreen();
78
79      while (true) {
80          // Tank drive - Axis3 = left stick vertical, Axis2 = right stick vertical
81          int leftSpeed = Controller.Axis3.position();
82          int rightSpeed = Controller.Axis2.position();
83
84          left_motor_a.setVelocity(leftSpeed, percent);
85          left_motor_a.spin(reverse);
86          left_motor_b.setVelocity(leftSpeed, percent);
87          left_motor_b.spin(reverse);
88
89          right_motor_a.setVelocity(rightSpeed, percent);
90          right_motor_a.spin(reverse);
91          right_motor_b.setVelocity(rightSpeed, percent);
92          right_motor_b.spin(reverse);
93
94          wait(5, msec);
95
96          // Conveyor belt control
97          conveyor.setVelocity(100, percent);
98          if (Controller.ButtonR1.pressing()) {
99              conveyor.spin(forward);          // pick up / place into high tube
100      } else if (Controller.ButtonR2.pressing()) {
101              conveyor.spin(reverse);          // place into lower tube
102      } else {
103          conveyor.stop();
104      }
105

```

```
106         // Descorer control
107         if (Controller.ButtonL2.pressing()) {
108             descorer.set(true);    // close
109         } else if (Controller.ButtonL1.pressing()) {
110             descorer.set(false);    // open
111         }
112
113         // Match loader control
114         if (Controller.ButtonDown.pressing()) {
115             match_loader.set(true);    // open
116         } else if (Controller.ButtonUp.pressing()) {
117             match_loader.set(false);    // close
118         }
119
120         wait(20, msec);
121     }
122 }
123
```

```

124  /*-----*/
125  /*                               Main                               */
126  /*-----*/
127  ✓ int main() {
128      // Initialize random seed
129      srand((int)(Brain.Battery.voltage(voltageUnits::mV)
130          + Brain.Battery.current(currentUnits::amp) * 100
131          + Brain.Timer.systemHighResolution()));
132
133      wait(200, msec);
134      Brain.Screen.clearScreen();
135
136      // Register competition callbacks
137      Competition.autonomous(autonomous);
138      Competition.drivercontrol(user_control);
139
140      // Run pre-autonomous
141      pre_autonomous();
142
143      // Idle loop - competition switch drives autonomous/user_control callbacks
144      while (true) {
145          wait(100, msec);
146      }
147
148      return 0;
149  }

```