

Engineering Notebook

3249V

Team Number



Infrared

Team Name

iTech Preparatory

School

5/02/2022

Start Date

01/20/2023

End Date

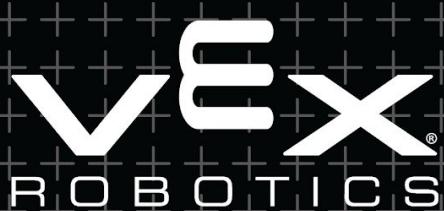
1

of 2

Book #

NOTEBOOK #2

v1.0.8.29.22



Resources

students.vex.com

Engineering Resources, Information on Notebooks, Videos, VEX Library, Teams Resources, and Scholarships



mentors.vex.com

Team and Mentor Resources, Mentor Professional Development, VEX Mentor Community and more



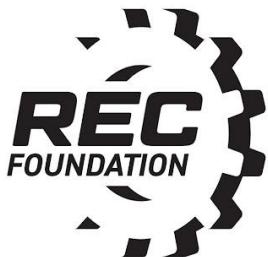
teams.vex.com

A Collection of Resources for Teams Provided by the REC Foundation



library.vex.com

Information on Building, Documentation, Troubleshooting, Coding, and other Educational Resources



About the REC Foundation

The REC Foundation's global mission is to provide educators with hands-on, student-led competition programs and educational resources to prepare future innovators for a diverse and inclusive STEM workforce. We see a future where all students design and innovate as part of a team, experience failure, persevere, and emerge confident in their ability to meet global challenges.

engineering.vex.com
notebooking.vex.com
coding.vex.com

Judging Rubric for Notebooks

vex.com
roboticseducation.org
[VRC 2022-2023 Game - Rules & Game Video](https://www.vex.com/vrc-2022-2023-game-rules-and-game-video)

Send suggestions and comments about these Digital Notebooks and Digital Parts to notebooks@vex.com



Table of Contents

Page	Linked Project Slides	Date
1	: Introductions: Koltin Hoffman	5/2/22
2	: Introductions: Aidan McCallum	5/3/22
3	: Introductions: Tyler Fields	5/5/22
4	: Introductions: Ethan Regan	5/6/22
5	: Introductions: Joshua Wilson	5/6/22
6	: Team Meeting#1	5/26/22
7	: Team Meeting #1 Continued	5/26/22
8	: Team Meeting #2	6/7/22
9	: Team Meeting #2 Continued	6/7/22
10	: Team Meeting #3	6/9/22
11	: Team Meeting #3 Continued	6/9/22
12	: Design process Explanation	6/10/22
13	: Design Process Explanation Continued	6/13/22
14	: Game Summary Section 1	8/31/22
15	: Game Summary Section 2	9/5/22
16	: Game Summary Section 2 Part 2	9/6/22
17	: Game Summary Section 2 Part 3	9/8/22
18	: Game Summary Section 2 Part 4	9/8/22
19	: Game Summary Section 3	9/9/22
20	: Game Summary Section 4	9/13/22
21	: Game Summary Section 4	9/13/22
22	: Team Goals	9/14/22
23	: Aidan McCallum's Research	9/14/22
24	: Aidan McCallum's Research Part 2	9/14/22
25	: Aidan McCallum's Research Part 3	9/15/22
26	: Aidan McCallum's Research Part 4	9/15/22
27	: Meeting #4	9/21/22
28	: Meeting #4 Continued	9/21/22
29	: Koltin Hoffman's Research	9/21/22
30	: Koltin Hoffman's Research Continued	9/21/22
31	: Meeting #5	9/22/22
32	: Meeting #5 Continued	9/22/22
33	: Ethan Regan's Research	9/22/22
34	: Ethan Regan's Research Continued	9/22/22
35	: Pre-Coding	10/2/22
36	: Pre-Coding: Odometry	10/2/22
37	: Meeting #6	10/4/22
38	: Meeting #6 Continued	10/4/22
39	: Tyler Field's Research	10/4/22
40	: Pre-Coding: Odometry Part 2	10/5/22



Table of Contents

Page	Linked Project Slides	Date
41	Meeting #7	10/5/22
42	Meeting #7 Continued	10/5/22
43	Josh Wilson's Research	10/5/22
44	Pre-Coding: Odometry Part 3	10/9/22
45	Pre-Coding: Odometry Part 4	10/11/22
46	Meeting #8	10/11/22
47	Meeting #8 Continued	10/11/22
48	Pre-Coding: Odometry Part 5	10/12/22
49	Meeting #9	10/12/22
50	Meeting #9 Continued	10/12/22
51	Design Process: Chassis Design Gearing and Side Skirt	10/13/22
52	Meeting #10	10/13/22
53	Meeting #10 Continued	10/13/22
54	Meeting #11	10/14/22
55	Meeting #11 Continued	10/14/22
56	Design Process: Chassis Design First Scrimmage	10/15/22
57	Pre-Coding: Odometry Part 6	10/15/22
58	Pre-Coding: Odometry Part 7	10/15/22
59	Meeting Formatting	10/17/22
60	Old Pre-Meeting Format	10/17/22
61	Old Post Meeting Format	10/17/22
62	New Pre-Meeting Format	10/17/22
63	New Post-Meeting Format	10/17/22
64	Meeting #12	10/18/22
65	Meeting #12 Continued	10/18/22
66	Pre-Coding Odometry Part 8	10/18/22
67	Pre-Coding Odometry Part 9	10/18/22
68	Pre-Coding Move to Point	10/18/22
69	Meeting #13	10/19/22
70	Meeting #13 Continued	10/19/22
71	Meeting #14	10/19/22
72	Meeting #14 Continued	10/19/22
73	Pre-Coding: Move to Point Part 2	10/19/22
74	Design Process: The Flywheel Version 1	10/19/22
75	Pre-Coding: Move to Point Part 3	10/20/22
76	Pre-Coding: Move to Point Part 4	10/21/22
77	Pre-Coding: Move to Point Part 5	10/21/22
78	Pre-Coding: Move to Point Part 6	10/21/22
79	Pre-Coding: Move to Point Part 7	10/21/22
80	Design Process: The Turret Version 1	10/21/22



Table of Contents

Page	Linked Project Slides	Date
81	Meeting #15	10/22/22
82	Meeting #15 Continued	10/22/22
83	Design Process: The Turret Initial Gearing	10/22/22
84	Design Process: Odometry Wheels First Designs	10/22/22
85	Pre-Coding: Move To Point Part 8	10/23/22
86	Pre-Coding: Odometry Part 10	10/23/22
87	Pre-Coding: Odometry Part 11	10/23/22
88	Meeting #16	10/24/22
89	Meeting #16 Continued	10/24/22
90	Pre-Coding: Odometry Part 12	10/24/22
91	Design Process: The Robot Initial Designs and Sketches	10/25/22
92	Design Process: The Robot Initial Designs and Image	10/25/22
93	Design Process: Intake Design and Efficiency	10/25/22
94	Design Process: Robot Design	10/25/22
95	Meeting #17	10/25/22
96	Meeting #17 Continued	10/25/22
97	Design Process: The Turret Version 1	10/25/22
98	Pre-Coding: The Aimbot	10/25/22
99	Pre-Coding: The Aimbot Part 2	10/25/22
100	November 19th Tournament Prep	10/25/22
101	Meeting #18	10/26/22
102	Meeting #18 Continued	10/26/22
103	Design Process: The Flywheel Version 2	10/26/22
104	Pneumatics Research	10/27/22
105	Meeting #19	10/27/22
106	Meeting #19 Continued	10/27/22
107	Design Process: The Turret Version 2	10/27/22
108	Design Process: How we Design	10/27/22
109	Design Process: Roller Mechanism Version 1	10/27/22
110	Game Strategy	10/27/22
111	Game Strategy: Strategy Page Format	10/27/22
112	Game Strategy: Strategies Page 1	10/27/22
113	Game Strategy: Strategy 1 Grep	10/27/22
114	Game Strategy: Strategy 2 Blitz	10/27/22
115	Game Strategy: Strategy 3 Grabby	10/27/22
116	Game Strategy: Strategies Page 2	10/27/22
117	Game Strategy: Strategy 4 Left Blitz	10/27/22
118	Game Strategy: Strategy 5 Skittles	10/27/22
119	Game Strategy: Strategy 6 Scraper	10/27/22
120	Notebook Formatting	10/27/22



Table of Contents

Page	Linked Project Slides	Date
121	Pre-Coding: The Aimbot Part 3	10/29/22
122	Pre-Coding: The Aimbot Part 4	10/29/22
123	Pre-Coding: The Aimbot Part 5	10/29/22
124	Pre-Coding: The Aimbot Part 6	10/30/22
125	Meeting #20	10/30/22
126	Meeting #20 Continued	10/30/22
127	Meeting #21	11/1/22
128	Meeting #21 Continued	11/1/22
129	October Monthly Update	11/1/22
130	Meeting #22	11/2/22
131	Design Process: The Logo	11/2/22
132	Meeting #22 Continued	11/2/22
133	Pre-Coding: The Aimbot Part 7	11/2/22
134	Pre-Coding: The Aimbot Part 8	11/2/22
135	Meeting #23	11/3/22
136	Meeting #23 Continued	11/3/22
137	Design Process: Tournament Flyers Version 1	11/3/22
138	Design Process: Tournament Flyers Version 1 Image	11/3/22
139	Design Process: The Chassis Version 2	11/7/22
140	Design Process: Wiring the Robot Version 1	11/7/22
141	Design Process: Wiring the Robot Port List	11/7/22
142	Meeting #24	11/7/22
143	Meeting #24 Continued	11/7/22
144	Design Process: Flywheel Mount Version 1	11/8/22
145	Meeting #25	11/8/22
146	Meeting #25 Continued	11/8/22
147	Design Process: Progress Check	11/8/22
148	Design Process: Odometry Wheels Version 2	11/9/22
149	Meeting #26	11/9/22
150	Meeting #26 Continued	11/9/22
151	Meeting #27	11/9/22
152	Meeting #27 Continued	11/9/22
153	Design Process: Flywheel Mount Version 1	11/9/22
154	Meeting #28	11/10/22
155	Meeting #28 Continued	11/10/22
156	Design Process: The Indexer Criteria and Constraints	11/11/22
157	Design Process: The Indexer Design 1: LEM	11/11/22
158	Design Process: The Indexer Design 2: SESGM	11/11/22
159	Design Process: The Indexer Design 3: RIM	11/11/22
160	Design Process: The Indexer Decision Matrix	11/11/22



Table of Contents

Page	Linked Project Slides	Date
161	Design Process: The Indexer Decision Matrix Explanation	11/11/22
162	Pre-Coding: Odometry Part 13	11/11/22
163	Pre-Coding: The Aimbot Part 9	11/11/22
164	Pre-Coding: The Aimbot Part 10	11/11/22
165	Design Process: The Chassis Version 3	11/12/22
166	Meeting #29	11/13/22
167	Meeting #29 Continued	11/13/22
168	Meeting #30	11/14/22
169	Meeting #30 Continued	11/14/22
170	Pre-Coding: The Aimbot Part 11	11/14/22
171	Design Process: The Flywheel Version 3	11/15/22
172	Design Process: Pneumatics and Turret Wiring	11/15/22
173	Meeting #31	11/15/22
174	Meeting #31 Continued	11/15/22
175	Game Strategy: Strategy Page 3	11/16/22
176	Game Strategy: Strategy 07 Kim	11/16/22
177	Game Strategy: Strategy 8 Bartholomew	11/16/22
178	Pre-Coding: The Aimbot Part 12	11/16/22
179	Pre-Coding: The Aimbot Part 13 New Turret Code	11/16/22
180	Meeting #32	11/16/22
181	Meeting #32 Continued	11/16/22
182	Meeting #33	11/16/22
183	Meeting #33 Continued	11/16/22
184	Design Process: The Flywheel Version 4	11/16/22
185	Pre-Coding: The Aimbot Part 14	11/16/22
186	Design Process: Roller Mechanism Version 2	11/16/22
187	Game Strategy: Strategy 9 Yoink	11/16/22
188	The Aimbot Part 15	11/16/22
189	The Aimbot Part 16	11/16/22
190	Pre-Coding: End of Pre-Coding	11/16/22
191	Pre-Coding: Code Backup 11/16/22 Pg1	11/16/22
192	Pre-Coding: Code Backup 11/16/22 Pg2	11/16/22
193	Pre-Coding: Code Backup 11/16/22 Pg3	11/16/22
194	Meeting #34	11/17/22
195	Meeting #34 Continued	11/17/22
196	Meeting #35	11/17/22
197	Meeting #35 Continued	11/17/22
198	Meeting #36	11/17/22
199	Meeting #36 Continued	11/17/22
200	Design Process: Odometry Wheel Suspension	11/17/22



Table of Contents

Page	Linked Project Slides	Date
201	Coding: Introduction	11/18/22
202	Meeting #37	11/18/22
203	Meeting #37 Continued	11/18/22
204	Coding: Driver Code Chassis Control Version 1	11/18/22
205	Coding: Driver Code Chassis Control Version 1 Continued	11/18/22
206	Pre-Tournament: State of the Bot Nov 19 Tournament	11/18/22
207	Meeting #38	11/19/22
208	Meeting #38 Continued	11/19/22
209	Meeting #39	11/21/22
210	Meeting #39 Continued	11/21/22
211	November 19th Tournament: Morning of Tournament	11/21/22
212	November 19th Tournament: Qualification Rounds	11/23/22
213	November 19th Tournament: Elimination and Reflection	11/23/22
214	Meeting #40	11/28/22
215	Meeting #40	11/28/22
216	Meeting #41	11/29/22
217	Post Meeting #41	11/29/22
218	Meeting #41 Details	11/29/22
219	Page Formatting Version 2	11/29/22
220	Page Formatting Version 2 Part 2	11/29/22
221	Page Formatting Version 2 Part 3	11/29/22
222	Pre-Meeting Format Version 2	11/29/22
223	Post Meeting Format Version 2	11/29/22
224	Meeting Details Formatting	11/29/22
225	Page Formatting Version 2 Part 4	11/29/22
226	Design Process Page Formatting	11/29/22
227	Page Formatting Version 2 Part 5	11/30/22
228	Coding Page Formatting	11/30/22
229	Strategy Page Formatting Version 2	11/30/22
230	Design Process Page 1: Odometry Wheels Version 2.1	11/30/22
231	Design Process Page 2: Chassis Version 3.1	11/30/22
232	Design Process page 3: Odometry Wheels V2.2 Plan	11/30/22
233	Pre-Meeting #42	11/30/22
234	Post Meeting #42	11/30/22
235	Meeting #42 Details	11/30/22
236	Design Process Page 4: Odometry 2.2 and Chassis 3.2	11/30/22
237	Pre-Meeting #43	11/30/22
238	Post Meeting #43	11/30/22
239	Meeting #43 Details	11/30/22
240	Design Process Page 5: Chassis Version 3.2	11/30/22



Table of Contents

Page	Linked Project Slides	Date
241	Pre-Meeting #44	12/1/22
242	Post Meeting #44	12/1/22
243	Meeting #44 Details	12/1/22
244	Meeting #44 Details Continued	12/1/22
245	Coding Page 1	12/2/22
246	Coding Page 2	12/3/22
247	Pre-Meeting #45	12/3/22
248	Post Meeting #45	12/3/22
249	Meeting #45 Details	12/3/22
250	Meeting #45 Details Continued	12/3/22
251	Coding Page 3	12/3/22
252	November Monthly Update	12/6/22
253	November Monthly Updated Continued	12/6/22
254	Pre-Meeting #46	12/6/22
255	Post Meeting #46	12/6/22
256	Meeting #46 Details	12/6/22
257	Meeting #46 Details Continued	12/6/22
258	November Monthly Update Continued +	12/6/22
259	Design Process Page 6: Indexer V2 Design	12/6/22
260	November Monthly Update Continued ++	12/6/22
261	Design Process Page 7: Flywheel V4.1 and V4.2	12/6/22
262	Design Process Page 8: Turret V3 and 3.1	12/7/22
263	Pre-Meeting #47	12/7/22
264	Post Meeting #47	12/7/22
265	Meeting #47 Details	12/7/22
266	Pre-Meeting #48	12/8/22
267	Post Meeting #48	12/8/22
268	Meeting #48 Details	12/8/22
269	Coding Page 4: Odometry Forward Code + Turning Code	12/8/22
270	Pre-Meeting #49	12/13/22
271	Post Meeting #49	12/13/22
272	Meeting #49 Details	12/13/22
273	Design Process Page 9	12/13/22
274	Design Process Page 10	12/14/22
275	Pre-Meeting #50	12/14/22
276	Post Meeting #50	12/14/22
277	Meeting #50 Details	12/14/22
278	Pre-Meeting #51	12/15/22
279	Post Meeting #51	12/15/22
280	Meeting #51 Details	12/15/22

Table of Contents

Page	Linked Project Slides	Date
281	Design Process Page 11: Flywheel V5	12/16/22
282	Pre-Meeting #52	12/20/22
283	Post Meeting #52	12/20/22
284	Meeting #52 Details	12/20/22
285	Design Process Page 12: Odometry Wheels V2.3	12/20/22
286	Pre-Meeting #53	12/21/22
287	Post Meeting #53	12/21/22
288	Meeting #53 Details	12/21/22
289	Design Process Page 13: Flywheel V6	12/22/22
290	Pre-Meeting #54	1/3/23
291	Post Meeting #54	1/3/23
292	Meeting #54 Details	1/3/23
293	Meeting #55	1/4/23
294	Pre-Meeting #56	1/4/23
295	Post Meeting #56	1/4/23
296	Meeting #56 Details	1/4/23
297	Pre-Meeting #57	1/5/23
298	Post Meeting #57	1/5/23
299	Meeting #57 Details	1/5/23
300	Meeting #57 Details	1/5/23
301	Pre-Meeting #58	1/6/23
302	Post Meeting #58	1/6/23
303	Meeting #58 Details	1/6/23
304	December Monthly Update	1/6/23
305	December Monthly Update Continued	1/6/23
306	Pre-Meeting #59 Tournament Day	1/7/23
307	Post Meeting #59	1/7/23
308	Meeting #59 Details	1/7/23
309	Meeting #59 Details	1/7/23
310	Meeting #59 Details	1/7/23
311	Meeting #59 Details	1/7/23
312	Team Reflection	1/7/23
313	Pre-Meeting #60	1/9/23
314	Post Meeting #60	1/9/23
315	Meeting #60 Details	1/9/23
316	Pre-Meeting #61	1/10/23
317	Post Meeting #61	1/10/23
318	Meeting #61 Details	1/10/23
319	Pre-Meeting #62	1/11/23
320	Post Meeting #62	1/11/23



Table of Contents

Page	Linked Project Slides	Date
321	Meeting #62 Details	1/11/23
322	Pre-Meeting #63	1/12/23
323	Post Meeting #63	1/12/23
324	Meeting #63 Details	1/12/23
325	Coding Page #5 Code Update + Flywheel Conversion Calc	1/12/23
326	Pre-Meeting #64	1/17/23
327	Post Meeting #64	1/17/23
328	Meeting #64 Details	1/17/23
329	Pre-Meeting #65	1/18/23
330	Post Meeting #65	1/18/23
331	Meeting #65 Details	1/18/23
332	Design Process Page 14 Expansion V3	1/19/23
333	Pre-Meeting #66	1/19/23
334	Post Meeting #66	1/19/23
335	Meeting #66 Details	1/19/23
336	Meeting #67	1/20/23
337	Pre-Meeting #68	1/20/23
338	Post Meeting #68	1/20/23
339	Meeting #68 Details	1/20/23

Introductions: Koltin Hoffman

I'm Koltin Hoffman and this is my second VRC robotics season. I'm hoping this will be fun and hard because I love challenges. I think that I might have an advantage over the competition with my knowledge and experience with fabrication, building cars, and motors. Two of my role models are my mom who got me into technical stuff and my dad who got me into building cars.

Introductions: Aidan McCallum

Hello my name is Aidan McCallum. This is my 4th year in Robotics and I am the coder of the team. I have been a coder for all my years of robotics since 6th grade and only skipped 8th grade due to covid year prevented me from finding enough people to form a team. My coding experience outside of robotics is not much but I have used Java, C#, Python, and C++ for different types of programs which have helped me code Robotics.

My previous team was 3249H and my team did really well for a freshman year and mostly rookie team. The only person on my team this year from 3249H is Ethan Regan. This year I mostly pulled together this current team from friends in robotics and friends outside robotics. I've always been interested in moving parts, and what I could do with something basic to solve complex problems. This is what I like about vex robotics and why I continued doing it for so long.

Introductions: Tyler Fields

Hello, I am Tyler Fields. I'm the driver for this team. This is my 1st year doing VEX robotics. I don't have much experience but I have driven Aidan's robot a couple of times for practice and I did robotics class in 8th grade. I decided to join robotics this year because my friend Aidan had been trying to convince me to join for a couple of years but I never was able to until this year. I'm looking forward to building, competing, and working with my team. I enjoyed building in my robotics class although I didn't get to do much because of online learning, now I'll be able to experience more of that.

Introductions: Ethan Regan

Hello, I am Ethan Regan. I am a builder and possible driver. This is my second year of VEX robotics. I was with team 3249H last year, as our builder/ driver. I enjoy fabrication, working with tools, and electronics, and the busyness of robotics. I did all robotics throughout middle school where I came to love the competitions and all the work that gets put in. We hope to get to worlds this year, I have high hopes for our team.

Introductions: Joshua Wilson

Hello, I am Joshua Wilson. I'm a builder. This year will be my 3rd year in VEX robotics. I was with team 4591M last year and was the main builder. My team last year went to worlds my first year and worlds last year. We also received the inspire award last year. The reason I joined robotics is that I'm good with building and problem-solving. I was invited to join 4591M by a friend on the team. I am hoping to go to worlds with this team and continue with this team. I have high expectations for myself and the team. I have outside experience with building by NHD and building with my family.

Meeting #1

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Tyler	Plans and Discussion	All Teammates Not Present	Define the Problem
Koltin	Plans and Discussion	All Teammates Not Present	Define the Problem
Aidan	Plans and Discussion	All Teammates Not Present	Define the Problem

Meeting #1 Continued

Post Meeting Info

Meeting Info
Location School: (VANCOUVER iTech Prep)
Attendees: Aidan Koltin and Tyler
Duration 4:05 - 5:30PM
Date: 5/26/22

Team Members	What got done	Unresolved Problems
Tyler	Figure out Formatting	Didn't get any research done
Koltin	Figure out Formatting	Didn't get any research done
Aidan	Figure out Formatting / Got Research Done	N/A

Meeting #2

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Ethan	Continue Design Research/ Summarize Section 3	N/A	Define the Problem
Josh	Learn Code, Rule Book Summarization	N/A	Define the Problem
Tyler	Learn Code, Summarize section 2 pt2	N/A	Define the Problem
Koltin	Summarize Section, Research	N/A	Define the Problem
Aidan	Teach Code, Summarize Game Manual	N/A	Define the Problem

Meeting #2 Continued

Post Meeting Info

Meeting Info
Location School: (VANCOUVER iTech Prep)
Attendees: All Members
Duration 4:05 - 6:00:pm
Date: 6/7/22

Team Members	What got done	Unresolved Problems
Ethan	Summarized Section 3	N/A
Josh	Worked on Summary, Learned Code	Section 1 Not finished
Tyler	Learned Code	Summarization Not Finished
Koltin	Worked on Section 4	Summarization Not Finished
Aidan	Teached Code and worked on Section 2 Summary	Summarization Not Finished

Meeting #3

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Aidan	Research and Final Design work	N/A	Generate Concepts
Tyler	Summarize	N/A	Define the Problem
Koltin	Decision Matrix	N/A	Design a solution
Ethan	Research	N/A	Generate Concepts
Joshua	Research and Summary	No laptop	Define the Problem

Meeting #3 Continued

Post Meeting Info

Meeting Info
Location School:
(VANCOUVER iTech
Prep)
Attendees: All Members
Duration 4:05 - 6:00:pm
Date: 6/9/22

Team Members	What got done	Unresolved Problems
Aidan	N/A	Research
Tyler	N/A	Summarization
Koltin	N/A	Decision Matrix
Ethan	Research	N/A
Joshua	N/A	Summerization

Design Process Explanation

The engineering design process is the core of VEX and all facets of engineering in general. Rather than being a one-time occurrence, the design process must be repeated many times to be very effective. It goes in the following order, though it is common to revert to an earlier step in the design process before completing the entire process when necessary.

Define the Problem:

Outline and understand the project's desired outcome; this is where you ask questions and set the criteria and constraints of the project. Many criteria and constraints will be set by outside sources, such as employers or rules of a challenge (VRC for example). However, some criteria will be much more project-specific than others. Some constraints are also universal, for example, the safety and effectiveness of the final product.

Generate Concepts:

In addition to generating ideas and brainstorming solutions to be used for completing the project; this section is where the data gathering will occur. Citations are absolutely necessary for this stage to show proper credit for others' work and original ideas. Research is an invaluable tool to generate unique ideas and solutions.

Design Process Explanation Continued

Design a Solution:

Take all the inspiration and the ideas that you gathered in the previous section, and put it into a design of your own. Creating an original design includes sketching out the real-life dimensions of the product in detail and to scale. Having multiple sketches from different views of the product will help communicate and improve the precision of the details of the unrealized design.

Build and Test:

Build a physical prototype of the product based off of the sketches that have been made. Once construction is complete, test its current effectiveness. If there are obvious improvements that need to be made, record them and physically adjust the design. Repeat this process as much as necessary until the next step can be achieved with the observations made in the physical tests.

Evaluate Solution:

Using data gathered from the built prototype, determine the effectiveness of the solution. Record the findings and use this information to decide whether to move to presenting the solution or go back to a previous step of the design process. If there are any major flaws that can be removed, or an improvement that can be made in addition to the current design, go back to generating concepts or designing solutions. Otherwise, continue to the final step of the design process.

Present Solution:

Make any last-minute adjustments or final changes to the design (should be cosmetic at this point), and rigorously test the solution to make sure it consistently meets expectations and performs without fail. Then, prepare for the presentation of the final product. Give a good and honest show of the design's capabilities.

Game Manual Summary Section 1

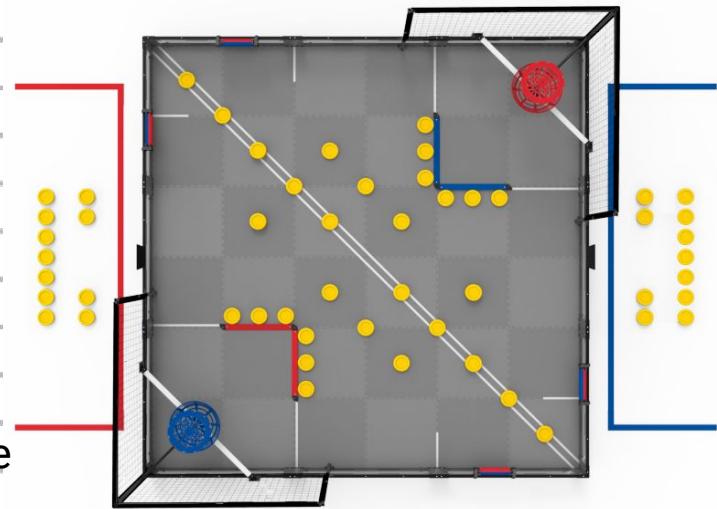
As our technological advancements continue to grow it leads to newer and bigger problems that are called “the STEM project”. The way schools teach STEM doesn't prepare students for the real world and once they realize how important the type of work is and they already decided that they are not interesting or not fun. If you don't have the skills or passion you can not make any progress in work and find it too challenging.

VEX robotics was made to make a solution to the problem, it provides the ability to make the STEM skills or further enhance those skills by working with a team, fixing problems, and traits of STEM. VEX robotics is not just for fun but for teaching how to apply the engineering design process and getting a mindset that inverters and other STEM workers use on a daily basis. Vex robotics is meant to teach and move onto lifelong skills to help the future leaders.

Game Manual Summary Section 2 Pt1

Section two of the game manual is about the game and its rules. Matches are set up where two alliances of two robots face off against each other with a Red Team and a Blue team.

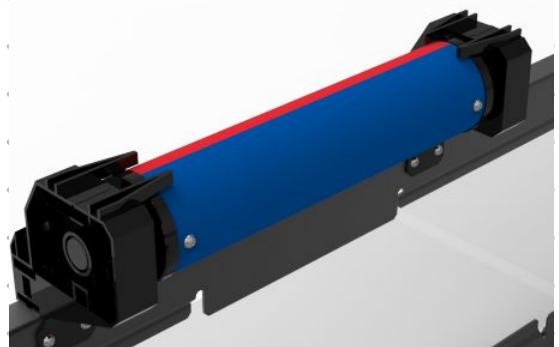
The field consists of 60 discs with 38 on the field. 8 for preloads and 14 for match loads with half of each going to each alliance. The field is separated diagonally and each opposing side has a high goal below each high goal is the opposing team's lower goal. In the non-goal-containing corners, there are 2 rollers in each corner with 4 rollers in total. Match loads are entered into the game by the loaders next to each low goal area, and there are two leaders on the field. Robots start on the borders tiles of the field indicated by the white lines.



The game is broken down into two phases the 15-second autonomous period and the 1:45 seconds driver period. In the Autonomous period, the robot is controlled by prewritten code and the team that scores the most points during the autonomous period gets a point bonus of 10 points. In the 1:45 second driver period, the Robot will be controlled by each team's driver to score points using discs, Rollers, and end-game expansion. Discs can be scored in the home goals of each team for 1 point in the low goal zone and 5 points in the High goal zone.

Game Manual Summary Section 2 Pt2

Rollers work on a racket and pawl mechanism to rotate in small increments. When a roller is scored by having your team's color shown at the top it is worth 10 points



The last method of scoring is end-game expansion. End game expansion occurs in the last 10 seconds of the game and in this period there is no horizontal expansion limit and the number of tiles a robot is touching will correspond to how tiles are claimed. For each tile claimed it is worth 3 points. Opposing alliances can claim the same tiles and get the same amount of points.

Game Manual Summary Section Pt3

Safety rules. If a team's actions are deemed unsafe or damage field equipment or a disk, that team may receive a disqualification. If a robot goes outside the playing field during a match, it will be disabled for the remainder of the match. While in the alliance stations during a match you must wear safety glasses with side shields.

Next is general game rules. All teams are expected to treat staff, volunteers, and fellow competitors with respect. If a team member is disrespectful/uncivil they may be disqualified from a match. Repeated violations can cause a team to be disqualified from the entire event depending on the severity. Adults may never help with the building or the programming of a robot. Students may have to demonstrate understanding of their robot to judges or event staff. The starting robot volume must be 18" by 18" by 18". All violations of this will have the robot removed until the situation is corrected. Robots may also not have any detaching parts and leave mechanisms on the playing field. Team members may not change teams minus some exceptions. During a match, each team can have 3 drive team members in the Alliance Station and they must remain there for the rest of the match. You may not: bring any cellular devices to the alliance station, stand on any sort of object during a match, or bring any additional materials. Controllers have to stay connected to the field for the duration of the match. During a match, you may also not make contact with the field or any field elements. During the autonomous period, you can't interact with the robot whatsoever. However, all rules still apply during the autonomous period. You are not allowed to create a robot with the intention of damaging, tipping, or entangling opposing robots. However, when judges must make a call regarding destructive intentions, the offensive robot gets the benefit of the doubt. You cannot intentionally make an opponent break a rule.

Lastly is specific game rules. During the start of a match, the robot may not be touching any field elements, disks (other than preloads), and tiles in the low goal. Robots may only expand vertically if its not within either “low goal” area. Any extensions must fit within a vertical cylinder 2’ in diameter. Match load disks may only be introduced after the driver controlled period starts. The match load disks may be introduced safely by a drive team member onto the loader to be retrieved by the robot.



Robots may only be in possession of 3 disks at once. Robots in violation of this whether intentional or accidental must stop all actions except trying to remove the disks. During the autonomous period robots may not contact anything within the opposing teams side of the autonomous line. All violations of this will result in the other team getting the autonomous bonus. Teams may not intentionally remove disks from the field.

Game Manual Summary Section 3

Ethan Regan.

Section 3; This section regards the limitations of your robot build. Inspections are performed before all tournaments and are done on a pass/ fail basis. Every robot must pass inspection in order to compete in a tournament.

Robots must begin each match within an 18" squared box, and all materials used to retain this limit must stay with the bot throughout the match. A maximum of **1 V5 brain, 8 V5 motors, 2 pneumatic cylinders, 1 battery, and up to 2 controllers are allowed**, none of these components can be modified.

1 robot is allowed per team, built with only vex parts (including allowed) parts such as:

*hot glue- only on wires.

***string less than 6.35mm thick**

*Contraptions to secure wires

*Rubber bands

*Zip ties

*nonfunctional decorations

*grease or lubrication

*Non shattering custom plastics may be used in moderation. **Plastics cannot exceed 0.070mm thick.**

Plastics such as lexan, acetron, and nylon may be used. Shattering plastics such as plexiglass are prohibited. **3d printed plastics are not permitted, except as non-functional decorations.**

Robots should have the following subsystems:

1. MRB: (chassis) the mechanical motion system including motors, wheels gears and other contraptions to move your robot.

2. Power control system: Includes a vex V5 legal battery, V5 Brain, and V5 motors, sensors, and connections.

3.additional functions: additional mechanisms to manipulate discs, rollers, or to aid in the earning of points in game

In order to pass inspection, robots must have subsystems 1 and 2

*Robots can not be intentionally built to destroy or damage other robots, or damage game field components. If a robot damages another competitor or the game field, a disqualification can be issued by the discretion of the attending referee.

*your Radio receiver must not be buried within the robot, and the receiving end of the module should not be in proximity to, or touching metal components.

Game Manual Summary Section 4

-Here are some match-affecting things that could cause a match replay.

Match replay, where a Match is played over again from its start, the replay must be agreed upon by both the Event Partner and Head Referee. And will only be issued in the most extreme circumstances. Some examples that may warrant a Match replay are Match Affecting “field fault” issues.

Discs or Field Elements not starting in the correct positions

Tape lines lifting

Field Elements detaching or moving beyond normal tolerances (not as a result of Robot interactions)

The Autonomous Period or Driver Controlled Period ending early

Field control disconnecting or disabling Robots. Note that this is sometimes confused with a Robot whose motors have overheated, or bent pins on a controller's competition port causing intermittent drop-outs. In general, any true field fault will impact both Alliances simultaneously, not one Robot at a time.

-And here are some definitions of how them game is ruled

Autonomous Points (AP) - The second basis of ranking Teams. An Alliance who wins the Autonomous Bonus during a Qualification Match earns ten (10)

Autonomous Points. In the event of a tie, both Alliances will receive five (5) Autonomous Points.

Autonomous Win Point - One (1) Win Point (WP) given to an Alliance that Owns two (2) Rollers and has Scored at least two (2) Discs in their Alliance-colored High Goal at the end of the Autonomous Period. Both Alliances can earn this Win Point if both Alliances accomplish this task.

Elimination Bracket - A schedule of Elimination Matches for eight (8) to sixteen (16) Alliances.

Elimination Match - A Match used in the process of determining the champion Alliance. Alliances of two (2) Teams face off according to the Elimination Bracket; the winning Alliance moves on to the next round. Section 4 The Tournament Overview VEX Robotics Competition Qualification and Elimination Matches are played in a Head-to-Head tournament format. Qualification Matches are used to rank Teams based on Win Points (WP), Autonomous Points (AP), and Strength of Schedule Points (SP). The top-ranked Teams then form Alliances to participate in Elimination Matches and determine the tournament champions. This section applies primarily to VRC Head-to-Head Matches. For rules specific to other types of Matches,

The volunteer VEX Robotics Competition tournament coordinator serves as an overall manager for the volunteers, venue, event materials, and all other event considerations.

Event Partners serve as the official liaison between the REC Foundation, other event volunteers, and event attendees.

Head Referee - An impartial volunteer responsible for enforcing the rules in this manual as written. Head Referees must have completed the REC Foundation Head Referee certification course (expected to be released in Summer 2022). Head Referees are the only individuals who may discuss ruling interpretations or scoring questions with Teams at an event.

Match Schedule - A list of Matches that is generated at the start of an event. The Match Schedule includes the predetermined, randomly-paired Alliances that will be competing in each Qualification Match and the expected start times for these Matches. The Match Schedule is subject to change at the Event Partner's discretion.

Practice Match - A Match is used to provide time for Teams and volunteers to get acquainted with the official playing field and procedures.

Practice Matches earn Teams zero (0) Win Points, Autonomous Points, and Strength of Schedule Points.

Qualification Match - A Match used to determine Team rankings for Alliance Selection. Each Qualification Match consists of two Alliances competing to earn Win Points, Autonomous Points, and Strength of Schedule Points.

Team Goals

Our team 3249V will aim high this year. Our main goal is to go to the VEX robotics Worlds Competition and hopefully win an Award. The pathway to Worlds is a difficult one. There are many ways to get to worlds, but not to get into State. To enter worlds you need to get any award except the judges' award or sportsmanship award at state. To get into State you need to get either the Tournament Champion award, the Design award, or the Excellence award.

Getting the Design award requires having the best notebook in the competition. Getting the Excellence award requires having a good interview, notebook, and a high ranking in the competition. This prized Excellence award is what all 5 of us strive for. Our school iTech has only had 2 excellence awards in the past so it will be a large achievement for us and our School.

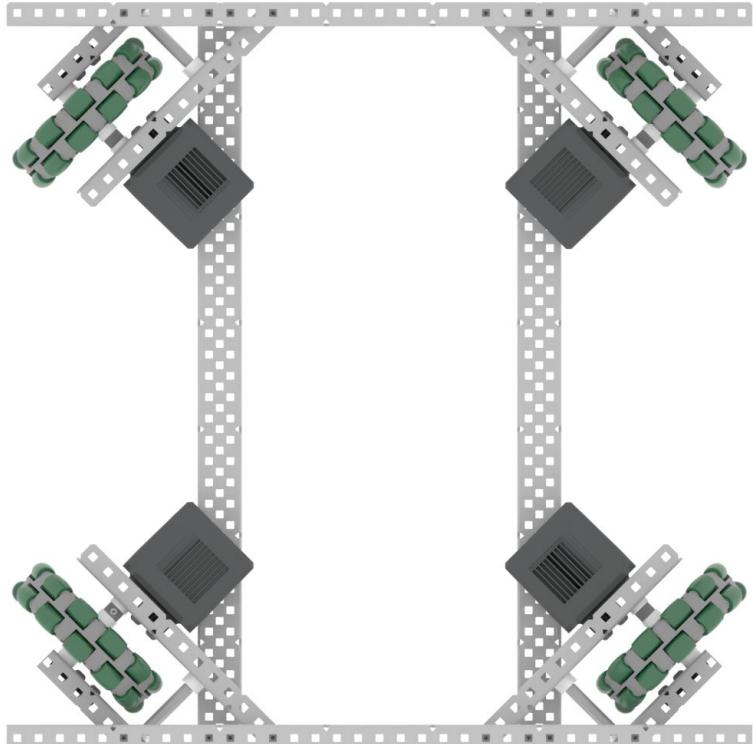
Aidan McCallums Research

I was lucky to watch the Reveal for both Middle school and high school I did some research and I have some concepts for my Final design. I will be going over all parts of the competition in sections

Chassis

In VEX robotics there are many Chassis types but here I will cover 3. The First design I will go over is the standard Tank drive that has 4 wheels at each corner of the bot all facing forward Like a Car would. Unlike a car, the wheels cannot rotate and have to rely on a style of movement Called tank drive which is where The left side of the robot and the right side of the robot goes opposite of each other to move left or right. To turn left the left side has to go backward and the right forwards and being opposite to turn right. This is a very simple drive train that the team will most likely use.

The second is the H drive. It is like the Tank drive but an extra motor is used to move left and right. One advantage is side-to-side movement while going forward, but having only one motor in the middle its power is limited. The disadvantage is that you can only use Omni wheels. I will go more in-depth about this when I talk about wheels later.



The last design I will cover here is the X Drive. This design allows for more speed and maneuverability which is desired in the current game. A disadvantage is external gearing will be harder to do because of how the wheels are positioned

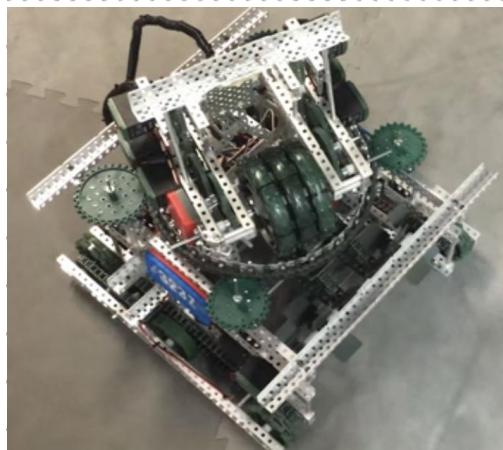
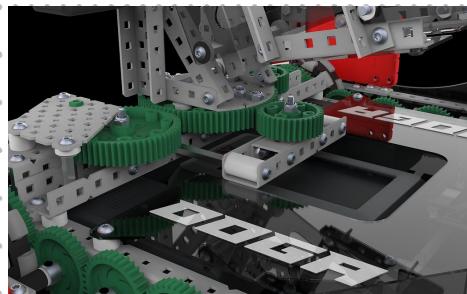
Lastly before getting to the next section I want to talk about wheels types. Unlike Tipping point or Turning Point, there is no platform to climb up on so there's no need to have friction wheels. All Omni wheels or Mecanum Wheels are the options. Mecanum wheels allow side-to-side movement which may be useful for some designs.

Aidan McCallums Research

Disc Scoring Now onto Disc manipulation. Discs are fairly large being 5.5" inches in diameter and 0.787" but extremely light only weighing 65 grams. There are multiple ways of scoring discs in spinup. I want to explain how I have seen most do it. The conveyer belt used last year to manipulate rings is by far the most efficient way of transporting discs within your robot. To score discs I would use a flywheel. The flywheel is a motorized part of your robot which spins at very fast speeds to launch discs into goals. I will go over this more when I make my final design.

Turret Design

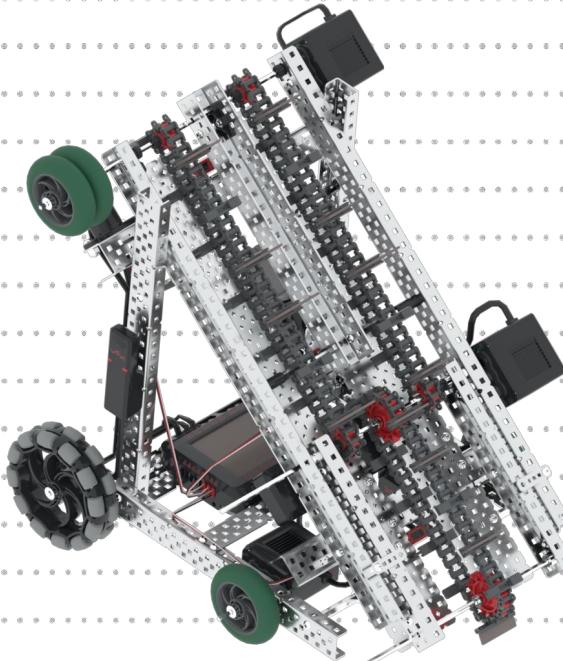
For a turret mechanism there is a lot to go over here. It is a overall complex mechanism that requires a significant portion of the bot to be designed around. The first design I want to look at is from the competition Turning Point by team 581A. This design works by rotating a claw and the claw has a launch mechanism that shoots a ball from its game. The design can be modified to shoot which then it can be used for this game.



The next design I want to cover is from another previous competition. The design was made by 323Z for the competition "nothing but net". It uses bent 1 by 1s with chains attached so it can be moved by gears. In the image above it, the motor gears are all connected to spin the most efficiently as possible. I personally like the design but it may not fit on the bot since a top and bottom flywheel may be needed in order to shoot a disc straight.

Aidan McCallums Research

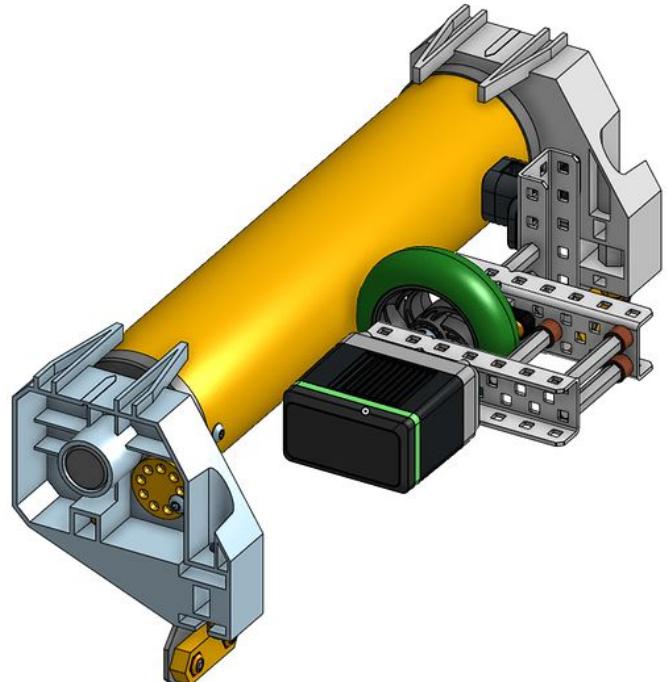
Roller Scoring



Rollers are worth 40 points and unlike discs, they can be easily unscored. Roller designs are very simplistic.

The first roller design I looked at is on this year's hero bot Disco. Disco is overall an ok bot but I want to look into its roller mechanism. It uses two small friction wheels to spin the rollers. It simplicity. It works using the same motor as the top rollers. One drawback is that you cannot hold a disc in the top roller while scoring a roller which may complicate win point autonomous.

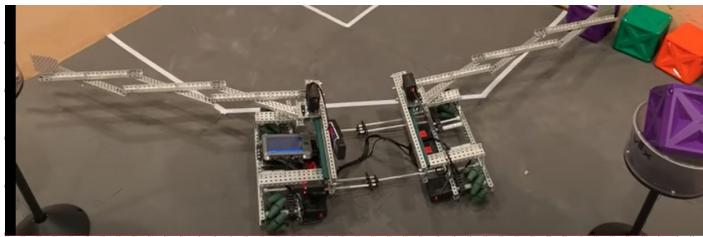
The second design I found was from Team 32511A. This is a simple attachment. This design is straight to the point with some sensors to tell color and a easy solution to implement. My one issue that I have with it is the usage of the motor. In this game when having a turret system motors are very important for control and using the same motor for another mechanism and this mechanism would work a lot better. The design here is obviously a one off design specifically for the roller.



Aidan McCallums Research

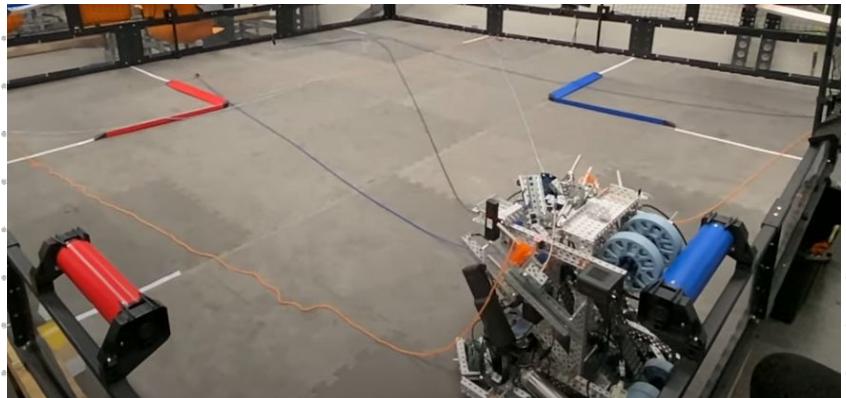
Expansion

For expansion this topic hasn't been cover much on the internet so I will only cover two Samples I found. The first design I looked at was in a video, by Team 834H in the competition Tower Takeover.



They used a linear slider to expand the robot itself and have scissor lifts on each side that extend outwards. This design would need a modification due to the scissor lift needing to touch the ground in order to count for scoring. But overall this design takes up too much space and motors in the robot just for the last 10 seconds of the game.

The second mechanism I looked into scores 75 points and is from team 67101C. The team utilizes rubber bands and a extended pneumatics piston to launch 5 strings that cover 25 tiles. I really like this design in its simplicity and results as it can cover the field in a second. The main drawback about its design as you need to be in a corner to utilize it which may be hard to get to within the last 10 seconds if other robots are pushing you around.



Meeting #4

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Koltin	Put his research in the notebook.	All Teammates Not Present	Do background Research
Aidan	Begin his decision matrix design	All Teammates Not Present	Create a prototype

Meeting #4

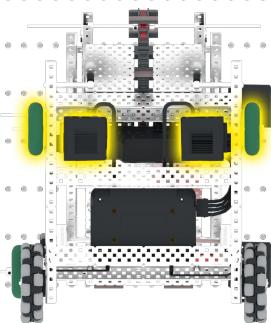
Post Meeting Info

Meeting Info
Location School: (VANCOUVER iTech Prep)
Attendees: Aidan and Koltin
Duration 4:05 - 6:00:pm
Date: 9/21/22

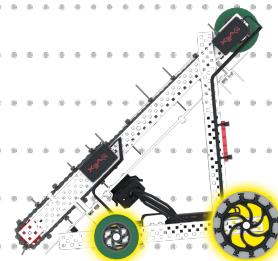
Team Members	What got done	Unresolved Problems
Koltin	Put research in notebook	N/A
Aidan	Talked with new teams, and worked on decision matrix	Decision matrix not finished

Koltin Hoffmans Research

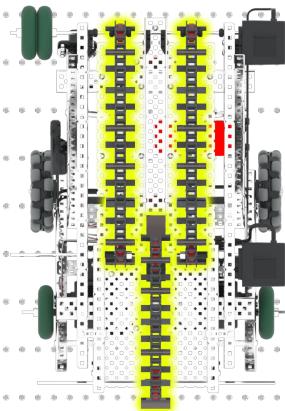
This year i have decided to research the code name disco bot by VEX, which has a 2-motor direct drive drivetrain. This makes for an easy to assemble and effective drive for the robot. Direct drive refers to having the shaft go directly from the motor to the wheels without using gears or a chain and sprocket system. The two motors power the front wheels making this a front-wheel drive robot. As shown in figure 1.



The drive wheels are 2.75" wheels, while the rear wheels are Omni Directional Wheels. Having the smaller 2.75" wheels in the front allows for the robot to tilt down, which makes picking up Discs easier. With the angle that the front is at it will help the robot get under the disk. As shown in figure 2.

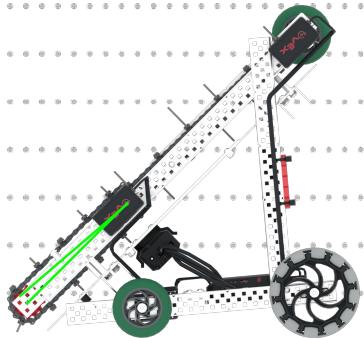


Disco has three main mechanisms; the intake, the conveyor, and the color roller. The roller on Disco is at the optimal height to make contact with color rollers and score them. The roller is powered by the top motor from the conveyor group and spins simultaneously when the conveyor is spun.



Koltin hoffman's research

Have you ever tried picking up a playing card off of a hardwood floor? Have you noticed that it is difficult to do unless you are underneath the object you are trying to pick up? This is why the intake has a floating ability, allowing it to grab and potentially get under the Disc, allowing for ease of picking them up.



Meeting #5

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Ethan	Do Research	N/A	Do background Research
Tyler	Do Research	N/A	Do background Research
Koltin	Working on decision matrix design	Rolled his ankle	Create a prototype
Aidan	Worked on Code	N/A	Create a prototype

Meeting #5 Continued

Post Meeting Info

Meeting Info
Location School: (VANCOUVER iTech Prep)
Attendees: Ethan, Tyler, Koltin, and Aidan
Duration 4:05 - 6:00:pm
Date: 9/22/22

Team Members	What got done	Unresolved Problems
Ethan	Put research in the notebook	N/A
Tyler	Finished Research	N/A
Koltin	Worked on Section 4	Decision Matrix design not finished
Aidan	Worked on Code	N/A

Research: Ethan Regan

Design.

Robots competing in vex spin up are required to move discs into the goals in order to score points. However in order to be competitive, robots must hoard and store discs, and be able to quickly fire them. Initially, vex members built a hero bot featuring a loader consisting of flaps and chains. This bot would be viable, yet horrible for competition. Many robotics teams built early designs consisting of large intake systems that take up valuable space in the robot. I believe a vertical intake system would leave viable space for large capacity ring storage, providing competitive advantage, along with the top wheel of my design to move the rollers. Our disk firing mechanism has been somewhat planned to be a flywheel launcher, with a pneumatic cylinder trigger. With my design of a vertical intake, optimal ring storage is reached. We have had team discussions and as of 6/9/22, we are thinking about 5 wheel drive, geared extremely fast to optimize efficiency. 9/20/22. Update; we have shifted into some more logistical game rules. 3 discs may be fired, so our robot must be able to pick up and throw discs quickly. As followed, I have split up my design proposal into 4 subsystems

Subsystem 1, Ring gatherer.

Rings will be the most important thing to gather. A full disc chamber (3) would mean certain victory. If we plan a map in a zigzag pattern to gather rings, we could take most of the rings on our side. The code should end at the rollers, in hopes other teams would not touch them for the match. As seen in designs such as ri24hr with their early season prototype, they have a fixed shovel style intake consisting of rotating wheels that guide the discs up to some sort of firing chamber. A 2 directional mechanism would also be possible, and there may be some competitive advantage

Subsystem 2, The launcher

As many other teams have pre planned, a flywheel is indeed a great way to ensure discs get into the goals. However the ingenuity challenge will be getting discs into the launcher. As the gatherer intakes the disks, we want to be able to hold onto them until we are ready to fire. We could store discs in the back half of the flywheel chamber and use a chebyshev's lambda mechanism to accelerate discs into the flywheel. Another way to move disks from storage to firing range is to use pneumatic pistons to push disks forward. Another thing to consider is aim. As we shoot discs from different angles, the launcher will also have to be angled appropriately

If we can figure out how to angle our flywheel mechanism on an x axis camber, we could use color sensors to auto aim and adjust speed so we could fire from different angles, almost anywhere on the field. This ease of aiming with programming could give us a strong competitive advantage. We would only need a driver to move to grab disks, and the robot would fire the discs itself.

Subsystem 3, the expander, and the spinner

Our robot may benefit from being able to spin the rollers in the last few seconds, rather than the first. We could move the roller with an extra wheel, powered by the ring gatherer. Also extra points for covering space on the field, so we could use pneumatic released strings fired from within the robot, using rubber band tension to launch these strings across the field in a net-like fashion.

Subsystem 4: the chassis

In order to move quickly around the field, the most important thing is going to be turning. A full omni setup would be most beneficial. I believe that we could use 6 wheel drive to outpush the competition and steal discs. If we were to use a transmission, we could draw some power to possibly turn the roller using a chain drive. One of the best advantages we could have is being heavy and able to push other robots off of discs. 6 wheel drive + transmission and an all aluminum frame would give us the ability to steal rings, push a bot away from rollers, and keep our trajectory if we make contact in autonomous

Pre-Coding

Introduction

Our first tournament is on November 19th. So coding during practice will be tight so I want to prepare as much as possible. This section is made prior to the bot being made and designed so everything in this code needs calibration. As when I am calibrating I will do entries logging errors, changes to overall code, etc. I will be going over Odometry, PiD's, Player control, Autonomous strategies, and Turret control and Aimbot. I will go over each subject on their own page as to do the topic proper representation.

VRC uses C++ for its coding languages. This year will be my second year using Vexcode V5 Pro and 4th year programming overall. Spin-up will be a real challenge as this year I will be programming an aimbot. An Aimbot requires constant and exact odometry code. The mathematics involved in all of this requires math beyond what I've been taught in schooling so far. By the end of this coding section I wish to have everything ready so my team can calibrate and use the code to its max.

Pre-Coding: Odometry

Overview

Odometry is by far the most complex thing on the robot. I will be going over my entire process of precoding it. All changes, etc. Odometry is the use of sensors to track position. It is mainly used in robotics so robots can know what to do and where to go.

One method of how you may experience odometry in everyday life may be your Cars Odometer tracking miles. Another way you may experience it is from a Roomba or another robotic cleaner. Roombas use a collection of sensors to navigate your home. It can detect cliffs with infrared sensors. It can detect walls from its bumper. From these sensors the roomba creates a maps of your home. Some Vex robots also use a collection of sensors to detect where it is at all times.

Why use Odometry? Not all teams use odometry. Last year it wasn't entirely needed as there wasn't a reason to use it other than a goforward PiD. 3249V last didn't use odometry instead they used a acceleration goforward function and a PiD turning with a inertial sensor. Also Odometry requires complex calculations and trigonometry which for some high school teams and nearly all middle school teams cannot perform due to not having the knowledge to do so. We need Odometry for creating an active aiming system or in slang terms "Aimbot".

Each team does odometry differently in some way. But at least one sensor is needed. Now before going on what I started I want to mention one way of doing odometry some teams may be using

Methods of Tracking

The GPS sensors gives all the tracking data needed, x, and y values and already calculates odometry. Everything seems correct on paper. Theres 3 main issues why we can't use the sensor. One being that the vex V5 system has a refresh rate of 10~ms which is too slow for PiD's. Second being that iTech doesn't have any sensors. The sensors are \$200 dollars. Third being that most tournaments in oregon don't have the Map strips needed for the sensor to work. All this makes the sensor not viable for the robot. I'll go over what sensors we are using on another page. As pre coding has some changes over time that include sensor changes.

Meeting #6

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Ethan	Sketches, Thursday	N/A	Brainstorm Ideas
Josh	Research, Can finish by today	N/A	Research the problem
Tyler	Research in virtual notebook Sketches	N/A	Brainstorm Ideas
Koltin	Transfer work to Virtual notebook/ Cad Design	No Laptop	Brainstorm ideas
Aidan	Sketches Wednesday	N/A	Brainstorm Ideas

Meeting #6 Continued

Post Meeting Info

Meeting Info
Location School:
(VANCOUVER iTech
Prep)
Attendees: All Members
Duration 4:05 - 6:00:pm
Date: 10/4/22

Team Members	What got done	Unresolved Problems
Ethan	First Sketches	Final design not complete
Josh	N/A	Research
Tyler	Research and began sketches	Sketches
Koltin	N/A	Decision matrix design
Aidan	Worked on final design	Final design not complete

Tyler Field's Research

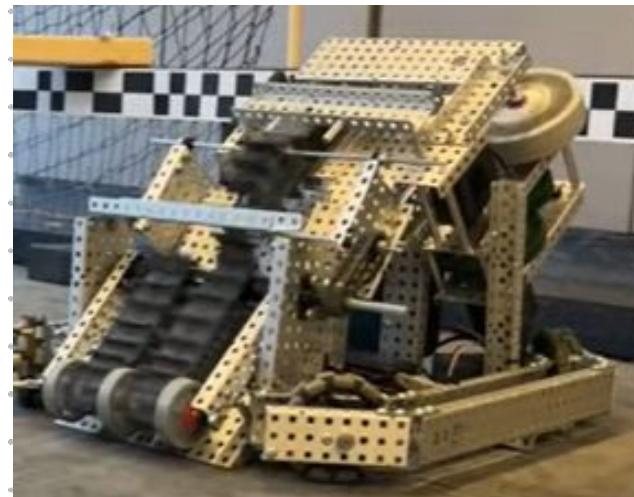
Mankato Robotics

I have decided to research 3 different robot concepts and analyze their pros and cons. The first robot I want to look at is the Mankato Arena Robotics Club 24-hour build. The robot has 4 Motor 257 rpm drive, 2 Motor 4200 rpm flywheel, 2 Motor 600 rpm intake/roller, 1 Piston indexer, and 1 Piston angle actuator. The robot has a large intake for disks that can pivot up and down to get over the disks easier. It has a flywheel with a storage container. The intake also doubles as a roller. It can store 3 disks and can fire them fairly fast as well. However the robot can't adjust its aim, it can't push disks, and it can't expand.



Millard North's Robot

The next robot I want to talk about is Millard North High 5069's 12-hour build. This robot features an X-drive, a floating intake, and a flywheel, and can hold up to 3 disks. However, it cannot turn the rollers nor can it expand. This robot also has a locked aim and shoots at a fairly slow pace.



4082B's Robot

The last robot I want to talk about is 4082B's robot. This robot has a 6 motor 300rpm drive, 1 motor 3600 rpm flywheel, 1 motor indexer, a 400 rpm intake, and a 66 rpm roller. Its design has a pivoting intake, a storage area for 3 disks, and a flywheel to launch said disks. It can accomplish all of the objectives of this game. It rotates fairly slowly so it can't aim very quickly and its expansion shoots 2 strings in the same general direction which means it has to get to one side of the field. Other than that it's a fairly good design.

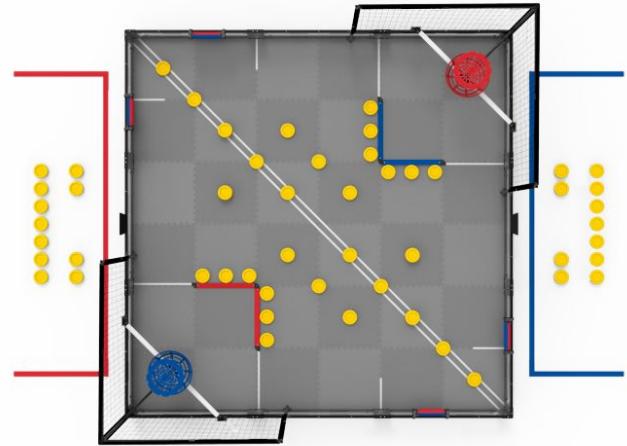


Pre-Coding: Odometry Part 2

Location Finding

Every team can create odometry differently or not use odometry at all. We decided to create an aimbot for disc scoring. Which for the first step of creating something of this complexity is to know where the goal is. Let's start off getting locations of everything we need

Let's identify parts of the fields that the robot might need to know. First being the autonomous line. The autonomous line contains 14 discs that either side can grab. But if the robot crosses the Autonomous line it will count as a penalty so we need to make sure the robot doesn't accidentally cross it during the autonomous period. Knowing the autonomous line during skills will help collect and score the 14 discs during the Programming skills. Next the robot should be able to identify where the rollers are located. If both rollers are scored as well as two discs in the teams high goal zone. The autonomous winpoint will be awarded. This winpoint will be extremely useful for climbing the leaderboards during Qualification rounds at tournaments.



Lastly for basic location finding will be the matchload ramp. During skills match loads are very important for maximizing skills points. During skills your team gets 14 discs which if scored correctly can give you 35 points.

Now for the most important locations to know. The high goal zone. Each high goal zone is opposite to your teams side. The code needs to also know the low goal zone incase the high goal zone is full of discs. I don't know if this will actually happen during a game but in a spin up simulation called XRC simulator the High Goal zone Filled up and we needed to use my teams low goal zone. In the game manual there are measurements for each and every element. Each element for both red team and blue team will be given a variable X and variable Y value. For vertical elements such High goal zones a Z value will be given. The reason why I am giving Z for a vertical value instead of Y for games like minecraft is that for 2d games X and Y are horizontal and vertical. Vex is a top down game so X would be along the lines of Judges to Audience and Y will be Red team to Blue team. Finally before graphing in the game manual 0,0 is on the bottom Right of the field with 140,140 being the top Left.

Meeting #7

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Ethan	Sketches and finalize chassis planning	N/A	Brainstorm Ideas
Josh	Research,	N/A	Research the problem
Tyler	Research in virtual notebook Sketches	N/A	Research the Problem
Koltin	Transfer work to Virtual notebook/ Cad Design	N/A	Brainstorm ideas
Aidan	Collaborating and conversing with other teams	N/A	N/A

Meeting #7 Continued

Post Meeting Info

Meeting Info
Location School: (VANCOUVER iTech Prep)
Attendees: All Members
Duration 4:05 - 6:00:pm
Date: 10/5/22

Team Members	What got done	Unresolved Problems
Ethan	Worked on chassis / Designed bot	Sketching Design.
Josh	Organize bin / Research	Research needed to be completed
Tyler	Worked on chassis / Designed bot	Sketching Design.
Koltin	Organize bin	N/A
Aidan	Worked on chassis / Designed bot	Sketching Design.

Josh Wilson's Research

Robot 80001B

i really like the design of this robot because of it launching mechanism and expansion mechanism. Its launching mechanism does not use a flywheel but a flinging mechanism that is loaded from the top and can launch 3 discs at a time. Its expansion shoots out 8 red strings that cover most of the field



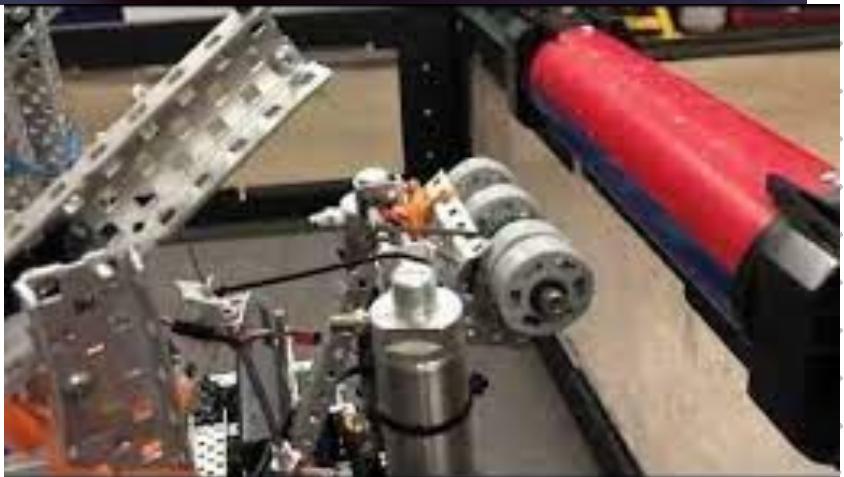
3150A Silver owl

its launching mechanism is in the middle of the robot and seems to have the disks fed from the middle up while having a helicopter like expansion that has to rods made out of modified plates



46V Passive Roller

their passive roller uses a rubber band that holds a hammer type head with plastic and mesh that pull down with pushed against the roller itself.



Pre-Coding: Odometry Part 3

Using the measurements given on the Vex Manual, will give us all our values for the objects in the odometry. Each point requires two variables for X and Y and for the high goals a 3rd Z will be needed. For the aimbot I will have thresholds so it doesn't run the PiD vertical correction and PiD horizontal correction when the robot isn't moving. Here is the code snippets below for locations.

Lines 85 and 116 are notes. For location finding on the middle line I'll use if then statements to check if its along the line. Z high, Z low, and Z Mid for the high goals at lines 108 - 110 and 113 - 115 are for the previously mentioned thresholds. I used float variables instead of double variables because these variables are static decimal points.

```
85 // Object coordinates for object locations from 0,0 in inches
86
87 float middleLineX1 = 0;
88 float middleLineX2 = 140.02;
89 float middleLineY1 = 0;
90 float middleLineXY = 140.02;
91
92 float blueDiscRampX = 0;
93 float blueDiscRampY = 72.20;
94 float redDiscRampX = 140.02;
95 float redDiscRampY = 72.20;
96
97 float northRollerX = 110.98;
98 float northRollerY = 139;
99 float westRollerX = 139;
100 float westRollerY = 110.98;
101 float eastRollerX = 29.43;
102 float eastRollerY = 1;
103 float southRollerX = 1;
104 float southRollerY = 29.43;
105
106 float redHighGoalX = 17.78;
107 float redHighGoalY = 122.63;
108 float redHighGoalZLow = 25; // lowest scoring point
109 float redHighGoalZMid = 30.5;
110 float redHighGoalZHigh = 35.5; // Highest scoring point
111 float blueHighGoalX = 122.63;
112 float blueHighGoalY = 17.78;
113 float blueHighGoalZLow = 25;
114 float blueHighGoalZMid = 30.5;
115 float blueHighGoalZHigh = 35.5;|
116 // Low Z and High Z is for aiming. EX if Turret aim is between Zlow < Yangle < ZHigh then Fire
```

Location Code

Methods Of Tracking

```
while(true) {
    velocityX += (inertialSensor.acceleration(xaxis)/0.0254);
    velocityY += (inertialSensor.acceleration(yaxis)/0.0254);
    // 0.0254 is for inches per second conversion
    positionX += velocityX*0.0015;
    positionY += velocityY*0.0015;
    threadLoopCount += 1;
    wait(1.5,msec);

    /* Psuedo code
    Get gyroscope date
    Get acelerometer and speed data a = v/t | x = t*v
    Translate acelerometer data to Position data
    wait 250 msec to loop again
    */
}
```

Now onto Sensor history.

Originally I was just going to use the Inertial Sensors

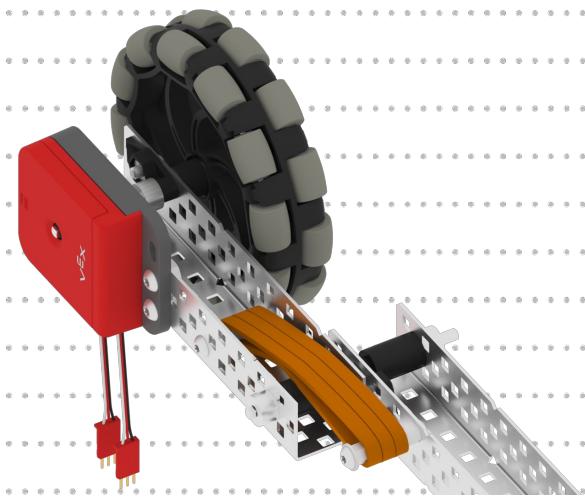
Accelerometer and gyroscope for odometry. On paper it would all line up. You would turn the Acceleration into velocity then turn that velocity into position (Code for calculations on the left). The Accelerometer gives X and Y acceleration so no extra calculations needed.

Pre-Coding: Odometry Part 4

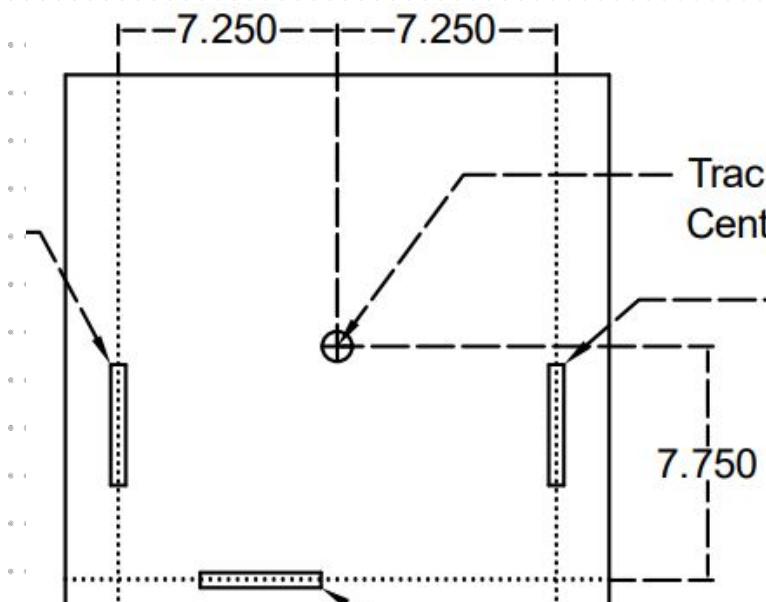
The issue with using the Inertial sensor was that it was a V5 sensor. As I mentioned before the V5 system has a minimum refresh rate of 10ms which was too slow for desired accuracy.

An approach last year I tried for partial odometry was using the encoder values of the vex V5 motor. From research it had a 5ms refresh rate which was double accuracy. To get distance from degrees I had to do $(1/360) * (\text{Left and right motor encoders averaged}) * \text{wheels circumference}$. On paper this would seem accurate. One issue that the Vex motors have when experiencing high resistance is motor slippage. Once motor slippage occurs all odometry is void. The calculations also miss the fact when getting pushed it can nudge one side more than the other and the calculations wouldn't be able to account for this.

When i was told about odometry for creating a GoForwardPiD function he mentioned using a drag wheel with a encoder. From looking at the VEX website for coding information about the encoder. It showed this design on the right. It uses a rubber band suspension to keep the wheel on the ground at all times. There wouldn't be any slippage due to not having any motor. The best part about it is that the encoder bypasses the V5 system background calculations so Its Refresh rate is 0.6ms~



Tracking Wheel Example

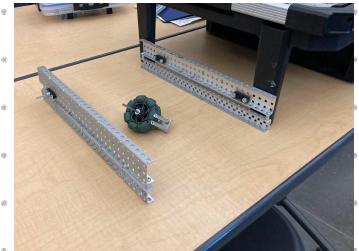


Tracking wheel Decision

When doing further Odometry research I found PDF by 5225 E-bots titlons on Odometry. The paper uses the previously mentioned track wheels so I'm on the right track. It also shows positions for drag wheels which I will show on the left here. The track wheel placement makes a lot of sense. Left and right can be used for Forward and turning tracking and the opposite wheel can be used when the robot is pushed

Meeting #8

Pre - Meeting Plan



Team Members	Tasks	Obstacles	Design Process step
Ethan	Build the chassis	All teammates not present	Build a Prototype
Josh	Organize the bin and finish research	All teammates not present	Background Research
Aidan	Working on the notebook and talking with other teams	All teammates not present	Build a Prototype

Meeting #8 Continued

Post Meeting Info

Meeting Info
Location School:
(VANCOUVER iTech
Prep)
Attendees: Ethan, Josh,
and Aidan
Duration 4:05 - 6:00:pm
Date: 10/11/22

Robot Today



Team Members	What got done	Unresolved Problems
Ethan	Built gearing system for the chassis	Chassis not built
Josh	I took all the screws out for organization	Not done with screw organization
Aidan	Improved odometry wheel design	Put code in the notebook

Pre-Coding: Odometry Part 5

Calculations

Now that we have how we are going to track motion. We need to transfer it our cartesian coordinate map. This process gave me a couple headaches. So lets describe how to get it. Our factors we have are Distance (D), Slope (M), Beginning X (X1), and Beginning Y (Y) and I need to get End X (X2) and End Y (Y2). I asked a friend and a family member and they couldn't help. I somehow stumbled upon the answer when reading one of my textbooks. To get End X you need to do $X1 + (\sin(\arctan(M)) * D)$. Now for end Y the calculations are $Y1 + (\cos(\arctan(M)) * D)$. The issue with Sine and Cosine is that they don't work past 180 degrees. To fix this I will add a If else statement so the Program to determine weather to make the result Negative or Positive. Here is the code for this is Below

```
if((180 < degHead) && (degHead < 360)) {  
    positionX += (dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(((encoderVL+encoderVR)/2 - diffrence) - lastEncoderL))-1);  
    positionY += (dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(((encoderVL+encoderVR)/2 - diffrence) - lastEncoderL))-1);  
}  
else {  
    positionX += (dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(((encoderVL+encoderVR)/2 - diffrence) - lastEncoderL));  
    positionY += (dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(((encoderVL+encoderVR)/2 - diffrence) - lastEncoderR));  
}
```

Odometry code

Encoder VL and VR are the current encoder values. In this screenshot lastEncoderL isn't used. It uses the calculations (lastEncoderL + lastEncoderR - pDiffrence). The reason I didn't show it hear so it would fit on the page. The variable difference is simply EncoderVL - EncoderVR. This calculation covered along with degHead which I will cover next.

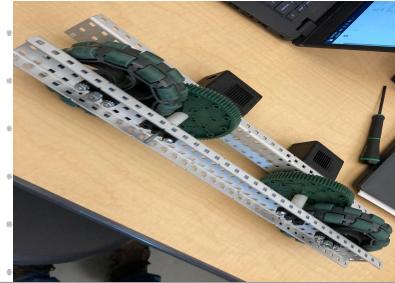
degHead is calculated by the turning odometry. The turning Odometry is taken From my last years turning code. It used a vague circumference of the bot and calculated degrees turning to inches. I will use similar calculations. The newer more accurate circumference is the wheels distance to tracking center. After Getting the circumference it does the following calculations below.

```
410     diffrence = encoderVL - encoderVR;  
411     degHead += (((2*M_PI*trackingOffsetL)/360)*((diffrence)*(dragWheelCirc/360)))*(180/M_PI);
```

$2\pi \text{TrackingoffsetL}/360$ get the circumference and gets the distance of each degree. Difference is simply how much it turned if it didn't turn at all the overall number will be 0. Next is the Drag wheel which is 2.75 inches in diameter to transfer into degrees. $180/\pi$ is to transfer the units into Degrees from Radians.

Meeting #9

Pre - Meeting Plan



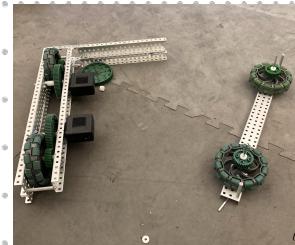
Team Members	Tasks	Obstacles	Design Process step
Ethan	Buildin' a robot	N/A	Build a Prototype
Josh	Organising the toolbox	N/A	N/A
Tyler	Buildin' a robit	N/A	Build a Prototype
Koltin	Organise the toolbox	N/A	N/A
Aidan	Do some programming and possibly some strategy	No Chassis/Leaving early	Building a Prototype

Meeting #9 Continued

Post Meeting Info

Meeting Info
Location School:
(VANCOUVER iTech
Prep)
Attendees: All members
Duration 4:05 - 6:00:pm
Date: 10/12/22

Robot Today



Team Members	What got done	Unresolved Problems
Ethan	Finished Side of Chassis	N/A
Josh	Organized Bin	N/A
Tyler	Started Side of Chassis	N/A
Koltin	Organized Bin	N/A
Aidan	Added code to the Chassis.	Strategy

Design Process: Chassis Design

Gearing and side skirt

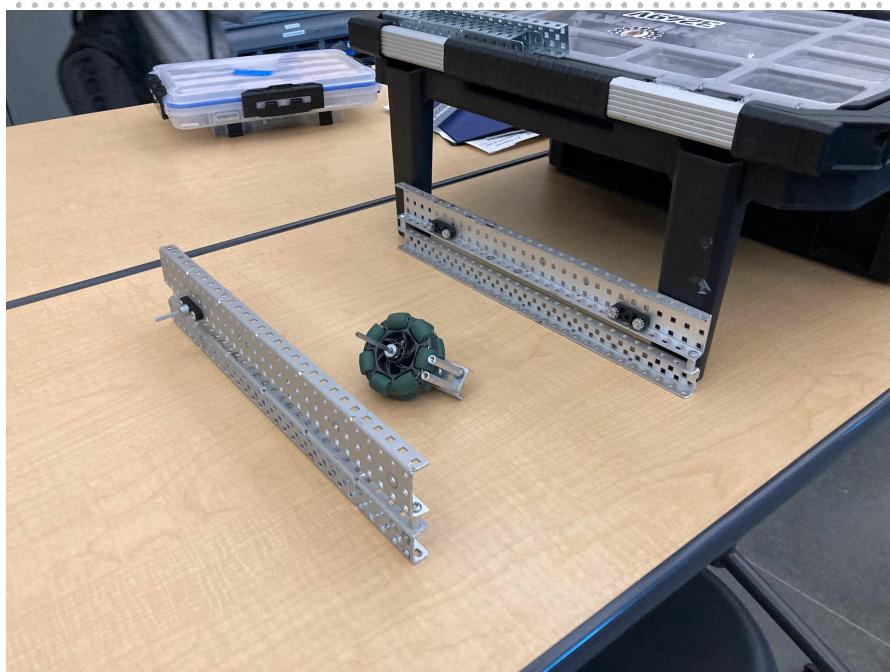


Gearing

As of 10/12, we started building the chassis. As in our design, we decided to gear for speed, and use full omni wheels. We are using a 3 by 35 c channel as our gearing frame, and a 60 to 36 tooth gear ratio.

Side skirts

The disks are quite the obstacle in the field, and could offset our odometry, so we have designed the robot's height with the idea of another c channel sitting a few millimeters off the ground to keep rings from entering the underside of the robot



Originally, we were going to use 30 by 2's but upon further designing, we decided to go with a full '35' frame, and keep our intake system on the inside of the robot. I spent most of practice experimenting with the practicality of different gear ratios, and decided on 60 - 36. This seemed reasonable for snappy acceleration while still being able to have a quicker than average speed.

Meeting #10

Pre - Meeting Plan



Team Members	Tasks	Obstacles	Design Process step
Ethan	Building chassis	N/A	Build a Prototype
Josh	Organizing box	N/A	N/A
Koltin	Organizing box	Leaving Early	N/A
Aidan	Making intake/Code	N/A	Build a Prototype

Meeting #10 Continued

Post Meeting Info

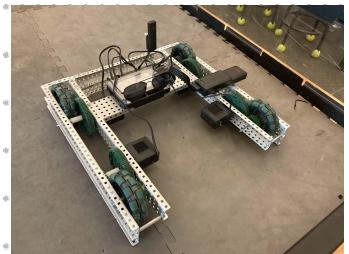
Meeting Info
 Location School:
 (VANCOUVER iTech
 Prep)
 Attendees: Ethan Josh
 Koltin and Aidan
 Duration 4:05 - 6:00:pm
 Date: 10/13/22



Team Members	What got done	Unresolved Problems
Ethan	First iTech Skrimage and First Chassis	
Josh	Organization	Organization
Koltin	N/A	
Aidan	First iTech Skrimage and First Chassis	

Meeting #11

Pre - Meeting Plan



Glasses provided by 4591V

Team Members	Tasks	Obstacles	Design Process step
Josh	Finish organization	Not all teammates present	N/A
Aidan	Gearing the Chassis	Not all teammates present	Build a Prototype

Meeting #11 Continued

Post Meeting Info

Meeting Info
Location School:
(VANCOUVER iTech
Prep)
Attendees: Aidan and
Josh
Duration 4:05 - 6:00:pm
Date: 10/14/22

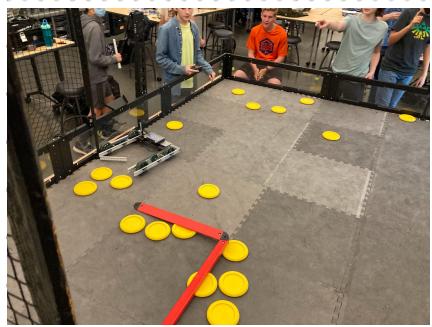
No photo D;

Team Members	What got done	Unresolved Problems
Josh	Organization, and Gearing the Chassis	Organization, After meeting photo
Aidan	Organization, Gearing the Chassis, and Creating the Intake	Intake

Design Process: Chassis Design

First Scrimmage

On meeting 10 we had the first scrimmage with 3249T. We both only had a chassis but it was a fun experience. We were only able to score low goal zones due to just having push bots. Tyler had to leave early due to him not having a ride home. So Ethan was the driver here. Our bot scored because our chassis goes very low to the ground.



End of Match for my 3249T



End of Match for my team

3249T's driver for today was Ashar one of my previous builders. He was the only one of his team here for the practice. He scored because he quickly installed a low C-Channel right before the scrimmage after I pitched the idea to him. For this scrimmage my team was on the Blue team and his team was on the red team.

During the match we pushed each other around and tried scoring discs. Ashar and Ethan both did a good job for what they had to work with.. The final score was 13 to 9 with my team winning. Afterwards I wrote some notes me and Ethan came up with.

First note which was a suggestion by 3249U's Joseph was to add locked omni wheels to reduce drift. Locked Omni Wheels are Omni-wheels with zip ties on the mini wheels to prevent the from moving sideways. From what I've heard Locked omnis have better friction than regular friction wheels.

The second note was to reduce turning speed. The robot moves ridiculously fast and turning is even faster. So a very slight amount of turn does a lot for the bot. This can easily be fixed in coding by halving the amount of input from the turn joystick.

Something I observed prior to the Scrimmage was the robot got stuck on the low goal zone brackets. I would have to talk to my others teammates on how we should address this problem.

Each note observed I feel like the problems will be fixed as the robot is more fleshed out. The drift may be fixed when the robot gets more weight and more driver experience may be able to counteract high turn speeds.

Pre-Coding: Odometry Part 6

For the last part of Odometry will be the back wheel. From my current understanding it allows the robot to tell when the robot is being pushed from the side. As it is placed on a different axis I need to change the calculations. For the X axis the back wheel will use Cosine instead of Sine. The same is for the Y axis. In the code I'll simply add the X and Y changes from the back wheel on the same line as the X and Y axis for other wheels. I will not show the line of code because of its length.

Debugging

Now how will the code run in alongside the program? The answer is Threads. Threads. The vex brain can run around 80 Threads simultaneously. Threads are like branches of a program that can run alongside the program without pausing other processes. My code will be using 2 Threads. One for the Odometry, and Another Thread for the Aimbot that can be toggled off and on. Threads in the program are placed after all the functions and right before autonomous.

Before recapping all of Odometry I want to briefly go over how I plan to calibrate and debug the code. I have two ideas for this. One is a basic approach I have done last year. At the end of last year I was making a goForward function and I needed to know its accuracy. So I used the controllers screen to tell me the distance the robot was moving and the robots heading. I got the distance from the motors encoders and the heading from the VEX inertial sensor. This sensor setup was extremely inaccurate for reasons I explained earlier. Going back to the screen setup. I can exchange the distance for X and Y, and just get heading from the odometry.

The second idea which is a far more difficult approach. I want to try to make a map of the field from previously mentioned variables with the bot on it as it does on the field. This is far more difficult especially and most likely will be shown only on the brain because the controller screen is too small. Ideally I can do both approaches first one for the controller and the second one for the Brain if I got some free time. Back to debugging.

Pre-Coding: Odometry Part 7

To code method one onto the controller. I initially thought of just putting the code in the Control code. Now thinking this wouldn't work because I will need to do more tests from the autonomous than in the control period. Putting a function into every function that prints the code would be tedious. In the end I decided to make a 3rd thread for the display.

Display Code

Here is a basic overview of the code on the right here. It first clears to screen to void all previously printed things. It creates a new line just to center the text. I have separate lines to print text and to print values as all values within "" will be text. I decided to include battery power for basic quality of life. Even without debugging it is useful to have. At the top of the thread is what the display should look like. The wait time for this function doesn't need to be ridiculously fast like any other odometry thread so it is just 10 milliseconds.

```
void thread3() { // Controller screen thread
    while(true) {
        /* Controller Screen Example
        / Position X: 19 Y: 18
        | Heading: 45
        | Power: 89%
        \
        */
        Controller1.Screen.clearScreen();
        Controller1.Screen.newLine();
        Controller1.Screen.print("Position X: ");
        Controller1.Screen.print(positionX);
        Controller1.Screen.print(" Y: ");
        Controller1.Screen.print(positionY);
        Controller1.Screen.newLine();
        Controller1.Screen.print("Heading: ");
        Controller1.Screen.print(degHead);
        Controller1.Screen.print("Power: ");
        Controller1.Screen.print(Brain.Battery.capacity());
        Controller1.Screen.print("%");
        Controller1.Screen.newLine();
        wait(10,msec);
    }
}
```

Now that I have defined how the robot displays its information. I can now explain how I can debug with this. When debugging the odometry code if god forbid its not correct. I simply cross reference the code with real measurements. Once I can prove the odometry calculations are correct. I can move onto other important calibrations required for Odometry

Meeting Formatting

We have used a basic Meeting format for when we have practice (Pages 60 and 61). It was the same format the last years 3249V was using. In our processes I've adapted it to add things such as pictures of the robot. Me and my teammates were doing an informal lunch meeting about the format and why we need to change it.

The overall format of both pages of meetings has changed. Specifically at the top. I moved anything that went outside the dotted lines into the dotted lines, to make the notebook closer to what a physical notebook look like. Now that the top of the page has two things. I put the heading and subheading on the left and the other element on the right. In pre meetings it is the robot at the beginning of the meeting, and in the post meeting is the meeting information.

The old format worked for basic meeting practices and was mainly paired with a images page. Our team decided to incorporate images into the Post meeting information page. We made space by shrinking the Unresolved problems and what got done columns.

In the Pre meeting notes. We merged the Tasks and obstacles column as most the time we don't have an obstacle for the day. In some of the new pages from the old format on the top right we kept the picture of the bot at the beginning of practice. This is to help track our progress as we build our robot. I removed this on the post meeting as to make more room for the meeting information.

Overall this change will help us be more organized and help track our progress better. The new images section will help us get more images in the notebook such as changes to a certain mechanism or mechanisms not yet attached.

Old Format

Meeting

Pre - Meeting Plan

Team Members	Tasks	Obstacles	Design Process step
Ethan			
Josh			
Tyler			
Koltin			
Aidan			

Example

Old Format

Meeting # Continued

Post Meeting Info

Meeting Info
 Location School:
 Attendees: All members
 Duration
 Date:

Team Members	What got done	Unresolved Problems
Ethan		
Josh		
Tyler		
Koltin		
Aidan		

Example

Meeting

Robot beginning of
Practice

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan		
Ethan		
Josh		
Koltin		
Tyler		

Meeting # Continued

Post Meeting Info

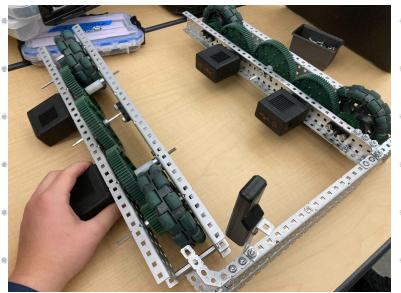
Meeting Info
Location School:
Attendees:
Duration:

Team Members	What got done	Unresolved Problems	Images
Aidan			
Ethan			
Josh			
Koltin			
Tyler			

Meeting #12

Robot beginning of
Practice

Pre-Meeting Plan

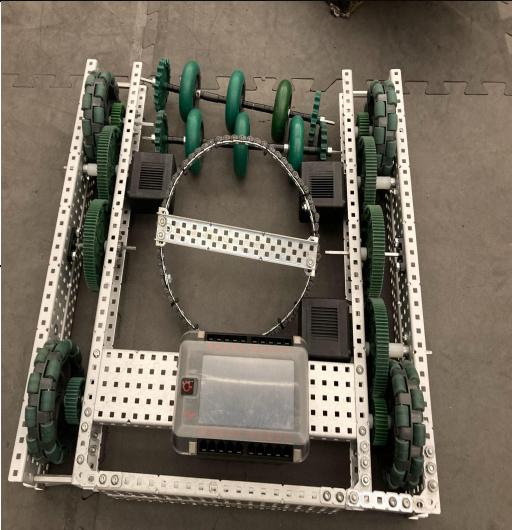


Team Members	Tasks and Obstacles	Design Process step
Aidan	Coding, working on flywheel	Build a Prototype
Ethan	Finish chassis and begin turret	Build a Prototype
Josh	Building odometry wheels	Build a Prototype
Koltin	Building odometry wheels	Build a Prototype
Tyler	Strategy and notebook	Build a Prototype

Meeting #12 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Preparatory
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Began working on flywheel	Not fully geared yet	
Ethan	Began working on the turret	No turning mechanism	
Josh	Worked on chassis	No odometry wheel	
Koltin	Worked on chassis	No odometry wheel	
Tyler	Worked on chassis	Chassis not fully redone	

Pre-Coding: Odometry Part 8

Recap

In this section of Pre-coding I covered all of the odometry code. I covered methods I thought that would work. Methods that I hope will work. How I will debug the code, etc. For the last part here I will go through all of the odometry code again and some changes I made after I put the code into here.

On the next page is all the odometry code. In order to fix the screen I had to port it into a google doc and shrink the font. The doc also makes text that is too long enter the next line so the longest of lines can still be represented.

Difference is EncoderL - EncoderR. Difference is used twice in the odometry. To tell how much the robot is turning. Difference is then used to make sure drift or turning is not added to the distance traveled and mess up X and Y.

I don't use pure encoder values and make a variable that represents the encoder values because of simplicity. Last values are used mainly to get the distance travelled. Distance is $\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}$ x1 and y1 are the previous encoder values.

As mentioned before the reason why I had to have angles between 180 and 360. The function sine and cosine don't work past 180 when it comes to negative values. Cosine only gives negative values and Sine only gives negative values past 180 degrees.

At the very end of the function the encoders are cleared and the loop can begin a new. The 1 msec wait is so the program doesn't do the calculations at a speed which would be incomprehensible to both the programmer and the robots sensors. It would create a unbelievable amount of wasted calculations. As said before the max speed of 3 sensors used here is ~0.6ms. I decided to use 1ms as it was simpler for me.

Last outside variable here is the drag wheel circumference. The drag wheel is just slang for odometry wheel. I plan to use a 2.75 inch small omni wheel. The circumference is just 2.75π put into a variable for ease of use.

Pre-Coding: Odometry Part 9

```
void thread1() { // Position thread If it ever breaks we dead
    double diffrence = 0;
    double pDiffrence = 0;
    double lastEncoderL;
    double lastEncoderR;
    double lastEncoderB;
    double encoderVL;
    double encoderVR;
    double encoderVB;
    lastEncoderL = encoderL;
    lastEncoderR = encoderR;
    lastEncoderB = encoderB;
    while(true) {
        encoderVL = encoderL.position(degrees);
        encoderVR = encoderR.position(degrees);
        encoderVB = encoderB.position(degrees); // what am i gonna do wit dis
        diffrence = encoderVL - encoderVR;
        degHead += (((2*M_PI*trackingOffsetL)/360)*((diffrence)*(dragWheelCirc/360)))*(180/M_PI);
        // tracking offset L and R should be the same no matter what
        if((180 < degHead) && (degHead < 360)) {
            positionX += ((dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(((encoderVL+encoderVR-diffrence)/2)-
            (lastEncoderL+lastEncoderR-pDiffrence)))^(-1))+((dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(encoderVB-
            lastEncoderB)^(-1))));
            positionY += ((dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(((encoderVL+encoderVR-diffrence)/2)-
            (lastEncoderL+lastEncoderR-pDiffrence)))^(-1))+((dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(encoderVB-
            lastEncoderB)^(-1))));
        }
        else {
            positionX += ((dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(((encoderVL+encoderVR-diffrence)/2)-
            (lastEncoderL+lastEncoderR-pDiffrence)))^(-1))+((dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(encoderVB-
            lastEncoderB))));
            positionY += ((dragWheelCirc/360)*(180/M_PI)*((sin(degHead)*(((encoderVL+encoderVR-diffrence)/2)-
            (lastEncoderL+lastEncoderR-pDiffrence)))^(-1))+((dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*(encoderVB-
            lastEncoderB))));
        }
        lastEncoderL = encoderL;
        lastEncoderR = encoderR;
        lastEncoderB = encoderB;
        pDiffrence = diffrence;
        encoderL.setPosition(0, degrees);
        encoderR.setPosition(0, degrees);
        encoderB.setPosition(0, degrees);
        wait(1,msec);
    }
}
```

Pre-Coding: Move To Point

Overview

Now that we know how we can track our movement. It's time to actually move! I will cover how we will move during the autonomous and the Skills Challenge. I'll cover how we will move during the control period after Pre-Coding.

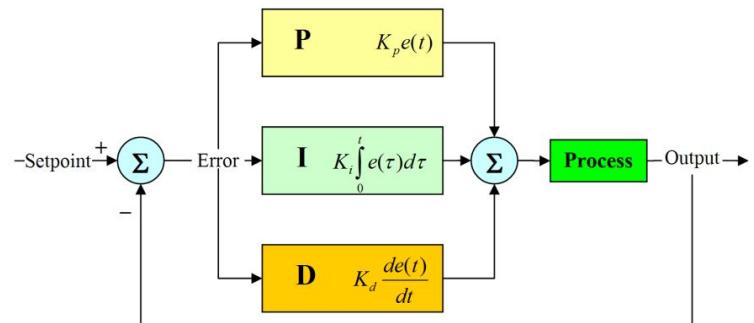
In this section I will cover how we will turn the chassis and move it back and forward to get to a location. `moveToPoint` is the main function in the program that goes from Point A to Point B. Point A and Point B are points on a place. This plane is our Cartesian Coordinate we discussed in the Odometry section.

`moveToPoint` is our primary move function as its so much simpler to code with and I can customize it to have a end heading. Curves within movement and other cool things. There will be other functions such as moving the turret, flywheel, etc. Something me or one of my teammates may do is have a drawable strategy program just like 3249V's team last year. `moveToPoint` is a function that can greatly simplify the coding one of my other teammates or myself has to do to accomplish this goal.

Explanation

The math in this section will dip into realms I fully don't understand yet. I am currently taking Algebra 2 and maybe precalc midway through the year so my mathematical knowledge is limited. The main component I will use to move is the PID controller. PID stands for Proportional Integral Derivative. A common word I will use throughout this section will be "Error". Error is not like computer error. In this case error can mean the distance it needs to move, or how much is left to move.

Let's simplify what the PID controller does. The PID is a control loop to maintain constant. It's used within most types of machinery. Here's an example. You need to keep the temperature of your house the same. The thing you have control over is your homes heaters.



Meeting #13

Robot beginning of
Practice

Pre-Meeting Plan

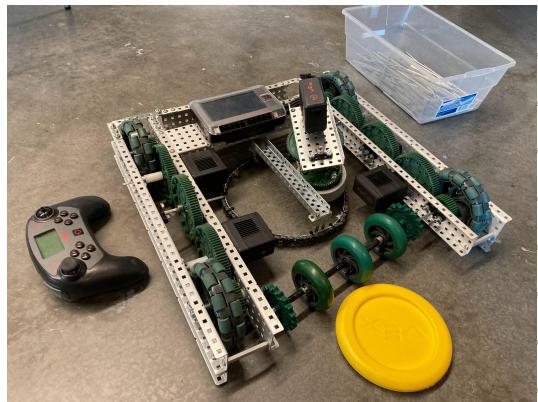


Team Members	Tasks and Obstacles	Design Process step
Aidan	Began flywheel, Begin making odometry wheels / No field or access to club part storages.	Build a prototype
Ethan	Work on Intake/Only there for 30 minutes	Build a prototype

Meeting #13 Continued

Post Meeting Info

Meeting Info
Location School: iTech Preparatory
Attendees: Aidan, and Ethan Duration: 10:35 - 2:05

Team Members	What got done	Unresolved Problems	Images
Aidan	Made Flywheel V1	Odometry wheels	 
Ethan	Began intake	Intake not fully complete	

Meeting #14

Robot beginning of
Practice

Pre-Meeting Plan

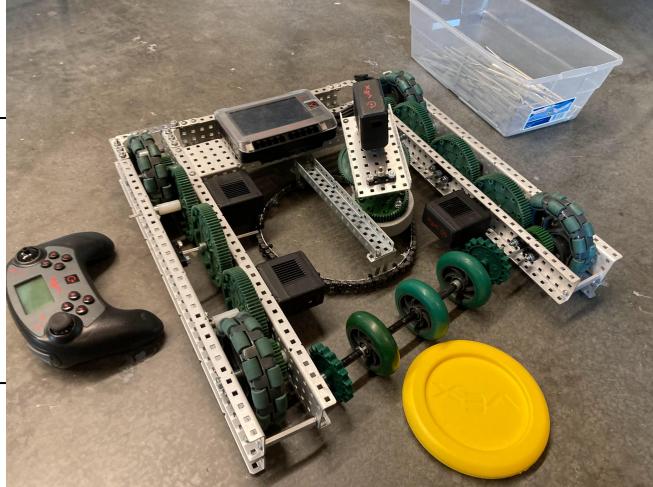


Team Members	Tasks and Obstacles	Design Process step
Aidan	Working on the flywheel / odometry	Build a Prototype
Ethan	Work on the turret and flywheel	Build a Prototype
Josh	N/A	N/A
Koltin	N/A	N/A
Tyler	Practice driving chassis	Build a Prototype

Meeting #14

Post Meeting Info

Meeting Info
Location School: iTech
Preparatory
Attendees: All Members
Duration: 2:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Fixed Chassis/ Odometry wheels	Odometry wheels not attached	
Ethan	Began working on intake	Intake not complete	
Josh	N/A	N/A	
Koltin	Creating custom Flywheel motor	No successful test	
Tyler	Odometry wheels	Odometry wheels not attached	

Pre-Coding: Move To Point Part 2

It's very cold outside so you want to keep the home warm. I set the thermostat to a cool 60 degrees because I hate the heat. A base Program you may write to adjust the temperature is If(temp<60) then {Heaters on} and if else(60<temp) {heaters off}. The temperature will constantly fluctuate between 59 and 61. It's never going to be on target. The PID is meant to fix that. The PID in vex is far simpler than one used in actual machinery. I don't fully understand how to calculate the PID but I for sure know how to calculate the vex one.

Each letter in PID has its own calculations and its own factor that I will need to calibrate. The P or Proportional scales proportionally to the Error factor. The calculations for Proportional is Error * pFactor. The I means integral. Integral scales with the error overtime. It mainly steps in when P cannot get to the exact point so I will kick in to make it as close as possible. The integral factor is the smallest of the 3 and is calculated with two lines. Integral = Integral + Error and then in when combined it is Integral*iFactor. The last factor will be the D factor or the Derivative. The derivative looks at the slope of the Error and then acts proportionally to that. More of a change means a stronger derivative. Derivative factor is calculated by (lastError-Error)*dFactor.

Coding

Now how does this translate to code? As shown in the example it's used to stabilize something. In the code I want to "stabilize" the robots position to a new X and Y coordinate. Yeah that was bad. Anyway to code this I need to define our variables. These variables are the same across all PID calculations but will vary function to function. PID's require a lot of calibration so a lot of tests are needed after Pre-Coding.

The variables the bot has when tracking movement is X,Y, and Degrees. Degrees will be used when making a turn PID but right now I will make a movePID. First 3 factors are going to be Kp Ki and Kd. These 3 are the changing factors between each function. They influence the output of the PID and can control its overall accuracy. In coding tests once Odometry wheels are on the robot I will show I will calibrate these. As of now they'll be set as 0. Next is the error variable. For move forward the difference between current position and desired position is the error. Next are proportional integral and derivative which I will leave blank. I described previously how the program calculates it. The last variables in the actual PID calculations are lastError and fixing. lastError is the last error of the cycle. The first loop will not have a lastError variable. Fixing is the Proportional calculations + the Integral and derivative.

Design Process: The Flywheel

Flywheel Version 1.

The first Flywheel is a really good first attempt at a flywheel in my opinion. It was geared for speed at 84:12 or in base terms 7:1. In basic it goes fast. The max speed which I never tested was 4200rpm. I never pushed the motor past 3600 rpm. Today when I tested it after working on it during my free periods I forgot to fully screw something in and it flew off. I almost lost my eye today if it wasn't the fact I was smart enough to wear safety goggles. Remember to stay safe.

This is the final design that I fully completed. On some images of the chassis I used two C-channels to put it together but it was too unstable for proper use. I remade it with 2 5x10 C channels and 3 inches for standoffs. I use one out of 4 of the schools grey flywheels as the flywheel. I used a 84 tooth low strength gear geared to a 12 tooth low strength gear that is connected to the flywheel. At meeting 14 Ethan replaced the low strength 12 tooth with a Metal high strength gear with the same number of teeth. During meeting 14 where I could test it my teammates were able to launch it from one side of the field to the other.



My teammate Koltin plans to copy something 3249U is doing by modifying the motors mechanical internals to spin without a flywheel. This is somehow legal because it doesn't directly modify the electronics. All parts used in the process are VEX parts. The motor without gearing spins at around 3600 RPM. I am personally skeptical on how it may be better as seen from 3249U testing it didn't shoot that far compared to the flywheel I made. Next tuesday when everyone is there I will see how it fairs compared to Flywheel Version 1. Right now the design for the flywheel is between my version one or Koltins direct drive 3600 rpm. This may all change once we get Flywheel weights and Ball bearings for shafts.

Pre-Coding: Move To Point Part 3

The fixing variable is set in RPM. This RPM goes straight into the 4 chassis motors and makes the chassis move. The chassis doesn't stop until the PiD is finished.

The PID won't stop until it has exactly reached its target. This here could be fairly troublesome as the PID may just make the smallest corrections that the motors cannot make to get an exact amount. I made some code to prevent this from happening. I made 3 different variables for this purpose in every PID. errorAverage, lastErrorAverage, and loopCount.

The error average doesn't do anything but be a average. It is calculated at the end of each PiD. The calculations for the average is $(\text{errorAverage} + \text{error} + \text{lastError}) / 3$. This average is cleared every 75 loops. The robot gets the loops from the loopCount variable. When it is cleared the last average is also recorded. At the top of the PID each loop before it does any calculations it checks if the lastErrorAverage is within 0.01 inches of the last error. Of course I use the absolute value of last error and lastErrorAverage so negative values can be used here. Once the loop is broken either by getting the exact value or by the averaged loop break. All the motors stop and the program continues as normal.

Debugging the PID

Before moving onto specific PiD coding I want to cover how I will debug the PID. From what I have heard and seen from 3249V last year it takes a lot of time. I have to calibrate each factor by itself and this will take a lot of time. There are in total 4 PID's which I have to calibrate each factor for each PID.

The methods to calibrate is simply to run the PID with a single factor. P is calibrated first by running tests and changing P very slightly to get some accuracy. Then I is calibrated the same way as P. Lastly D is calibrated in the same way as P and I. If I were on a much harder time crunch say the october 29th tournament I would use a PI control algorithm so I can at least have a basic autonomous and skills.

I have designed a simple strategy that incorporates all 4 PID's. The four PID's are movePID, turnPID, turretXPID, and turretZPID. turretXPID and turretZPID are apart of the auto aim which needs priority. movePID and turnPID are the second priority due to only being involved with autonomous functions.

Pre-Coding: Move To Point Part 4

Coding movePID

The movePID is simple. The function starts with the parameters of X and Y. Error is then calculated using the distance formula. X1 and Y1 are current position in this case. After fixing is calculated all 4 motors are set the velocity. The left motors turn forward and right motors turn in reverse. The reason why right motors go in reverse is that they're placed mirror to left motors. 1.5 seconds later the PID does its job then the loop repeats until it's on target or the error break system goes in place.

```
204 void movePiD(double X, double Y) {  
205     double Kp = 0;  
206     double Ki = 0;  
207     double Kd = 0;  
208     double error = (sqrt(pow(X - positionX,2)+pow(Y- positionY,2)));  
209     double proportional = 0;  
210     double intergral = 0;  
211     double derivitave = 0;  
212     double lastError = 1;  
213     double fixing = 0;  
214     double errorAverage = error;  
215     double lastErrorAverage = 0;  
216     double loopCount = 0;  
217     while (abs(error) <= 0) {  
218         if ((abs(lastErrorAverage) - 0.01) < (abs(lastError) < (abs(lastErrorAverage) + 0.01))) { // preventing infinite correct loop  
219             | break;  
220         }  
221         proportional = error*Kp;  
222         intergral = intergral + error;  
223         derivitave = error - lastError;  
224         fixing = proportional+(intergral*Ki)+(derivitave*Kd);  
225         leftFrontMotor.setVelocity(fixing,rpm);  
226         leftBackMotor.setVelocity(fixing,rpm);  
227         rightFrontMotor.setVelocity(fixing,rpm);  
228         rightBackMotor.setVelocity(fixing,rpm);  
229         leftFrontMotor.spin(forward);  
230         leftBackMotor.spin(forward);  
231         rightFrontMotor.spin(reverse);  
232         rightBackMotor.spin(reverse);  
233         lastError = error;  
234         wait(1.5,msec);  
235         error = (sqrt(pow(X - positionX,2)+pow(Y- positionY,2)));  
236         errorAverage = (errorAverage+error+lastError)/3;  
237         loopCount += 1;  
238         if (loopCount > 75) { // This is so previous larger errors don't break the infinite loop fix  
239             | lastErrorAverage = errorAverage;  
240             | errorAverage = 0;  
241         }  
242     }  
243     leftFrontMotor.stop();  
244     leftBackMotor.stop();  
245     rightFrontMotor.stop();  
246     rightBackMotor.stop();  
247 }
```

Pre-Coding: Move To Point Part 5

Coding turnPID

turnPID has some complexity compared to the movePID. The only parameter is degr. degr is the next desired heading. Error is calculated by degHead - degr. After fixing is calculated the program decides weather it's best to turn left or right. It decides this by seeing if degr is less than 180 degrees than it turns left. If it is more than that it turns right. Instead of changing motor direction. The program is given negative velocity values. Afterwards the robot does the same left forward and right reverse to turn. Of course now there are negative velocities. Afterwards the code does its error break system. The Code is on page 78 because It doesn't fit in one page.

Coding turretXPID

This function differs a lot from the other two. The function does the normal PID calculations. Its single parameter is degr which is set by the autoAim Thread. After fixing it decides weather to turn Left or Right in the same way the turnPID does. After that it makes the turretXMotor spin for 5 msec. This process in plan doesn't require an encoder so I use the Motor encoders which refresh at 5ms. The PID itself doesn't repeat the autoAim thread makes it repeat. I will explain the autoAim thread after this pre-coding section. As of now this may be confusing to most who read. I will reference this explanation later on in the notebook. The code is on Page 79.

turretZPID has been removed due to switching from having a vertical control motor for aimbot to controlling the flywheels velocities for the ambot. I will cover this more on the autoAim section. It acts similar very turretX with some limits. The limits being that the turret can aim directly into the robot because that would break the turret. As well as it can't go more than around 60 degrees without breaking the size constraints. I will post its old code on the same page as turretXPID

```

void turnPID(double degs) {
    double factorP = 0;
    double factorI = 0;
    double factorD = 0;
    double error = degHead - degs;
    double integral = 0;
    double derivative = 0;
    double pError = 1;
    double fixing = 0;
    double errorAverage = 0;
    double pErrorAverage = 0;
    double loopCount = 0;
    while (abs(error) <= 0) {
        if ((abs(pErrorAverage) - 0.01) < (abs(pError) < (abs(pErrorAverage) + 0.01))) { // preventing infinite
        correct loop
            break;
        }
        integral = integral + error;
        derivative = error - pError;
        fixing = (error*factorP)+(integral*factorI)+(derivative*factorD);
        if (degs > 180) { // if it is better to turn left it will, this is to make the function more efficient
            leftFrontMotor.setVelocity(-fixing,rpm);
            leftBackMotor.setVelocity(-fixing,rpm);
            rightFrontMotor.setVelocity(fixing,rpm);
            rightBackMotor.setVelocity(fixing,rpm);
        }
        else {
            leftFrontMotor.setVelocity(fixing,rpm);
            leftBackMotor.setVelocity(fixing,rpm);
            rightFrontMotor.setVelocity(-fixing,rpm);
            rightBackMotor.setVelocity(-fixing,rpm);
        }
        leftFrontMotor.spin(forward);
        leftBackMotor.spin(forward);
        rightFrontMotor.spin(reverse);
        rightBackMotor.spin(reverse);
        pError = error;
        wait(1.5,msec);
        error = degHead - degs;
        errorAverage = (errorAverage+error+pError)/3;
        loopCount += 1;
        if (loopCount > 100) { // This is so previous larger errors don't break the infinite loop fix
            pErrorAverage = errorAverage;
            errorAverage = 0;
        }
    }
    leftFrontMotor.stop();
    leftBackMotor.stop();
    rightFrontMotor.stop();
    rightBackMotor.stop();
}

```

```

void PiDTurretZ(double degr)
{
    double factorP = 0;
    double factorI = 0;
    double factorD = 0;
    double error = 0;
    double intergral = 0;
    double derivitave = 0;
    double pError = 0;
    double fixing = 0;
    error = turretHeadingZ - degr;
    intergral = intergral + error;
    derivitave = error - pError;
    fixing = (error*factorP)+(intergral*factorI)+(derivitave*factorD);
    if (degr > 180) {
        turretMotorZ.setVelocity(fixing,rpm);
    }
    else {
        turretMotorZ.setVelocity(-fixing,rpm);
    }
    turretMotorZ.spin(forward);
    pError = error;
    wait(5,msec);
    turretMotorZ.stop();
    error = turretHeadingZ - degr;
}
void PiDTurretX(double degr)
{
    double factorP = 0;
    double factorI = 0;
    double factorD = 0;
    double error = 0;
    double intergral = 0;
    double derivitave = 0;
    double pError = 0;
    double fixing = 0;
    error = (turretHeadingX - degr);
    intergral = intergral + error;
    derivitave = error - pError;
    fixing = (error*factorP)+(intergral*factorI)+(derivitave*factorD);
    if (degr > 180) {
        turretMotorX.setVelocity(fixing,rpm);
    }
    else {
        turretMotorX.setVelocity(-fixing,rpm);
    }
    turretMotorX.spin(forward);
    pError = error;
    wait(1.5,msec);
    turretMotorX.stop();
}

```

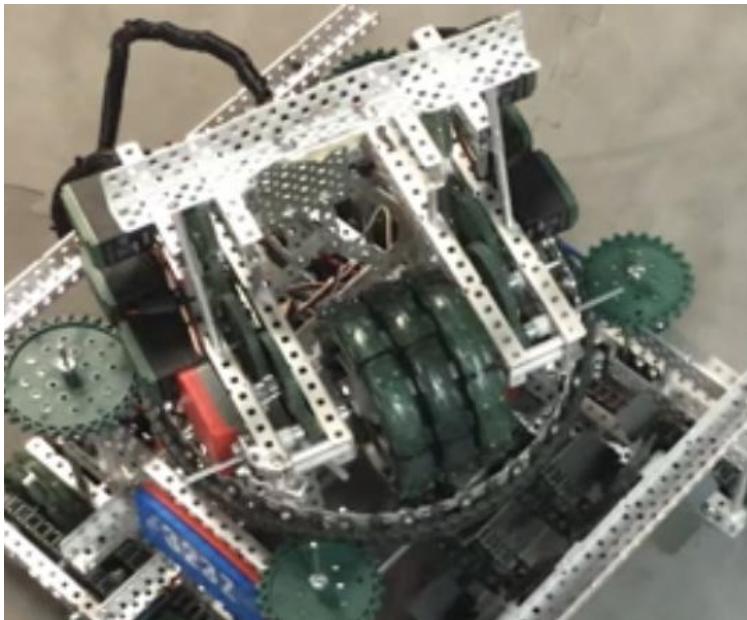
Copy and
paste from
the code
archive ->

Code from
Program ->

Design Process: The Turret

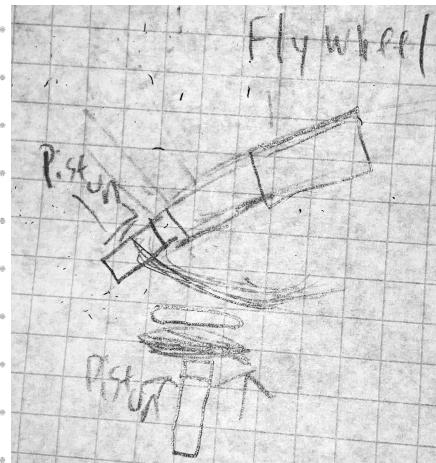
Version 1

This is as of now our current turret on the right. This initial build was built by our builder Ethan Regan. It's based off of the design I researched. This design is from 323Z during the "Nothing but net" Competition. Inside of this turret is a ball launcher and a basic flywheel built with the 393 Motors. My team will make a design that uses the V5 system and the Flywheel. On the left you can see the beginnings of the chain wheel. The design will use one V5 motor with either direct gearing, torque gearing, Or



speed gearing. It all depends on the weight of the flywheel, turret, and roller mechanism. If it is light enough I'll attempt speed gearing. If its too heavy I'll do light torque gearing to maintain flywheel speeds. Our chassis is very fast so that may be an issue for the Aimbot. A concern Ethan and Tyler have expressed is wiring. Specifically how wiring will run through the turret for motors and sensors without damaging it. An idea I came up with that may fix the length issue is having the wires go below the loading platform connected to a loose shaft

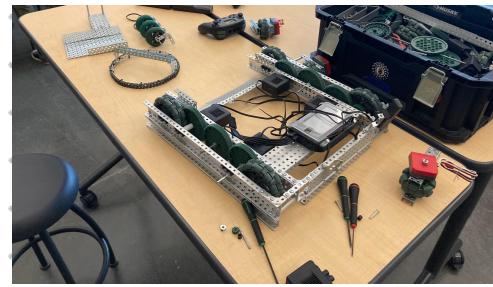
that spins with the turret. Talking about the loading Mechanism. I came up with a loading mechanism that makes can 360 degree aimbot possible. It uses a two stage pneumatic trigger mechanism. The first trigger Raises a disc up to a latch in the flywheel. The next piston pushes the disc into the flywheel. I made a quick sketch on the right of the concept. It is not too scale. I'll try completing the turret next practice when I had 4 hours.



Meeting #15

Robot beginning
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Make the Turret	Create a Prototype

Design Process: The Turret

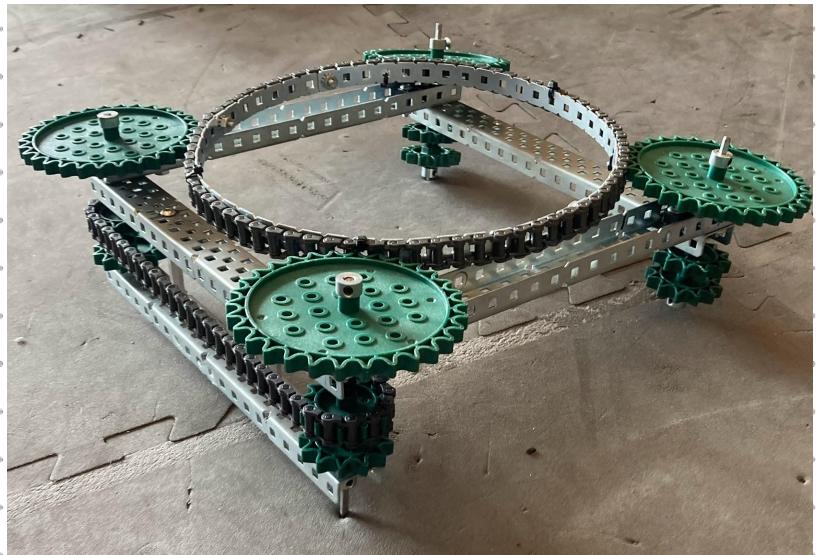
Initial Gearings

Today's practice (Practice 15) made a good starting point for what the turret will begin to look like. The size of the turret's turntable has expanded quite a bit. I put an old image on the left and a new image below for comparison. Along with the sizing up of the turntable. I was able to make the turret wheel much more circular. I did this by simply turning the 4 sprockets. The 1 by 1s are very bendable so over time with use it will become more and more circular.



The gearbase for the turntable comprises of 4 gears chained together with one motor turning the whole thing. The frame is made up of 20 hole Steel C channels. I use 30 tooth sprocket gears to turn the mechanism and 12 tooth sprocket gears to turn the chain system. There are 2 12 tooth sprockets for each 30 tooth sprocket gears. Each 12 tooth (seen below) is spaced with a 4.6mm spacer. In the image below. On the left side is the more complete version with spacers and its overall a much better build. The two sprockets on the right have unintentional Camber.

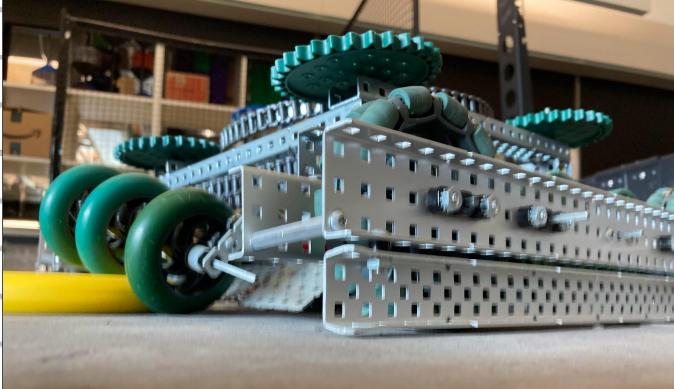
I plan for either Ethan or I to fix the other sides and chain the gearbase back together. Earlier during practice I was able to chain the sprockets together. The camber prevented all the sprockets and chains from spinning correctly. Once the Gearbase is finished we will move onto creating the loading mechanism for discs to enter the flywheel.



Meeting #15 Continued

Post Meeting Info

Meeting Info
Location School: iTech Preparatory
Attendees: Aidan
Duration: 9:00 - 1:00

Team Members	What got done	Unresolved Problems	Images
	Half of the turrets chain system is made.	No motorization or turret bed made yet.	
Aidan			 

Design Process: Odometry Wheels

First Designs

If the V5 Brain is the brains of the robot. The Odometry wheels are its senses. It allows the robot to where it is going and where things are from the Program. As mentioned in the Odometry section of Pre-coding mechanisms are small yet extremely important. I will be going over all aspects of the wheels themselves here.

- There is in total 3 Odometry in the Robot. The Left wheel, Right wheel, and the Back wheel.
- Each are all identical. The key components are the of course the Optical Encoders, and the Wheels. The wheels used here are 2.75 inches in diameter. I am doubtful of this measurement as I have heard that Joseph from 3249U last year measured the 4 inch Omni wheels to be 4.125 so I need to measure it myself. Each Odometry wheel has 1 of both these wheels and sensors. The structure is built from 2 10 hole C Channels. Each channel is separated by 2, 3 inch standoffs. At the other ends of the structure is the encoder and wheel. The encoder is is connected by 1 screw. A shaft goes through the middle that holds the wheel inplace. Opposite to the



Encoder is a shaft bearing to ensure smooth turning.

The placement of each wheel is still being determined and will be placed after we have the chassis fully built. Left and Right have to be opposite to each other and the same distance from the center. The back wheel needs to be placed at any point along a different axis of the chassis's wheels. Overall this may be the finished design of the Odometry wheels prior to the November 19th Tournament

Pre-Coding: Move To Point Part 8

Programing moveToPoint

Now to cover the main part of this section. The function moveToPoint. Move to point utilizes a mixture of movePID and turnPID. The Function is small only spanning 9 lines, but mighty code is extremely useful. For quickly creating an autonomous with only a couple lines of code.

The function contains 3

Parameters. X is the desired X position, Y is the desired Y position.

Final Deg is if I want the robot to have a final heading. The first If statement checks if the

```
288     void moveToPoint(double X,double Y,double finalDeg) {  
289         if (findAngleMove(X,Y) != degHead){  
290             turnPiD(findAngleMove(X,Y));  
291         }  
292         movePiD(X,Y);  
293         if (finalDeg != null) {  
294             turnPiD(finalDeg);  
295         }  
296     }
```

Robot is facing the next desired position. If it isn't the robot will use turnPID to correct its heading. Next it uses movePID to move to the wanted position. Lastly the second IF statement checks if a final heading has been set. A final heading may be used if the robot is going to a roller. Where the robot can only turn it from certain angles. The NULL Variable checks if the value is set to nothing.

I do have somethings I would want to add in the future such as controlled drift or speed limiting. Controlled drift is a concept I've thought of the have the robot both turn and go forward at the same time. This would work by having one side of the Chassis move faster than the other side. The Robot would be able to move in arcs instead of normal straight lines. I programed the Odometry with the arc movements in mind.

Recap

This Concludes the formal end to Pre-Coding: Move to Point. In this Pre-Coding section I covered the PID, and how I will code and debug it. What a PID is, and the Function of it. The final thing I covered is the code above on how I will quickly make autonomi. Going forward I will cover reference this pre-coding section a lot when calibrating the PID's. If any major changes I have in the code before pre-coding ends. Pre-Coding in general ends once the robot has full functionality and I can begin to fully calibrate and test it.

Pre-Coding: Odometry Part 10

Odometry Data Reorganization

I have updated the Odometry code to better optimize the Auto Aim Code. In Pre-Coding Part 3 (Page 43). I show the very lengthy code of the location finder. I have made this significantly shorter. Before I explain how. I want to explain why I did/didn't do this.

I am inexperienced when it comes to overall coding. I only know what I have done in the past or have learned from Mentors. This year I want to do more with my code. Last year everytime I had a good basis for code I didn't know how to make it better so I assumed it is what everyone used. This was a horrible assumption leading to inaccurate and bad autonomous. My second bad assumption was thinking it would be accurate each time. I got embarrassed from this when talking to a group full of juniors/seniors with coding experience. I have hopefully learned from my mistakes and will improve this year.

I improved location data code by transferring the long chunks of X and Y data or related data. From single float values to arrays. Arrays are used to store multiple values. In the array you can have single dimension arrays that have Keys and Values or Multi-Dimensional arrays to store Keys that have Keys and values. Each Key uses an ID set by the program. Let's make this explanation easier with a example. Let's make a line of code. `Float Numbers[4] = {10,20,30,40};` The number in brackets is the amount of ID's or Keys they're are. The ID's start at ID:0. If I pull Key 2 I will get the value of 30.

The Code has 3 Arrays. One Two-Dimensional Array, and Two One Dimensional Array. The Two Dimensional Array has the X and Y coordinates. The first of the two single dimension arrays has the Z values of the High Goals. The last array is the internal offsets from the Odometry wheels. On the next page I will put tables of all 3 of these arrays so I can reference them when coding.

I added these arrays to help me when coding the Aimbot. The roller is planned to be placed on the back of the turret and a method of switching to roller scoring is to set the aimbots target to the roller selected. The array can also make the code more flexible without creating functions that are only used once or twice.

Pre-Coding: Odometry Part 11

Location Data Array

Name	ID's	X	Y
Blue Ramp	0	0	72.20
Red Ramp	1	140.2	72.20
North Roller	2	110.98	130
West Roller	3	139	110.98
East Roller	4	29.43	1
South Roller	5	1	29.43
Blue Goal	6	122.63	17.78
Red Goal	7	17.78	122.63

```
89 float coordinateLocations[2][8] {  
90 | {0,140.2,110.98,139,29.43,1,122.63,17.78},  
91 | {72.20,72.20,139,110.98,1,29.43,17.78,122.63}  
92 };
```

Z Coordinates Array

Name	ID's	Z
Low Goal Point	0	25
Center Goal Point	1	30.5
High Goal Point	2	35.5

```
99 float zCoordinates[3] {25,30.5,35.5};
```

Offsets Array

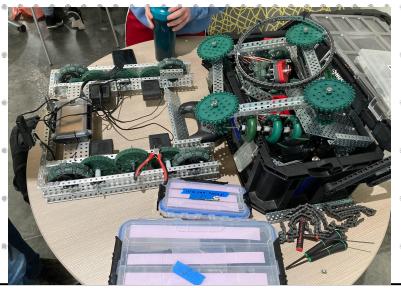
Name	ID's	Distance
Left Wheel Offset	0	7in*
Right Wheel Offset	1	7in*
Back Wheel Offset	2	7in*

```
108 float offsets[3] {9,9,9};
```

Meeting #16

Robot beginning of
Practice

Pre-Meeting Plan

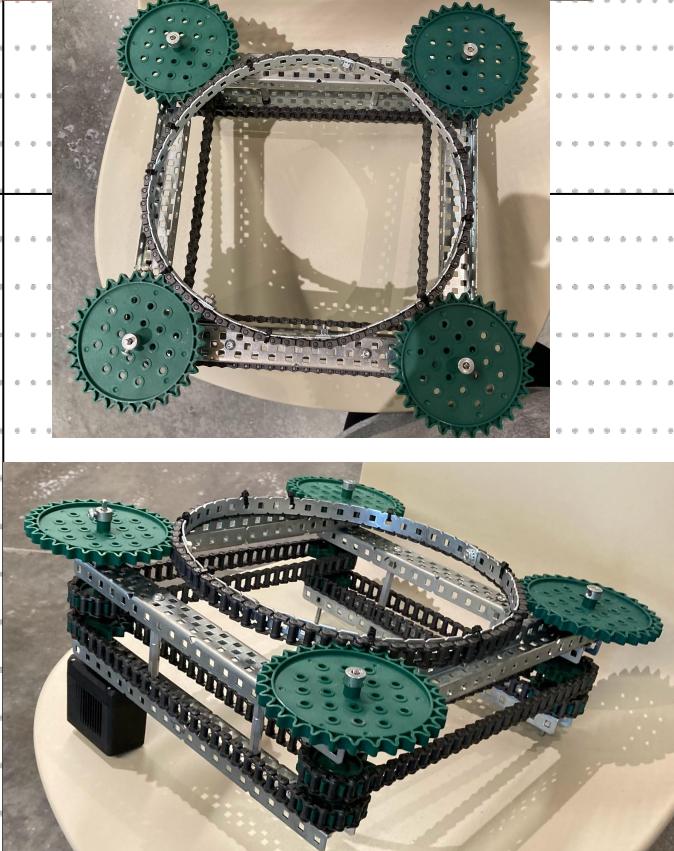


Team Members	Tasks and Obstacles	Design Process step
Aidan	Work on Turret / No field or access to club part storages.	Build a prototype
Ethan	Work on Flywheel/Only there for 30 minutes	Build a prototype

Meeting #16 Continued

Post Meeting Info

Meeting Info
Location School: iTech Preparatory
Attendees: Aidan, and Ethan Duration: 10:55 - 1:10

Team Members	What got done	Unresolved Problems	Images
Aidan	Made the Turret.	Needs to correct the Chain Circle length	
Ethan	Began Flywheel V2	No motorization or gearing.	

Pre-Coding: Odometry Part 12

Getting Velocity

In the Odometry I realized that were missing velocity. Velocity itself isn't needed for coordinate systems but it will be extremely useful for the aimbot. Getting velocity itself is easy math wise. The Velocity Formula is $V = \Delta D / \Delta T$. Distance in the Odometry Thread is measured in inches. The Odometry thread runs at the speed of 1 ms. The Velocity would equal inches per m/s. It's a very small unit of measurement but It can be adapted for the auto aim. I am going to separate velocity into X and Y so if needed I can get velocity relative to something like a goal.

```
332     velocityX = (pPositionX-positionX); // in/ms not in/s or m/s  
333     velocityY = (pPositionX-positionX); // Time = 1 so dividing isn't needed
```

The code here is very simple. It does the velocity formula to get the robots velocity. As said in the code dividing by one isn't necessary because the refresh time of the function is 1 m/s. The code is placed directly after the position finding. The code for the previous position is placed before the position finding but after the turning. This will hopefully be the last change to the odometry code before Coding with the robot.

Design Process: The Robot

Initial Designs and Sketches

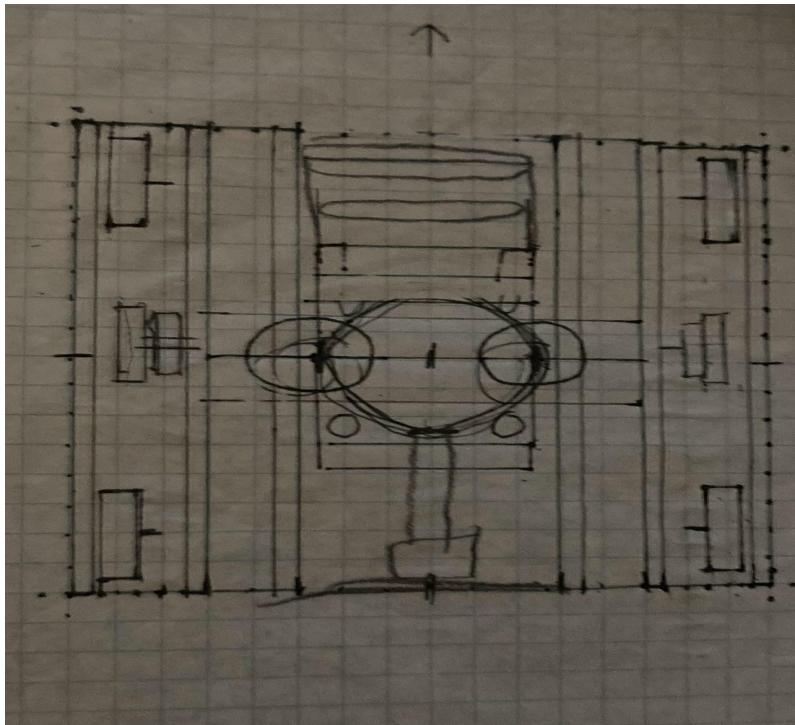
The robots initial design is rough at best. I should've covered this after meeting 7, but I lost the paper. I finally found the paper and now I can get notes about it. The initial designs were designed by Ethan, Tyler, and I during the first half of meeting 7. We chose to do this over a decision matrix because our original tournament that was our plan was the October 29th tournament. At the time we were working off the physical notebook and if we didn't do this design planning we wouldn't have a robot till maybe January. My team has unreliable communications systems so working at home would be near impossible.

During our discussion Tyler took notes mainly. He was mainly keeping the conversation understandable to him and worked out basic logics. Most the notes are actually in use and I will show photos on the next page..Here I will cover scrapped ideas and why we scraped them. In total there are 8 notes and 2 are scrapped.

The first note that was scrapped was the shared intake idea. The shared intake concept used the motors of the chassis for the intake. I pitched the idea while also pitching a dual intake idea which would be difficult for loading discs into the turret. The shared intake would lower the amount of torque the chassis would have. Now with the 333rpm speed geared chassis we would've been easily pushed around.

The second concept we scrapped was the Vertical motor control. This was in theory a basic way of controlling the turret with a auto aim. We scrapped this in favor of a Intake motor.

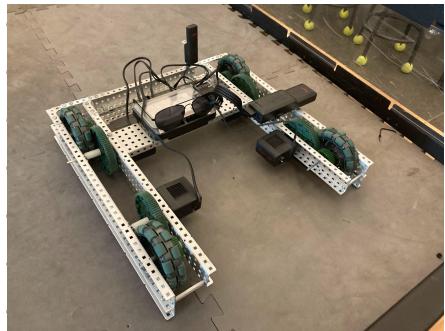
Last concept which wasn't in the notes but in the sketch was the drag wheels being in the wheel compartments of the chassis. This now can't be a possibility due to having a gear train running through it. It still is technically possible but its really tight.



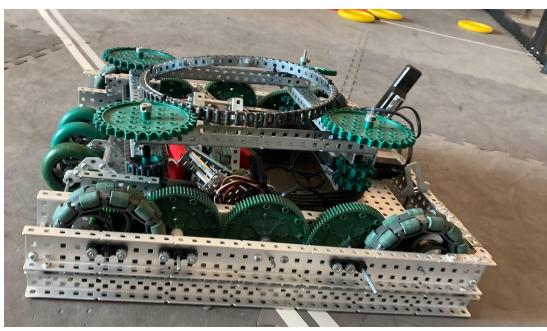
Design Process: The Robot

Initial designs and Images

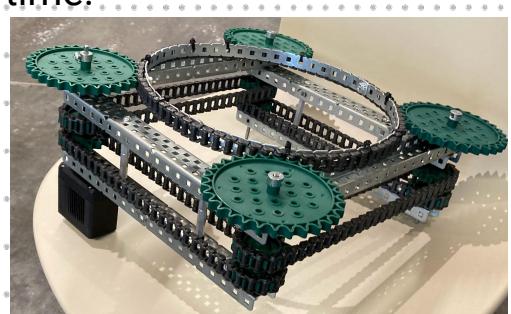
Note #1: Normal Drive.
Normal drive was chosen because it was simpler to code and to built with.



Note #2: Wheel Guards.
Wheel Guards/ Side skirt was added to prevent discs from jamming the robot.



Note #5: Sprocket circle Turret. Sprocket circle turret was the best and most simple design we had at the time.



On the sketches and designs we haven't built everything for an image so I will explain them here. Note #4 Single intake. The Single intake is just using a conveyor belt to load discs below the turret. This is somewhat true now but we are using a wheeled system then that feeds into the conveyor belt system.

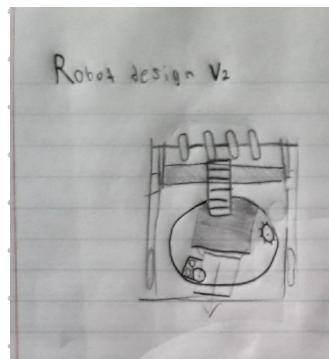
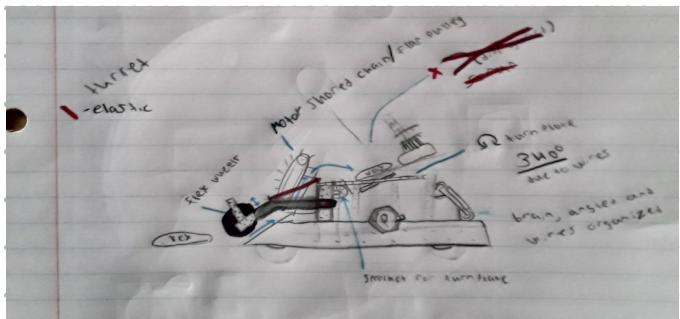
Note #5 Weak Pneumatic. Below the turret wheel we plan to have a pneumatic load the discs into the flywheel. The weak pneumatic will allow the pneumatic to barely lift up 3 discs all the way. The 3 discs on top of each other will load the one on top into one the flywheel. This concept may be scrapped in the future.

Note #6 Roller is a counterweight to the flywheel. This concept is still in place and up for planning. It will allow the turret to still be able to fire 360 degrees and still score rollers. The design I plan will use color sensors (if we have them). If not I will either try to find a sensor to detect the change in color or potentially the screws hitting the teeth.

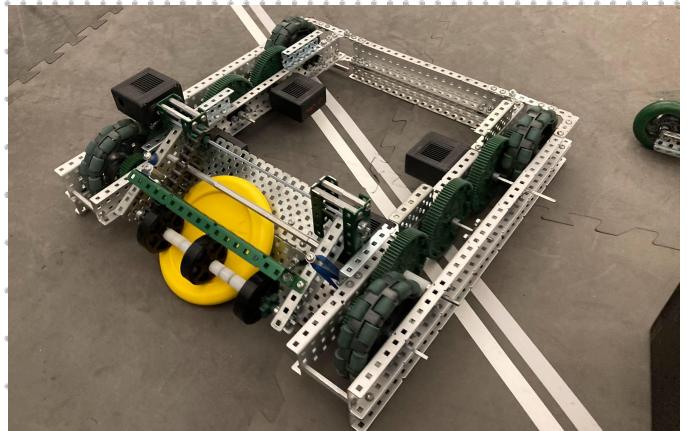
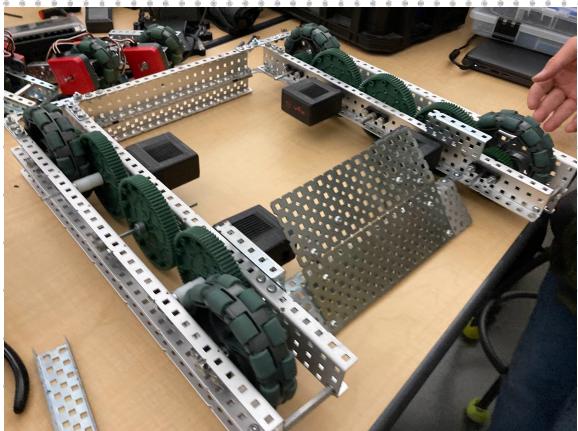
Its really nice to see that some of our concepts have come to fruition. The robot will of course change and some of these systems may not even be on our state bot and potentially even our worlds bot. If we even go to worlds of course.

Design Process: Intake design and efficiency

As the turret's rotating plate is being innovated and fitted to the robot, it is time to put our design for the intake system into action. We will be using all of our available flex wheels, hovering approximately a quarter inch off the ground suspended by a system of rails attached from a small tower on the side of the rotating plate hardware. The entire system of chains, flaps and wheels will be powered by one motor, directly connected to the flap system and chain driven down to the flex wheels. Our intake system will be able to move up and down to adjust for the height and stacks of discs.



The arm holding the intake is locked in place by a bottom bump guard, and held taught by rubber bands. An evident issue with a suspended intake will be chain tension. We have a design comprised of shaft collars, standoffs, chain cogs, and rubber bands to ensure the chain will keep constant contact with its gears despite its range of movement. As a disc enters the intake system, it will be slid directly back and up into the turret's turntable. We will slide the discs back on a plate, and have a triangular set of guides to ensure the proper flow of discs. As seen in team ri24h's intake explanation, the most resistance free way of guiding discs is to use spacers freely spinning on a screw. As simple as it seems it will be an excellent way to passively guide rings without causing jams or slippage.



Design Process: Robot Design

Loading a 360 degree spinning turret has been one of our biggest challenges so far. We have done multiple re - iterations of the system, and as of now, we are shooting for a system to load rings by moving them up and over the turret ring, then we will be able to shoot from a 360 degree angle, but only load from 340 of those degrees. The system will have a pin assembly to stall a disc while the turret's flywheel assembly is in front of the loading area. Once the disc is in, it will be dropped onto a plate, between a flywheel and a pneumatic trigger, and will be able to fire when the disc gets situated into the firing plate. The intake system will be a long arm mechanism with 4 flex wheels, tensioned by a rubber band system so that the discs are absorbed into the robot with little resistance, and a motor shared flap-chain to bring the discs up at an extreme angle, around 45 degrees, and a final wheel to force the disc into place.

Images pending

Meeting #17

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Turret Testing	Evaluate the Solution
Ethan	Finish Flywheel mark 2	Build a Prototype
Josh	Chassis redesign	Build a Prototype/ Evaluate the Solution

Meeting #17

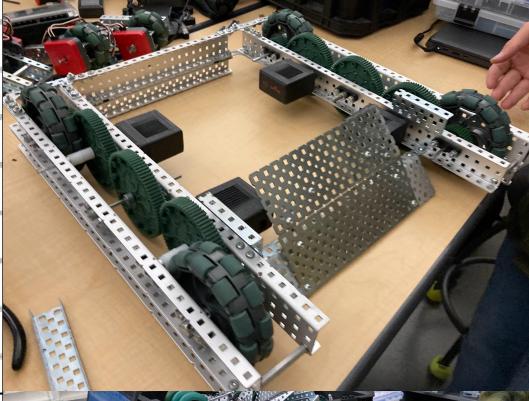
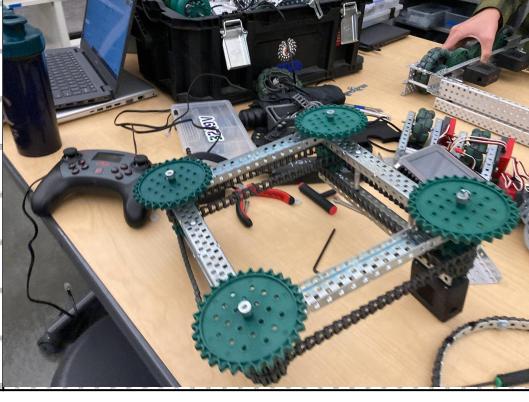
Post Meeting Info

Meeting Info

Location School: iTech Preparatory

Attendees: Aidan, Ethan, and Josh

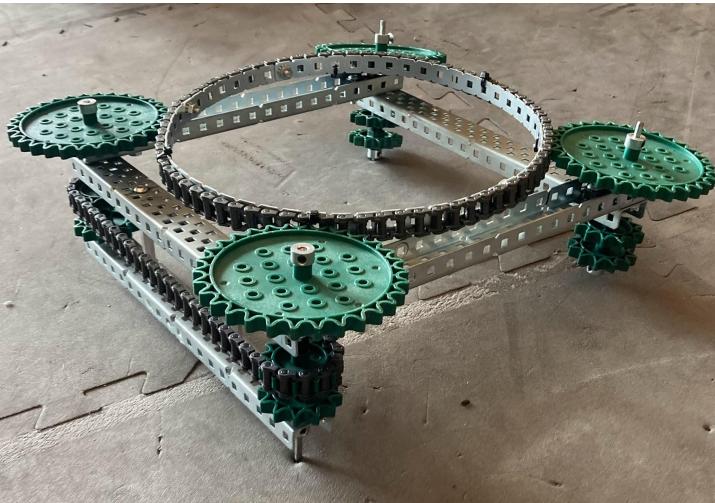
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Tested turret and got all pneumatics	Odometry wheels not attached	
Ethan	Worked on the intake system and flywheel	Intake not complete	
Josh	Took brain off chassis and modified Flywheel.	N/A	

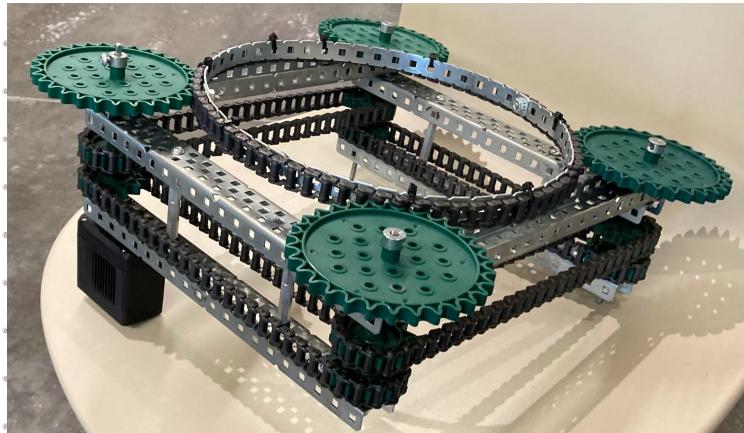
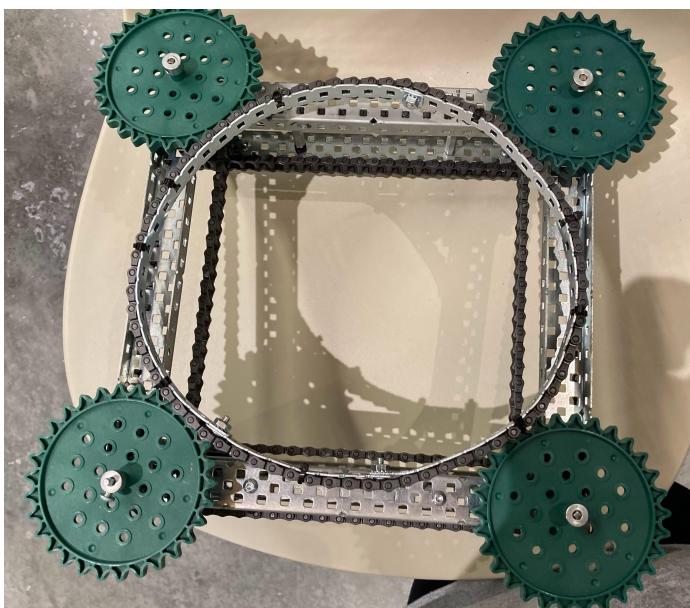
Design Process: The Turret

Version 1

The turret is somewhat complete. On the left is what it was on practice 15 when it was initially built, and below is the image of it complete on practice 16. Now that all the chains are attached I can begin working on the secondary parts of the turret. Part 1 will be the Flywheel + Roller Mount. The Flywheel Mount needs to be at 35 degrees and it will attach onto the the wheel itself. I plan to replace some of the 1 by 1s with thin 5x5s to allow more



- Stability in the Turret wheel and add connections
- Part 2 of this will be the bottom disc platform. A disc will sit there and when a pneumatic is activated the disc will launch into the flywheel barrel. The flywheels barrel can hold one disc through a rubber band hinge lock system. I will explain this more in depth when I have the design itself.



During practice today I modified the Turret a small bit. I changed the length of the 1 by 1s in order for it to fully fit. I tried speeding the Turret up by changing the motor from a normal motor which runs at 200 RPM. To a 600 RPM speed motor. The Motors already low torque couldn't handle the turret.

Something I noticed at practice is that the screwed parts of the 1 by 1's make the flywheel unstable. I might try to zip tie the connections to allow the flywheel to spin more smoothly. This may decrease overall structural stability but with the improvements I mentioned previously here It could offset it.

Pre-Coding: The Aimbot

Overview

Everything in Pre-coding has led to this challenge; the Aimbot. The Aimbot is how the robot will control the turret. It will know where it is at all times from the Odometry. The turret will use its PID's to maintain an optimal shooting angle. The Flywheel will adjust itself to cover the distance. The Aimbot is not all scoring discs. I can change its target. I can change its target to the rollers to score. The Aimbot will be my most difficult and rewarding venture into coding in all my years of vex. It will challenge my mathematical understanding and further expand it. Now for the question. What is an Aimbot?

Explanation

Lets simply what Aimbot means first. It comprises of two words; Aim, and Bot. The Aim for this word means to have constant aim on a target whatever it may be. The bot means its computer controlled. I have no input on our Aimbot other than the ability to select targets and the Odometry values. Aimbot is in shooter videogames a common slang for a type of exploit or hack in slang. Aimbots give the user an unfair advantage by letting a computer aim a weapon at enemies so the player doesn't have to. Of course this type of hacking is banned in all but most video games. The VEX Robotics Competition isn't a video game. Its real life and all the rules are in a game manual. It states that Aimbots aren't banned in this game so I will pursue. As of now I do consider myself having not enough experience to fully create such a complex program. By the tournament I wish to say otherwise. Now why would I want to attempt such a challenge.

An aimbot by definition gives the user an advantage over others. It gives me an advantage on those who either lack the experience to build and program one or those who have ways to make up for lacking one. An Aimbot requires having a Turret bot with at minimum 180 degree turning. An Aimbot is a burden to code and only having 180 degrees of turning wouldn't be suffice for the effort it takes. The full 360 degrees of rotation is the way success. Now to finish the explanation I will explain where it would function and how much it would control; of the robot.

Pre-Coding: The Aimbot Part 2

All sensory parts of the robot are made specifically for the Aimbot except for the potentiometer which I will cover during coding. There are in total 7 Sensors that deal with the aimbot 3 Motors that control it, and two pneumatics that manipulate the discs. In total 7 out of the 20 registered objects on the robot aren't dedicated to the Aimbot. Lets begin on how each of these 13 objects will control the Aimbot. In this section I will go over how all of these 13 objects are used.

I have covered the What, Why, Where, and now there's one last thing. How? How will I code this complex process? First let's start with organization. I will create two Threads. These will be named Thread 3 and Thread 4. Threads 1 and 2 were explained in Odometry (Display Thread is now Thread 2). Thread 3 Will cover the aiming systems and the target selection. Thread 4 will cover Auto-firing and control the intake and trigger systems. Let's start by covering Thread 3 which is by far the most complex.

The Aimbot has multiple levels of aiming. In a 3d game autofiring can be easily handled because of a Video input. Whenever an enemy is detected on your screen you will aim to it. Simple as that. The robot unfortunately doesn't have eyes. So it can't see. The robot does have a way of knowing where it is. The Odometry can serve for this purpose. We have location data in coordinates which the robot can use as a target. For the explanation and what I keep in mind while coding I will use the goal as a target. The first step is aiming the robot.

Coding

First we need to get the angle to the goal from the turret. I did this by getting the angle of a slope which is $\arctan(m)$. I put the code below for this. The slope is the line between the goal and the robot. I explained how this is calculated previously in Odometry. We need to move the turret. I will be using the previously created turretPID. This was mostly explained in the Pre-coding section Move to point. Next we need to find how we will get a disc into the goal.

```
double findAngle(int selector, double x = 0) { // selector uses coordinate list
    // m=(y2-y1)/(x2-x1) slope formula
    x = atan((positionY - coordinateLocations[1][selector])/(positionX - coordinateLocations[0][selector]));
    return(x);
}
```

November 19th Tournament Prep

In 25 days we will have our first tournament. The award our team will be aiming for is the Excellence award. This is by far the hardest award by far due to how many good teams that will be there. A secondary award my team might actually be able to get is the Tournament Championship award. This award is going to be the main focus of all 37 Teams attending the tournament. Out of the 37, 3 of those will be iTech teams. So I personally won't know what to expect.

In preparation for the tournament I plan to look at the results of the Oregon League matches and the upcoming Sandy October 29th tournament. On a spreadsheet I will get the ranks and scores of all the teams from those events. Combined with potentially getting the tournament sheet from 3249U I could get a lot of data for picking our alliance member. I plan to ask for 3249U's tournament data in exchange for our tournament data at the November 19th tournament which they aren't attending.

The robot as it is now cannot compete in any tournaments. This practice we are redesigning the chassis to include Odometry wheel locations and a better placement of the brain. The Turret is ready but nothing can be mounted to it and it can't fully mount to the chassis. The flywheel needs to have a motor and be weighted down more before it is fully ready. The roller system hasn't been built at all. The only points we could get is low goal zone goals or play a really hard defense or really hard aggression.

Now we are on track on having a bot to compete in the tournament in full operation. We have 11 scheduled practices left until the 19th so I have high hopes that we can make it. I need to talk to Tyler for how we can manage strategy and skills. For the other two iTech teams I can see at least one of them having a fully built bot. I will go over all of this again into more detail on the Monthly Update. When I get time I will create the spreadsheet and give it its own page. Personally I feel like we will be ready for whatever challenge that comes up at the tournament when the time comes

Meeting #18

Robot beginning of
Practice

Pre-Meeting Plan

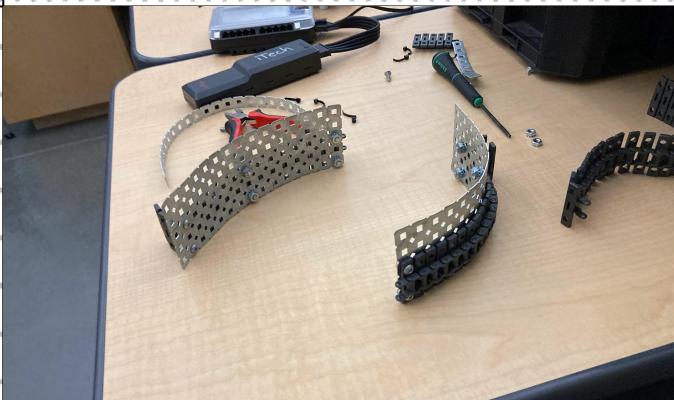
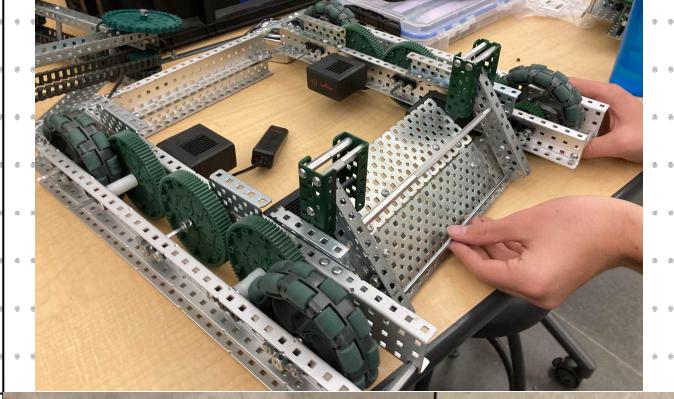
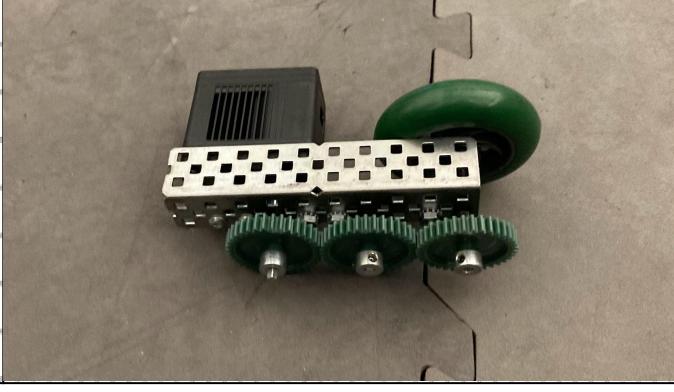


Team Members	Tasks and Obstacles	Design Process step
Aidan	Redesigning the turret	Design a Solution
Ethan	Building the intake system	Create a Prototype
Tyler	Designing the roller system	Design a Solution

Meeting #18 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Attendees: Aidan Ethan
and Tyler
Duration: 4:05 to 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Began turret redesign		
Ethan	Created intake mount		
Tyler	Created a prototype roller		

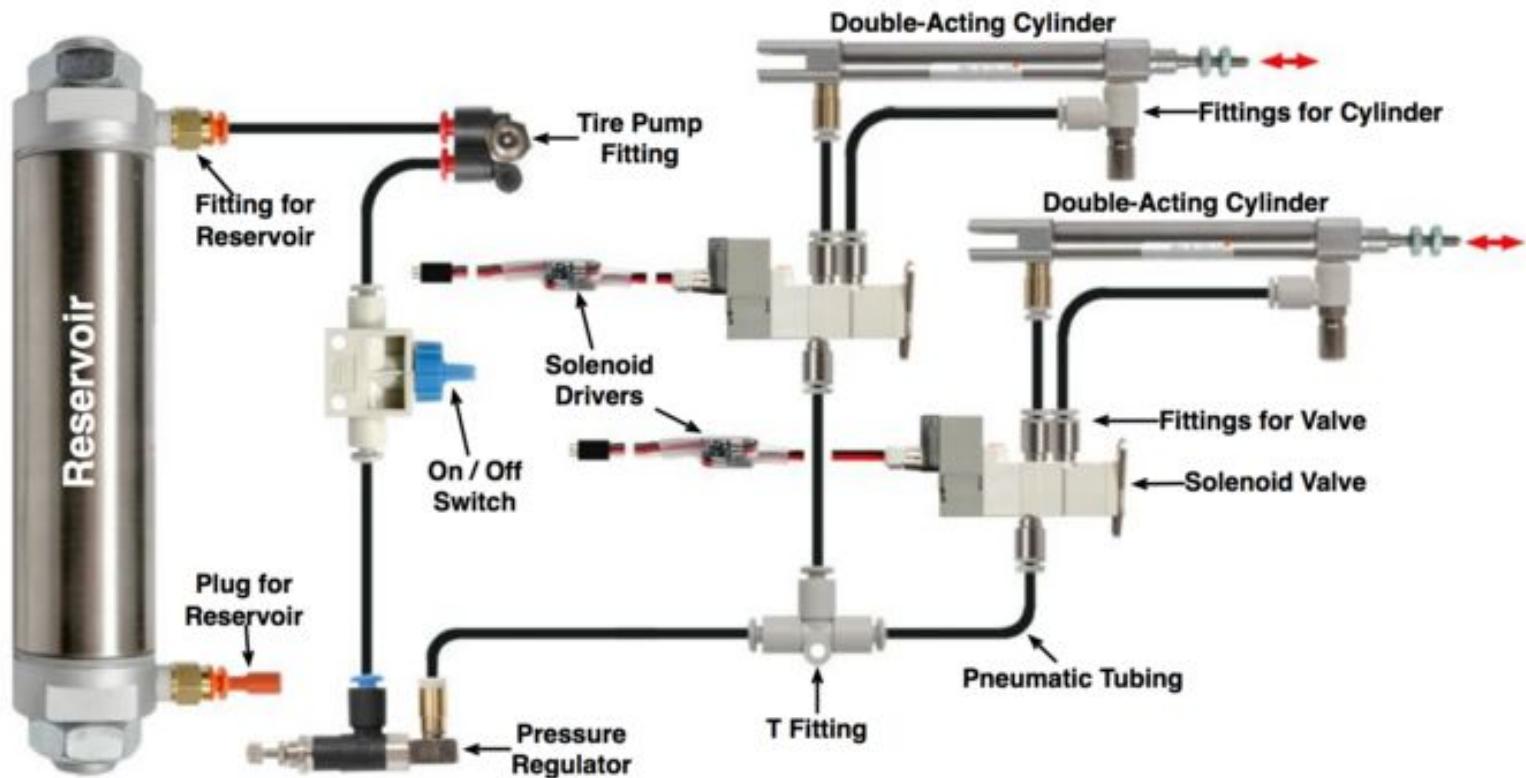
Design Process: The Flywheel

Version 2

Our first flywheel did it's job. It was a 1 motor drive 4200 rpm flywheel. It had issues, like the ejecting of parts at high rates of speed, loss of speed with every disc shot, and motor strain leading to overheating. With what we know now, we will be using a faster gear ratio + a torque motor to reduce speed loss and achieve higher rpms, we will be adding weight to the flywheel itself to deduce speed loss as well, and we will double the previous gear ratio to achieve higher speeds. One thing the builders will stress is the use of Nylock nuts, as opposed to Keps nuts, as the previous design basically rattled itself apart. We also learned to use a high speed washer setup (metal, plastic,metal) to reduce friction heat. Our old unwashed design got hot to the touch from just seconds of running, we need bearings to stay cool for the entire match to avoid damage.

Pneumatics research

We will be using a pneumatic cylinder to trigger the discs into the flywheel, and another pneumatic cylinder to release our expansion system. Our pneumatics specialist has been recently switched schools, and we now get to learn the pneumatics system. I have taken classes in electronics here at itech and i can say that I am fluent in understanding wiring diagrams. This one is especially easy to understand.



(Credits to vex community forum.)

I will continue to develop the intake system this meeting (10/27/22) and if I have time, tyler and I will work to assemble and install the pneumatic system

Challenge to overcome: we will be using 2 pneumatic cylinders, we will have to figure out how to rig the tubes together. I believe we can do this with a second switch and fitting.

Meeting #19

Robot beginning of
Practice

Pre-Meeting Plan

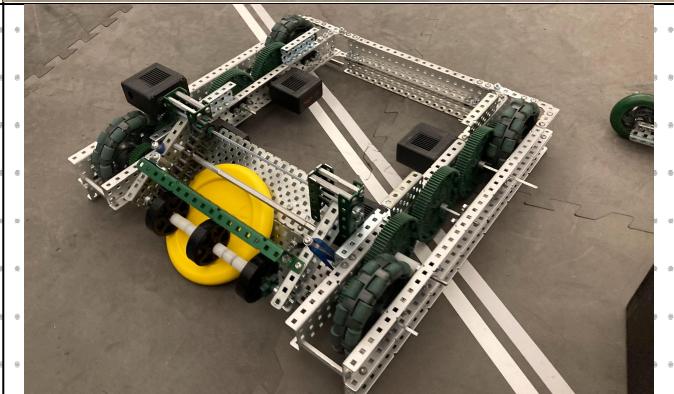


Team Members	Tasks and Obstacles	Design Process step
Aidan	Redesigning the turret	Create a Prototype
Ethan	Building the intake system and take pneumatics inventory	Create a Prototype
Tyler	Designing the roller system	Create a Prototype

Meeting #19 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Attendees: Aidan Ethan and Tyler
Duration: 4:05 to 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Resized the turn table to fit new turret wheel.	No work on the Turret wheel	
Ethan	Created the Mouth of the Intake/ surveyed and organized our pneumatics system	N/A	
Tyler	Almost finished with roller design	N/A	

Design Process: The Turret

Version 2

I have thought of a way to improve the turret. This modification will allow for easier mounting of the flywheel, and allow the turret to spin much faster. It will increase the overall stability of the structure of the Turret wheel.

First Change I will make is replacing the chain and 1by1 circle with tank tread. Tank tread when turned inwards can form a circle and keep its shape. Tank Tread also has screw holes on the ends of it so I can attach parts to with without interfering with the turret. I will raise the sprockets so the turn wheel doesn't touch. I am about to have practice and tell you the changes. 10/26/22

I began the new the design. I was able to get two 15 x 5 plates attached to the belt. Using the 1by1's to finish the loop is really annoying so I instead I will make the whole circle 5 wide plates.

I have to refit the sprockets on the turntable due to tank treads being much wider than basic chain. I got longer shafts for 3 of the corners and on one of the corners. The shaft was long enough to simply move upwards to fit the resizing. iTech robotics has currently a issue when it comes to spacers mainly lacking them. I found some old rubber shaft collars and decided that will do.

I put 3 rubber shaft collars as spacers on each corner. I tested with some tank thread and this worked.



Now all that is left is to finish the turret wheel with a mount for it. On top of the mount I will put the flywheel on there. We don't plan to put the roller on the flywheel anymore. I don't fully knowing how cords will act on the flywheel so for now we want to minimize what is going to be in the flywheel.

The last thing I need to add is going to be a hinge latch system in the mount. The latch system will be a passive rubber band system that allows the discs to go into the flywheel but not fall back down into the turret system.

Design Process: How we Design

In the notebook we don't show much we really should be showing in our notebook. Most of our designs are discussed through talking amongst each other. We sometimes do sketches that we lose the paper to. We usually do post entries on our design process after creating a part of the robot.

We work to improve our communication with the notebook and write more down in the notebook. Currently we try to do entries either during classes or at home. As we are on a time crunch for the November 19th tournament we don't have much time to program and notebook during then. Of course when the robot is built sometime at the end of 2 weeks from now we will have a lot of time for notebooking and coding.

Unlike previous years we actually see the need and concern for having a really good notebook. The notebook at some times can be more important than the bot for teams to qualify. However at our first tournament our focus is on creating the Robot. The turret design is something really rare to see this early in the season.

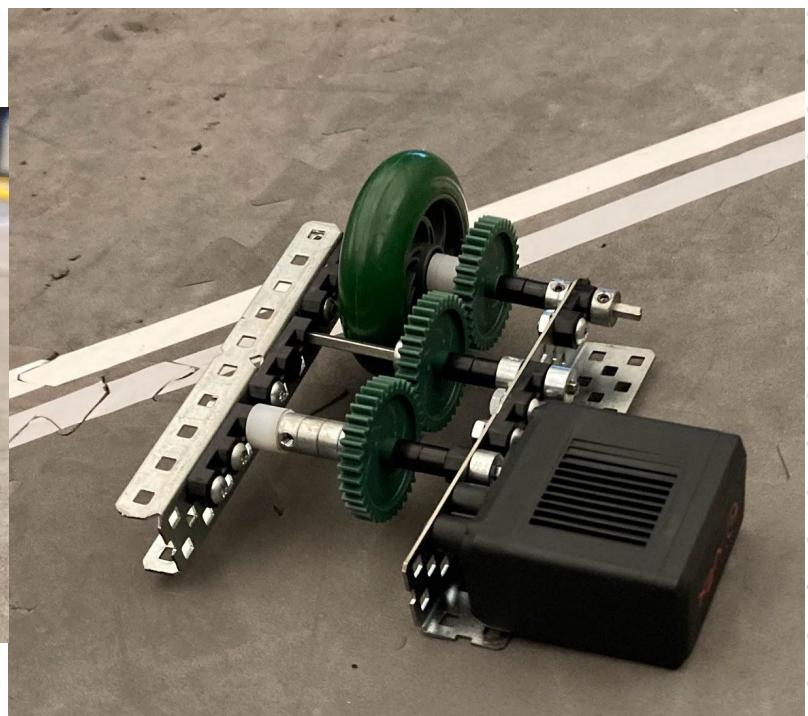
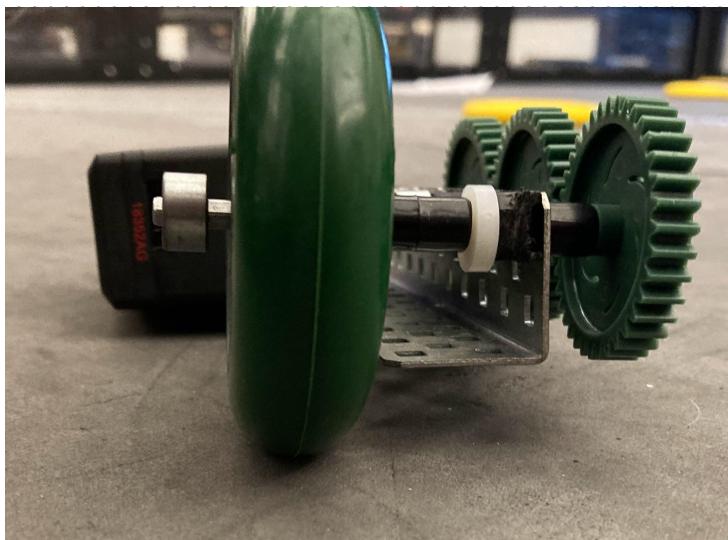
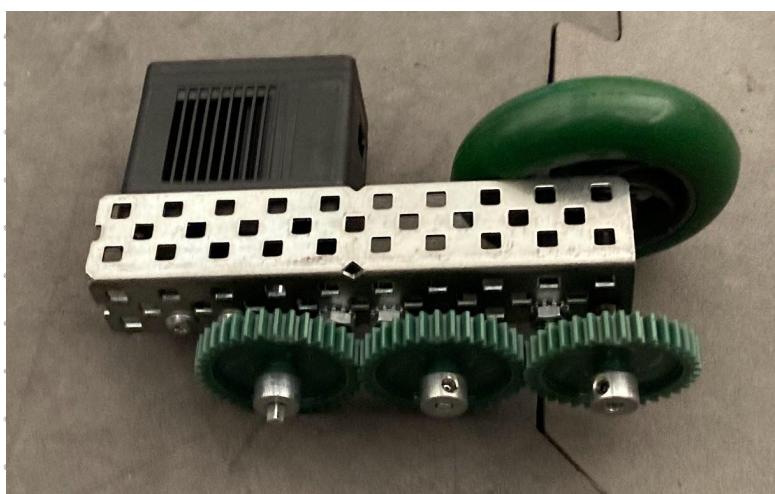
Of the teams that are going to the 19th tournament. 23 of them are teams that are going to the earlier October 29th tournament. Making the list of turret teams even rarer. This will leave around 11ish teams other than us and the other iTech teams. I would say our chances to go to state this next tournament from the Tournament champions title is very likely.

I will try to keep the notebook as up to date as possible. Like I said the robot is the priority during meetings. Our focus will change after the tournament to the notebook as after December other teams will have bots that can rival the turret design such as Catapults and Slingshots. The turret design is a really high risk but extremely high reward design.

Design Process: Roller Mechanism

Version 1

I was tasked with building the roller mechanism for the robot. I am very inexperienced so building this was a big learning experience. This was the first design I had to create and build by myself so there was quite a bit of trial and error. I tried to make it as small as possible so it could fit better on the robot. With that in mind I made my first design which was a motor with a gear train with a wheel at the end. This was very unstable and the shafts and gears wobbled around a lot. I quickly learned from this that the shafts need at least 2 support points for it to be stable. My solution for this was to have 2 C channels so I could make 2 different support points. This made it a lot more stable. To improve the design I want to make it thinner by making the wheel outside the C channels and making the motor go inside the C channels. This way the wheel is more isolated and the motor is at a more convenient location.



Game Strategy

Overview

Strategy is everything on the field. Strategy is something that really varies between individual teams. For our upcoming tournament we are going to be trying for either the Tournament Champions Award or the Excellence Award. These two awards require a strong performance on the field. Which means having a good strategy.

There are two phases in the game. The Autonomous Phase and the Player Control Phase. As I learned from last year a good autonomous could secure the win. Our bot this year is much faster than the other bots so we can use our speed to our advantage against other teams in this period.

The Turret and Auto Aim is a huge asset to have in this game. In the 15 seconds of the autonomous of autonomous with the speed of our robot and the auto aim we could potentially score almost all the middle discs while also securing the Autonomous win point. This will leave the discs on our side of the field and their side which we can rush.

Formatting

The formatting for this Section will be different than most other sections. In the google slides notebook each field page takes a whole page. Explaining a strategy for sure will not take a whole page so we will have strategy pages briefly explaining the strategy and its purpose. I will also show the coding aspects next to each Strategy. Strategy pages will be actively updated pages so date will be shown when strategy is created. Strategy format on page ##

Coding

I would like to briefly cover the coding aspects of strategy. Not in terms of actual coding but strategy selection. The robot can only hold up to 8 different programs and presumed 8 autonomous programs. I plan to extend this amount with the use of a potentiometer. Last year at the Willamette February 13th tournament. Me and Ethan saw a team that used a potentiometer to select their strategies. I plan to use this concepts to expand how many strategies each program can hold.



Game Strategy: Strategy Page Format

Strategy ##, Date Created:##/#/#/## Name: #####. Points: ## AWP: [Yes/No].
Program: [1-8], Potentiometer: [1-8] Page: ##. Strategy [Strategy Name] is a
[Autonomous or skills] program. [Brief Description of Strategy]. [Why the strategy
exists]. [Potential Uses]. Created By [Author].

INSERT CODE HERE

Strategy ##, Date Created:##/#/#/## Name: #####. Points: ## AWP: [Yes/No].
Program: [1-8], Potentiometer: [1-8] Page: ##. Strategy [Strategy Name] is a
[Autonomous or skills] program. [Brief Description of Strategy]. [Why the strategy
exists]. [Potential Uses]. Created By [Author].

INSERT CODE HERE

Strategy ##, Date Created:##/#/#/## Name: #####. Points: ## AWP: [Yes/No].
Program: [1-8], Potentiometer: [1-8] Page: ##. Strategy [Strategy Name] is a
[Autonomous or skills] program. [Brief Description of Strategy]. [Why the strategy
exists]. [Potential Uses]. Created By [Author].

INSERT CODE HERE

Game Strategy: Strategies Page 1

Strategy 01, Date Created:10/27/22 Name: Grep. Points: 45 AWP: yes. Program: 1, Potentiometer: 1 Page 113. Team: All Grep is a Autonomous program. It starts by rolling the roller and then swiftly collecting and shooting as many disks on our side as possible to make the opposing team have less disks to take. It finishes near a 3 stack on the autonomous line for you to potentially take if you deem fit. This strategy is a low risk way to get a lot of disks in at the very start. It can be used when our teammates don't have an autonomous. Created By Tyler Fields.

```
//X = 111 Y = 132.5 // Grep
moveToPoint(85,109.3,null); moveToPoint(60,84,null);
moveToPoint(43,91.8,null); moveToPoint(48,72.3,null);
moveToPoint(37,60,null); moveToPoint(30.3,43,180);
```

Strategy 02, Date Created:10/27/22 Name: Close Blitz. Points: 50 AWP: Yes. Program: 1, Potentiometer: 2 Page: 114 Team: All. Blitz is a Autonomous program. It quickly starts by getting the roller and then it rushes to the center to grab some of the discs. It then loops back and grabs a 3 stack and shoots those off lastly it grabs 3 discs but not fire them off due to time constraints. This strategy can be used when our teammates don't have an Autonomous and a quick way to get early points.. [Potential Uses]. Created By Aidan McCallum.

```
//X = 111 Y = 132.5 // Blitz
moveToPoint(111,110,null); moveToPoint(82,82,null);
moveToPoint(82,106,null); autoFire = false;
moveToPoint(28,50,null);
```

Strategy 3, Date Created:10/27/22 Name: grabby. Points: 75 AWP: Yes. Program: 1, Potentiometer: 3 Page 115. Strategy grabby is an Autonomous program. The goal of this strategy is to get as many points as possible. Its a good way to get a lot of points early on. This strategy can be used when our teammates don't have an Autonomous and a quick way to get early points. Created By Tyler Fields.

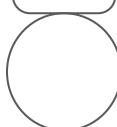
```
//X = 111 Y = 132.5 // Grabby
moveToPoint(117,117,null);moveToPoint(94,94,null);
moveToPoint(52,109,null);moveToPoint(59,81,null);
moveToPoint(37,59,null);moveToPoint(36,86,null);
```

Key:

Starting Position

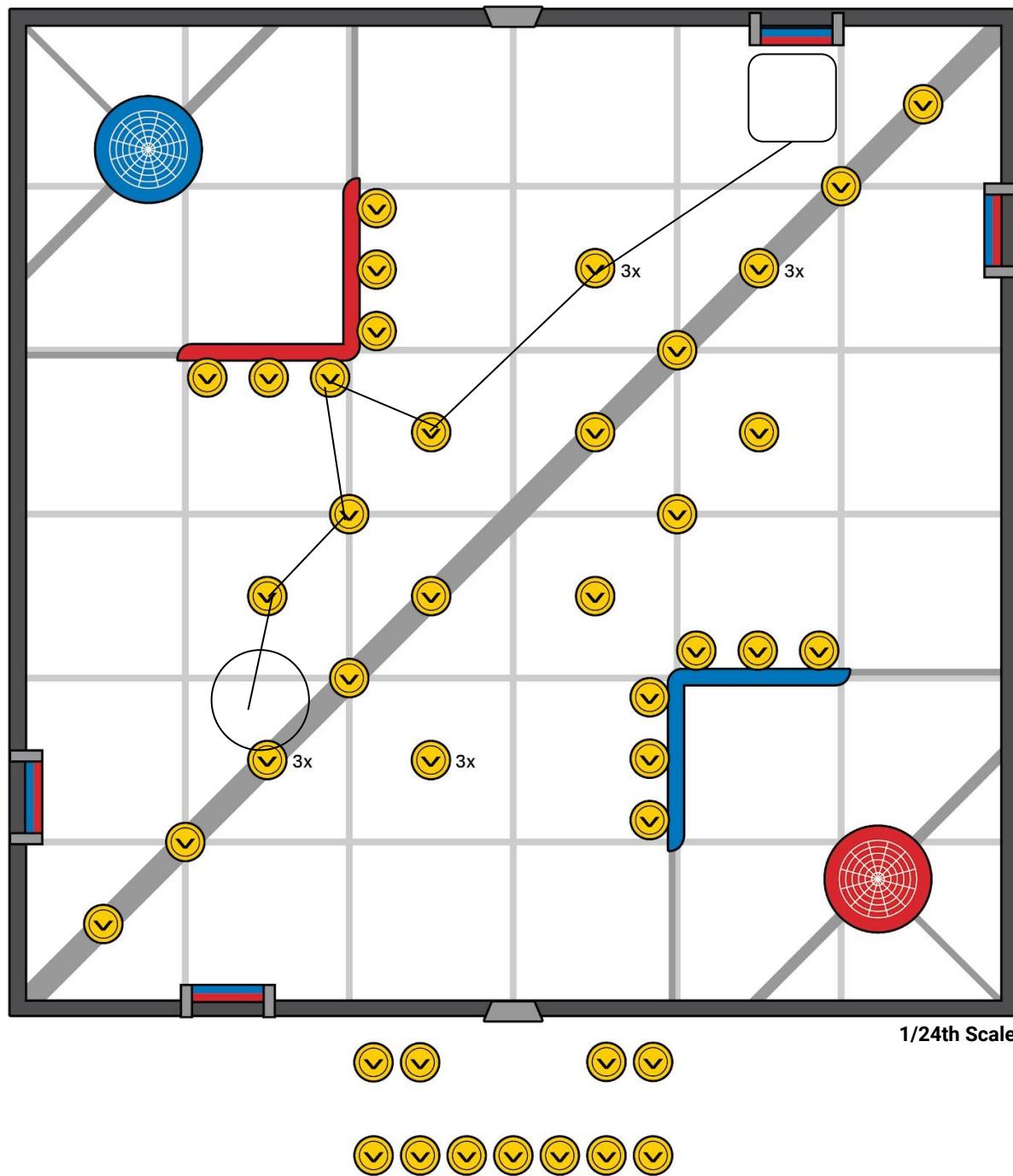


End Position



Strategy: 01

Name: Grep

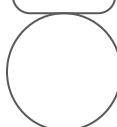


Key:

Starting Position

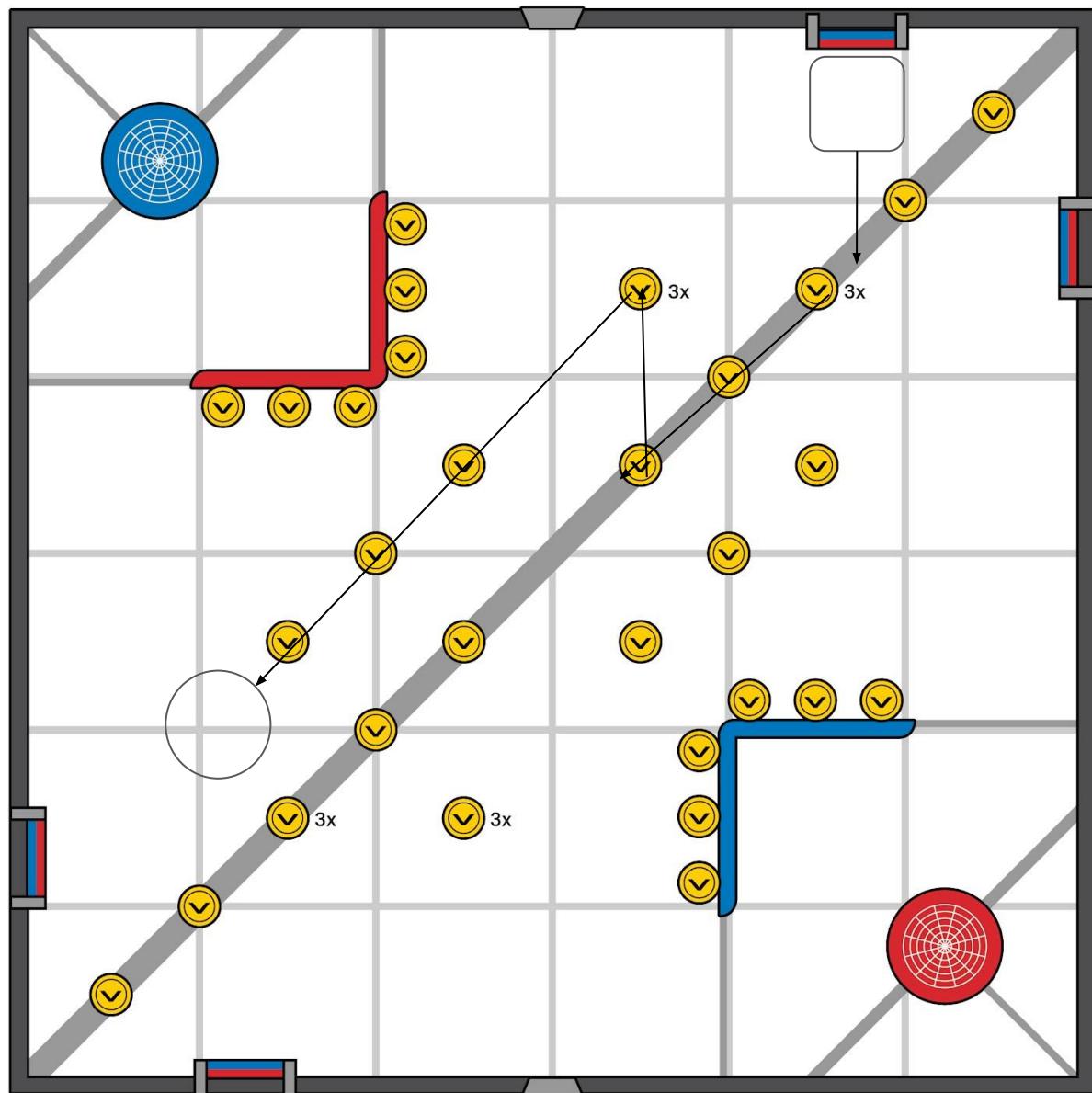


End Position



Strategy: 02

Name: Blitz

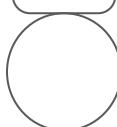


Key:

Starting Position

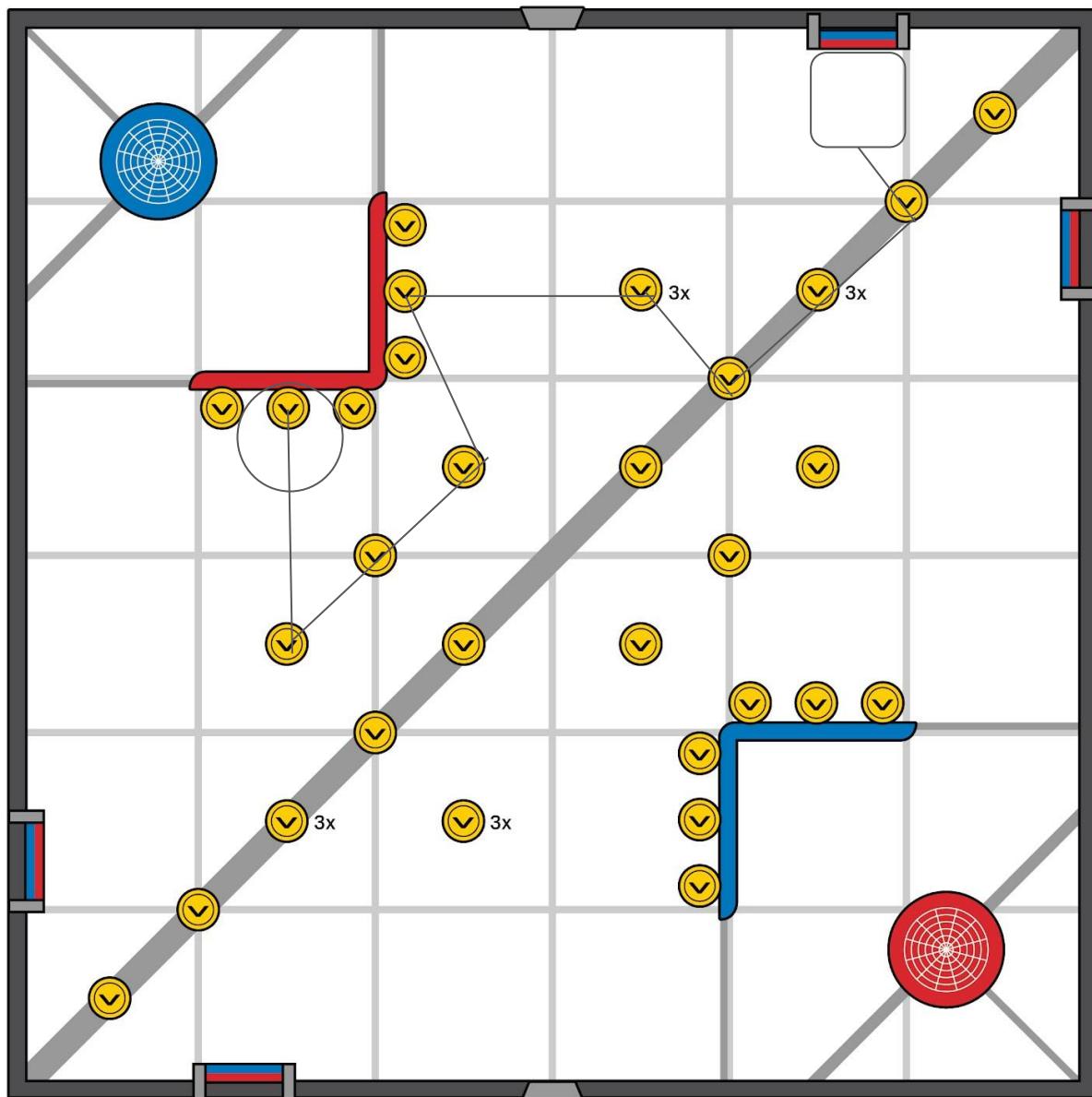


End Position



Strategy: 03

Name: Grabby



Game Strategy: Strategies Page 2

Strategy 04, Date Created: 10/27/22 Name: Left Blitz. Points: 45 AWP: Yes. Program 1, Potentiometer: 4 Page 117. Strategy Left Blitz is a Autonomous program. It is a variation of the strategy Blitz used on the left side. The strategy is a more teammate strategy friendly version of Blitz. Most teams don't program using the left side so its can work with teams that have a more Basic Autonomous. Created By Aidan McCallum.

NOT PROGRAMMED YET

Strategy 05, Date Created: 10/27/22 Name: Skittles. Points: 264. Program 8 Page 118. Skittles is a Skills program. It starts getting both rollers and a disc. It then travels down the middle line to get the 14 discs there. Lastly it travels down both sides of the field to get the 6 discs on each side. Lastly it runs to the loading ramp where we loading ramp where we put discs into the robot. Created By Aidan McCallum.

NOT PROGRAMMED YET

Strategy 6, Date Created: 10/27/22 Name: Scraper. Points: 55 AWP: Yes. Program 1, Potentiometer: 5 Page 119. Scraper is a Autonomous program. Scraper is a low risk left side Autonomous that grabs the discs on the L of the low goal zone. This strategy should work well with another team if they have another Autonomous. It can be used if the alliance has a riskier autonomous like blitz. Created By Aidan McCallum.

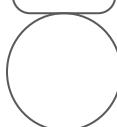
NOT PROGRAMMED YET

Key:

Starting Position

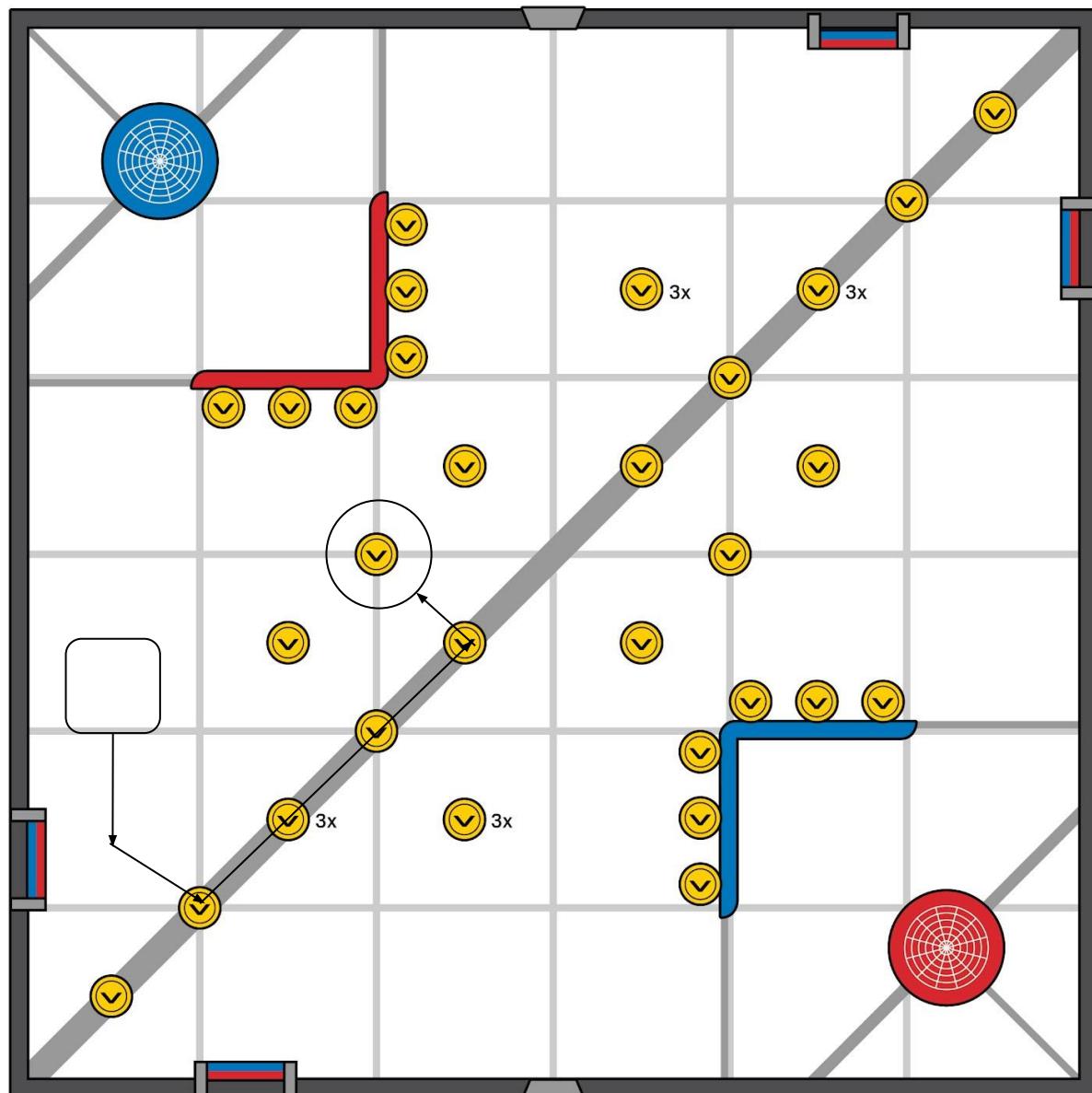


End Position



Strategy: 04

Name: Left blitz

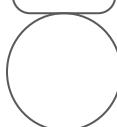


Key:

Starting Position

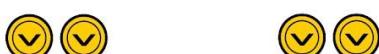
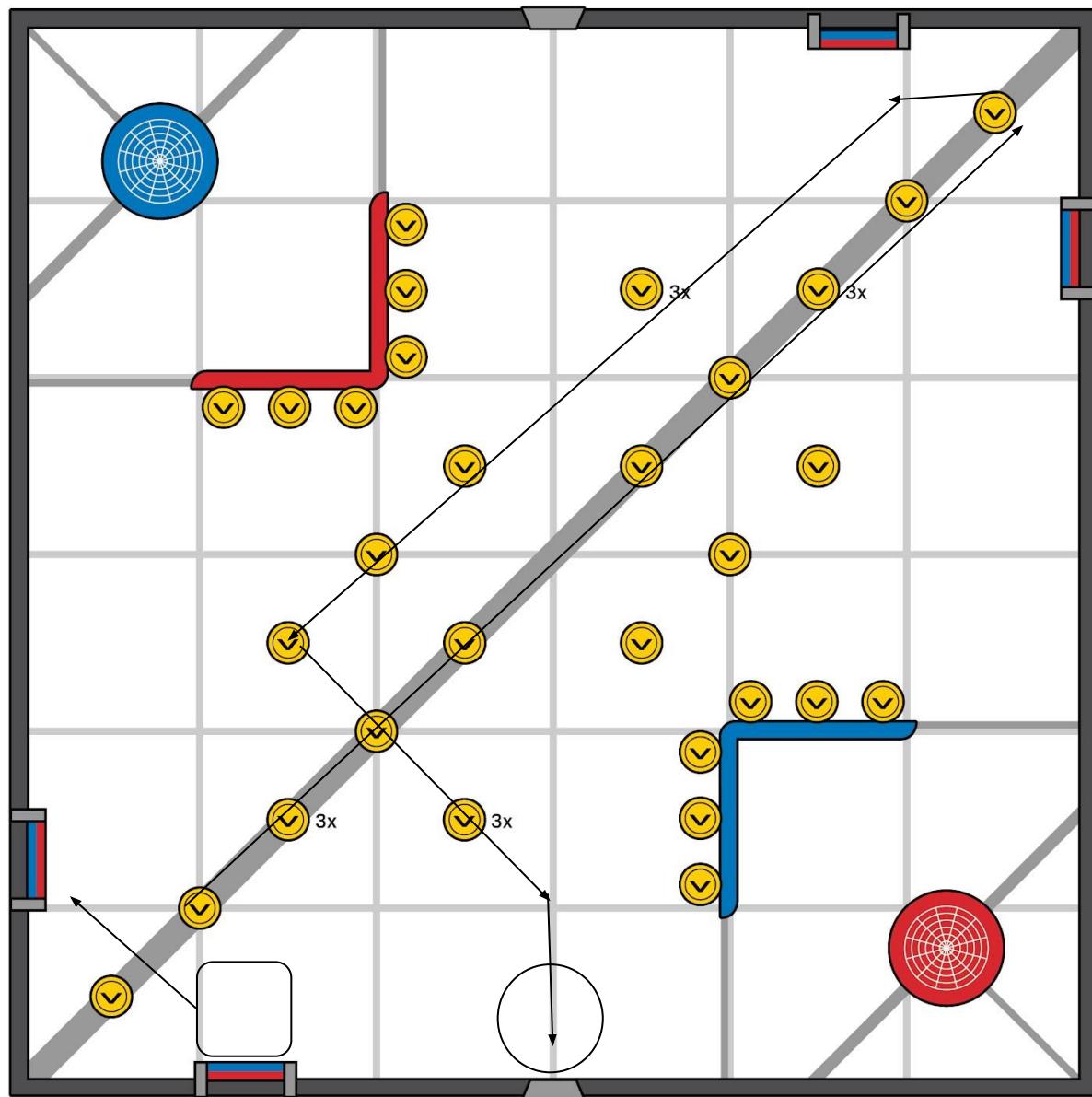


End Position



Strategy: 05

Name: Skittles

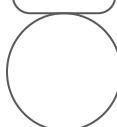


Key:

Starting Position

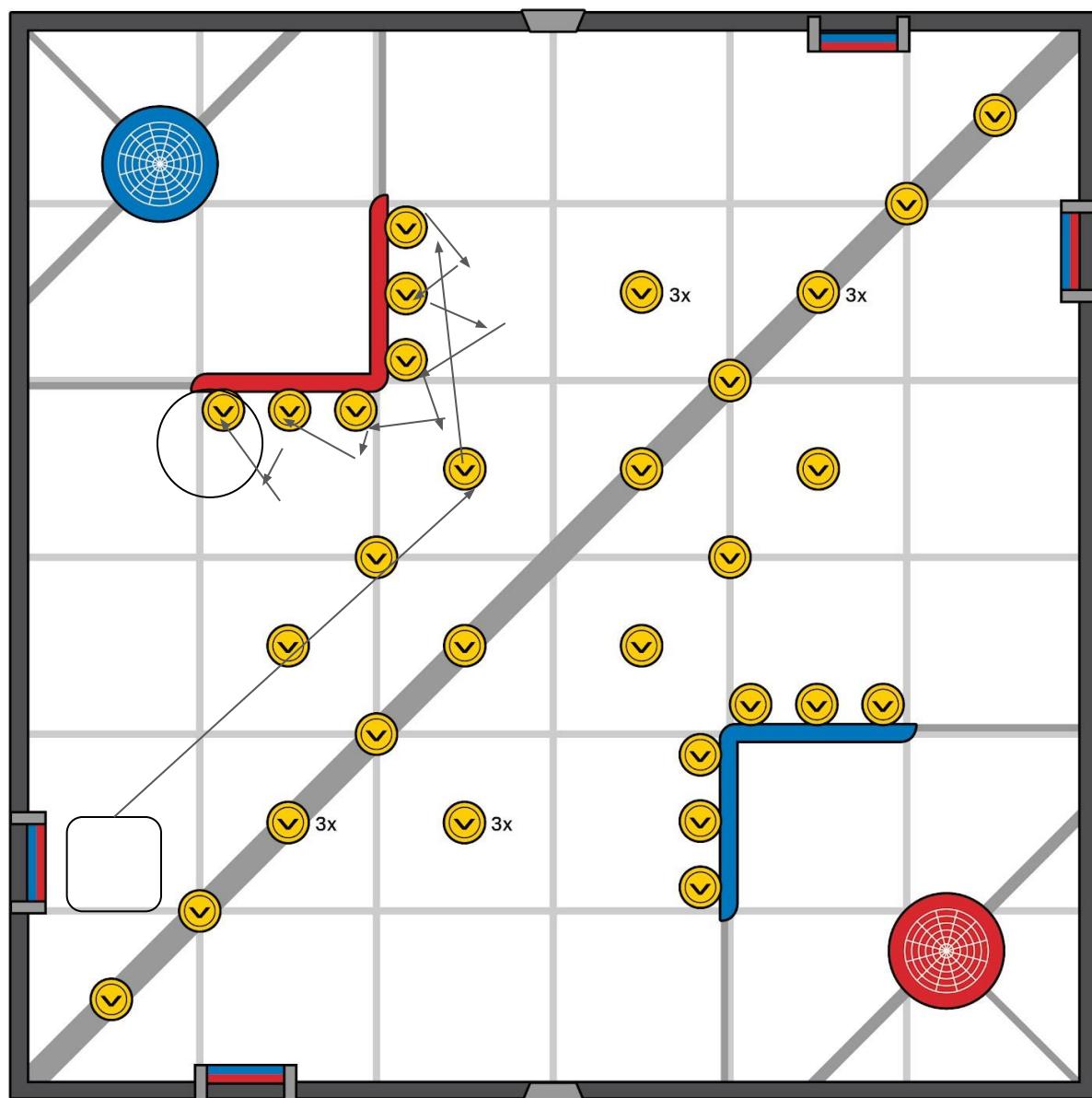


End Position



Strategy: 06

Name: Scraper



1/24th Scale



Notebook Formatting

Formatting is a useful tool to have. Having a way of consistency between pages will make it easier for the reader to read the notebook. This organization is not global for all Sections such as Strategy or Meeting Notes. Its primary usage for topics on what is closer to what a typical engineer is used for such as Design Process or Pre-Coding.

Let's start at the top of each page. The title should explain what the Project is and what section of a project the page covers. An example being "Pre-Coding: Odometry Part 8". The Title is always in Font size 35 unless the text is too long. Which will be scaled down. The lowest the font is allowed to go is Size 30. The title should always be centered as well.

Next, let's cover subtitles. Subtitles should be at the top of the page below the Title, unless it's apart of a multi page project where a subtitle. Exceptions to this rule are non-engineering related pages such as this one. The subtitle at the top of the page should always be in size 20 font and centered. Subtitles in the middle of the page are on the very left same font size. Subtitles should not be more than 5 words.

The main body of a page takes up most of the page. It should always be in 15 point font. Paragraphs should cover a mthein idea. Each paragraph must be at least 5 sentences in length. Each beginning of the sentence should always have a topic sentence. Essentially having the basic rules of a paragraph.

When formatting images make sure it the image can be seen properly. It shouldn't be too small to notice but not too big to take up most the page. If text can rap around a image it has to use a separate text box that can somewhat line up with overall page formatting. Images should NOT be in the middle unless it spans left to right. Images can be tricky at times so try your best when putting them in.

In overall formatting everything needs to be readable. We shouldn't have text overlap with text. Images covering text. Text cover session notes, etc. Font sizes shouldn't be smaller than font size 10 because its unreadable. For the font type we will use the default font Roboto as set by VEX. Readability is a necessity for winning any awards that the notebook needs.

One last thing to note. In the "Project" Section at the bottom of the page we put the name of the overall formatting. This is mainly because different projects will have different types of content such as "Design Process" and "Pre-Coding".

Pre-Coding: The Aimbot Part 3

Before we continue I want to explain the original auto aim. Originally I used a two motor turret system. Motor 1 controlled the X axis and Motor 2 controlled the Z axis or vertical axis. It was much easier to code as I had to create a second PID to calibrate the Z axis so the Flywheel is directly facing the goal at all times. The idea is mainly scrapped with some remnants of its existence in the code. The reason for it being scrapped was because we didn't have enough motors to operate it along with a chassis, intake, roller, and flywheel mechanisms.

I plan on getting a disc into a goal by changing the flywheel speed. Unlike the previous method this is by far not a simple task. As observed from testing the flywheel's velocity the flywheel's speed will not be the disc's eject speed. The laws of physics prevent this from happening in most cases. We need to set the flywheels speed to a certain amount to get the distance it needs to travel.

Before I figure the code out. I want to know if the flywheel can even support shooting a disc from all the way across the field. I will be using the formula for angular momentum $L = mvr\sin\theta$. L in this case is the estimated flywheel speed. M is mass. V is velocity, r is radius, and $\sin(\theta)$ is the contact angle. I don't know if this is the actual formula to find this. this is the best thing I could find that I could understand. The main concern is that I cannot use anything relating to a second object for this. If I find that I'm wrong later then I will redo this on a later page.

Let's set the formula up. M is equal to 0.20lbs. The flywheels radius is 1.5* (stationary). The angle of contact is roughly 70 degrees. Velocity requires its own calculations. The formula to find velocity in this case we need angular velocity first. Angular velocity changes in rotations / change in time. This can also be described as RPM. The flywheel is geared at the extremely fast speed of 4900rpm. First i need to transfer RPM to rad/s or radians for a second. it comes to be 513.1268 rad/s. Now we do Rad/s*Radius to get velocity $513.1268 * 1.5 = 769.6902$.

Now let us input the full equation. $L = 0.2\text{lbs} * 1.5\text{in} * 769.6902\text{in/s} * \sin(70) = 178.69682204$. To see if this is enough I will use the online calculator Omni calculator. Specifically the calculator they made for Projectile motion. We need to input the velocity. The angle which for now we will put as 35 degrees, and an initial height which I will input 1'2" as a placeholder for now. The final distance came to be 9 ft... I need to talk to Ethan about this before he continues building the flywheel.

Pre-Coding: The Aimbot Part 4

We now have a mechanical issue that needs addressing. I will assume that we have a flywheel that can be launched across the field. When addressing any flywheel related information other than radius and launch angle which won't change. I will use the variable name instead if needed. For now, I will try not to cover anything relating to flywheel until we have a solution.

Auto-Firing System

I will now cover the Auto-firing system until we have a fix for the flywheel. The Autofiring system is a thread independent from Autonomous or Driver control like odometry. Unlike the Odometry thread the Autofiring thread actively controls portions of the bot. It regulates the amount of discs the robot is intaking and expelling.

First let's start off analyzing the Auto-Firing systems purpose. Its primary purpose is to have more internal automation. The driver shouldn't have to remember to use the intake. It's my job as the programmer is to make the driver's job easier. Of course the driver wouldn't be able to drive without a programmer. The driver also suggests on what the robot should do. So did the Programmer create the Driver role, or did the Driver create the Programmer role? I personally will never know.

The Auto-firing system is one of many ways on making the robot easy to use. It eliminates the need of controlling the internals by automating them. Without it you would need to remember to activate the intake, put a disc into the turret, put a disc into the flywheel, and fire. All steps these without putting 3 discs into the robot. The Autofiring thread accomplishes all of this. It requires 3 limit switches to accomplish this.

Coding the Autofiring System

The first line sensor is placed near the mouth of the intake, between the mouth of the intake and the turret table. When a disc hits the limit switch a boolean which is a value that is either true or false.

```
void intakeRead(){
    intakeDisc = true;
    discLoad += 1;
    waitUntil(!intakeSensor.pressing());
    intakeDisc = false;
}
```

Pre-Coding: The Aimbot Part 5

This boolean is called "intakeDisc". At the same time, a variable called "discLoad" is increased by 1. I will get back to why that's there later. Once the disc enters the turret mechanism and the sensor is unclicked "intakeDisc" is set to false. The code is on the previous page (122) on the bottom right. The reason why it is in a function rather than in the thread is something I'll explain later.

The second sensor is located at the bottom of the turret. This is where a disc can be placed before a pneumatic puts it into the Flywheel mechanism. Like the Intake Sensor it will

```
472 void turretRead(){  
473     turretDisc = true;  
474     waitUntil(!turretSensor.pressing());  
475     turretDisc = false;  
476     if ((intakeDisc = true) && (flyWheelDisc = false)) {  
477         triggerControl();  
478     }  
479 }
```

detect if a disc is there and when the disc moves. Unlike the intake sensor, it doesn't add another disc to the counter. Something unique to the sensor is if a disc is in the

```
440 void triggerControl() {  
441     trigger = true;  
442 }
```

intake and there is no disc in the flywheel. It will automatically fire the pneumatic so there can be space in the turret for the next disc. code above for the sensor. The code is on the left for the trigger control .

The third and final sensor part of the internal sensors is the flywheel sensor. It is placed technically in the turret but in a place where the pneumatic can push it into the flywheel itself, shooting it into the goal. Like the previous two functions, this one detects

```
443 void triggerControl2() {  
444     if ((lockOn = true)) {  
445         trigger2 = true;  
446         discLoad -= 1;  
447         wait(25,msec);  
448         trigger2 = false;  
449     }  
450 }
```

```
480 void flyWheelRead(){  
481     flyWheelDisc = true;  
482     if ((lockOn = true)&&(autoFire = true)) {  
483         triggerControl2();  
484     }  
485     waitUntil(!flyWheelSensor.pressing());  
486     flyWheelDisc = false;  
487 }
```

whether a disc is in the mechanism or not. This function has a special addition to it. If the autoFire is enabled, and the turret is locked onto the goal with correct flywheel velocities and correct angle. The Pneumatic fires and sends a disc into the fast-spinning mechanism which will launch it into the high goal. After it does this the program

removes one from the overall disc count. The code is on the right and the trigger control on the left.

Pre-Coding: The Aimbot Part 6

I have covered the sensors and the internal disc loading systems; now it is time to cover the thread as a whole. Before covering the code I would like to mention its unique positioning in the code. The thread is placed below the control periods functions and right before the autonomous. Threads normally are placed after the PIDs and before the control period functions

Thread 4 is very small. If you notice the code before is not within the thread. That is intentional because of the `sensor.pressed()` command requires a callback. The callback in this case is the previously mentioned sensor functions. The thread checks if the robot has 3 discs and stops the intake if that is the case. Unlike the other threads, the refresh rate is very high compared to Thread 1 or 3. The reason why is that there aren't any calculations here. It's mostly just sensor inputs. The thread doesn't control any fine movements just binary pneumatics.

```
512 void thread4() { // auto Fire Thread
513     intake.setVelocity(100,percent);
514     while (true) {
515         intakeSensor.pressed(intakeRead);
516         turretSensor.pressed(turretRead);
517         flyWheelSensor.pressed(flyWheelRead);
518         if ((discLoad = 3)){
519             intake.stop();
520         }
521         else {
522             intake.spin(forward);
523         }
524         wait(25,msec);
525     }
526 }
```

Flywheel Programming

The Flywheel will most likely have a fix on either Monday or Tuesday. For now, I will refer to flywheel speed as "flyWheelVelocity" and flywheel mass as "flyWheelVelocity". Finding the required flywheel velocity will take a lot of steps. First I will get the distance. Let's assume we are in the center of the field which is 6 ft in every direction. Using the distance formula we are roughly 8.55ft away assuming in this case the goal is in the very corner (large assumption here). Next, we need to find a way to get time. I talked with one of my friends named Will who is in an advanced physics class. We talked about ways of getting time only to end up setting a base time. The base time was decided to be 0.61 seconds from the middle of the field. The time of 0.61 will NOT be the time for all distances. I will be using a ratio of 8.55:0.61 for all distances. Yes, I know this is a vague assumption to use but whatever. I will update this when I find a better way to get a time measurement.

Meeting #20

No images today

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Chassis V2 / No field or access to club part storages.	Build a prototype
Ethan	Troubleshoot intake/ improve chain system	Build a prototype

Meeting #20 Continued

Post Meeting Info

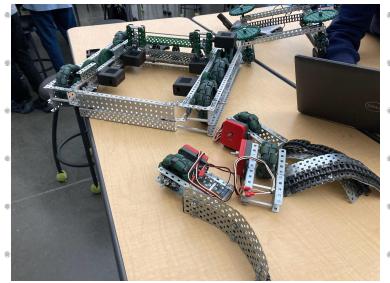
Meeting Info
Location School: iTech Preparatory
Attendees: Aidan, and Ethan Duration: 10:55 - 1:20

Team Members	What got done	Unresolved Problems	Images
Aidan	Find issues in chassis/Get aimbot Calculations, create turret wheel	No middle wheel, No Odom wheels,	No images today
Ethan	Worked on intake	The flaps still have to be geared down to the intake wheels, and the	

Meeting # 21

Robot beginning of
Practice

Pre-Meeting Plan

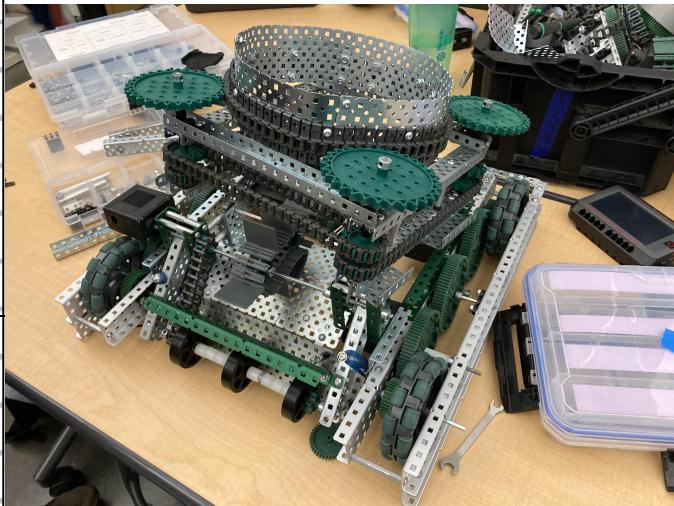
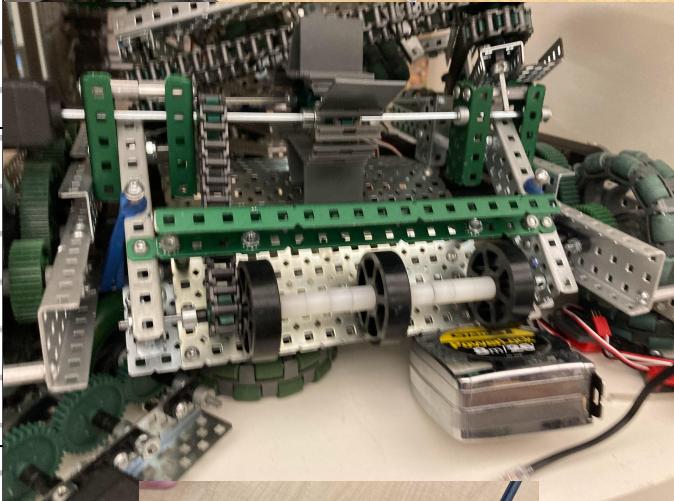
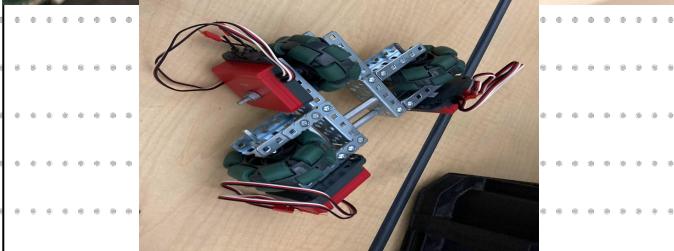


Team Members	Tasks and Obstacles	Design Process step
Aidan	Work on the turret wheel	Create a Prototype
Ethan	Finalize the intake and install odometry system	Create a Prototype
Josh	Mount the turret	Create a Prototype
Tyler	Mount the turret	Create a Prototype

Meeting #21

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Turret wheel almost done		
Ethan	Intake almost done/ Odometry mount		
Josh	Turret mount brackets	No turret mounted	
Tyler	Designed logo		

Project Meeting #21 Continued

Name Ethan Regan

Date 11/1/22

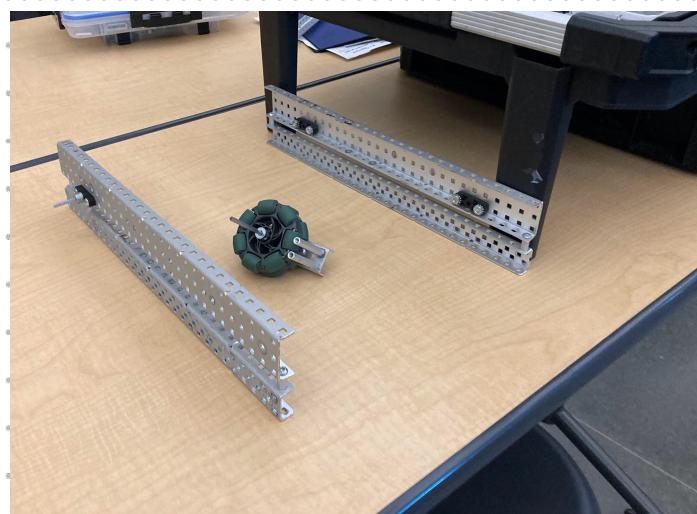
Page 128

October Monthly Update

This month of October was successful. This is the month we started building. The notebook has expanded by 93 pages which are around 3 pages a day. The code has mostly been written this month to be tested later. The progress we made this month was exponentially more than in September. Before I go into more detail I will try to do these each month.

The Robot

The Robot has changed a lot this month. From not existing to what we have on the bottom. The robot on day of building is on the left here. It has the beginnings of the chassis and a work in progress odometry wheel. On the bottom we have the turret on the bot but not mounted, chassis mostly complete. The beginnings of the intake are shown here. The image is somewhat old so it isn't accurate.



The Code

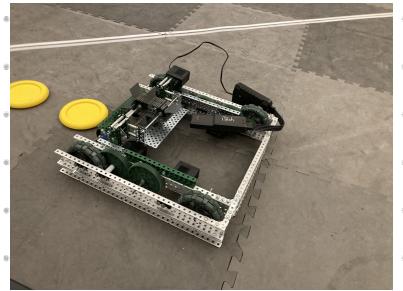
The code has changed a lot. I have been coding a small bit before making entries defining the basis of the odometry. I have mentioned some of this code like the original idea to use the inertial sensor for Odometry. The aimbot should be coded by next week and we may begin the coding sections of the robot.

The Notebook

The notebook has expanded a lot since the start of the month. This month we covered a good balance of pages ranging from strategy and coding to basic formatting. A lot of the foundational parts of the notebook were put into it. This next month will be somewhat slower notebook-wise as we enter into a long testing phase.

Meeting #22

Robot beginning of
Practice



Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Coding and turret wheel	building/design
Ethan	Finalize and mount odometry wheels, then continue on intake presuming the turret is mounted	building/design
Josh	get the turret mounted	building/design
Tyler	Research notebook changes	Research the problem

Design Process: The Logo

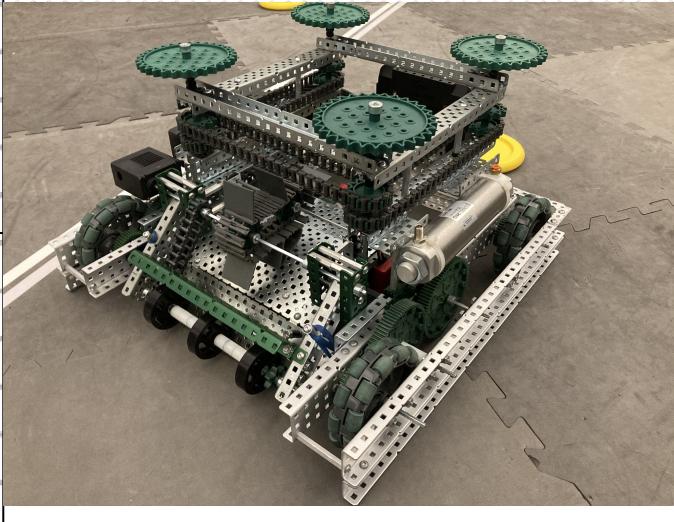
We decided to make a logo for the team. We made it based on the Costco brand Kirkland's logo. We decided to do that because of an ongoing joke that our team is Kirkland brand engineering. I made the logo using photoshop. To make the logo I first found the font closest to the "Kirkland" part of the Kirkland logo. I used the font Kanit to type out infrared, put it behind a black background, and then screenshotted it to make it a png and put it into photoshop to edit. I then cut out the middle of the a to insert the red swoosh from the Kirkland logo. I made a new photoshop file to put the Kirkland logo in to cut out the red swoosh and put it into our logo. Once I had the white "Infrared" part of the logo I had to do the red cursive part. There are no fonts that look cursive and are connected so I found a cursive font generator. I picked one that looked similar to the "signature" part of the Kirkland logo. I put engineering into it and pasted the result into the photoshop file.



Meeting #22

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Found and fixed issues in code. Stabilized turret wheel		
Ethan	Mounted odometry wheels and turret		
Josh	N/A	N/A	
Tyler	Researched the Changes		

Pre-Coding: The Aimbot Part 7

So I got some formulas! I talked with my physics teacher and he was able to give me some help with my coding. First I got a formula to get was to get the change in X that uses velocities but without the Time variable. The second tip he gave me was that I needed to use rotational torque to calculate the transfer of energy.

The formula I got was very useful. It can calculate the x distance from velocity, launch angle, and vertical travel. Here is the formula below. I need to somehow figure out a way to get V. I have X, Yo, Yf, G, and launch angle. My first attempt was to try to put it into an algebra calculator. It did not work at all. To see some of the calculator's failings check below the formula image. Now some of these failings were my fault but in general, the velocity was usually short by 2 feet. The way I double-checked this was by using omnicalculator. So even if I was slightly off the overall formula I was given was still off by a lot.

$$\Delta X = \frac{-V_0^2 \cdot \sin(2\theta)}{2 \cdot g} \left[1 + \sqrt{1 + \frac{-2 \cdot g \cdot (Y_0 - Y_f)}{V_0^2 \cdot \sin^2(\theta)}} \right]$$

↳ where: $g = -9.806 \text{ m/s}^2$

V_0 = initial launch velocity Y_0 = initial height
 θ = launch angle Y_f = final height

$$x = \sqrt{\frac{d^2 v^2 g + c v - c u - c d^2 v + d v \sqrt{d^2 v^2 g^2 + 2 c v g - 2 c u g - 2 c d^2 v g + c^2 d^2}}{u - v + 2 d^2 v}}$$
$$\sqrt{\frac{v \sin(2\alpha) + z^2 g \sin^2(\alpha) - z \sin^2(\alpha) \sin(2\alpha) - z \sin(\alpha) \sqrt{z^2 g^2 \sin^2(\alpha) + 2 v g \sin(2\alpha) - 2 z g \sin^2(\alpha) \sin(2\alpha) + \sin^2(\alpha) \sin^2(2\alpha)}}{-y + 2 z \sin^2(\alpha)}}$$

Pre-Coding: The Aimbot Part 8

When I was trying to use the algebra calculator I made the large mistake of putting the code in without checking the calculations. I did this yesterday night right before I went to sleep and realized I should've checked if it was right. In the morning my fears were confirmed. Below is the longest line of code I have written and also my second biggest mistake when it came to coding this year.

```
ejectVelocity = sqrt(pow((sin(flyWheelAngle)),2)*pow(findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2)*gravity+((sin(2*flyWheelAngle)*turretOffsetZ)-(sin(2*flyWheelAngle)*(zCoordinates[1]))-(sin(2*flyWheelAngle)*pow(sin(flyWheelAngle),2)*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]))+((sin(flyWheelAngle))*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target])))*sqrt(pow((sin(flyWheelAngle)),2)*pow(findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2)*pow(gravity,2)+2*(sin(2*flyWheelAngle)*turretOffsetZ*gravity)-2*(sin(2*flyWheelAngle)*(zCoordinates[1])*gravity)-2*(sin(2*flyWheelAngle)*pow(sin(flyWheelAngle),2)*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]))*gravity)+pow(sin(2*flyWheelAngle),2)*pow(sin(flyWheelAngle),2))/(zCoordinates[1]-turretOffsetZ+(2*pow(pow(sin(flyWheelAngle),2),2)*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target])),2)));
```

I'm not going to go into any detail about this. If you are wondering where the formula was. It was the first failed formula from part 7.

Now how am I going to calculate this now? Well, I am still in the process of figuring that out. I got around 17 days to do so, so wish me luck! My process so far is to decode it myself. The $\frac{1}{2}$'s part of the formula came from a quadratic formula: $yf=g/2*t^2+Vo*t+yo$. This was turned into a time formula. This method of getting time was later used in the formula $x=Vx*t$ or $x=Vd*cos(a)*t$.

From this really basic formula $X=Vod*cos(a)*t$ I think I can somewhat get time out of it with surrounding factors. I also can use the previous formula to get yf to get time and velocity as well. I can merge the two equations to try and get time and velocity. First I need to transfer the X and Y values into the distance. The robot has both X and Y values being the X of the robot's position in X and Y and the Z from the vertical position. I can merge $\Delta X=vd*cos(a)*t$ into $\Delta D=V*t$ by removing the conversion from d to x.

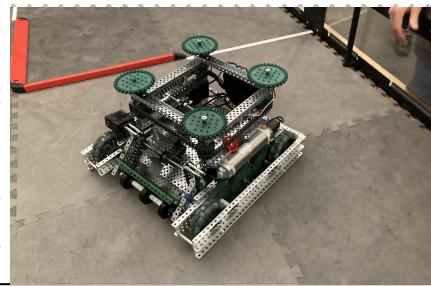
Onto the Other formula. yf can be transferred to D by using the distance formula. Yo can be converted to Do by doing $Y*sin(a)$. The answer to Formula one requires ΔD so I will remove the Do . The two formulas combined look like this: $\Delta Df=Vo*t=g/2*t^2+Vo*t$.

Let's use some theoretical calculations now. Let's assume the distance is 8ft and the angle is 35. Let's see if I can try and get either velocity or time. When I get my result I'll write back here.

Meeting #23

Robot beginning of
Practice

Pre-Meeting Plan

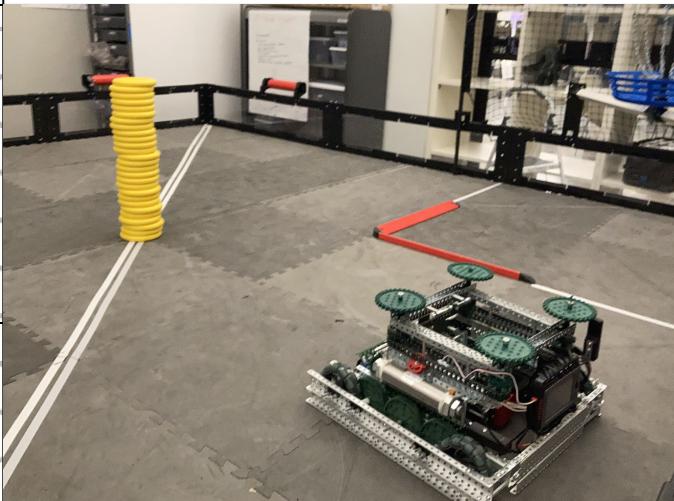
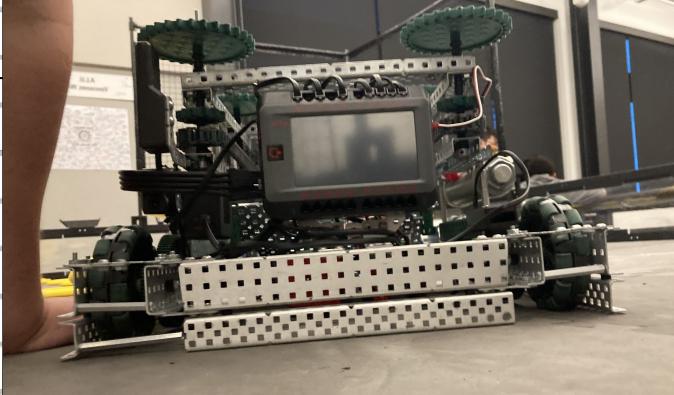


Team Members	Tasks and Obstacles	Design Process step
Aidan	Coding	Build a prototype
Ethan	Fix odom wheels	Build a prototype
Josh	Began designing a Expansion mechanism	Design a solution
Tyler	Round out the turret wheel	Build a prototype

Meeting #23 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Got threads to work in the code	Chassis cannot function under current load	
Ethan	Brain, Battery, and radio mounted	Chassis cannot function under current load	
Josh	Began designing the expansion mechanism	Chassis cannot function under current load	
Tyler	Rounded out turret wheel so it can be properly mounted	Chassis cannot function under current load	

Design Process: Tournament Flyers

Version 1

Tournament flyers are a great way of presenting your team to others. It invites includes a formality to be had with other teams. It helps your team gets recognized. It may even help get the Excellence award. I want to make one for 3249V as 3249V has been made in the past.

Before we begin something you may wonder is why this is in the Design Process. Well, this is the last and most crucial step of it; the step being "Present the solution". The fliers can help communicate to those who want to alliance with your robot. This is like finding a manufacturer for your product. Of course in this case the robot isn't finished and will never be. This flyer may go through multiple iterations as we get closer to the tournament.

For the design, I made I went with a basic color theming and the theme of neon in my head. I chose neon because I thought it relates to the theme of infrared. Mainly the idea of lights. I used mostly red and purple throughout the flyer. The red comes from the infrared name. Purple comes from the iTech high school colors Purple. Another reason is that my school's colors are Green and Purple and my coach Ms. Hagin would get mad at me if I didn't have any purple.

On the title, I put the 3249 in Neon green and purple with the Minimized V. Next to it is the full Infrared engineering logo. I next divided the flyer into 3 sections. "The Team", "The Robot", and "Key Notes". The team section will have a photo when we take it with our new lab coats on Tuesday. The team section also lists the Team members, Team roles, and our years in vex.

The robot section contains the specs of our robot. These currently are the planned specs we wish to have at the tournament. Such as the roller mech and the Flywheel speeds. Lastly, I finished it off with special notes about our teams such as the aimbot and the ease of strategy.

Something felt like it was missing when I was finished so I decided to make a background. I first made a diamond, I made more inside each other with each becoming smaller. I made the color pinkish red trying to mix Purple and red. Before adding it to the main poster I set the transparency to 20%. I am very happy with how this first poster looks so far, and I hope to make it better in the future.

3249V

INFRARED

Engineering

The Team

Aidan - 4 Years - Coder

Ethan - 2 Years - Builder

Tyler - 1 year - Driver

Josh - 3 Years - Builder

Key Notes

The Aimbot

The Aimbot is the Keystone of the Robot. The robot through the use of Odometry can automatically aim and fire at the goal. The Robot is built around this and programmed for this intent. The robot can take around 0.5 seconds to get a disc from the ground and into the air

The Robot

360 Degree Turret

Dual Motor 19 Knot Flywheel

4.09mph Chassis

Two Passive Rollers

Catapult String Expansion

Banana Capacity of ##

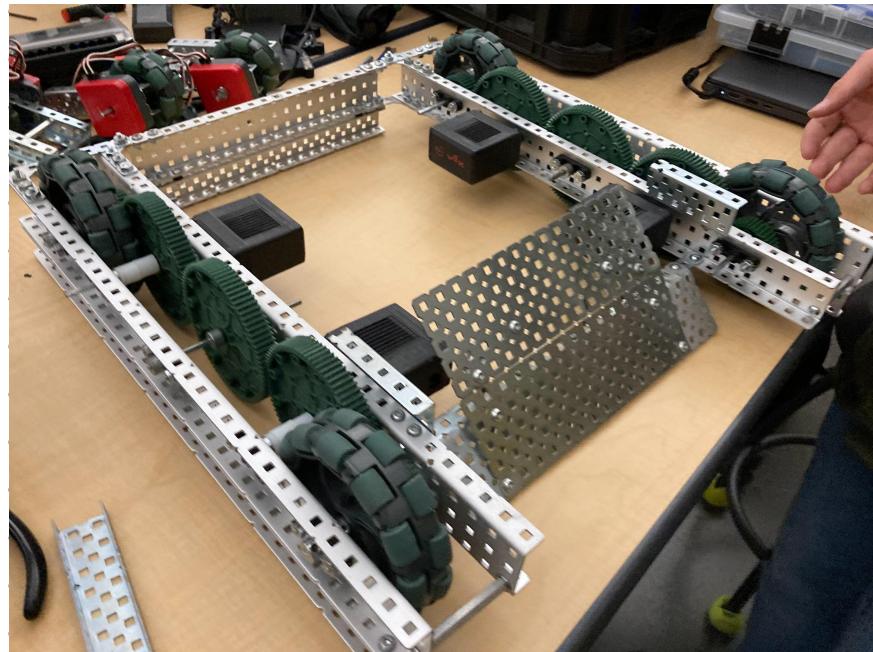
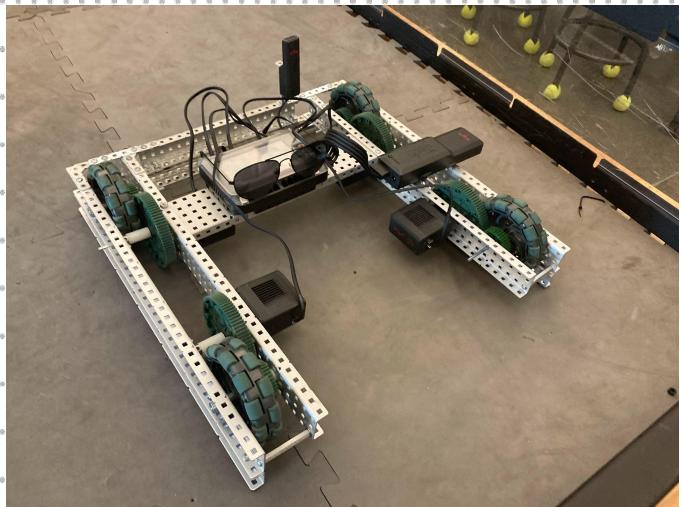
Flexible Strategy

We have many strategies. If our strategies conflict another program can be made within 5 minutes. If you're our alliance member come talk to us to make sure we can cooperate to our best ability.

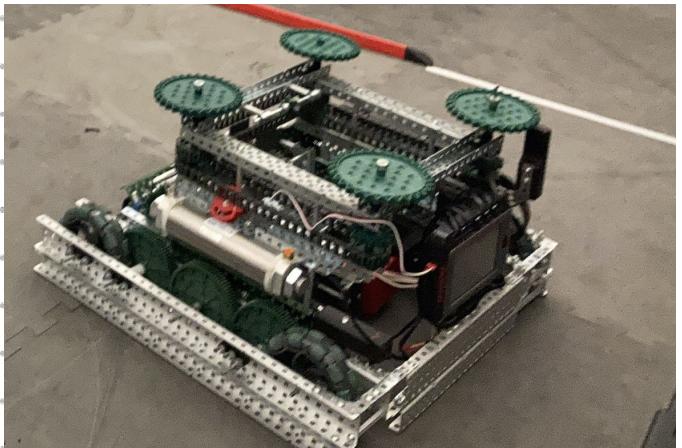
Design Process: The Chassis

Version 2

Chassis version two is an upgraded version of chassis version 1. Its upgrade is very simple yet not that different. On the left is chassis version 1. The brain is placed in the center and the battery is placed in a temporary spot. It has messy wiring and basic speed gearing. The only thing that will stay the same throughout the game will be the chassis U frame. On the bottom here is the chassis Version 2



Version two changes the location of the brain and battery which I will get to later. It also contains a 3rd middle gearing. This gear combines the torque from both motors on each side. This allows the chassis to overpower bots if needed. It contains the base of the intake ramp as well. The brain and battery were changed to be on the turret so we can fit more items in the chassis such as odometry wheels.



Not until recent testing that this will be our chassis final design. Well in recent testing the weight of the turret slowed the robot down significantly. We all concluded that this was the fault of the gearing. I'll discuss our next plan to fix this in version 3 where we will change the gearing.

Design Process: Wiring The Robot

Version 1

The Robot unfortunately needs to be wired. This is the job of the coder. Why is it the job of the coder? Well the coder sets the ports of all the motors, sensors, and 3-wire inputs and outputs in the code. To help my team wire the robot I am creating a diagram they can reference back here.

First how does wiring work?

Every electronic on the robot needs to be connected to the brain in order to receive commands. Or in cases like the Odometry wheels information is provided to the brain. The Vex V5 Brain has in total 21 Smart ports and 8 3-wire sensors. 20 of those smart ports are on the front and one on the side. We also have 8 extra 3 wire sensors due to me buying a V5 3 wire Expander. The robotics club didn't have any so I had to buy one for my team. Wiring setup on page 141.



Now why should I use the 3 wire sensors over v5 sensors? Well the 3 wire sensors have faster response rates. Another reason why is that my robotics club doesn't have any v5 sensors for odometry other than the inaccurate inertial sensor and vision sensor.

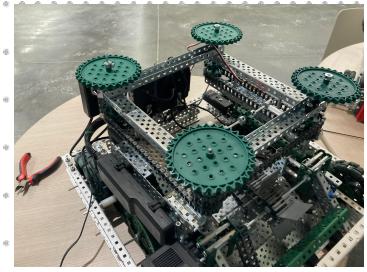
Port	Device	Name
1	Motor	leftFrontMotor
2	Motor	leftBackMotor
3	Motor	rightFrontMotor
4	Motor	rightBackMotor
5	Motor	flywheel
6	Motor	turretMotor
7	Motor	flywheel2
8	Motor	intake
10	3-Wire Expander	expander

3-Wire Port	Device	Name
A	Potentiometer	PotentiometerA
B	Pneumatic	expansion
C,D	Encoder	encoderR
E,F	Encoder	encoderL
G,H	Encoder	encoderB
ExA	Limit Switch	intakeSensor
ExB	Limit Switch	turretSensor
ExC	Trigger	trigger1
ExD	Trigger	trigger2
ExE	Limit Switch	flyWheelSensor

Meeting #24

Robot beginning of
Practice

Pre-Meeting Plan

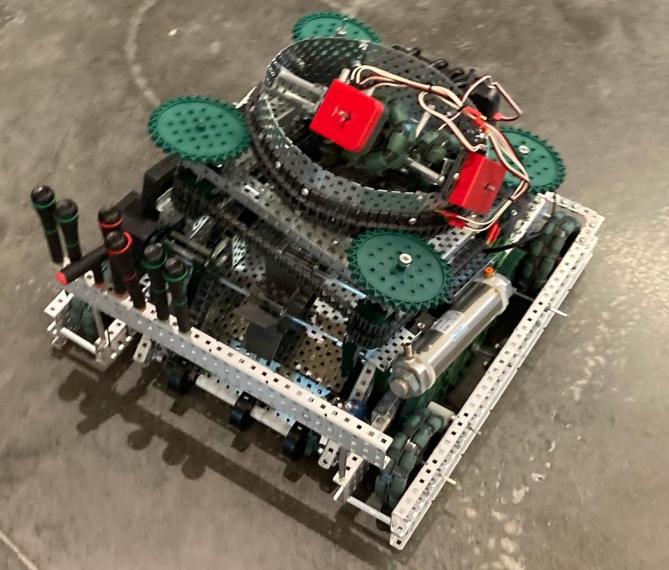


Team Members	Tasks and Obstacles	Design Process step
Aidan	Figure out issue with the Robots Chassis	Evaluate the Solutions

Meeting #24 Continued

Post Meeting Info

Meeting Info
Location School: iTech
prep
Attendees: Aidan
Duration: 10:55 - 1:15

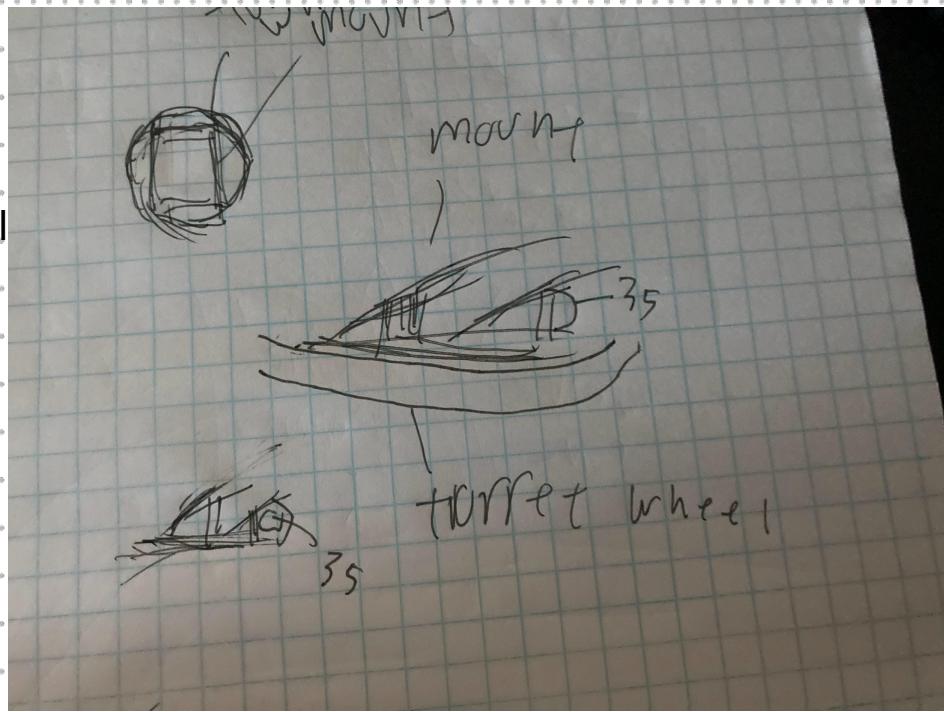
Team Members	What got done	Unresolved Problems	Images
Aidan	Found and somewhat fixed an issue with the chassis camber. By installing a front channel	The camber is still there and needs to be fixed	

Design Process: Flywheel Mount

Version 1

The turret wheel is almost finished. Meaning it is almost time for the flywheel mount to be made. The flywheel mount will be placed on the current itself and will serve as the junction point to where the flywheel is to be mounted. It will contain the flywheel trigger system. The flywheel trigger system will contain a pneumatic piston as well as the flywheel Limit switch.

The design has two parts to it. The connecting part is two c-channels that point inwards towards each other. They're connected by a 3rd and 4 channel that goes on top of the other two c-channels. They are slightly shorter than the other two c-channels to prevent them from touching the turret wheel. Another thing is that they're on the outermost part of the mount. They're not present in the middle sketch but are present on the top part of the paper on the right.



This will still allow for a disc to go through the mount without the mount being unstable.

The next part is angled portion of it. It will contain 4 angled U-channels. These U-Channels will be small and have 2 connection points. A connection point on first part of the mount, and a second point using a small c-channel most likely 3-5 long. The angled portion will be very thin only being able to have a single disc in it as well as width of the flywheel.

The flywheel should be able to be mounted easily to the flywheel mount. Simply by sliding the new flywheel into the mount. The flywheel has to be mounted fairly close to the clip as the total range of a pneumatic is 2 inches. I also have to make some space in the clip for the limit switch. I'll try to start building during the upcoming practice 25.

Meeting #25

Robot beginning of
Practice

Pre-Meeting Plan

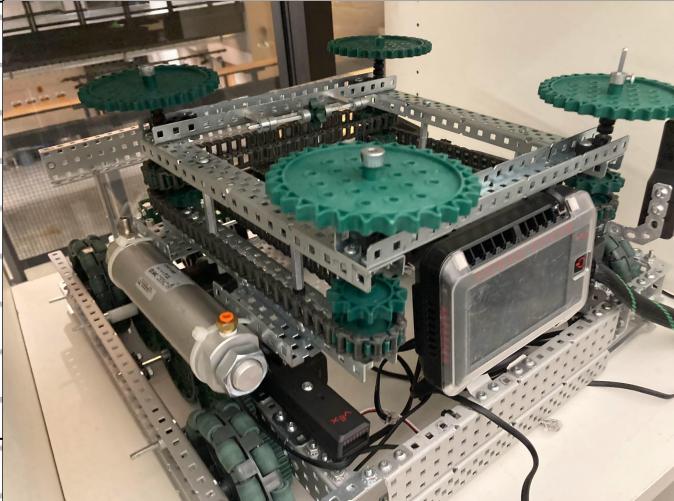
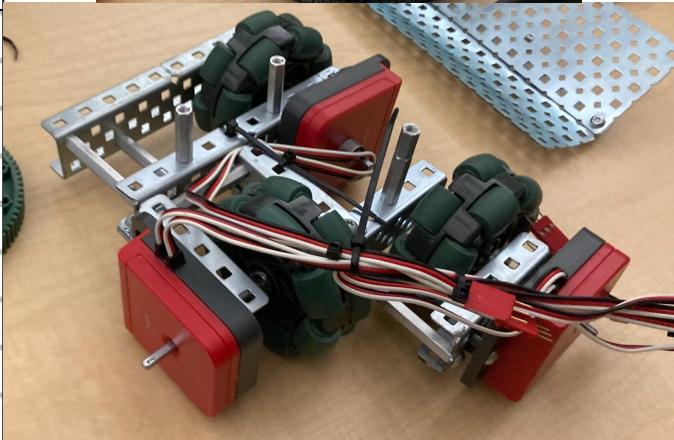


Team Members	Tasks and Obstacles	Design Process step
Aidan	Create the Flywheel Mount	Build a prototype
Josh	Chassis Redesign	Build a prototype
Tyler	Chassis Redesign	Build a prototype

Meeting #25 Continued

Post Meeting Info

Meeting Info
Location School: iTech Prep
Attendees: Aidan, Josh and Tyler
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Began Mount for Turret	Mount not complete	
Josh	Began changing the gearings the chassis so our chassis can support the weight of the bot	Chassis V3 isn't complete	
Tyler	Created Locked Omnis	Chassis V3 isn't complete	

Design Process: Progress Check

Upcoming tournament Progress check

As the tournament creeps closer we need to make sure we are on track. They're are as of now 10 days till the november 18th league and 11 days for our November 19th tournament. To stay on track Tyler and I have made a checklist that we plan to complete by the 19th. Here is the list below

- Indexer V1
- String Launcher V1
- Chassis V3
- Rollers V2
- Flywheel V3
- Odometry Wheels V3
- Reorganize Toolbox
- Team photo w/clothing*
- Poster Redesign*
- Aimbot Fully coded

All the objectives with * next to it are optional.

This seems daunting to complete. Well it is, its a near redesign of the entire robot 10 days before a tournament. To help complete the goal we plan to do work sessions at Tyler's house so we have more than 2 hours to work on the robot at once.

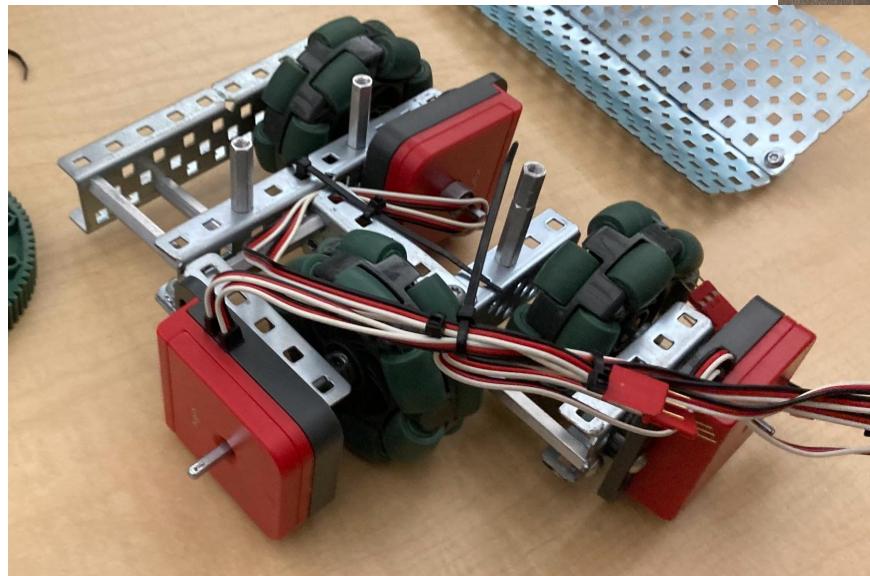
These out of school work sessions are planned to take place wednesday and friday. Wednesday we plan to get sketches and plans for Friday. On friday we have a day off because of veterans day. So we will spend the free day building portions of the robot. If everyone can make it to the Friday work session we might be able to completely finish the robot by the time we finish. If not we can try to complete it all by Thursday or maybe not at all. If everyone cannot make it we may do heavy notebook focus in order to get the Design award at the Competition.

Design Process: Odometry Wheels

Version 2

Odometry wheels V2 is a far more compact version of the Odometry wheels V1. On the left are the old odometry wheels. On the left you can see the old odometry wheels, and below you can see the new odometry wheels. The reason why we made V2 was so it can be mounted easier. We plan to mount it to the back of the robot with a suspension system. We used an extra 5x2 C-channel to mount the left wheel. The right and back wheels were mounted together by the open parts of the c-channels.

In theory this still should work in the code. If my team can get the correct measurements than I can see it working.



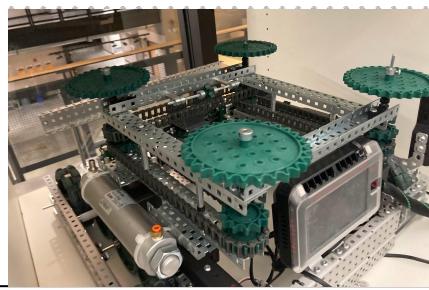
If it doesn't work. We need to redesign the odometry wheels into Version 3. In my observations when messing around with the odometry wheels is that my Odometry code is flawed. Specifically the backwheel code. I forgot that the backwheel can track both Turning and overall movement. So I need to adjust the code to attune for that. My plan to fix it is doing the degree code for the

back wheel. Checking if it matches with the left and right Wheel I will create a "backDiffrence" variable which will cancel out the movement code. If the turning is higher than the left and right wheel I will make the left over go into the movement

Meeting #26

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Create the Flywheel Mount	Build a prototype
Ethan	Make the Indexer	Build a prototype
Josh	Continue making chassis Version 3	Build a prototype
Tyler	Creating a Roller design	Design a prototype

Meeting #26 Continued

Post Meeting Info

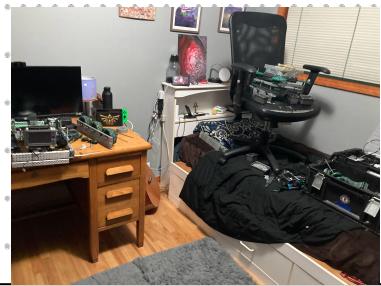
Meeting Info
Location School: iTech Prep
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Created Flywheel Mount V1. I also talked to 3249U's jonah and got calculations for the Aimbot		
Ethan	Began creation of Indexer	He will be gone till either sunday or tuesday	
Josh	Began Chassis V3		
Tyler	Created the designs for Roller mech v2		

Meeting #27

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Create Version 2 of the Flywheel Mount and the flywheel. / No access to any parts, storage bins, or anything we would have at Itech	Build a Solution
Tyler	Organize the bin and continue chassis Version 3 / No access to any parts, storage bins, or anything we would have at Itech	Build a Solution

Meeting #27 Continued

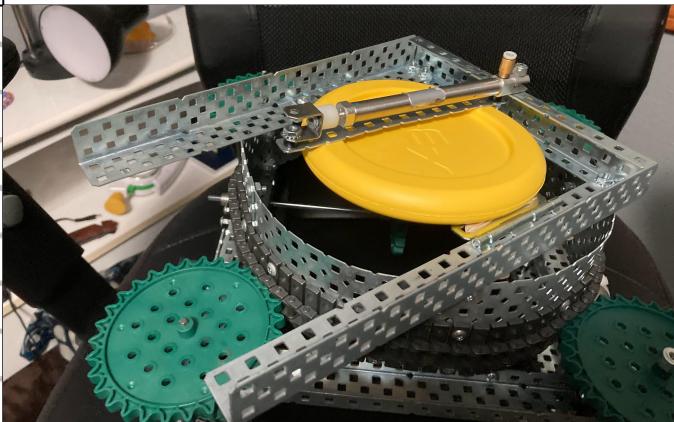
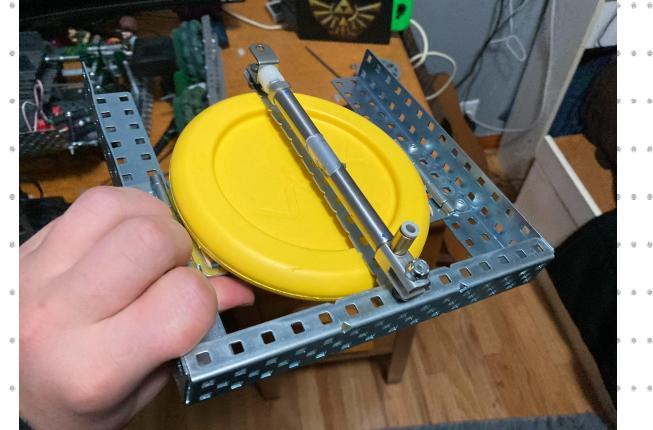
Post Meeting Info

Meeting Info

Location: Tyler's house

Attendees: Aidan and Tyler

Duration: 6:30 - 9:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Improve the Flywheel mount creating a Version 2	We didn't have the materials required for The Flywheel itself	  
Tyler	Organize the robot and work on create chassis version 3	We needed to get 60 tooth gears for spacing instead of the	

Design Process: Flywheel Mount

Version 1 Continued

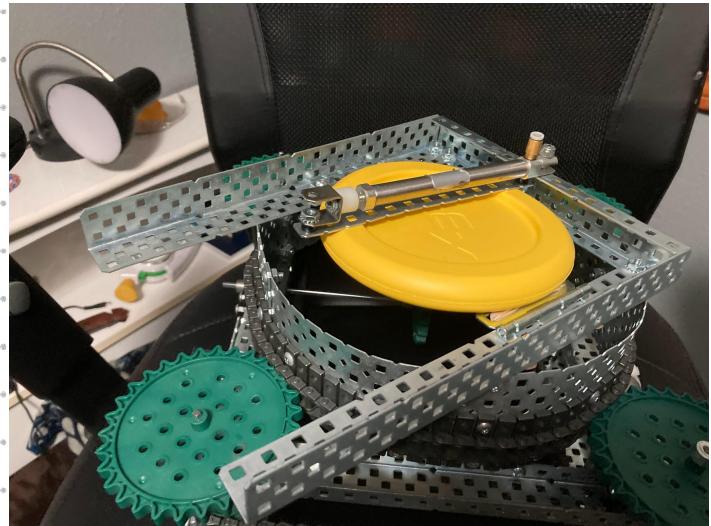
Today at practice 26 I was able to create the basic idea for the flywheel mount. It wasn't complete but it was still an iteration. I made the slot where the disc would go and get shot out of.

The design so far is simple. Its two 20 long angle brackets screwed together with a connecting C-Channel. The c-Channel is 15 holes wide which is wide enough for a disc to go through. The disc stays in the mount using a one way latch system. It's made by using a hinge two rubber bands and a $\frac{1}{4}$ inch screw.

It works by having a hinge positions 2 holes deep in the Angle bracket so I can only go in one direction. The two $\frac{1}{4}$ inch screws are put in with two Keps nuts on each screw. The rubber band then goes through the hinge and both screws. The rubber band kind of folds like a U shape. So when the disc passes all the way through it is locked in. I got the rubber locking mechanism from 3249V's Sean last year on his experimental 7 motor drive. If you can't tell it was a joke bot made at the end of the year.

The pneumatic here acts as a trigger mechanism. It is also very unstable in its position. I don't know much about how to mount a pneumatic but I know this isn't one of them. At the end of the pneumatic is a 9 long 1 by 1's that spans underneath the pneumatic. It acts as a hook shape so so it can properly fit in the size constraints of the turret. At the end is a 1 long single piece used as a trigger piece nylocked to the 1by 1.

Overall this design can be improved especially by how the pneumatic was mounted and its overall length. The brackets are fairly long so it's hard to actually mount a flywheel. In verison two I plan to cover this and the pneumatic as well as the actual purpose of the mount. I hope to have version two ready by the end of practice 28.



Meeting #28

Robot beginning of
Practice



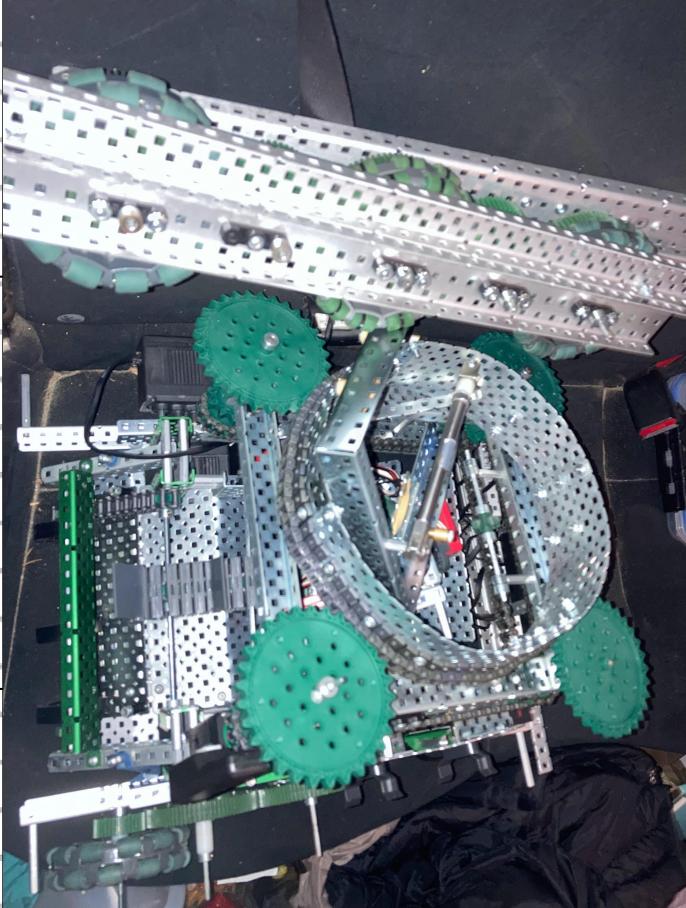
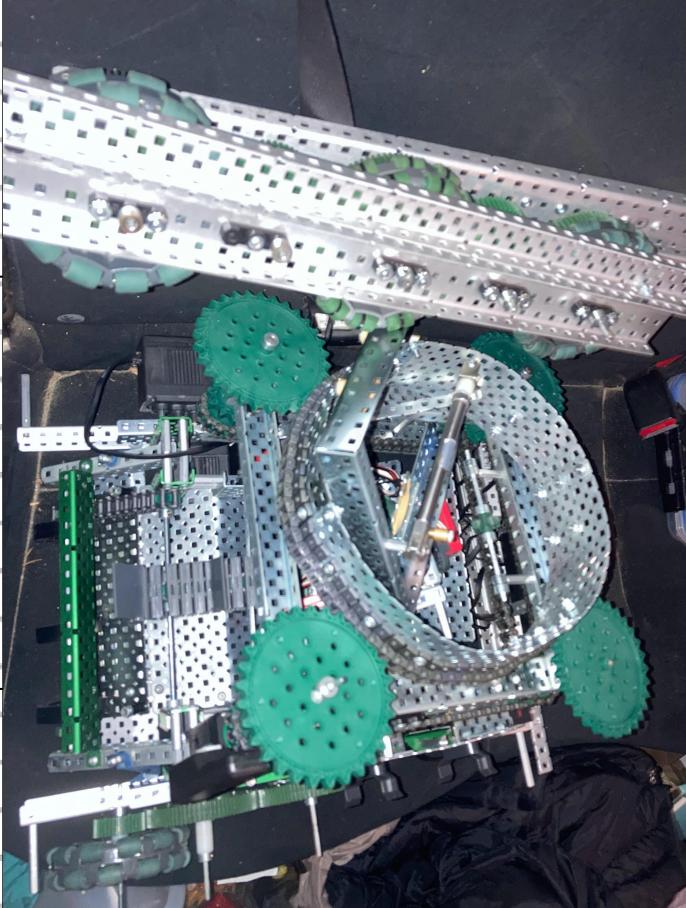
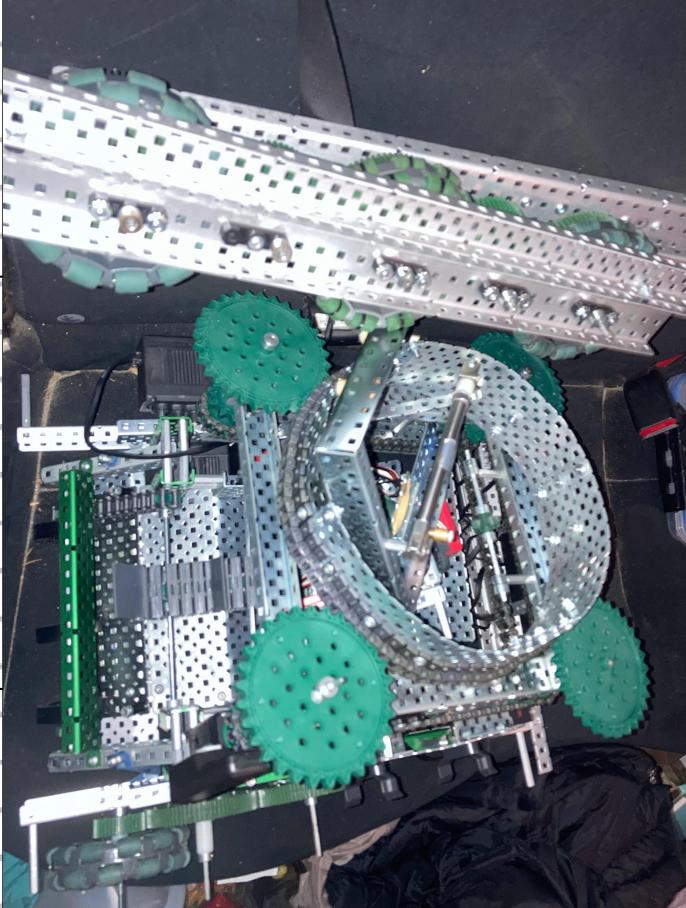
Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Continue working on the flywheel mount. At 5:30 we will all collect materials for Sundays build session	Build A prototype
Josh	Continue working on Chassis Version 3. At 5:30 we will all collect materials for Sundays build session	Build a Prototype
Tyler	Continue designing the roller mechanisms. At 5:30 we will all collect materials for Sundays build session	Design a Prototype

Meeting #28 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Continued working on flywheel mount version 2. I got parts for a potential string launcher, indexer, roller mech, and any design we may need.	Ethan hasn't made a CAD of his indexer design. So I plan on making some designs on my own.	
Josh	Worked on chassis version 3. One side is almost complete	Chassis still cannot drive	
Tyler	Continue designing his roller mechanism.	Had some issues designing a roller mech.	

Design Process: The Indexer

Criteria and Constraints

The indexer is literally the heart of the robot. It “pumps” discs from the intake to the flywheel. Now we need to have this design prior to Tuesday as the robot desperately needs to have something of an autonomous. Ethan was supposed to send me a CAD file of the autonomous wednesday night. He for some reason wasn’t able to send it so I am forced to make a design for Sunday.

The indexer is something we never had a solid idea of. Other mechanisms on our robot either had one good design or we could only find one design. I have three designs for the indexer so a decision matrix is needed. I will be going over each design in as much depth as possible.

Before covering designs I need to have some criteria and constraints before explaining the designs. The criteria and constraints will be based off of basic values. Such as overall quality requirements that will make the system Viable for gameplay. The ability to reach a specific point, and its legality in the VEX rules.

Criteria:

- Must Extend a minimum of 5 inches
- Must load a disc in less than a second
- Has efficiently use the Air in the pneumatics
- Required to be consistent with failures being mechanical problems only
- Compatible with match loads and Preloads

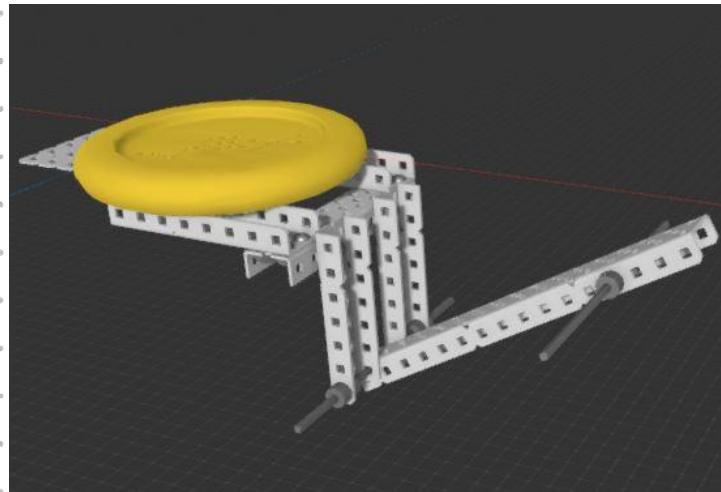
Constraints:

- Must follow all Vex rules
- Can only use a single piston and or potentially the intake motor.
- At least 30 uses per match with the other mechanisms.
- Has to fit within an elevated center of the robot without touching the ground.
- Required to have a spot to insert a limit switch for the Auto firing systems.
- Has to be open enough for a disc to be taken out at the end of the match.

Design Process: The Indexer

Design 1: Lever Extension System (LES)

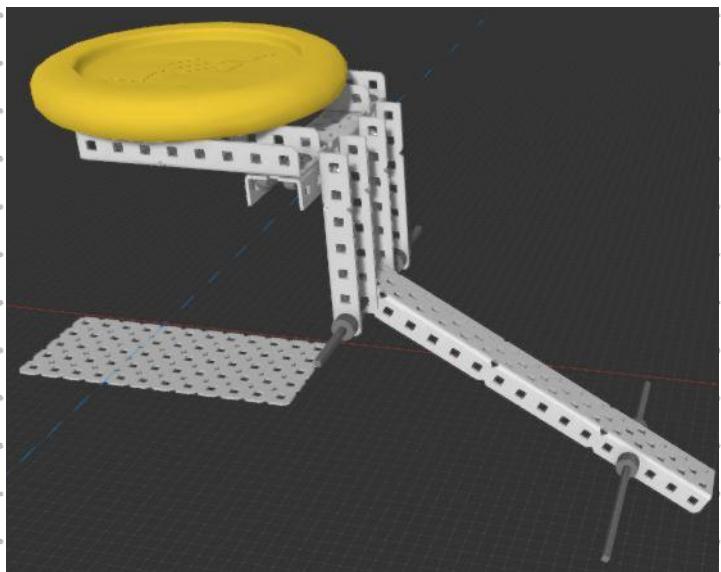
The first design is a simple lever mechanism. This mechanism was based off of a design from the Last years 3249V where they designed a pneumatic rake system which pulled a rake system up and down. The concept is similar here where a pneumatic will push a c-channel downwards. On the right I CAD'ed the design so you can see a demonstration of this.



Indexer Down

In this system pivot point is close to the pneumatic which would push two inches down. Which is reflected to be around 7-8 inches in this model. The downside of this is that more energy is needed to push the disc platform up. Which would result a increased need for pressure.

The design on the right wouldn't be its actual design if built. Everything except the lever would stay the same just a different lever. As you can see the system will just fall downwards. To fix this I would use a linear slider. I didn't include the linear slider or a pneumatics because the program I used called Probot doesn't include pneumatics, and linear sliders yet.



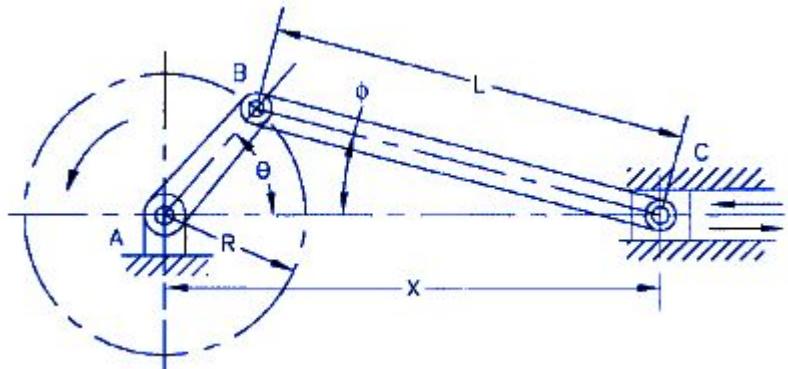
Indexer Up

Overall this design in my opinion may not be a good approach. Its bulky and requires a lot of power. Its one upside of the Lever extension system its its speed. If the power was adjusted correctly. The system could feed 2 discs a second assuming that the pneumatics have a response rate of a quarter second. Honestly this is a good first attempt at a indexer design on my part and I will still include this in the decision matrix.

Design Process: The Indexer

Design 2: Steam Engine Slip Gear (SESGM)

Researching broad things in google may not give results that you were looking for. Somehow this time it yield a success. When searching the term "Piston mechanism" I found a mechanism that actually may yield success. From the website Engineers edge. I found an image of an image of what they describe as a "Piston slider crank mechanism". It utilizes a motor to turn a object on a slider back and forth transferring the rotational movement into linear movement. It also describes the calculations to calculate linear velocity, distance, acceleration, etc.



Why would a mechanism like this be useful? Well I could reverse its function. Instead of having a wheel drive a piston I will have a piston drive a wheel. Kind of like a Steam engine or a motor. Now why would having a spinning wheel or gear be useful? Well I could transfer this rotational energy into a linear slider.

A Pneumatic can only extend 2 inches. So I need to make the most of this range. According to the Vex website a 60 tooth gear is 2.5 inches in diameter. This is farther than the actual range of the pneumatic, but it only needs to be in contact with the Screw holes of the gear not the full 2.5 inches in diameter. So in theory the pneumatic should be able to reach the full range it screwed properly.

The circumference of a 60 tooth gear is roughly 7.85 inches. Which fits the criteria. Now I should explain now why there is such an increase in distance. The pneumatics overall turn radius is 1 inch ($2/2$) or roughly 6.28 inches.



This would normally be fine if we are not accounting the track going downwards. This is the Slip gear part of the name. Half of the gear will need to be shaven off so it can go downwards so now we have 3.925 which is no longer enough to fit the criteria. On the same shaft as the gear I will add a 84 tooth gear which will add as a slip gear. And 84 tooth slip gear would have a total distance of 5.5 inches which is in the criteria.

Design Process: The Indexer

Design 3: Reverse Intake Mechanism (RIM)

Ratchet gears only go in one direction. So when a motor is turning against them they won't turn. An idea pitched to me by 3249U's Joseph was using the intake motor as a indexer when the motor is reversed to turn a ratchet gear. I wanted to mention this idea as it may actually work and save a pneumatic piston.

To make the design you have to have a single motor gear together the indexer and the intake system. On the indexer you need a ratchet gear to prevent it spinning when the intake is sucking in discs. When the indexer is in use the intake motor is reversed using a linear slider with gear track to send a disc to the flywheel. Its range is depended on the overall gear track so it is theoretically infinite. Its overall speed may be hindered from how many things are connected to the motor.

In my opinion it's not a bad idea. It can be applicable to the small amount of bots that have a turret. For the previously mentioned auto-firing system the system won't work. The robot utilizes the intake, turret, and flywheel as a storage space for a disc. So reversing the intake would just eject the disc inside of the intake and potentially the turret as well. From the programming this would cause the disc limiter to fail and the intake to stop working overall.

Even fixing the code I think the base 200 rpm would strip the gear or bend the shaft really quickly. But there are some upsides. The spare piston could be used to add another piston to the future expansion mechanism meaning more coverage, or more potential shots that the turret can fire. Joseph who went to the October, 29th Sandy tournament mentioned to me that it would take 14 discs to win a round. So we want as many chances to hit 14+ discs in.

Design Process: The Indexer

Decision Matrix

Design:	Lever Extension Mechanism	Steam Engine slip Gear mechanism	Reverse Intake Mechanism
Range (3 if it the mechanism can reach the flywheel 0 if not.)	3	3	3
Speed 1-3 (overall how fast)	3	2	1
Efficiency 1-3 (How much power is used)	1	3	2
Pre-loads and Match Loads (1 if one, 3 if both, 0 if none)	3	3	1
Uses 1-3 (how many uses can be in a match)	1	3	2
Total	11	14	9

Design Process: The Indexer

Decision Matrix Explanation

On the decision matrix I want to explain some of the scores and why I set some of those scores. First of all for all the scores that ranked 1-3 I didn't have any repeat numbers unless I truly thought they tied each other. Each design had their strengths and weaknesses and were each different.

First I want to cover the "Efficiency" and "Uses" scores. Mainly why the SESGM beat out the RIM. The RIM already incorporates drawing from a long motor system that will be running the whole match which may overheat the motor prior to the match ending. Where the SESGM is using a pneumatic piston. Pneumatics are more limited but to my knowledge they don't overheat. The only limit is how much air is in the reservoirs.

Now i want to explain the Preloads and match loads scores. Mainly why the RIM got a 1. Depending on how we process match loads we may either put them in for the robot to pick up or shove them directly into the flywheel. If we go with the pick up plan the RIS cannot score match loads with needed efficiency.

Build Planning

Overall the SESGM was the best design out of the 3. Now it needs to stand the test of its practicality when built. It does require some vertical space so I will need to angle the intake upwards more so it can fit exactly the 5 inches. The remaining 0.5 inches is to make up for the 35 degree angle.

In the sunday practice I won't be able to build the indexer. We don't have any linear slides or gear tracks in our bin. We also don't have any spare HS 84 tooth gears that we can grind down most of it. So the next practice I have access to my robotics clubs storages I will take the necessary parts. I really hope this design can work as our permanent indexer. I will try running tests at practices; but the November 19th tournament will really test it.

Pre-Coding: Odometry Part 13

Fixing the Back Wheel Code

An issue has came up in the odometry code once again. This time this is a oversight I made while creating it. I discussed the problem solution somewhat on page 149 but I will redress it here. In my observations when messing with the Odometry wheels V2 I saw that when turning the back wheels spins a lot. In the current odometry code the back wheels account for any side to side movement so when turning the position tracking could be thrown off by a ginormous amount.

To fix this issue I plan to use a similar turning calculation for the back wheel. Instead of getting degrees I want to see how many degrees turned in the back wheel is required to achieve the same number of revolutions. If it isn't the same the back wheels encoder values that are remaining will be added to the X Y calculations.

```
338     backWheelDifference = encoderVB - offsets[0]*difference/offsets[2];
```

New back wheel difference. It functions very similarly to the difference variable

```
347 if((180 < degHead)) {  
348     positionX += ((dragWheelCirc/360)*(180/M_PI)*((cos(radians(degHead))*(((encoderVL+encoderVR-difference)/2) - (lastEncoderL+lastEncoderR-pDifference)))*-1))+  
349     ((dragWheelCirc/360)*(180/M_PI)*((sin(radians(degHead))*((encoderVB-backWheelDifference) - lastEncoderB)*-1)));  
350     positionY += ((dragWheelCirc/360)*(180/M_PI)*((sin(radians(degHead))*(((encoderVL+encoderVR-difference)/2) - (lastEncoderL+lastEncoderR-pDifference)))*-1))+  
351     ((dragWheelCirc/360)*(180/M_PI)*((cos(degHead)*((encoderVB-backWheelDifference) - lastEncoderB)*-1)));  
352 }  
353 else {  
354     positionX += ((dragWheelCirc/360)*(180/M_PI)*((cos(radians(degHead))*(((encoderVL+encoderVR-difference)/2) - (lastEncoderL+lastEncoderR-pDifference)))))+  
355     ((dragWheelCirc/360)*(180/M_PI)*((sin(radians(degHead))*((encoderVB-backWheelDifference) - lastEncoderB))));  
356     positionY += ((dragWheelCirc/360)*(180/M_PI)*((sin(radians(degHead))*(((encoderVL+encoderVR-difference)/2) - (lastEncoderL+lastEncoderR-pDifference)))))+  
357     ((dragWheelCirc/360)*(180/M_PI)*((cos(radians(degHead))*((encoderVB-backWheelDifference) - lastEncoderB))));  
358 }
```

Incorporation of the backwheel difference variable in odometry code.

Pre-Coding: The Aimbot Part 9

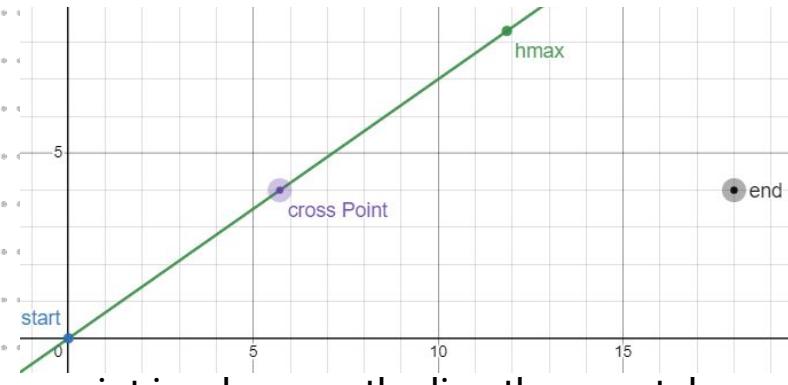
After many days of banging my head against the wall, I got it. The aimbot calculations are finally in my hands. Now before I get to that I have to first discuss my failed attempts at the aimbot calculations. I tried a lot of paper calculations which all of them failed. Each attempt I will give their own section.

$\Delta Df = Vo*t = g/2*t^2 + Vo*t$ Method

I discussed this at the end of the last page of the Aimbot section. This method failed really quickly. Now reading over this you can easily see why. In the calculations you can see that I used $Vo*t$ twice. Which would prove without background knowledge that $g/2*t^2 = 0$ meaning $Vo*t = Df$. Which is what I was looking for. I didn't do any paperwork with this. In future methods when I do have paperwork I will include the paper on a separate page for ease of understanding.

Slope Method

The slope method as I call it was my closest to actual calculations that I got before help. Slope method is a bit hard to explain in words So I will use a graph to help illustrate. The blue point will be our start, and the black point will be our end. In these calculations I didn't use any vex given measurements just random points I selected to end with. The only thing that is in actual use is the angle of 35 Degrees. Using $\tan(35) = y/x$ gives me the slope of the line. I tried used this to scale how fast the flywheel needs to go. The cross point is where on the line the y matches the y of the goal. Hmax was gotten by using the midpoint formula. Then get the y value from the $\tan(35) = y/x$ to get the height.



My thinking was to get time from how long it took for a disc to fall from hmax to the end point. Then to go from there to get the overall velocity. I tried this twice and both times failed. I first tried on desmos and I don't have the original calculations. My second attempt I partially did on paper so I will post that on page 164.

$$\theta = \dots$$

at cross point

$$V_0 = \dots, X_f = \dots, X_0 = \dots$$

$$D = \dots$$

$$t = t_1, t_2, \dots, t_n = \dots$$

$$x_m = (c_0 + x_s)/2$$

$$\Delta x_1 = \dots$$

$$c_0 = y_f/\theta$$

$$\Delta y_1 = \dots$$

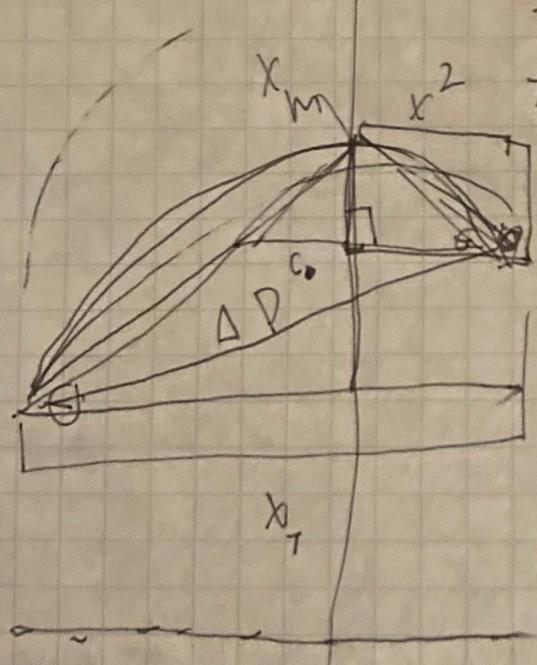
$$\Delta D = \dots$$

$$\Delta x_2 = \sin(\theta) \cdot d_2$$

$$\Delta d_2 = \sqrt{(x_m - x_s)^2 + (y_2 - y_0)^2}$$

$$\Delta y_2 = \frac{645(\theta) \cdot d_2}{\pi m}$$

$$\Delta j_1 = \dots$$



$$t_1 = \dots$$

$$t_2 = \sqrt{\frac{2g(c_0 - h_m)}{g}}$$

$$y_f = h_m$$

$$t_1 \quad t_2 = \sqrt{\frac{2g(c_0 - h_m)}{g}}$$

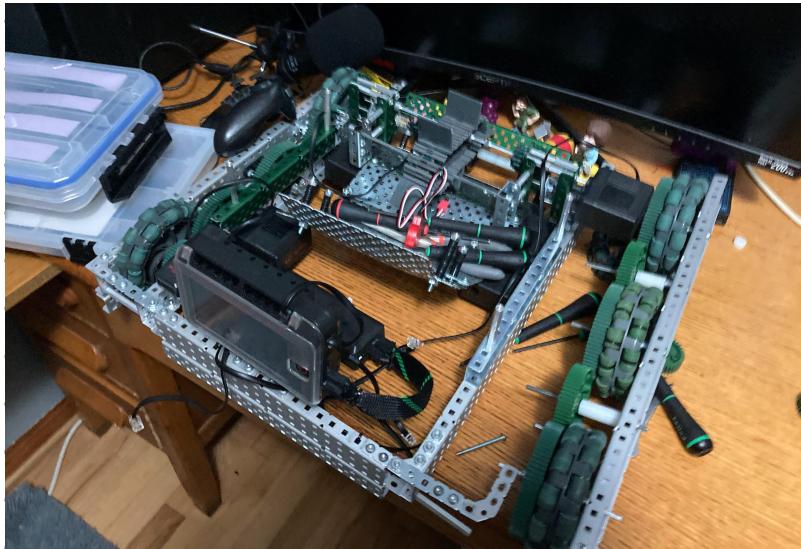
$$V_f^2 = V_0^2 + 2ax + \Delta x$$

$$0 = V_0^2 + 2ax + \Delta x + h_m$$

Design Process: The Chassis

Version 3

The Chassis couldn't support the robot. To be more specific the gearing couldn't support the robot which now requires a chassis redesign. The main objective of this redesign is to remove the speed gearing and replace it with a standard gear train. A secondary objective we planned to do is to add a middle wheel which will function to prevent drift. As well as to allow us to potentially go over the low goal zone.



On the left is the chassis as of Meeting #26. On the right part of the chassis is the new gearing setup. We are switching the 84:36 with a 1:1 setup with 60 tooth gears as idlers. The reason behind 60 tooth idlers instead of 36 tooth idlers is most the distance on the outside was mainly covered by the wheel itself not that actual gearing. On the left side of the chassis you can still kind of see what I mean.

Below is an image of the new gearing system. Something new you may observe is that there's now a middle wheel. Due to how the intake goes into the robot the middle and the back wheel from now on will be motorized. The front wheel (on the left in this image) is really close to the intake so it would be impractical to set it there.

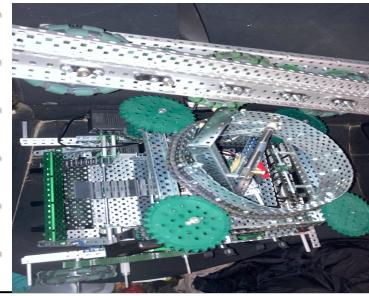
I would like to address something special about the middle wheel. Back when we had our first scrimmage and first tested the chassis. Joseph suggested to adding a Middle wheel to prevent drift. At the time with our speed gearing would prevent due to sizing. The middle wheel is special to the other wheels because it is a locked omni. As described before locked omnis have more friction then standard friction wheels but require more effort to create spending an entire practice to make two



Meeting #29

Robot beginning of
Practice

Pre-Meeting Plan

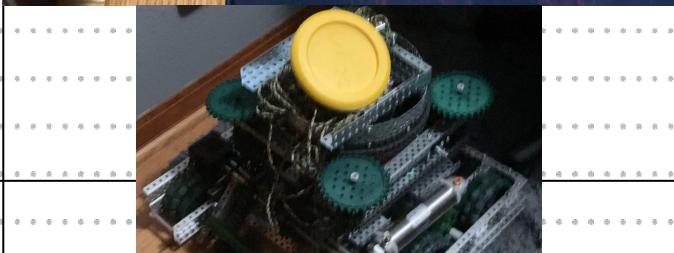
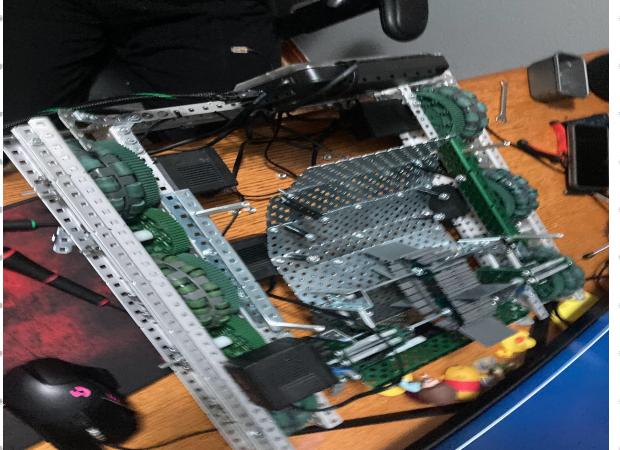


Team Members	Tasks and Obstacles	Design Process step
Aidan	Create the flywheel, then add the flywheel to the turret.	Build a Prototype
Tyler	Fix the chassis, then create roller mechanism	Build a Prototype

Meeting #29 Continued

Post Meeting Info

Meeting Info
Location: Tyler's house
Attendees: Aidan and Tyler
Duration: 3:00 - 9:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Create the framework for the Indexer, and Flywheel. Also created flywheel mount v2	I didn't have enough materials for the flywheel	  
Tyler	Finished Chassis version 3	No roller mechanism made. This needs to be made prior to comp	

Meeting #30

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Create flywheel Version 3. Then if there is any remaining time mount the odometry wheels	Build a Prototype

Meeting #30 Continued

Post Meeting Info

Meeting Info
Location: iTech Prep
Attendees: Aidan
Duration: 10:55 - 2:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Began the gearbox for the flywheel.	No solid position to put the flywheel on the turret.	

Pre-Coding: The Aimbot Part 11

The Aimbot is almost complete now. I got the new calculations with the help of 3249U's coder Jonah. The new calculations have seem to work when comparing them to my old final distance without time formula. The image below created by Jonah is the formula to calculate the velocity required to shoot a disc into the goal from a stationary point. At the bottom of the image is the full formula. If its hard to see it says

" $V_0 = \sqrt{g^*d^2/(2*\cos^2(\varphi))} \cdot (h_g - h_o - d \tan(\varphi))$ ". φ or Phi is the launch angle. H_g is the goal height, and H_o is Launch height. This should all be enough for me code the aimbot so lets start using it.

First we need to seperate the velocity into 3

A handwritten derivation of the Aimbot velocity formula. It starts with the formula $T = \frac{d}{V_0 \cos \varphi}$. Below it, the total distance d is shown as the hypotenuse of a right triangle with vertical leg $V_0 \sin \varphi$ and horizontal leg $t \tan \varphi$. The total time T is then expressed as $T = \frac{d}{V_0 \cos \varphi} = \frac{d}{\sqrt{V_0^2 \cos^2 \varphi}} = \frac{d}{V_0 \cos \varphi} = \frac{\sqrt{V_0^2 \cos^2 \varphi}}{V_0 \cos \varphi} = \frac{\sqrt{V_0^2}}{\cos \varphi} = \sqrt{V_0^2} \cdot \frac{1}{\cos \varphi} = \sqrt{V_0^2} \cdot \sec \varphi$. The final formula is $V_0 = \sqrt{(z \cos \varphi)(h_g - h_o - d \tan \varphi)}$.

Velocities. V_1 will be our flywheel speed. V_2 will be our pneumatic speed, and V_3 will be our relative speed, finally V_0 will be our total velocity. The formula to get V_0 is $V_1 + V_2 + V_3$.

We have V_2 , and now we V_0 . Lastly we now we need to get V_3 . To get relative velocity compared to the goal we need velocity across X and Y. In Odometry part 12 I added velocity to the Odometry code so we have X and Y velocity. Reversing the Odometry code and replacing robot heading with angle to the goal in the formula gives the relative velocity. Finally we will do $V_0 - V_3 - V_2$ to get V_1 . Code for all of these calculations below.

```
449 relativeVelocity = sqrt(pow(velocityXsec,2)+pow(velocityYsec,2)+2*velocityX*velocityYsec*cos(90-findAngle(target)));
450
451 totalDistance = sqrt(pow(findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2)
452 +pow(tan(radians(flyWheelAngle))*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2));
453
454 ejectVelocity = sqrt((gravity*totalDistance/(2*cos(radians(flyWheelAngle))))*(zCoordinates[1]-turretOffsetZ-totalDistance*tan(radians(flyWheelAngle)))))
455 -pneumaticSpeed-relativeVelocity; // inches per second
```

Design Process: The Flywheel

Version 3

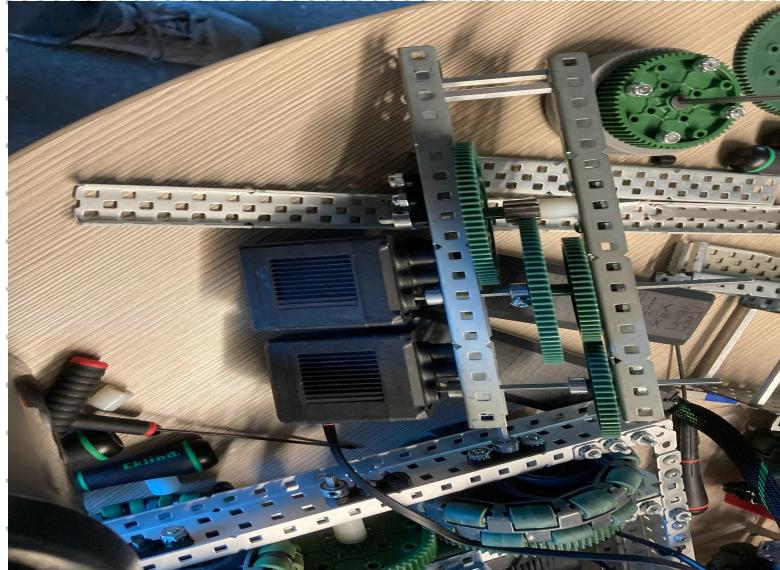
The tournaments coming close so it's time to create the flywheel again. Using the experience from flywheel Version 1, and the build practices from Version 2. The flywheel Version 3 will be the final version before the tournament. Version 3 is Designed to be very fast. With 4900rpm to launch the discs at approximately 30mph's, and two Torque motors for a faster recovery. Along with these speed changes the Flywheel will be heavier. So we the flywheel can recover faster and so the motors don't have to constantly be running to keep speed with its momentum.

On the left is the new gearbox. Which will be attached along the flywheel

Mount. This gearing setup is still is not complete. The current goal for me next practice is to finish the gearbox and attach it to the mount. Its is far compact than the Flywheel version 1 which allows it for easier mounting. The Flywheel will be connected by a sprocket chain system which I hope to work.

The Gearbox consists of two 25x2 c-channels. Connected by two, two inch standoffs on the end of it. The motors are going to be at the top for easily accessibility. Each shaft connection has other the motor connection has a nylocked bearing. In the system I try to keep everything as close to a c-channel with possible and not kept in place by shaft collars so it cannot be unstable and fly off.

I plan to have this design finished by next practice. If there is any time remaining I may mount the flywheel to its mount. I really hope this design works because I we don't have time to screw up anymore. We only have 4 days left so everything we need to do has to be ready as fast as possible.



Design Process - Pneumatics and Turret wiring

In order for our indexer to work seamlessly, efficiently, and not take a motor from another part of the robot, we will be using pneumatic pistons to push our gear in an up and down motion, moving 1 disc up at a time. We will have to make sure we have enough air to last throughout the match, so we will have to use 2 cylinders to hold enough air to work. We may need 20 to 40 movements per match, and the tanks may not hold enough air to last the round we will assist the contraption with rubber bands to take some strain off the piston.

Turret wiring and Tubing

Our turret will require 2 motors and a pneumatic cylinder to be rotating throughout the match. We will have to have wires running to these systems, and the obvious problem of restricting or twisting is present. We are fortunate enough to have mesh wire casing, and we can use this to run wires to our motors and pistons in one casing, and restrict it with zip ties so we do not expand past our 180 degree limitation.

Meeting #31

Robot beginning of
Practice

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Finish the Flywheel Version 3. if there is time left the second job is to create the flywheel version two and assemble the Flywheel, the Flywheel Mount, and the turret wheel.	Build a Prototype
Ethan	Create the indexer. If there is remaining time mount the turret base to the chassis as well as the odometry wheel. If time allows, begin pneumatic assembly	Build a Prototype
Josh	Create the expansion mechanism	Build a Prototype
Tyler	Create the Roller Mechanism	Build a Prototype

Meeting #31 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Flywheel Gearbox was complete.	The flywheel gearbox is too powerful for the motors to handle so a redesign is needed	
Ethan	Remounted turret and began work on the indexer	Indexer V1 (SESGM) would not be able to fit within the robot so a new design is needed.	
Josh	Cut string into 10 different pieces for both red and blue teams	Josh took the string home after he and Tyler cut them so he can fray the ends.	
Tyler	Cut string into 5 different pieces for both red and blue teams	Needs to work on creating a roller design.	

Game Strategy: Strategy Page 3

Strategy 7, Date Created: 11/15/22 Name: Tyler Fields. Points: 70 AWP: No. Program: 2, Potentiometer: 1 Page: 176. Strategy Kim is a Autonomous program. In this the robot immediately shoots one of its match loads and takes another preload to shoot. This strategy can be used as a good way to get a bunch of early points with disks. Created By Tyler Fields.

NOT PROGRAMMED YET

Strategy 8, Date Created: 11/15/22 Name: Tyler. Points: 915 Program: Physical, Page: 177. Strategy bartholomew is a skills strategy. The goal is to circle the field to get all the rollers, as many disks as possible, and end up at a good spot for our string expansion. Created By Tyler Fields.

N/A

Strategy 9, Date Created: 11/16/22 Name: Yoink. Points: 70 AWP: no. Program: 3, Potentiometer: 1 Page: 187. Strategy yoink is an Autonomous program. The goal of this strategy is to grab as many disks as possible without worrying about other things. This may be useful to prevent the other team from being able to score through high goal. Created By Tyler Fields.

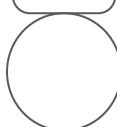
NOT PROGRAMMED YET

Key:

Starting Position

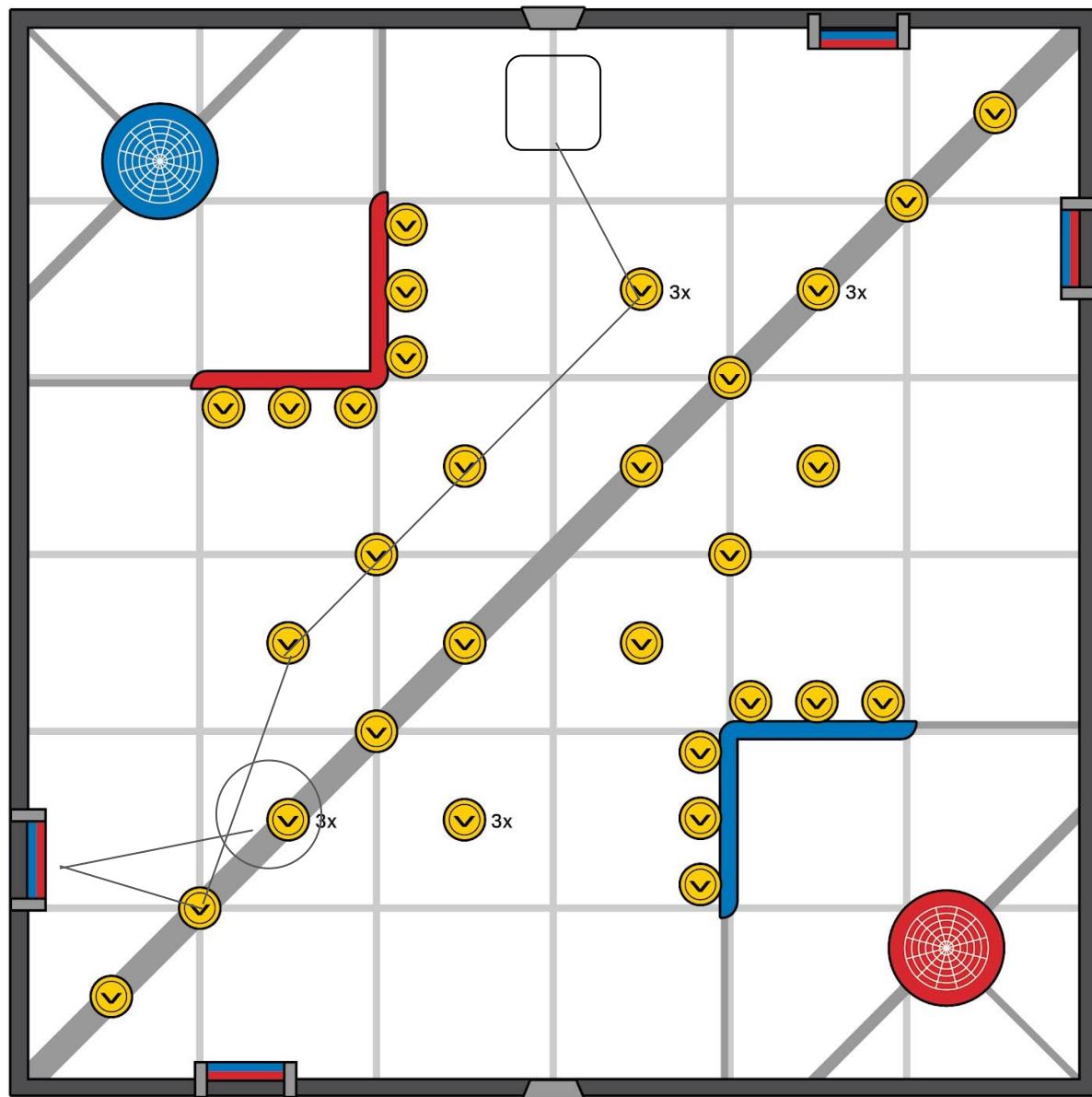


End Position



Strategy: 07

Name: kim

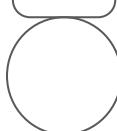


Key:

Starting Position

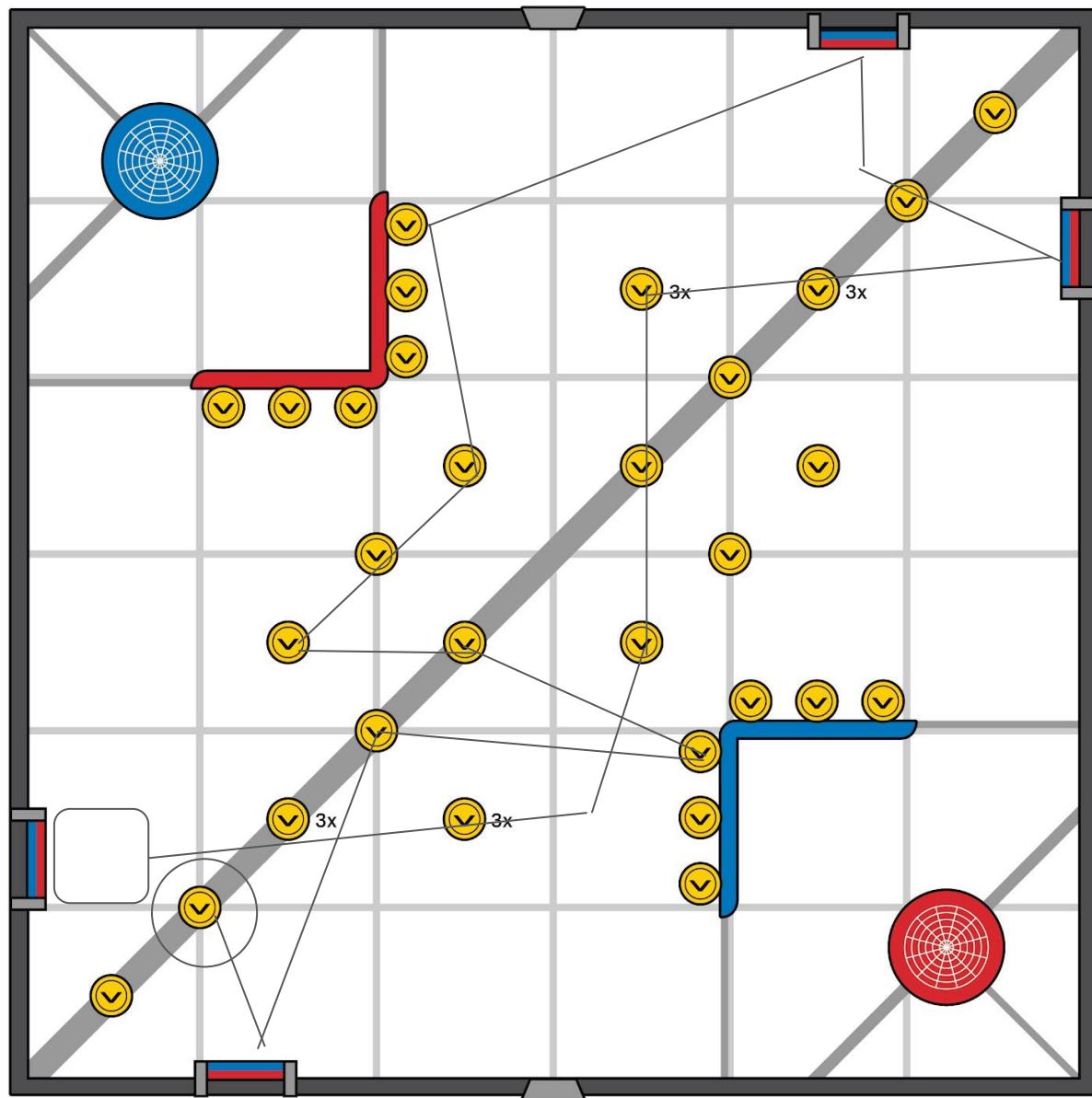


End Position



Strategy: 08

Name: bartholomew



Pre-Coding: The Aimbot Part 12

Having the required linear speeds is useful but not helpful for controlling the flywheel. The flywheel speeds are controlled by changing the RPM so we need a way to transfer Linear speed. Luckily VEX actually gave a really useful solution. Vex gave a set of formulas I could use. I put them on the left. Both formulas are energy formulas. Energy = Energy so I can do E_{linear} = E_{rotational}. I can reformat this as $\frac{1}{2} \cdot \text{mass} \cdot \text{Velocity}^2 = \frac{1}{2} \cdot \text{rotational inertia} \cdot \text{angular velocity}$. In the code I solve for RPM or angular velocity so that will be our desired outcome. If I reformat the equation to get angular velocity I get angular velocity= $\sqrt{\text{mass} \cdot \text{Velocity}^2 / \text{Rotational Inertia}}$.

$$E_{\text{Linear}} = \frac{1}{2} mv^2$$

$$E_{\text{Rotational}} = \frac{1}{2} Iw^2$$

The output in angular velocity comes out in rad/s. In the calculations I add a conversion from rad/s to rpm. Then take this final rpm and put it into both flywheel motors. Code for this below.

```
flywheelVelocity = (((sqrt(flyWheelMass*pow(ejectVelocity,2)/(flyWheelMass*flyWheelRadius)))/49)/(2*M_PI))*60;  
flyWheel.setVelocity(flywheelVelocity,rpm);  
flyWheel2.setVelocity(flywheelVelocity,rpm);
```

Turret PID

I have modified things in turret PID. I Changed it from firing once at a time with the function to now being its own task. A Task is a process in the program that runs asynchronously in a thread. I changed it to a task instead of a function because a function would force the rest of the aimbot thread to wait for the turret PID to fully complete. It also allows me to add a changing error variable that allows the PID to run with a moving robot. The changing error variable is the angle of the turret to the high goal. New PID code on page 179. The new code for the turret below.

451

task myTask = task(turretPID);

```
470  
471     if((aimBot=true && (totalTurretRotation >= 360*4))) {  
472         // Turret movement very simple :D  
473         if ((findAngle(target) - turretThreshold) <= turretHeading <= (findAngle(target) + turretThreshold)) {  
474             lockOn = true;  
475         } // decides if the turret is within acceptable limits  
476         else if((activePID = false)&&((findAngle(target) - turretThreshold) <=! turretHeading <=! (findAngle(target) + turretThreshold))) {  
477             turretError = findAngle(target);  
478         }  
479     }
```

Pre-Coding: The Aimbot Part 13

New Turret PID Code

```
int turretPID()
{
    double factorP = 0.625;
    double factorI = 0.005;
    double factorD = 0.25;
    double error = 0;
    double integral = 0;
    double derivative = 0;
    double lastError = 1;
    double fixing = 0;
    double errorAverage = error;
    double lastErrorAverage = 0;
    double loopCount = 0;
    activePID = true;
    while (abs(error) >= 0) {
        error = (turretHeading - turretError);
        integral = integral + error;
        derivative = error - lastError;
        fixing = (error*factorP)+(integral*factorI)+(derivative*factorD);
        if (turretError > 180) {
            turretMotorX.setVelocity(fixing,rpm);
        }
        else {
            turretMotorX.setVelocity(-fixing,rpm);
        }
        turretMotorX.spin(forward);
        lastError = error;
        wait(1.5,msec);
        error = turretHeading - turretError;
        errorAverage = (errorAverage+error+lastError)/3;
        loopCount += 1;
        if (loopCount > 75) { // This is so previous larger errors don't break the infinite loop fix
            lastErrorAverage = errorAverage;
            errorAverage = 0;
        }
    }
    activePID = false;
    turretMotorX.stop();
    return(1);
}
```

Meeting #32

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Fix the issue in the flywheel where it cannot spin smoothly	Build a Prototype

Meeting #32 Continued

Post Meeting Info

Meeting Info
Location: iTech Prep
Attendees: Aidan
Duration: 10:55 - 2:00

Team Members	What got done	Unresolved Problems	Images
Aidan	I fixed the issue with the flywheel. The bearing flats weren't properly snapped on to the c-channel so it make it turn inconsistently.	The flywheel itself needs to be installed	

Meeting #33

Robot beginning of
Practice

Pre-Meeting Plan

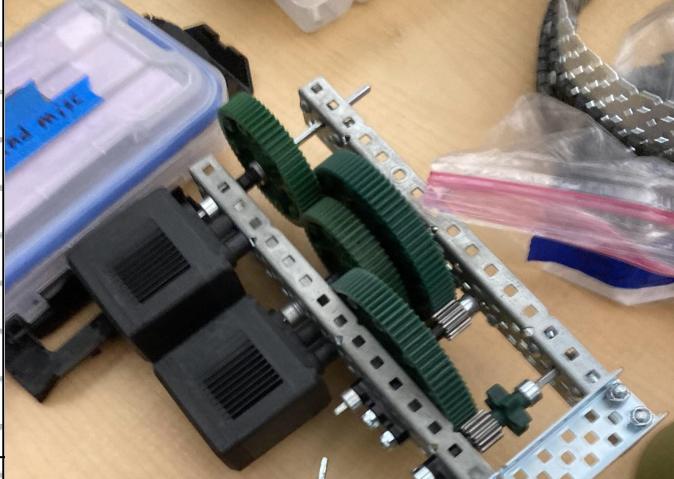
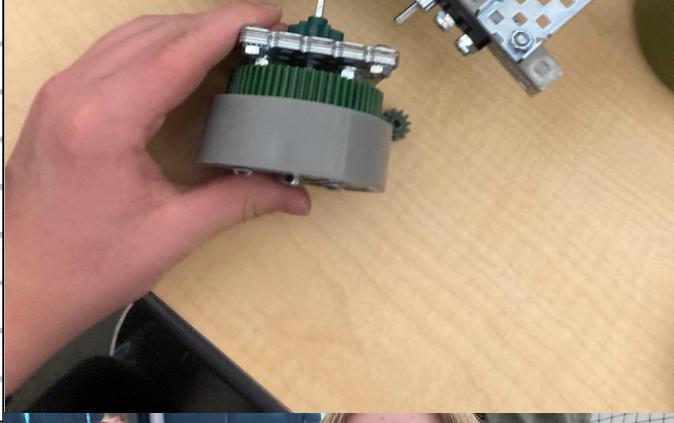
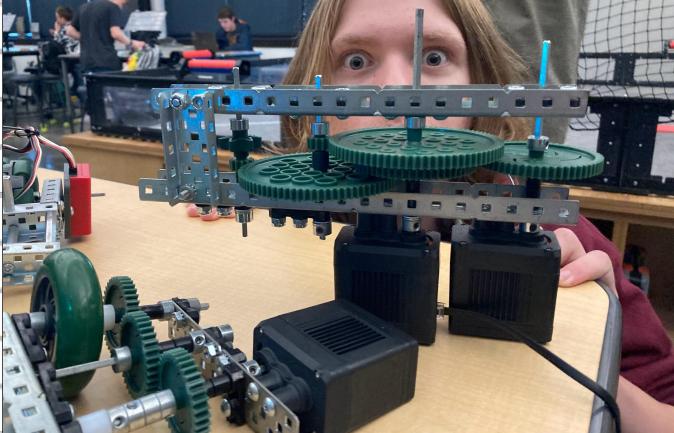


Team Members	Tasks and Obstacles	Design Process step
Aidan	Refit the flywheel with high strength gears creating Flywheel Version 4. If time allows I will try and setup the Flywheel	Build a Prototype
Ethan	Create the Indexer and if time allows mount the odometry wheels.	Build a Prototype
Tyler	Create the second iteration of the roller mech by using the intake motor to spin the rollers	Build a Prototype

Meeting #33 Continued

Post Meeting Info

Meeting Info
Location School: iTech Prep
Attendees: Aidan, Ethan and Tyler
Duration: 4:05 - 6:00

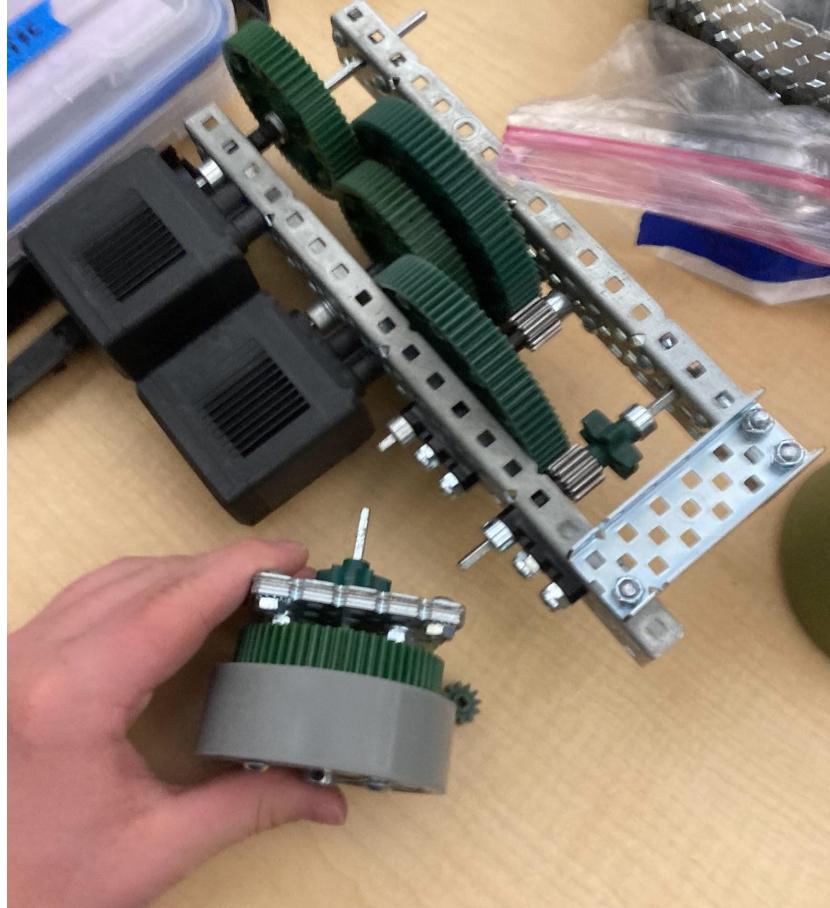
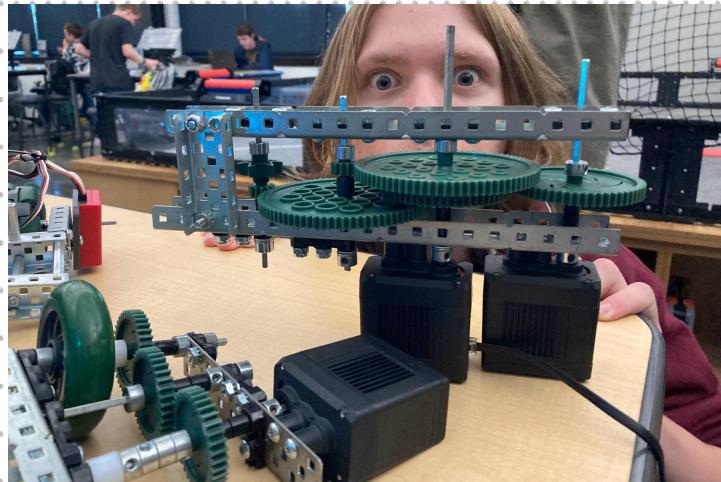
Team Members	What got done	Unresolved Problems	Images
Aidan	Flywheel Version 4 is complete and spins smoother than the previous version 3 with the same speed. Also I added 5x5 plates as weights bringing the flywheel up to ~0.5lbs	The flywheel needs to be connected so we can begin firing	
Ethan	Worked on the indexer platform, The intake motor was stripped so we had to replace it.	The indexer needs to be built to put discs into the turret	
Tyler	Began work on the roller mech putting the framework in place to connect a wheel to turn the rollers	The roller mechanism and the string mechanism needs to be on the robot so we can at least score 52 points in skills	

Design Process: The Flywheel

Version 4

The flywheel has some problems. The LS gears kept chipping against each other and didn't spin smoothly. If this version was in an actual match we would've broke our flywheel in the autonomous period. So we replaced it with HS gears. These have proven in tests of verison 4 far more effective and clean. Even with single motor testing it has proved to be smoother than Version 3.

On the left is a picture of the flywheel V3 with Tyler in the background. Ignoring Tyler we can see that the last system was very thin. In tests it was prone to gear slipping and was very loud. It did serve as a good basis for V4 shown below. The gearing is thicker which reduces slippage and also improves overall stability. In the middle the spacing is very thin. I had to use



Metal washers and thin spacers so everything doesn't scrape against the metal sanding down both the gears and the metal.

The second part of verison 4 is the flywheel itself. I used 6 5x5 plates to add around 0.6lbs which is a significant weight increase. On thursday I will try to get a more specific value on Thursday. I will be using my dads food scale which measures in ounces. The next step to go along is to assemble the flywheel mount, Flywheel gearing and flywheel, as well as the turret wheel (a lot of wheels). This will hopefully be accomplished on either Meeting #34 or #35.

Pre-Coding: The Aimbot Part 14

Recap

I am done with almost all the Aimbot code prior to having a programmable robot. The robot will become programmable if Ethan can actually mount the Odometry wheels. There are factors which I want to cover after the tournament. Such as redoing the energy calculations to include the discs energy.

In this section I covered what an Aimbot is and what it covers. The code for the Aimbot, What the Auto firing system is and the code for the Autofiring system, and finally my failed and succeeded calculations. Going forward after the Tournament I will have to look over all the code from Odometry to Aimbot. I also need to talk to Joseph and Jonah about potential factors I might need to redress. I am a sophomore who just started physics so I don't know everything to when it comes to these types of problems. On page ## and ## is the entire Aimbot Code spanning Threads 3-4 as well as The autofiring Systems. There are some minor updates things that I have changed such as some organizational changes. As stated before code will change in the Coding Section.

Overall I am kind of happy with how this section went. This section has caused me the most grief and most issues. I know for sure my code is wrong in some way and It will fail. But i hope it will be at least somewhat accurate. Accurate enough hopefully for 40 to 50% accuracy. This is definitely low but hopefully enough to score at least 14 discs in a match. In the end it all comes down to the coding section and testing. I hope it works but I don't have much confidence in the calculations and code. I hope for the best at the tournament.

Design Process: Roller Mechanism

Version 2

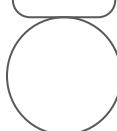
For version 2 of the roller I made a roller that moves with the intake system. I first measured the height of the roller and then measured where id need to place it on the robot. I then used some 3" standoffs on top of the logo on the front of the robot. I used the 4" green wheels because it was the best for getting good traction on the roller. I used gears and a chain attached to one of the shafts on the intake. I was originally going to use a passive rolled design, that is a roller that does not function using a motor. However we scraped this idea for now because our first tournament was coming up soon and I'm not the best builder and making things like a one way gear would take a lot of extra time to figure out for me. However we did not completely scrap the idea, in the future we could possibly change to a passive design if we deem fit.

Key:

Starting Position

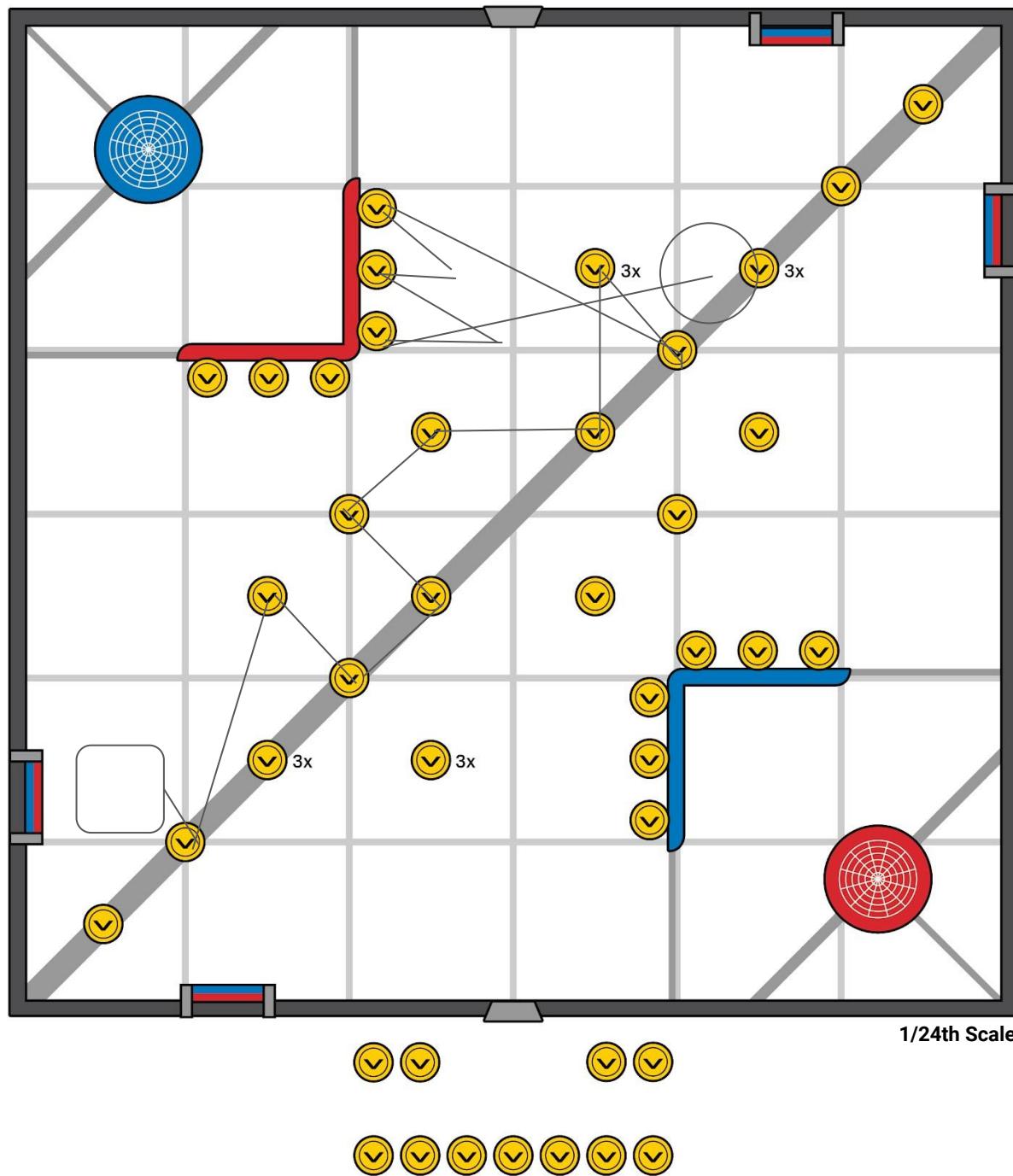


End Position



Strategy: 09

Name: yoink



Pre-Coding: The Aimbot Part 15

```
325 void expansionControl() {
326     expansion = true; // no need to retract this is a one time spring mechanism
327 }
328 void triggerControl() {
329     trigger = true;
330 }
331 void triggerControl2() {
332     if ((lockOn = true)) {
333         trigger2 = true;
334         discLoad -= 1;
335         wait(25,msec);
336         trigger2 = false;
337     }
338 }
339
340 void intakeRead(){
341     intakeDisc = true;
342     discLoad += 1;
343     waitUntil(!intakeSensor.pressing());
344     intakeDisc = false;
345 }
346 void turretRead(){
347     turretDisc = true;
348     waitUntil(!turretSensor.pressing());
349     turretDisc = false;
350     if ((intakeDisc = true) && (flyWheelDisc = false)) {
351         triggerControl();
352     }
353 }
354 void flyWheelRead(){
355     flyWheelDisc = true;
356     if ((lockOn = true)&&(autoFire = true)) {
357         triggerControl2();
358     }
359     waitUntil(!flyWheelSensor.pressing());
360     flyWheelDisc = false;
361 }
```

Pre-Coding: The Aimbot Part 16

```
435 int thread3() { // auto aim thread
436     float flyWheelMass = 0.6; // in lb
437     float flyWheelRadius = 1.5; // in inches
438     //float flyWheelContactAngle = 70; // needs calibration
439     float flyWheelAngle = 35;
440     double totalDistance;
441     double totalTurretRotation = turretHeading; // to make sure that we don't go overboard and twist/rip wires
442     //float maxTotalTurretRotation = 720;
443     float pneumaticSpeed = 8;// in inches per second idk how this is that fast
444     double relativeVelocity;
445     double ejectVelocity;
446     double flywheelVelocity;
447     double velocityXsec = velocityX*1000;
448     double velocityYsec = velocityY*1000;
449     if ((redTeam = true)) {
450         target = 7;
451     }
452     else {
453         target = 8;
454     }
455     task myTask = task(turretPID);
456     while(true) {
457         turretHeading += degHead - lastdegHead;
458         totalTurretRotation += degHead - lastdegHead;
459         if (turretHeading >= 360) {
460             turretHeading = turretHeading - 360;
461         }
462         else if (turretHeading <= 0){
463             turretHeading = turretHeading + 360;
464         }
465     }
466     velocityXsec = velocityX*1000; // inches per ms to inches per second
467     velocityYsec = velocityY*1000;
468     relativeVelocity = sqrt(pow(velocityXsec,2)+pow(velocityYsec,2)+2*velocityX*velocityYsec*cos(90-findAngle(target)));
469
470     totalDistance = sqrt(pow(findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2)
471     +pow(tan(radians(flyWheelAngle))*findDistance(positionX,positionY,coordinateLocations[0][target],coordinateLocations[1][target]),2));
472     ejectVelocity = sqrt((gravity*totalDistance/(2*cos(radians(flyWheelAngle))))*(zCoordinates[1]-turretOffsetZ-totalDistance*tan(radians(flyWheelAngle)))));
473     -pneumaticSpeed-relativeVelocity; // inches per second
474
475     flywheelVelocity = (((sqrt(flyWheelMass*pow(ejectVelocity,2)/(flyWheelMass*flyWheelRadius)))/49)/(2*M_PI))*60; // somewhat accurate I hope
476     flywheel.setVelocity(flywheelVelocity,rpm);
477     flywheel2.setVelocity(flywheelVelocity,rpm);
478     if(aimbot=true && (totalTurretRotation >= 360*4)) {
479         // Turret movement very simple :D
480         if ((findAngle(target) - turretThreshold) <= turretHeading <= (findAngle(target) + turretThreshold)&&(flywheelVelocity-1 <=flywheel.velocity(rpm) <= flywheelVelocity+1)) {
481             lockOn = true;
482         } // decides if the turret is within acceptable limits
483         else if((activePID = false)&&(findAngle(target) - turretThreshold) <=! turretHeading <!= (findAngle(target) + turretThreshold)) { // prevents infinite PID's
484             turretError = findAngle(target);
485         }
486     }
487     else {
488         turretHeading = totalTurretRotation;
489         turretError = degHead;
490     }
491     this_thread::sleep_for(1);
492 }
493
494 int thread4() { // auto Fire Thread
495     intake.setVelocity(100,percent);
496     intake.spin(forward);
497     while (true) {
498         intakeSensor.pressed(intakeRead);
499         turretSensor.pressed(turretRead);
500         flyWheelSensor.pressed(flyWheelRead);
501         if (discLoad != 3) {
502             intake.setVelocity(100,percent);
503         }
504         else {
505             intake.setVelocity(0,percent);
506         }
507         //if ((discLoad = !3)){
508         /* */
509         else {
510             intake.stop();
511         } */
512     }
513     this_thread::sleep_for(25);
514 }
515 }
```

Pre-Coding

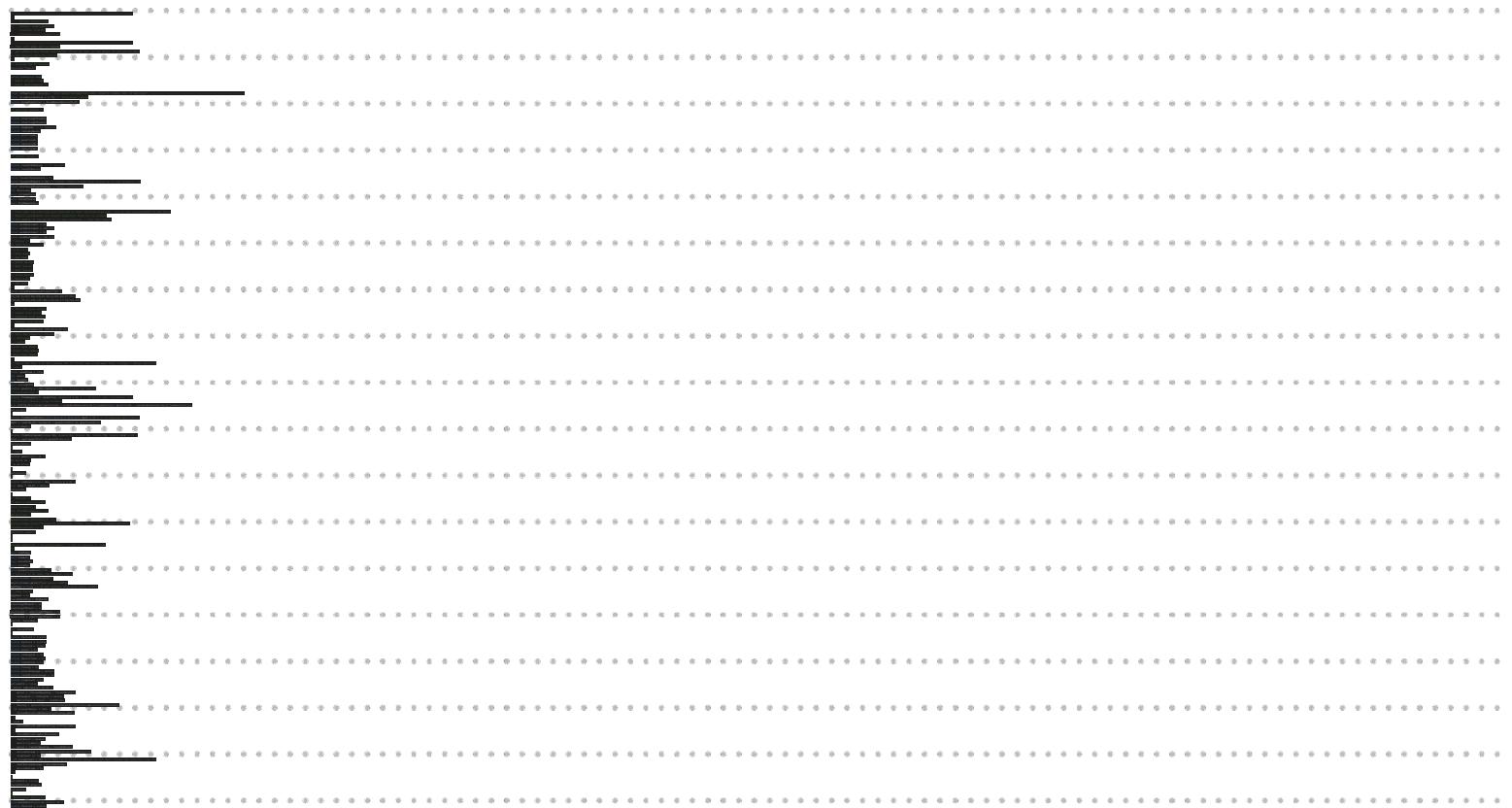
End of Pre-Coding

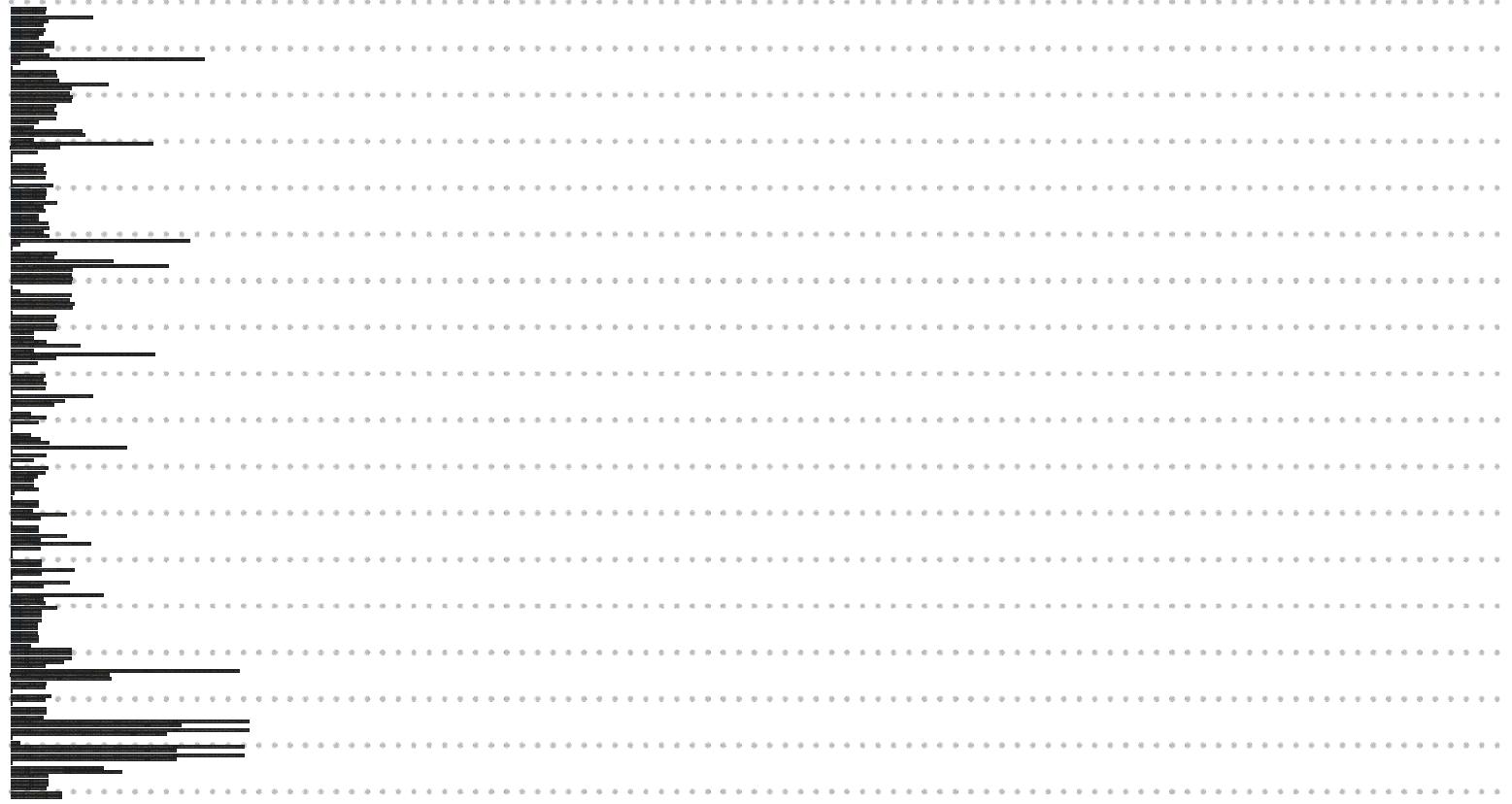
After 40 pages this is the end of the Pre-Coding section. Tomorrow the Odometry wheels will be mounted, turret will be completed (hopefully), and everything needed to program the robot installed. The robot together will be at my control and I need to make sure it all functions properly.

I didn't cover everything I said I would. I didn't choose to cover things such as Driver control because that's something that deserves to be in Coding. I will need to cover MoveToPoint and Aimbot as these are two subjects I either need to improve or need or calibrate.

On pages 191 - 193 will be a copy and paste of the entire code at 1 point font (Yes this is a bad way of page filler). This acts as a backup incase I lose the code and its backups, or if I am not present at a tournament or meeting.

The main objective of Pre-Coding is to create a base and it did its job well. The total section dates from September 2nd to November 16th. It covered the Odometry, PID's and the Aimbot as well as early strategy code like the Potentiometer code in the strategy pages. As I continue gaining experience and facing problems I the code will further innovate and Improve. I really hope that in the future I will look back and laugh at my stupid mistakes re-reading this. Or not and I was right from the start and made a good base for the future. Only time will tell.





Project Pre-Coding

Name Aidan McCallum

Date 11/16/22

Page 192

[REDACTED]

[REDACTED]

Meeting #34

Robot beginning of
Practice

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Work on the flywheel and try to have it mounted	Build a Prototype
Josh	Create a design for the expansion mechanism	Create a Design

Meeting #34 Continued

Post Meeting Info

Meeting Info
Location: iTech Prep
Attendees: Aidan and
Josh
Duration: 11:00 - 12:05

Team Members	What got done	Unresolved Problems	Images
Aidan	Fixed flywheel weight to the flywheel		
Josh	Decided a string design based on a strategy		

Meeting #35

Robot beginning of
Practice

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Assemble and mount flywheel. If time allows test accuracy of odometry wheels.	
Ethan	Mount odometry wheels and finish indexer	
Josh	Create Expansion mechanism with strings	
Tyler	Finish the roller mechanism	

Meeting #35 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Began combining the flywheel mount, and flywheel gearbox		
Ethan	Finishing up the intake and shaved excess shafts off the chassis so we fit size constraints		
Josh	Decorated robot and created a basic string mech design		
Tyler	Assisted team with tasks		

Meeting #36

Robot beginning of
Practice

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
Aidan	Assemble the flywheel.	Create a Prototype
Ethan	Finish intake+indexer	Create a Prototype
Josh	Create expansion mech	Create a Prototype
Tyler	Mount the rollers	Create a Prototype

Meeting #36 Continued

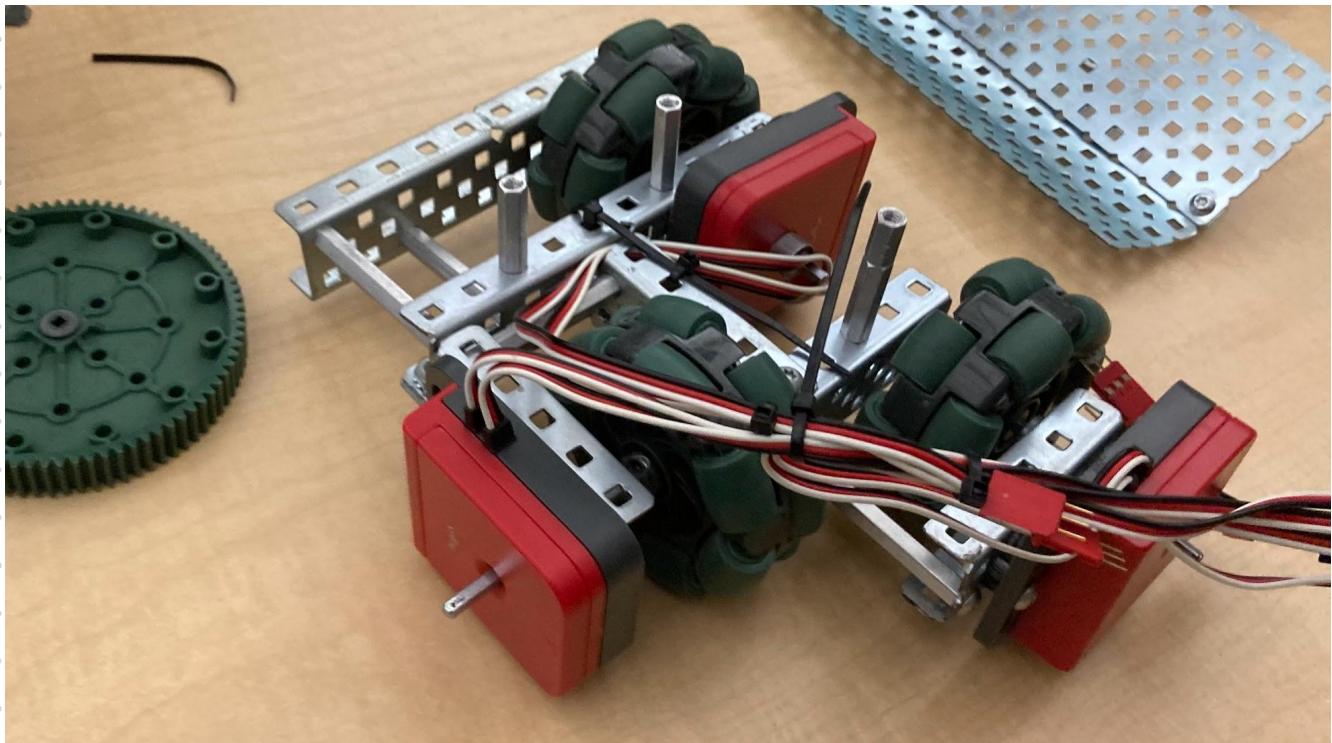
Post Meeting Info

Meeting Info
Location School: Tylers House
Attendees: All Members
Duration: 4:05 - 6:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Tested and almost complete Flywheel V5	We were missing stands offs and plates required to finish	
Ethan	Created the last part of the intake prior to the indexer	Our single battery ran out of juice	
Josh	Researched other expansion mechanisms and found a design better than the previous plan.	N/A	
Tyler	Fray the ends of the strings.	We left our roller mech in the cubby on accident.	

Design process: Odometry Wheel Suspension

The odometry wheel suspension system will be crucial for the auto aim system to work. We will be using L brackets and rubber bands to keep pressure on the ground, the odometry wheel brackets are quite large and spacious, and we will need to be using shafts to keep the bracket mount from shifting around in the robot. The sliding L bracket system should slide up and down, and be steady enough to not twist, as this would throw off the entire robot's aiming system.



Chassis mounting will be a quite difficult task for this system. We plan to use a C channel spanning horizontally throughout the inside of the robot, and mount the bracket to the chassis. If we have succeeded in this design, the bracket should droop a few inches when picked up, remain in contact with the ground at all time, and also not obstruct the height of the wheels

Coding: Introduction

The chassis is now almost fully drivable and now needs code for it. This section will be going through the entire session only ending when the notebook is finished. A majority of the code now will be testing, evaluating, changing the code, testing again, and reevaluating. changing the code again until the code is perfected. Which will take a lot of time.

Coding will span many different parts of the robot. Majority of the parts are calculations. An example being the complicated Aimbot Code. It needs improvement and sophisticated calculations such as loss from heat and friction. As I improve my physics skills and calculations the robot's accuracy will improve as well.

Coding includes rigorous testing as well. From basic debugging to fine tuning. Testing will be a big portion of how my time is spent coding. Testing is a big portion of the design process as well. I wouldn't see the code ready until late season or potentially state. As with the robot everything can be improved.

Meeting #37

Robot beginning of
Practice



Pre-Meeting Plan

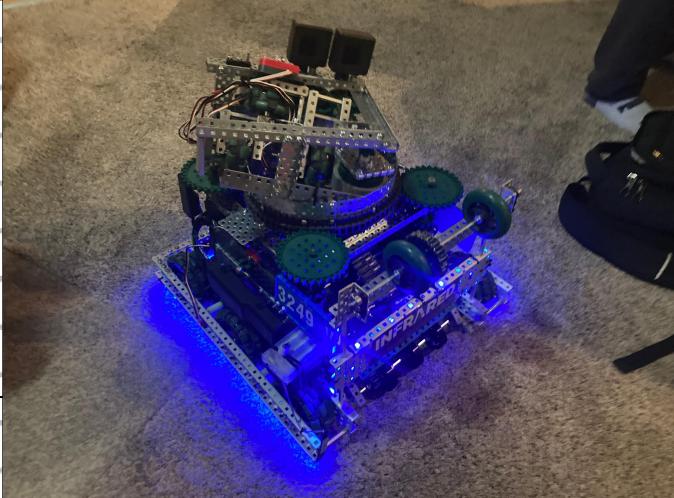
Team Members	Tasks and Obstacles	Design Process step
Aidan	Program and test PID's and Notebook if spare time.	Create a Prototype
Ethan	Mount the odometry wheels, Mount the Indexer, and assemble flywheel.	Create a Prototype
Tyler	Mount the rollers, Assist others in tasks	Create a Prototype

Meeting #37 Continued

Post Meeting Info

Meeting Info
Location School: Tylers House

Attendees: All Members
Duration: 5:30 - 10:00

Team Members	What got done	Unresolved Problems	Images
Aidan	Attempted to mount the flywheel and turret but failed	Turret and flywheel won't be ready for the turret	
Ethan	Mounted the turret and indexer onto the robot	He didn't mount the odometry wheels	
Tyler	Created a second poster design and added a roller mount		

Coding: Driver Code

Chassis Control Version 1

The core of the Robot's control is the driver code. It controls the whole bot for 1 minute and 45 seconds of a match. It allows the driver to control the robot using the controller. Its essential for competing in the Vex Robotics Competition. I have used the same Chassis driver code some change for the past 3 years and now 4 so it's near perfected.

The code was originally taught to me by one of the previous iTech coaches Jessie. He taught me this one on one since most teams at the time were using Tank drive which Ms.Hagin taught. The original code used in the old ROBOTC was a single joystick control which my team wanted. At the time it gave a slight edge over other robotics teams. Over the years I've modified it to suit my needs. From converting to VEXcode Pro and changing it from single joystick control to arcade control. The most recent iteration was switching it from linear increase to Exponential fraction increase (I'll explain later).

Arcade Control which is what we used last year Uses the Left joystick to go Forward and back. And the right Joystick to go Left and right. The separated joysticks prevent Human error from having the robot drift left or right. It also helps with doing slight movement adjustments. In situations like roller scoring or feeding the intake precise movements are critical. The code below is how the robot is controlled.

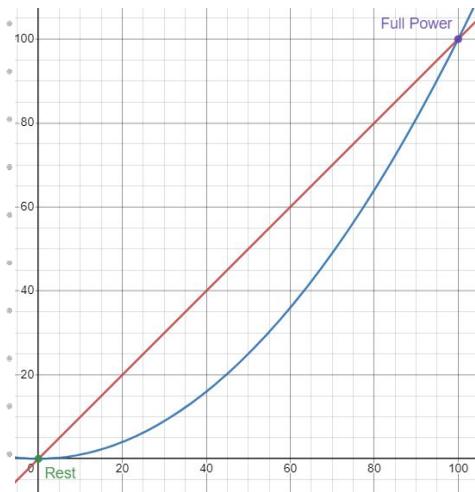


Basically the velocity is the left joystick added to the right joystick. In the left joystick it adds both values. In the right joystick it minus both values so the turning is always negative. The right side being negative and the Left side being positive allows the Robot to move left and right.

```
557 leftFrontMotor.spin(forward,(Controller1.Axis3.position() + Controller1.Axis1.position())^2/100,velocityUnits::pct);  
558 leftBackMotor.spin(forward,(Controller1.Axis3.position() + Controller1.Axis1.position())^2/100,velocityUnits::pct);  
559 rightFrontMotor.spin(forward,(Controller1.Axis3.position() - Controller1.Axis1.position())^2/100,velocityUnits::pct);  
560 rightBackMotor.spin(forward,(Controller1.Axis3.position() - Controller1.Axis1.position())^2/100,velocityUnits::pct);
```

Coding: Driver Code Part 2

The most recent change I made last year was changing the linear increase to a exponential fraction. This addition allowed the controller more precision while moving. Let me explain how it works. Instead of increasing the motor velocity based raw joystick movements I made it so it's exponential and divided by 100. Its nearly impossible to be at exactly 50% speed. With the code at 50% Joystick movement it is at 25% Chassis speed. On the left is a graph explaining how it works. Red is the joystick movement. Blue is the Chassis speed.



Controller Rumble

An observation I made last year during scrimmages is that the controller were sometimes caught off guard by when the game started. A simple fix which helped a lot was adding a short rumble at the start of the Driver period and sometimes at the 30 second mark. For context a big event happened at the 30 second mark last year which could score a penalty if we didn't watch out. Code for this below.

545

Controller1.rumble(rumbleShort);

Expansion Control

Expansion while not being ready yet can still be coded. It utilizes a callback command which calls back to a simple function. The callback is an event that checks if the controller button B is pressed. When it is pressed and it is in the last 10 seconds of the match the expansion it will activate the function. The function is a control function that activates both pistons and 0.5 seconds later they're deactivated. Code below.

```
if (timeRemaining <= 10){  
    Controller1.ButtonB.pressed(expansionControl); //pneumatic  
}  
// Chassis Control
```

```
void expansionControl() {  
    expansion = true; // no ne  
    expansion2 = true;  
    wait(500,msec);  
    expansion = false;  
    expansion2 = false;  
}
```

Pre-Tournament: State of the Bot

November 19th North Marion Tournament

Yea we don't know what will happen. I am uncertain about our robot as with our entire team. The code hasn't had enough time to test. The pneumatics aren't calibrated or even ready to be used, and overall our bot needs more testing if we had one more month, one more week, or even just one more day we could've had a semi-working robot.

Our roller system and intake is all good. Our pneumatic systems are like I said untested and requires testing. Josh wasn't at the meeting today we don't have a expansion mechanism. The turret is a whole story on its own. As well as the flywheel.

We removed the piston from the flywheel mount to have a flywheel above the mount so we fit within size constraints. Which still wouldn't at all. The flywheel gearing doesn't even spin anymore after I "fixed" it. Its whole mess I don't want to even get into because at this point I don't know what's the problem. The turret when spinning derails itself from its chains when spun. Its doing horribly.

Our goals at this point is to have a good interview and aim for the design award. I am confident in our notebook and that it can go against most if not all teams at the Tournament. iTech has been known to have a notebook a step above every other notebook so we are hopeful. I was told by Joseph that a competitor we may have is 2990U for the design award.

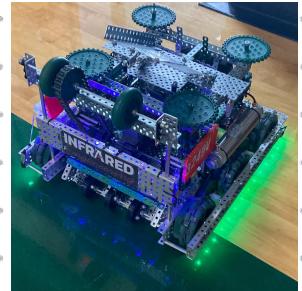
After this Tournament we may just take apart our robot and start from scratch. Most of the robot needs a redesign which we lack any time for. The chassis has a horrible camber problem which we need to address and other issues that I already addressed earlier. The main goal post tournament is to rebuild, refine, and perfect the robot. We really hope the qualify at this tournament and then go to the 14th of january Sandy tournament with a full turret. As well as fixing all the issues we have above

Well I am hopeful for tomorrow. My main role is to research other bots and improve our own both. And maybe code if the odometry wheels are mounted Ethan. Other than that I need to calm down from the stress of today and prepare for tomorrow. I need to go to bed snow.

Meeting #38

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Scout out robots to see what our competition is. Talk to future alliance partners and talk to enemy alliances.	Present the Solution
Ethan	Assemble and fix parts of the robot. Firsts things first is to install the Odometry wheels.	Build a Prototype
Josh	Help Ethan with the robot and record matches for the team.	Build a Prototype
Tyler	Drive at competition matches and help ethan with the robot	Present The Solution

Meeting #38 Continued

Post Meeting Info

Meeting Info

Location School: North Marion High school
Attendees: All Members
Duration: 7:30am - 6:30pm
Date: 11/19/22

Team Members	What got done	Unresolved Problems	Images
Aidan	I had to reprogram the autonomous due to not having Odom Wheels. I was also on drive team helping scout other teams	We need to improve the notebook and Interview by adding more notes about the engineering design process ** We took a break after the tournament to rest	
Ethan	Fix issues such as chassis drift, and install the flywheel	Flywheel never worked and we lost our finals matches	
Josh	Recorded a Majority of our Matches	Odom wheels didn't fit because we never tested them	
Tyler	Drove the robot at the competition winning 1:5 and getting 55 in skills.	We needed more drivers testing	

Meeting #39

Robot beginning of
Practice

N/A

Pre-Meeting Plan

Team Members	Tasks and Obstacles	Design Process step
All	Reflect on the Tournament and plan our next step.	Evaluate the Solution

Meeting #39 Continued

Post Meeting Info

Meeting Info
Location School: iTech
Prep
Attendees: All Members
Duration: 10:55 - 11:25

Team Members	What got done	Unresolved Problems
All	We made a Evaluator/Designer system to redesign our robot and create a task priority list. Evaluator list: Chassis - Aidan/Tyler Intake - Ethan Roller - Tyler Turret - Aidan Expansion - Josh Flywheel - Aidan Task priority list Chassis /w odom wheels Rollers Intake Turret Flywheel Expansion	

Nov 19th Tournament

Morning of Tournament

Going into the tournament after failing to create the turret I personally came in nervous hoping for the design award. At that point we only had a roller mechanism and chassis with no Odometry wheels. So going into it we had nothing we originally planned for. When we got there I began scouting while Ethan attempts to get the flywheel working. We didn't have a robot that we thought could qualify, that combined with the time crunch I didn't have a spreadsheet prepared for the tournament. Our first matches were first posted after the Drivers Meeting.

Q 5 10:41 AM	<u>3249V</u> 27508C	74923C 64900D
Q 12 11:20 AM	16689A 1812Y	<u>3249V</u> 1812X
Q 29 12:36 PM	27508A 27508D	<u>3249V</u> 64900B
Q 40 1:56 PM	64900A <u>3249V</u>	64900C 1460M
Q 48 2:35 PM	1460W 51581D	<u>3249V</u> 27508E

Once we got our times and matches we decided a good time for interviews was between matches 12 and 29. We wanted to get the interview out of the way as quick as possible while also having enough time to make a half decent interview.

Nov 19th Tournament

Qualification Rounds

We were Q5 for our first match which gave us little prep time, or so we thought. Once we were queuing it took 20-30 minutes for our match to start. It gave me time for a basic autonomous. It consisted of pushing our 2 preloads into the low goal zone. When doing a single test of this to make sure it was all going the

```
wait(1000,msec);
leftFrontMotor.setVelocity(50,percent);
leftBackMotor.setVelocity(50,percent);
rightFrontMotor.setVelocity(50,percent);
rightBackMotor.setVelocity(50,percent);
leftFrontMotor.spinFor(forward,5,rev);
leftBackMotor.spinFor(forward,5,rev);
rightFrontMotor.spinFor(forward,5,rev);
rightBackMotor.spinFor(forward,5,rev);
```

correct direction I found out one of the reasons our chassis is drifting. I forgot to add a true value so each motor spun one at a time and I found out the rightBackMotor wasn't connected. I fixed the code and Ethan made a quick repair which unfortunately didn't fix the problem as a whole. We won the match 82:26 which was unfortunately our only win.

The second match was a hard loss. The match started off in 2 vs 1 with us having two Robots. In the autonomous we got stuck on our low goal zone and had our alliance member unstuck us. Their bot got caught on a disc afterwards. In which we couldn't help them. Our opponent 16691A had a catapult and was one of the best teams in oregon at the time so they swooped us up. We lost 106:20.

After the match we had our interview which in my opinion was really bad. I knew the notebook wouldn't be able to stand on its own so the interview was the best bet. I tried to talk more on the bot, and tried to talk about the notebook. I stopped myself halfway through transitioning in fears of speaking too much. We didn't talk about the notebook, and overall had a decent interview, at most we got a 3 on it.

The third match was close only losing it due to losing from the autonomous bonus. The round was going to be our win otherwise. Our Fourth match was honestly a fair match. The main issue was the chassis drifting so we couldn't fully drive straight. If we had a straight driving chassis we might've actually won the round. The last match we might've had a chance with winning from a good autonomous with a good end position. The issue again was the chassis drifting preventing us from taking the win.

Nov 19th Tournament

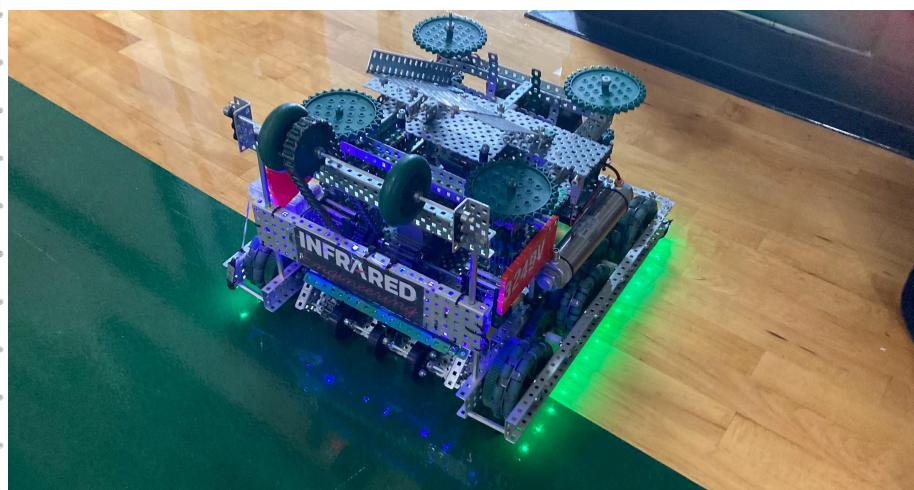
Elimination Rounds

We didn't come into the tournament expecting to win, so I didn't take the effort to make a spreadsheet for alliances, and we were 36th place so it didn't matter. We were able to partner with 64900D the 16th place team. In this time everyone tried getting the flywheel on and the odometry wheels mounted. The odometry wheels due to a lack of testing, couldn't be fitted onto the bot, and the flywheel wasn't working due to a lack of testing.

The round was a loss at 72:26 with barely anything scored just a couple discs and expansion. In the round tyler used the wrong autonomous we drove right over the line and lost the autonomous bonus. We didn't have the potentiometer code fully ready at this point. We were doing decently and we almost had a single roller. In the end it futile/

Reflection

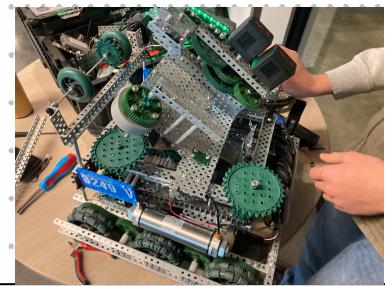
What could we do better? Testing was something we should've done. Nothing on that robot was reliable and failed at least one time during our matches. Another thing we could've had on was the Odometry wheels. If we had the odometry wheels and they've been proven to work we could've gone 5:1. A complete reverse of our 1:5 win loss. In the end we didn't qualify and lost the notebook award. In the future we need to cover more of the design process such as actual designing, brainstorming, etc. We will try to improve the notebook as well being more objective and going more with a Problem/Solution style. Our next Tournament is the January 7th Tournament so we hope to qualify there.



Meeting #40

Robot beginning of
Practice

Pre-Meeting Plan



Team Members	Tasks and Obstacles	Design Process step
Aidan	Dismantle parts of the robot	Going back to the beginning of the process

Meeting #40 Continued

Post Meeting Info

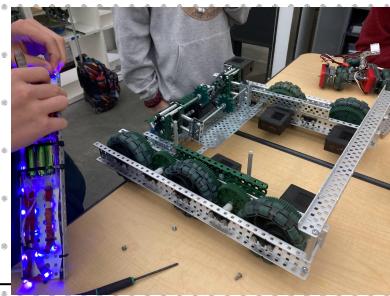
Meeting Info
Location School: iTech
Prep
Attendees: Aidan
Duration: 10:55 - 12:45

Team Members	What got done	Unresolved Problems	Images
Aidan	Unmounted the turret and removed the back part of the Chassis		

Meeting #41

Robot beginning of
Practice

Pre-Meeting Plan

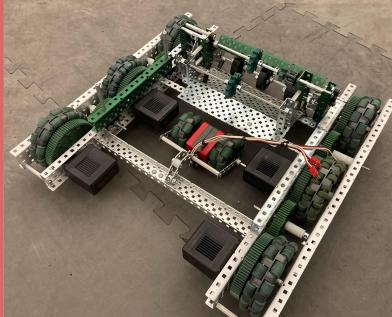
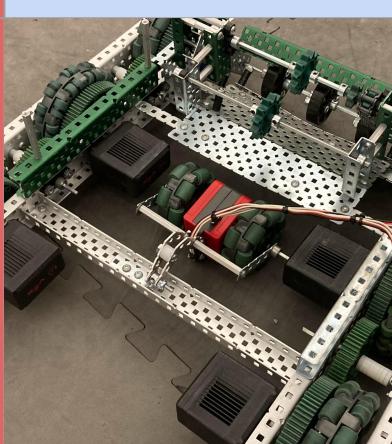
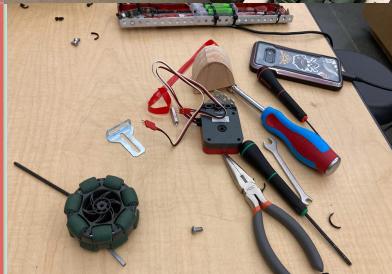


Team Members	Tasks and Obstacles	Design Process step
Aidan	Make a new meeting format and new notebook formatting	Design a Solution
Ethan	Rebuild the Chassis	Build a Solution
Josh	Rebuild the chassis and attack odom wheel	Build a Solution
Tyler	Reorganize Box	N/A

Post Meeting #41

Name: Aidan	Meeting Tasks 1, New Meeting Format 2, New Page Formatting	Tasks Completion: 1, 100% 2, 33% Time Worked: 4:05pm - 6:00pm	Next Tasks: Begin Odom Wheel Testing and PID testing
Name: Ethan	Meeting Tasks: 1, Rebuild The Chassis 2, Create and Attach Odometry Wheels	Tasks Completion: 1, 50% 2, 40% Time Worked: 4:05pm - 6:00pm	Next Tasks: Finish Odom Wheels (Thursday)
Name: Josh	Meeting Tasks: 1, Dismantled old Odometry wheels	Tasks Completion: 1, 100% Time Worked: 4:05pm - 6:00pm	Next Tasks: N/A
Name: Tyler	Meeting Tasks: Organize the Toolbox	Tasks Completion: 1, 50% Time Worked: 4:05pm - 6:00pm	Next Tasks: Continue Organizing

Meeting #41 Details

Name: Aidan	Task: New Page Formatting	Details: After reviewing our notebook and receiving tips from Joseph, I decided to redo our notebook so we have a better chance at design award.	N/A
Name: Ethan	Task: Rebuild the Chassis	Details: At the Tournament the chassis had drift issues from the wheels rubbing against the chassis combined with other issues it needed to be rebuilt.	
Name: Ethan	Task: Attach Odometry Wheels V3	Details: At the tournament we didn't have the odometry wheels, meaning we had no planned strategies and no quick creation. This lead to our repeated losses	
Name: Josh	Task: Dismantle Odometry Wheels V2	Details: The Odometry Wheels V2 didn't fit with the chassis at the tournament so we had to dismantle them.	
Name: Tyler	Task Organize Toolbox	Details: Our toolbox is a complete mess and requires organization.	

Page Formatting Version 2

Overall Formatting Changes

New Color Coding for Each page for Organization and Aesthetic purposes.

I decided on adding color. In projects the color coding actual means something. In non organized sections or in meeting notes it's just used for aesthetics. Light green will be subtitles. Light blue will be image places or misc. Red mean Problem, and Blue will be Solution. Purple will be overall changes. If the page doesn't require Problem and Solution Red and blue will be used to note specific changes.

Colored Text Boxes instead of blank paragraphs for Organization.

Instead of having subheading with paragraphs I switched it to colored boxes. Each box will have a heading in 20 point font with a sentence that describes the basic description, and in 15 point font below will be details. There will always be some space between each box. Another benefit to this is the modularity of the page.

Meeting Note Changes

New Task based Meeting Formats instead of a basic Todo list for better Time management.

I changed the format with completing tasks in mind. On the old format it was a Todo during practice, Then a What did you do. I changed it to a more task based format. Asking questions like "What tasks got done?", "What is the deadline of this task?", and "What is your next tasks?" Is a good way to stay on in robotics. The reason we failed at the tournament was because of our unorganized time management. Which prevented us from completing our work.

Added a "Previous task" Section

The Previous task section was added so incase if someone was gone one day and someone had to finish they're task. The pre-meeting notes will credit both of them.

Page Formatting Version 2 Part 2

Added a Task Deadline in the Pre-Meeting notes so we have a set timeframe to complete things

As mentioned before time management was one of our issues at the tournament. Having a task deadline creates a pressure and a reminder that we need to get things done on time. Along with this I have plans to make a monthly calendar with weekly updates, so we know if we are either on time, need to catch up, or ahead of our goals.

Added a “Secondary Task” Section In the Pre-Meeting Notes to maintain productivity.

The secondary task section is for when the First task is either near completion, or is a really quick thing to do. We have a secondary task in order to maintain workflow.

Adds a Task recap and Task Completion in the Post meeting Notes so we can track our Progress better.

In the old meeting notes we vaguely described what we did from a day to day basis. It was vague and not thorough enough for the notebook. We added a percentage based completion so we have a idea on progress and if we will meet our deadlines. We recap your tasks so we know what you did that day. We number the tasks you did then in the completion space the task number will correlate to the percent complete.

Added a “Next Task” Section In the Post Meeting Notes for better planning

This change serves two purposes, (1) The next task will help remember what things need to be done. (2) If someone is missing and their deadline is coming up. A person can do their task for them if they have a non-urgent task themselves. An example is tomorrow's meeting where Ethan will be absent from the practice that day. One of my task requires the Odometry wheels, for this case I will do Ethans Next task instead of my Task, once I'm done I go back to my task.

Page Formatting Version 2 Part 3

Replaced the Images section and The “What Got Done” Section with a new Meeting Details Page for Better Spacing.

A main thing I observed with our old format is the amount of resizing and font changing. I want to reduce the use of that to the point where its only necessary to not waste a page. An example of resized so we didn’t have to use a second page is Meeting #41. Another reason I added a details page is so I can expand it how I like. The new Box system is modular allowing me to potentially use a second page for Task Details

Removed the Pre Meeting Image for Cleanliness

The Pre meeting image was a addition to help track progress, but was a bit awkward in its position. With the new Details section its far better to look from other meetings detail pages instead of looking at pre and post meeting photos.

Removed Meeting Info Section and replaced it with “Time Worked” in Post meeting notes.

This was a add in from the old meeting formats which didn’t work well for the new format. Instead of having it in the top right I added a “Time Worked” inside the same section as “Task Completion.

Added a Page format Copy and Paste to prevent wasting time when setting up meeting pages.

Pre Meeting Format on Page 222, Post Meeting Format on Page 223, and Meeting Details Format on Page 224. Remember to set up Meetings in that specific order.

Pre-Meeting Format

Name:	Previous Task:	Current Task:	Task Deadline	Secondary Task

Post Meeting Format

Name:	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:
Name:	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:
Name:	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:
Name:	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting Details Format

Name:	Task:	Details:	Images
Name:	Task:	Details:	Images
Name:	Task:	Details:	Images
Name:	Task:	Details:	Images

Page Formatting Version 2 Part 4

Design Process Page Format Changes

Created a The First Format for the Design Process.

Never made a format for the Design Process Pages in the past. I made one for the two scenarios. When something on the build has been changes like during a build day, or when there is a problem. When there is a problem two boxes will be made. The red problem and the blue solution, then a changes+image section is added when the solution has been solved in the build.

Iterative Version naming system will be changed so we don't have a different version for minor changes or incomplete changes.

Usually when I make a page on a Design I give it a version number. An example being Chassis V3 or Turret V2. I also have incomplete builds as version such as Flywheel V3. To clear future confusion Non-complete mechanism, or a small change is made, we will use the Previous Version number and add a 1 in the decimal place.

Example Design Process Page Format

The format for The design process change will be put on page ###. Keep in mind that on the format page is a copy and paste of everything that will be used in the project.

Coding Page Format Changes

Created a The First Format for the Coding Section

The first format of coding is a lot like the Design Process Format. There are some notable changes. Number 1 being that it works off of Test logs and not simple Problems + Solutions. A majority of coding requires many tests. So I thought solving code using test logs would be more effective than simply using Problems + Solutions.

Design Process Page Formatting

Mechanism Version

Image Placeholder

Changes

Details

Problem:

Problem details

Solution:

Solution details

Page Formatting Version 2 Part 5

Example Coding Page Format

The format for The Coding will be put on page 228. The format page is a copy and paste of everything that will be used in the project.

Strategy Page Format Changes

New Table design to Fit more Programs on Each Page.

The old Format only held 3 strategies on each page and had a code entry underneath. This setup was inefficient for how much space that strategy took. For better spacing I included a more compact table.

Removed Code Snippet from Strategy Entry for Spacing

I removed the coding snippet for two reasons. (1) To allow for more space and more strategy entries on each strategy page. (2) I usually forget to make strategy code, so when I am testing the code I will be adding the code snippet along with it.

Example Strategy Page Format

The format for Strategy Pages will be put on page 229. The format page is a copy and paste of everything that will be used in that project.

Coding Page Formatting

Code Sub-Section

Code Snippet

Description/Changes

Reason for Code/ Reason for Change

Log Number	Code being Tested	Accuracy	Problems?	Solution?

Strategy Page Formatting V2

Strategy Number and Name	Program and Potentiometer	Points Scored	Brief Description	Date Created, Author, and Page Number	Tested and Worked?

Design Process Page 1

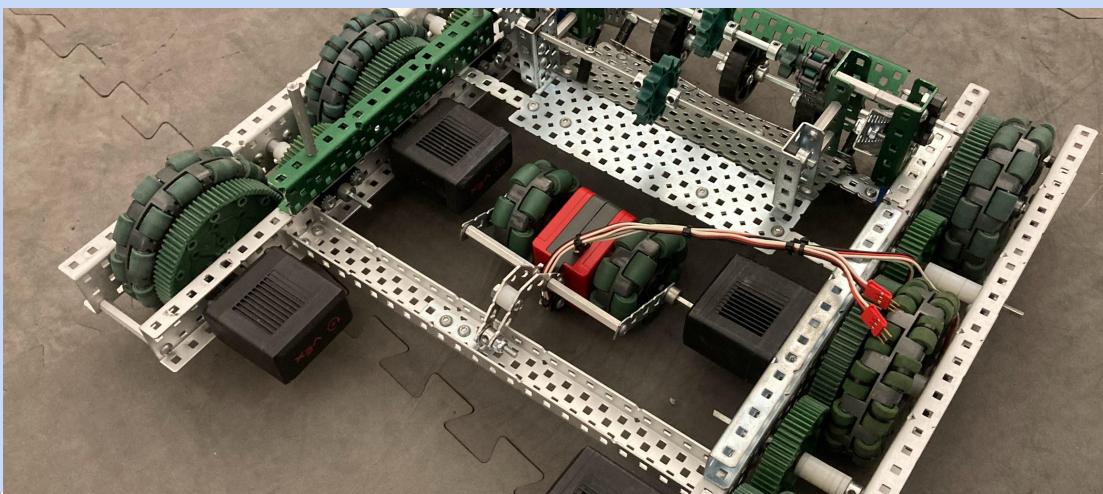
Odometry Wheels Version 2.1

Problem: Odometry Wheels Don't Fit

The original odometry wheel mounting bracket was too large and flimsy, and there was no way to securely mount the system.

Solution: More Integration of the Odometry wheels instead of a Modular Installation.

As we rebuild the chassis, we are going to be integrating the odometry wheels into the chassis first, so we are sure to have the system built in properly, and to prevent error in the future. We will center the turning wheels and attach the drift wheel to the back.



Changes: improved and integrated mounting suspension

We installed a 24 2x1 C-Channel in the middle. Then attached a 90 degree bracket down to a back to back encoder, both wheels are free spinning, as the shaft does not extend fully through both encoders. Both sides are braced, and connected to the 90 degree bracket, to ensure a smooth up and down motion to contact the field at all times. The bracket can droop about a quarter inch, and is pushed down towards the field at all times due to the rubber band, while still being free floating. This may provide an advantage as the suspension will allow the bot to pass over the barrier on the field without getting caught on the odometry wheels

Design Process Page 2

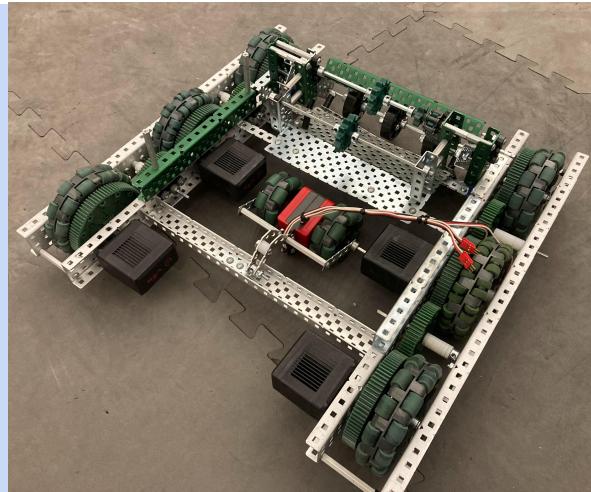
Chassis Version 3.1

Problem: The Chassis Was drifting during the Tournament.

During the tournament a big issue we ran into was the chassis drifting. This caused multiple problems and indirectly lead to losses. Now after the tournament It is time for us to fix the issue

Solution: Rebuild the Right side of the chassis to prevent wheels rubbing against the Metal.

We found when lightly spinning the chassis that the right wheels rubbed against the metal causing drift. Now we need to rebuild the right side of the chassis.



Changes: Added Spacing to each side for more Wheel Room

We added washers on each side to slightly expand the amount of room the wheels had. This hopefully plans the drift problem for the future. Another issue we ran into that this change fixed is that the inner part of the wheel casing didn't align to a C-Channel. Now with the slightly changed spacing we can secure the back portion far better.

Design Process Page 3

Odometry Wheels Version 2.2 Plan

Problem: Odometry wheels are free floating and flimsy

The wheels sit independently and may shift around during competition

Solution: Rubber band suspension system

We will add a rubber band and standoffs to keep tension on the system at all times



Plan: Add rubber band and Standoffs for Suspension.

The plan for the next practice is to add suspension for the odometry wheels in order to stay on the ground at all times

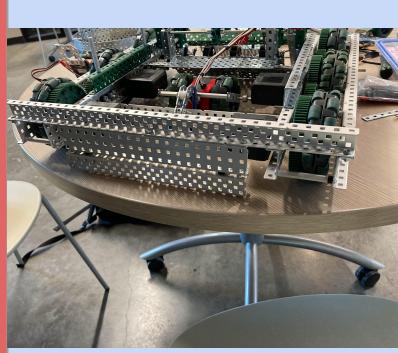
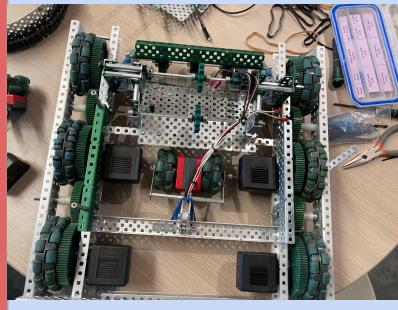
Pre-Meeting #42

Name: Aidan	Previous Task: Make new page Format	Current Task: Install a back c-channel onto the Chassis to complete Chassis redesign	Task Deadline 12/1/22	Secondary Task Test Odometry Wheels
Name:Ethan	Previous Task: Install Odometry Wheels	Current Task Finish Installing Odometry Wheels	Task Deadline 12/1/22	Secondary Task N/A
Name:Josh	Previous Task: Dismantle The old odometry wheels	Current Task Create the Expansion Mechanism	Task Deadline 12/16/22	Secondary Task N/A
Name:Tyler ABSENT	Previous Task: Organize box	Current Task: ABSENT	Task Deadline: ABSENT	Secondary Task: ABSENT

Post Meeting #42

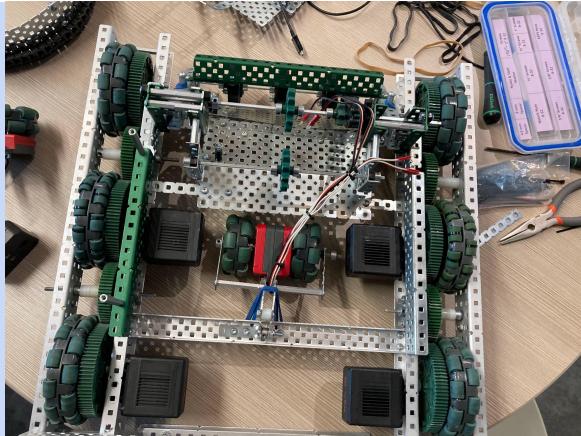
Name: Aidan	Meeting Tasks: Install 1, C-channel onto the back of the Chassis to finish New chassis	Tasks Completion: 1, 100% Time Worked: 30 Minutes	Next Tasks: Test Odometry Wheels in coding
Name: Ethan	Meeting Tasks: 1, Install Odometry Wheels	Tasks Completion: 1, 66% Time Worked: 30 Minutes	Next Tasks: Finish Install Odometry wheels
Name:Josh	Meeting Tasks: Create Expansion Mechanism	Tasks Completion: 1, 20% Time Worked: 30 Minutes	Next Tasks: Finish Expansion Mechanism
Name: Tyler	Meeting Tasks: ABSENT	Tasks Completion: ABSENT Time Worked:	Next Tasks: Finish Organizing Toolbox

Meeting #42 Details

Name: Aidan	Task: Finishing Chassis Redesign	Details: Installed the back part of the chassis in order to complete the chassis redesign. And begin making the chassis drivable again.	
Name: Ethan	Task: Install the Odometry Wheels	Details: Installing the rubber band suspension system detailed in page 232	
Name: Josh	Task: Create an Expansion Mechanism	Details: took a pneumatic nut as a launch weight. Then tied our string to a expansion mech we researched.	

Design Process Page 4

Odometry Wheels Version 2.2



Change: Executed plan to add rubber band and Standoffs for Suspension on the left and right Odometry wheel

We added Rubber bands and standoffs on the left and right odometry wheels in order to maintain contact with the ground at all time.

Chassis Version 3.2

Problem: The Chassis has Camber which prevents it from driving Straight.

Along with the spacing problem the chassis has camber along its outer edges which prevented it from driving straight in the tournament.

Solution: Replace the Bendy 90 degree angle gussets with a C-Channel to connect the back

Setting up a C-channel facing inward should be enough to fix our camber issue.

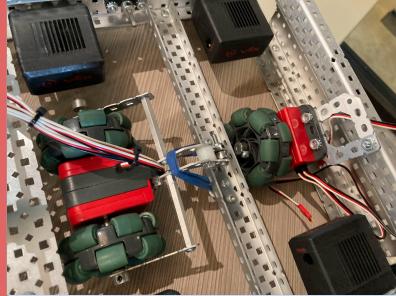
Pre-Meeting #43

Name: Aidan	Previous Task: Create New Notebook Format	Current Task: Finish Odometry Wheels (Ethans Task)	Task Deadline 11/29/22	Secondary Task Begin Odometry Testing
Name: Tyler	Previous Task: Organize Box	Current Task Continue Organizing Box	Task Deadline 1/7/22	Secondary Task N/A
Name: Josh	Previous Task: Create Expansion Mechanism	Current Task: Setup Lucid spark	Task Deadline 11/29/22	Secondary Task: Continue working on expansion,
Name: Ethan ABSENT	Previous Task: Rebuild the chassis	Current Task: ABSENT	Task Deadline ABSENT	Secondary Task ABSENT

Post Meeting #43

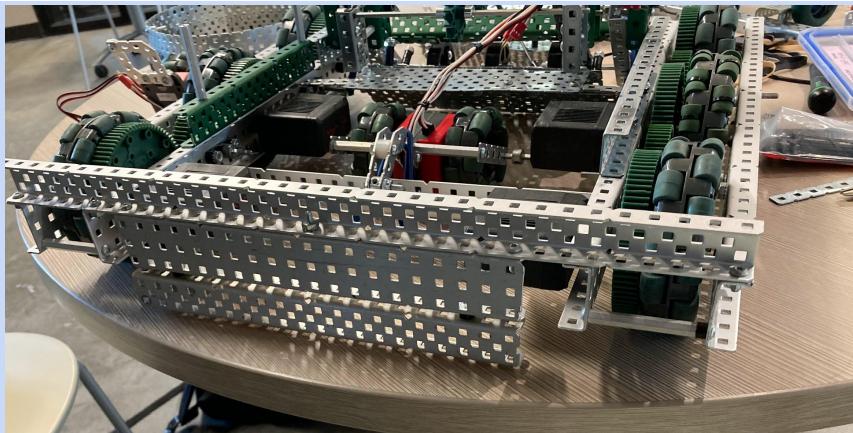
Name: Aidan	Meeting Tasks: 1, Install Odometry wheels	Tasks Completion: 1, 90% Time Worked: 2 Hours	Next Tasks: Test Odometry wheel code
Name: Josh	Meeting Tasks: 1, Setup Lucidchart 2, Continue creating an Expansion Mechanism	Tasks Completion: 1, 50% 2, 30% Time Worked: 2 Hours	Next Tasks: Continue working on the expansion Mechanism
Name: Tyler	Meeting Tasks: 1, Organize the toolbox	Tasks Completion: 1, 90% Time Worked: 2 Hours	Next Tasks:
Name: Ethan ABSENT	Meeting Tasks: ABSENT	Tasks Completion: ABSENT Time Worked:	Next Tasks:

Meeting #43 Details

Name: Aidan	Task: Install Odometry wheels	Details: Installed the back odometry wheels and added suspension.	
Name: Josh	Task: Setup LucidChart	Details: Setup the Lucid chart and shared it with everyone so we can have a task board.	N/A
Name: Josh	Task: Create Expansion Mechanism	Details: Setup the framework of how the launch mechanism will work	
Name: Tyler	Task: Organize the Toolbox	Details: Put unneeded things away and all that's left is to get a magnet to pull all the screws from the bottom of the toolbox.	N/A

Design Process Page 5

Chassis Version 3.2



Changes: Added the back C-channel as well as the attached the old low clearance back so we can Push discs.

First we screwed on the C-channel to both sides of each wheel casings with a $\frac{3}{8}$ th inch screws. Then I installed the old low clearance section that we used in the tournament so we can score discs without possession. This method worked at the tournament fairly well. It also acts as a barrier so we don't have a disc stuck in the middle of our robot during a match.

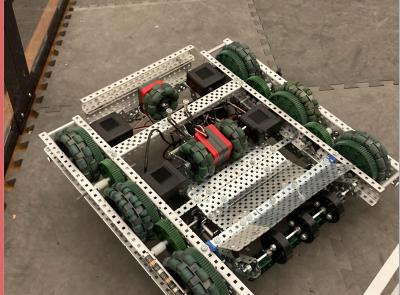
Pre-Meeting #44

Name: Ethan	Previous Task: Odometry wheel mounting bracket	Current Task: Work on the flywheel.	Task Deadline N/a	Secondary Task Make pneumatics work
Name: Aidan	Previous Task: Install the Odometry wheels	Current Task: Test the code and begin PID calibration	Task Deadline N/A	Secondary Task Setup planning for the month
Name: Tyler	Previous Task: Organize the Toolbox	Current Task: Finish Organizing the toolbox	Task Deadline: N/A	Secondary Task: Begin working on the new turret design
Name: Josh	Previous Task: Create expansion mech	Current Task: Continue working on the expansion mechanism	Task Deadline: 12/16/22	Secondary Task: N/A

Post Meeting #44

Name: Ethan	Meeting Tasks: 1, Install Odometry wheels 2, rebuilt turret frame	Tasks Completion: 1, 100% 2, 65% Time Worked: 4 hrs	Next Tasks: finish and mount turret frame, then work on the indexer.
Name: Aidan	Meeting Tasks: 1, Test the code 2, begin PID calibration	Tasks Completion: 1, 25% 2, 0% Time Worked:	Next Tasks: Calibrate Odometry Code
Name: Tyler	Meeting Tasks: 1, Organize the toolbox	Tasks Completion: 95% Time Worked:	Next Tasks: Work on turret or flywheel
Name: Josh	Meeting Tasks: 1, Create the expansion mech	Tasks Completion: 45% Time Worked:	Next Tasks: Continue making expansion mech

Meeting #44 Details

Name: Aidan	Task: Test the code	Details: I ran some basic tests to get the debug thread working so I can begin calibrating the bot	N/A
Name: Aidan	Task: Calibrate PIDs	Details: We had some issues with Odom wheels and people were running scrimmages so i couldn't use the field	N/A
Name: Ethan	Task: Install Odometry Wheels	Details: We reinstalled the back odometry wheel so it is more stable. Then we had to wire the whole thing together	
Name: Ethan	Task: rebuild the turret frame	Details: Remade the turret frame with aluminum	

Meeting #44 Details

Name: Josh	Task: Create Expansion Mechanism	Details: Made the top framework of the expansion mech	
Name: Tyler	Task: Organize tool box	Details: Pretty much finished with organizing the toolbox.	N/A

Coding Page 1

Log Number: 1	Code being Tested:Thread 2	Accuracy: N/A	Problems: Words were not on screen	Solution: Add a set cursor line to the code
Log Number: 2	Code being Tested:Thread 2	Accuracy: N/A	Problems: Values were not Displayed	Solution: Separated out values and words on the table

Thread 2 - Debugging Thread

```
424 Controller1.Screen.clearScreen(); Controller1.Screen.setCursor(1, 1);
425 Controller1.Screen.print("X: "); Controller1.Screen.print(positionX);
426 Controller1.Screen.newLine();
427 Controller1.Screen.print("Y: "); Controller1.Screen.print(positionY);
428 Controller1.Screen.newLine();
429 Controller1.Screen.print("Heading: "); Controller1.Screen.print(degHead);
```

Coding snippet of Thread 2 (Figure 1)

Added Solutions for Log 1 and Log 2 into the Code

I first added a setCursor on line 424 (Figure 1) to set it to the top left corner of the controller code so all the code is on the screen. Then on lines 425, 427, and 429 I separated out values and text so the controller brain properly displays them.

Log Number: 3	Code being Tested:Thread 2	Accuracy: N/A	Problems: Controller wasn't maintaining connection with Brain	Solution: update radio and controller firmware
------------------	-------------------------------	---------------	---	---

Coding Page 2

Log Number: 4	Code being Tested: Thread 2	Accuracy: N/A	Problems: Undefined Error 03800CE8	Solution: Re-downloaded code
Log Number:5	Code being Tested: Thread 2	Accuracy: N/A	Problems: No issues Thread 2 is good to go	Solution: N/A
Log Number:6	Code being Tested: Thread 1	Accuracy: N/A	Problems: the turn code is not working despite the robot turning.	Solution: Revert to old turn code Calculations
Log Number:7	Code being Tested: Thread 1	Accuracy: 0%	Problems: X and Y start with X -201629274 and Y 52503992.07	Solution: Run preauton code whenever I use the odom
Log Number:8	Code being Tested: Thread 1	Accuracy: 0.1%	Problems: inches calc is way off	Solution: Remove the radians translation as it is not needed

X, Y Odometry Code

```

378     diffrence = encoderVL - encoderVR;
379     lastdegHead = degHead;
380     degHead += (((2*M_PI*offsets[1])/360)*((diffrence)*(dragWheelCirc/360)))*(180/M_PI);

```

Removed Radians Translation

Removed unnecessary radians translation that messed up the code

Pre-Meeting #45

Name: Aidan	Previous Task: Test code	Current Task: Do Odometry tests	Task Deadline 12/8/22	Secondary Task Do PID testing
Name: Ethan	Previous Task: mount turret	Current Task: Finish and mount turret	Task Deadline 12/16/22	Secondary Task Spray paint the 6 inch, 4 mm standoff
Name: Josh	Previous Task: Work on Expansion Mech	Current Task: Work on Expansion Mech	Task Deadline 12/16/22	Secondary Task
Name: Tyler	Previous Task: Continue Organizing Bin	Current Task Contiune Organizing	Task Deadline N/A	Secondary Task

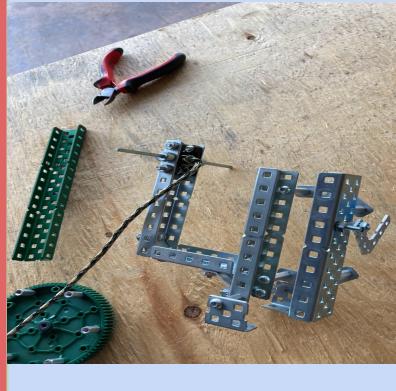
Post Meeting #45

Name: Aidan	Meeting Tasks: 1, Test Odometry Wheels 2, Test PID	Tasks Completion: 1, 35% 2, 5% Time Worked: 4 hours	Next Tasks: Continue Testing Odometry Wheels
Name: Ethan	Meeting Tasks: 1, Finish and Mount Turret 2, Spray paint the 6 inch,4 mm standoff	Tasks Completion: 1, 50% 2, 100% Time Worked: 4 hours	Next Tasks: Create The new Intake
Name: Josh	Meeting Tasks: 1, Work on Expansion Mech	Tasks Completion: 50% Time Worked: 3 hours	Next Tasks: Continue working on Expansion
Name: Tyler	Meeting Tasks: 1, Organize Tool Bin	Tasks Completion: 100% Time Worked: 2 Hours	Next Tasks: Continue Organizing the Toolbin

Meeting #45 Details

Name:Ethan	Task: mount the turret frame	Details: I built the frame at home, and i mounted the frame to the robot	
Name:Ethan	Task:Spray paint the 6 inch,4 mm standoff	Details: Ethan Spray Painted our abnormally long stand-off for decoration.	
Name:Aidan	Task: code for auton with new odom wheels	Details: I did some testing of the Odom wheels. Code tests in Code Entries Pages 2 - 3	N/A
Name:Aidan	Task: Test PID	Details: I did some minor PID tests that failed. Check code entries for full details	N/A

Meeting #45 Details

Name: Tyler	Task: Organize Tool Bin	Details: Finally finishing the Organization.	N/A
Name: Josh	Task: develop the expansion mech	Details: Worked on the end of the Expansion Mechanism	

Coding Page 3

Log Number:9	Code being Tested: Thread 1	Accuracy: 45%	Problems: degree code is incorrect but moving forward is fine	Solution:
Log Number:10	Code being Tested: PID move	Accuracy: N/A	Problems: The motors weren't wired in because we were building	Solution: plug the motors in...
Log Number:11	Code being Tested: PID move	Accuracy: no	Problems: Something tangled in the motors	Solution: Get the thing out
Log Number:12	Code being Tested:	Accuracy: N/A	Problems: overshot	Solution: I don't know

November Monthly Update

Overview

Page Overview

In this Update I will cover how we were at the beginning of the November, the tournament, and how we are at the end of November. I will have a Notebook section, Build section, and a Code section for each part.

The updates for now on will be Time/Event based and not Part based.

When doing monthly team updates it best for us to record events. As well as the beginning and end of the month. An example of an important event would be a large. The change is simply to better record turning points in the team.

Basic Summary of the Month

This month a lot of things happened but the main event was the tournament. We started off the month from a very successful October. We design a Logo to give our team a Identity other than the Robot. We worked a lot on notebook entries and we even cad'd some design Plans. We did a last week sprint to get the notebook to 200~ pages so we had a shot at design.

Then the tournament hit and we were decimated getting 1:4. Our rebuild plan was going to happen on the tuesday prior to thanksgiving, but our Coach was sick. So left with less time we weren't able to fully redesign our robot and was forced to just build iteratively off of what we have. After the tournament we saw our flaws and began redoing everything, even the notebook.

November Monthly Update Continued

Beginning Of the Month



Beginning of the Month Robot: Bed for Improvement

At the beginning of the month we had the the old turret frame. The old 4 wheeled chassis. Overall just the beta design of our bot before the tournament. Not much works but it serves as a bed for our improvement.

Beginning of the Month Code: Preparation Almost Complete.

At the beginning of the month, the code itself is not fully complete. I was close to the final Trajectory code but other than that we had the basis for the Odometry, Aimbot, and PID's

Beginning of the Month Notebook: Really Promising

At the beginning of last month we were at page 129. So far other than poor formatting it was really promising for a notebook. Honestly the notebook was the once thing we excelled at the time.

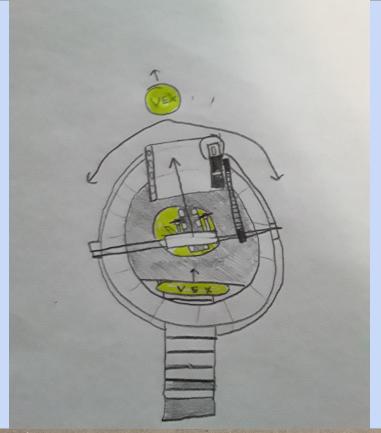
Pre-Meeting #46

Name: Aidan	Previous Task: Test the PID	Current Task: Work on the Notebook:	Task Deadline: 1/7/23	Secondary Task: Test code
Name: Ethan	Previous Task: Mount the Turret	Current Task: Make the New Intake	Task Deadline: 12/16/22	Secondary Task: Make indexer Platform
Name: Josh	Previous Task: Work on the Expansion Mechanism	Current Task: Work on expansion	Task Deadline: 12/16/22	Secondary Task: Research Decorations
Name: Tyler	Previous Task: Organize the Box.	Current Task: Create the New flywheel	Task Deadline: 12/16/22	Secondary Task

Post Meeting #46

Name: Aidan	Meeting Tasks: Work on the Notebook	Tasks Completion: N/A Time Worked: 2 hours	Next Tasks: Test Odom Wheels Again
Name: Ethan	Meeting Tasks: 1, Make the New Intake, 2, Make indexer Platform	Tasks Completion: 1, 45% 2, 5% Time Worked: 2 hours	Next Tasks: Make Indexer Platform
Name: Josh	Meeting Tasks: 1, Worked on Expansion 2, Researched LEDs	Tasks Completion: 1, 30% 2, 100% Time Worked: 2 hours	Next Tasks: Work on Expansion
Name: Tyler	Meeting Tasks: 1, Begin Flywheel	Tasks Completion: 1, 5% Time Worked: 2 hours	Next Tasks: Begin working on the Flywheel

Meeting #46 Details

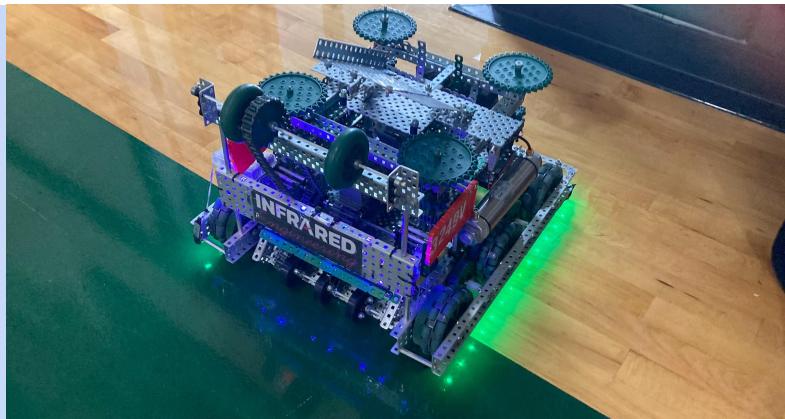
Name: Aidan	Task: Work on the Notebook	Details: Work on the already late monthly update for November	N/A
Name: Ethan	Task: Make the New Intake	Details: Started by replacing the sprocket gears with a flywheel to make the disc go upwards faster.	
Name: Ethan	Task: Make indexer Platform	Details: Sketched out a design for the new indexer	
Name: Josh	Task: Work on expansion	Details: Figured out how to fit the pneumatics for the expansion	

Meeting #46 Details

Name: Josh	Task: Research LEDS	Details: Josh researched LED's to buy for decoration	N/A
Name: Tyler	Task: Build Flywheel	Details: Began building the flywheel but had to dismantle it due to issues.	N/A

November Monthly Update Continued +

The Tournament



The Tournament Robot: Mistakes and Drifting

I already did 5 pages on the tournament, I will just cover the bot as of then. Nothing functioned except the rollers, intake, and Chassis kind of. The rollers were wobbly and needed a solid foundation. The intake was useless because no turret. And the Chassis was drifting. The bot was full of errors and needs fixing after the tournament wehe we were stripped it clean and made a new base

The Tournament Robot: No Odom Wheels

From what I knew at this point the Code at this point was ready for the fully functioning bot except with some turret calibration. If the Odometry wheels were attached we might've won more matches

The Tournament Robot: Failed Interview

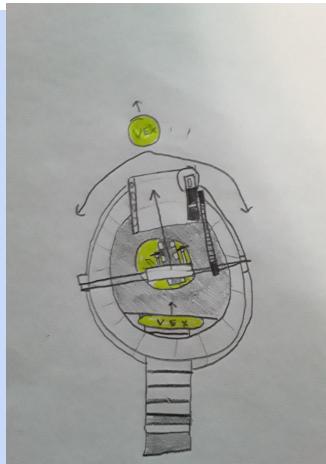
We just reached 200 pages and we were really hoping to win design. Our interview was really bad. If I were to improve it we would talk more about the code, and the notebook. Explaining that we didn't come ready and needed more time until we would be fully ready. If we followed some tips Joseph gave us we might've gotten design award.

Design Process Page 6

Indexer V2

Problem: Our original design was flimsy, overcomplicated, and not likely to work

Solution: We will be using an extended intake to get discs up to a platform

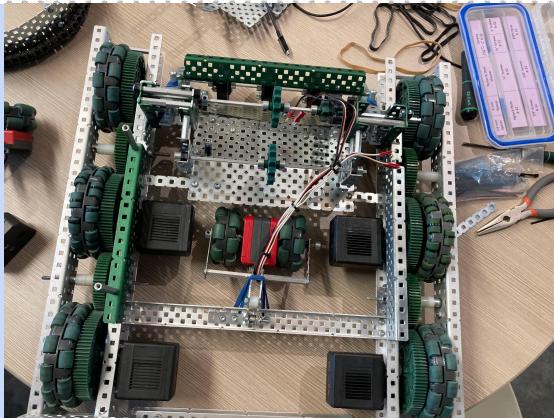


Plan: New design to the entire indexer which removes a Piston

inside the turret wheel. From there, an omni wheel geared to the flywheel will be constantly spinning in the same direction as the flywheel, and be pushing the discs into the launcher. This solution will allow full 360 shooting with no worries about pneumatics losing air. This design is the breakthrough I have been looking for since the turret was conceptualized

November Monthly Update Continued ++

End of Month



End of Month Robot: Return and Rebuild

After the tournament and after thanksgiving weekend we stripped most the bot down except from the Intake and started redesigning. We were able to put the Odometry wheels on before the end of the month.

End of Month Code: Code Testing Begins

Now that the Odometry wheels are connected to the robot. Throughout the month of December we can begin doing Odometry, PID, and Aimbot Testing.

End of Month Notebook: Innovated and Improved

This is by far the strongest new start we have after the tournament. With the new format we solved a lot of issue we had with our previous formatting. It also just looks a lot more interesting to the eye.

Overall Reflection of the Month

November was somewhat of a successful month. The tournament has helped our team and has shown us our cracks and our failings, but at the cost of a quick redesign. Our next Tournament is January 7th and we have to make a quick recovery with the turret or else we will fail there as well. And yes I know this is late

Design Process Page 7

Flywheel V4.1

Problem: Our original had too much Resistance

The Original design had too much resistance to the point where it's nearly impossible to spin and oy the gears without stripping the gears.

Solution: Redo the flywheels Structure

The reason for redoing the whole structure is because the amount of curving added too much resistance.

Change: Dismantle the Old Flywheel

This is an older change made nearly after the tournament. But we dismantled it so we can work on other things and not have it in the way. No picture because there isn't anything to show

Flywheel V4.2

Problem: In the new Structure the Motors weren't able to be properly wired.

Tyler accidentally put the two motors two close together so the front motor couldn't connect a Motor.

Solution: Give the Motors more spacing, and use bigger gears to connect each one.

Giving more spacing allows us the motors to be properly wired

Change: Dismantle the New Flywheel Design because of a lack of time

By the time the issue was pointed out it was 15 minutes left in practice and we needed to clean up.

Design Process Page 8

Turret V3

Problem: Our old Design was too Heavy.

The Old turret was extremely heavy to the point where it slowed the robot down. It was even heavier than the Chassis

Solution: Remake the entire turret base out of Aluminum.

Aluminum is a lighter material than steel so it should be light enough to be usable

Change: Implemented Previous Solution and worked

We replaced the steel base of the turret to an aluminum base and added more green parts.

Turret V3.1

Problem: The Turret Had some Camber

There is camber on the turret which may or may not affect it in the long run.

Solution: Add a Chain around the top section so it's all on the same Level while spinning

The top chain section should help with the camber by having them all spin on the same axis with some differences

Solution: Fix the Spacing on the Back Right Sprocket Gear

The back right sprocket gear was missing a spacer making it uneven.

Pre-Meeting #47

Name: Aidan	Previous Task: Work on the Notebook	Current Task: Code Testing The Odometry Wheels	Task Deadline: 12/16/22	Secondary Task: Notebook
Name: Ethan	Previous Task: fix intake	Current Task Design and build the new indexer and platform	Task Deadline Prototype before winter break	Secondary Task - build mounting bracket for expansion
Name: Tyler	Previous Task: begin making the flywheel	Current Task: Make the Flywheel	Task Deadline: 12/16/22	Secondary Task
Name: Josh	Previous Task: create the expansion mech	Current Task: Work on Expansion	Task Deadline	Secondary Task

Post Meeting #47

Name: Aidan	Meeting Tasks: Notebook	Tasks Completion: N/A Time Worked: 2 hours	Next Tasks: Notebook or if possible Odomotry wheel testing.
Name:Ethan	Meeting Tasks: Worked on the the indexer	Tasks Completion: 35% Time Worked: 2 hours	Next Tasks:
Name:Josh	Meeting Tasks: Expansion Mech	Tasks Completion: 40% Time Worked: 2 hours	Next Tasks:
Name:Tyler	Meeting Tasks: Create the flywheel	Tasks Completion: 17.5% Time Worked: 2 hours	Next Tasks:

Meeting #47 Details

Name: Aidan	Task: Work on the notebook	Details: Updated parts of the notebook	N/A
Name: Ethan	Task: Work on the Indexer	Details: Began created the design made for the Indexer	
Name: Josh	Task: Work on the Expansion Mech	Details: Attached pneumatic to launch mechanism	
Name: Tyler	Task: Create the flywheel	Details: started making the framework on the flywheel itself.	N/A

Pre-Meeting #48

Name: Aidan	Previous Task: Work on the Notebook	Current Task: Code: Simplify The code	Task Deadline: 12/16/22	Secondary Task: Notebook
Name: Ethan	Previous Task: Start creating indexer	Current Task Develop the turret	Task Deadline 12/16/22	Secondary Task - reinforce bot
Name: Tyler	Previous Task: Make the Flywheel	Current Task: Make the Flywheel	Task Deadline: 12/16/22	Secondary Task
Name: Josh ABSENT	Previous Task: Work on Expansion ABSENT	Current Task: ABSENT	Task Deadline	Secondary Task

Post Meeting #48

Name: Aidan	Meeting Tasks: Simplify code, Organize the Cubby	Tasks Completion: 20% 95% Time Worked: 2 hours	Next Tasks: Notebook or if possible Odometry wheel testing.
Name:Ethan	Meeting Tasks: Worked on the the indexer	Tasks Completion: 40% Time Worked: 2 hours	Next Tasks: Make turret-indexer connection work
Name:Tyler	Meeting Tasks: Create the flywheel	Tasks Completion: 40% Time Worked: 2 hours	Next Tasks:
Name:Josh	Meeting Tasks: ABSENT	Tasks Completion: ABSENT Time Worked:	Next Tasks: work on expansion

Meeting #48 Details

Name: Aidan	Task: Simplify Code	Details: The code was too complex for bug testing so I made it easier for myself	N/A
Name: Aidan	Task: Organize the Cubby	Details: I finally organized our cubby so we can put our stuff in it.	
Name: Ethan	Task: Worked on the the indexer	Details: Added a turret platform and a ramp to the turret	
Name: Tyler	Task: Create the Flywheel	Details: Continued designing the flywheel.	N/A

Coding Page 4

Odometry Forward Code

```
387     pPositionX = positionX;  
388     pPositionY = positionY;  
389     positionLR = (((encoderVL+encoderVR-diffrence)/2) - (lastEncoderL+lastEncoderR-pDifference));  
390     positionB = (lastEncoderL+lastEncoderR-pDifference);  
391     if((180 < degHead)) {  
392         positionX += (dragDegrees*(cos(radians(degHead))*positionLR+dragDegrees*(sin(radians(degHead))*positionB)))*-1;  
393         positionY += (dragDegrees*(sin(radians(degHead))*positionLR+dragDegrees*(cos(radians(degHead))*positionB)))*-1;  
394     }  
395     else {  
396         positionX += dragDegrees*(cos(radians(degHead))*positionLR+dragDegrees*(sin(radians(degHead))*positionB));  
397         positionY += dragDegrees*(sin(radians(degHead))*positionLR+dragDegrees*(cos(radians(degHead))*positionB));  
398     }
```

Simplified Code

I simplified the odometry code in order to quickly debug calculation problems if I made any errors.

Odometry Turning Code

```
376     difference = encoderVL - encoderVR;  
377     lastdegHead = degHead;  
378     degHead += (2*M_PI*offsets[1])/360*(difference)*(dragWheelCirc/360)*(180/M_PI);  
379     //degHead = (2*offsets[0]*difference*dragWheelCirc*180)/pow(360,2);  
380     backWheelDifference = encoderVB - offsets[0]*difference/offsets[2];  
381     if ((degHead >= 360)) {  
382         degHead = degHead-360;  
383     }  
384     else if ((degHead <= 0)) {  
385         degHead = degHead+360;  
386     }
```

Fixed Turning Code by going back to the previous code and recalculating it. I simplified the odometry code in order to quickly debug calculation problems if I made any errors.

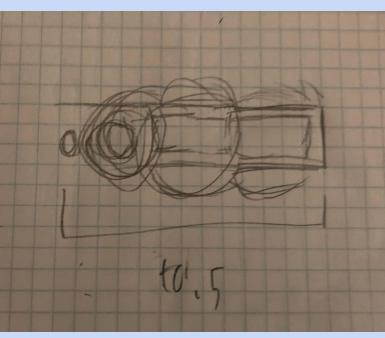
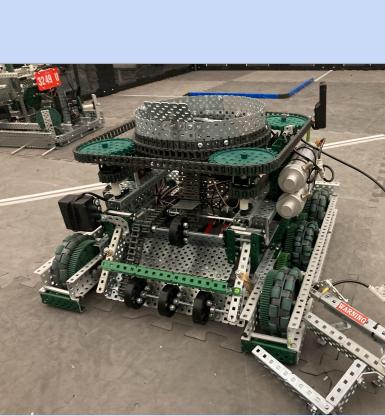
Pre-Meeting #49

Name: Aidan	Previous Task: Organize the Cubby	Current Task: Design the flywheel	Task Deadline 12/16/22	Secondary Task: work on code
Name: Ethan	Previous Task: Worked on the the indexer	Current Task: Work on developing the turret	Task Deadline 1/7/22	Secondary Task
Name: Tyler	Previous Task: Create The flywheel	Current Task Create new odom wheels	Task Deadline 1/7/22	Secondary Task
Name: Josh ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #49

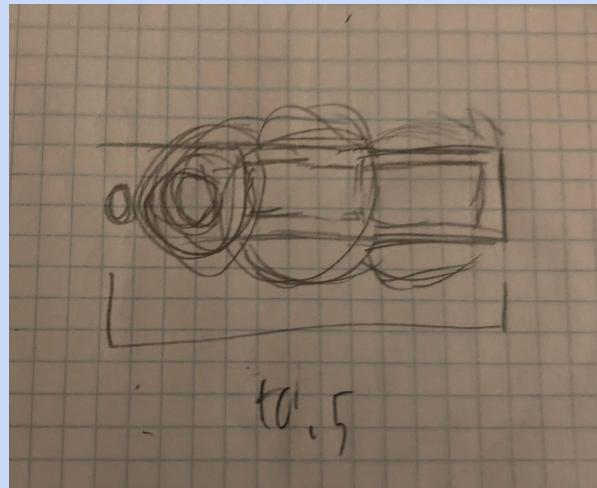
Name: Aidan	Meeting Tasks: Design the flywheel	Tasks Completion: 95% Time Worked: 5 hours	Next Tasks:
Name: Ethan	Meeting Tasks: Build the turret	Tasks Completion: 37% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Create a new odom wheel design	Tasks Completion: 95% Time Worked: 2 hours	Next Tasks:
Name: Josh ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #49 Details

Name: Aidan	Task: Design The flywheel	Details: I started designing our new flywheel with a focus on higher momentum	
Name: Ethan	Task: Work on developing the turret	Details: worked on getting the new intake to work.	
Name: Tyler	Task: Create a new odom wheel design	Details: Our current design is flimsy and it needs to be redone.	N/A

Design Process Page 9

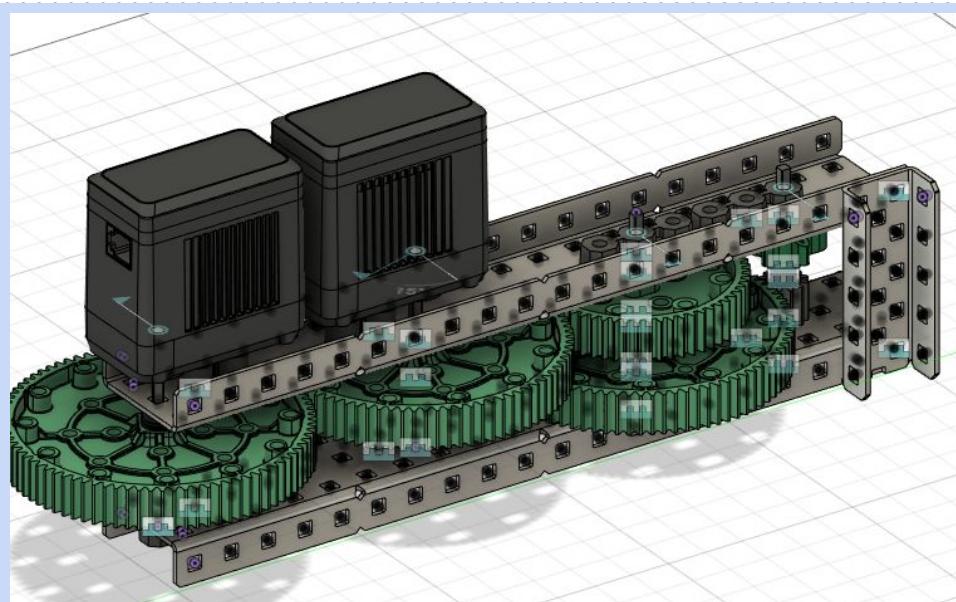
Flywheel V4.3 Sketch



Plan: New Sketch of the Flywheel Aimed for 1960RPM with a 1lb flywheel

The sketch above is a basic design for a new flywheel gearbox . The main idea behind of the design is that it focuses on increasing the flywheels mass for momentum to allow more discs to be shot per second.

Flywheel V4.3 CAD

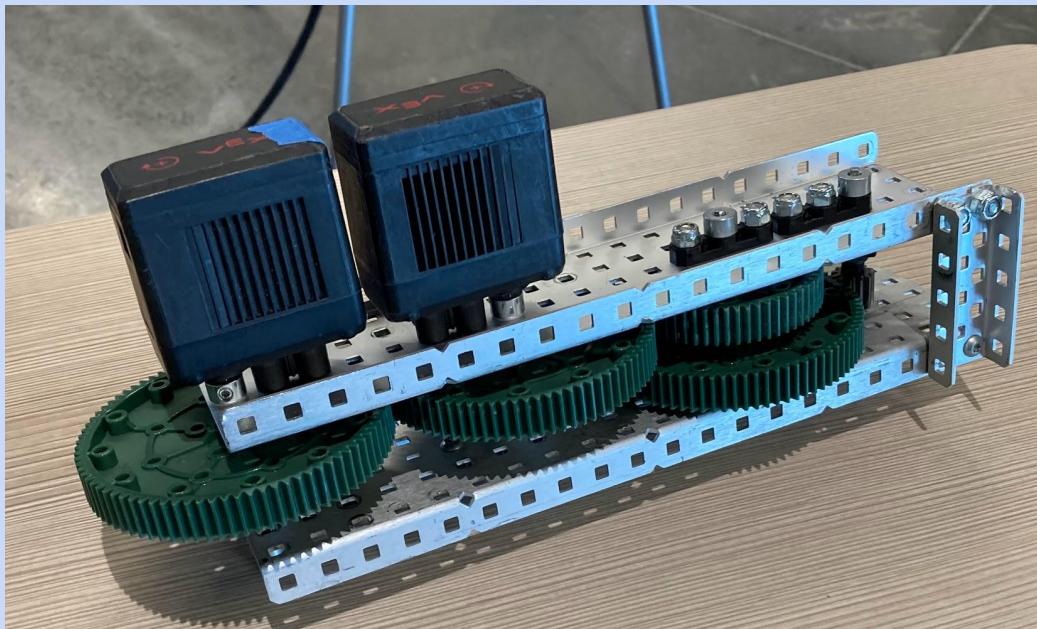


Design Process Page 10

Plan: Cad'd the Design so I can create the gearbox faster

I made the CAD for 2 reasons. One is to figure out the correct dimensions and materials, Two is because of a time constraint on getting materials. I am planning to build this during lunch and free periods so in order to complete this task I need to get all the materials during lunch because the Robotics room has classes which I'm not allowed to disturb.

Flywheel V4.3



Problem: Our club doesn't have any 20 long 3-wide C-Channels

Solution: Use 20 long 5-wide C-Channels Instead

Change: Build the Cad Design for the Flywheel.

I built the new gearbox for the flywheel. The building process went really smooth other than the fact of spacing which I forgot to include in the cad. Another issue which is the replacement of the 3-wide c channel to 5-wide c channel. This was a necessary change because at the Time I couldn't find any 20 hole 3-wide c-channels and I couldn't modify things during my lunch period.

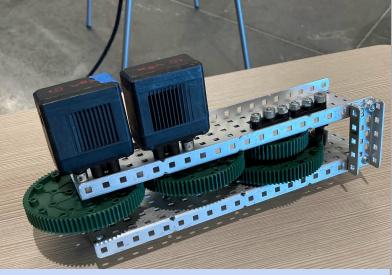
Pre-Meeting #50

Name: Ethan	Previous Task: Work on the turret	Current Task: Attach the flywheel	Task Deadline 12/15/22	Secondary Task
Name: Tyler	Previous Task: Create a new odom wheel design	Current Task Attach the flywheel	Task Deadline 12/15/22	Secondary Task
Name: Aidan	Previous Task: Design the flywheel	Current Task: Build flywheel Gearbox	Task Deadline 12/14/22	Secondary Task
Name: Josh ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #50

Name: Ethan	Meeting Tasks: Attach the flywheel	Tasks Completion: 90% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Attach the flywheel	Tasks Completion: 90% Time Worked: 2 hours	Next Tasks:
Name: Aidan	Meeting Tasks: Build flywheel Gearbox	Tasks Completion: 100% Time Worked:3 hours	Next Tasks:
Name: JOSH ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #50 Details

Name: Aidan	Task: Build Flywheel Gearbox	Details: After Cading the new flywheel gearbox I built it over my free periods. As I won't be able to make it to Robotics practice today	
Name: Ethan and Tyler	Task: Attach the Flywheel Gearbox	Details: Attaching Aidans Flywheel as well as adding it to the ramp	No image

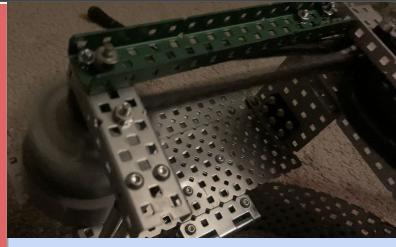
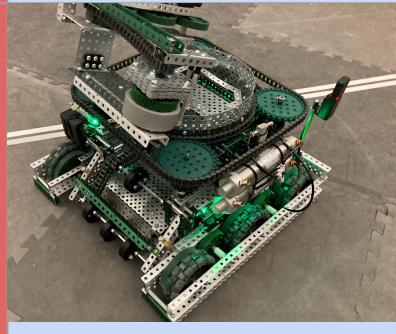
Pre-Meeting #51

Name: Aidan	Previous Task: Build the flywheel gearbox	Current Task: Build the Flywheel	Task Deadline 12/15/22	Secondary Task
Name: Ethan	Previous Task: Attach the flywheel	Current Task Build the flywheel	Task Deadline 12/15/22	Secondary Task
Name: Tyler	Previous Task: Attach the flywheel	Current Task: Build the Flywheel	Task Deadline 12/15/22	Secondary Task
Name: Josh ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #51

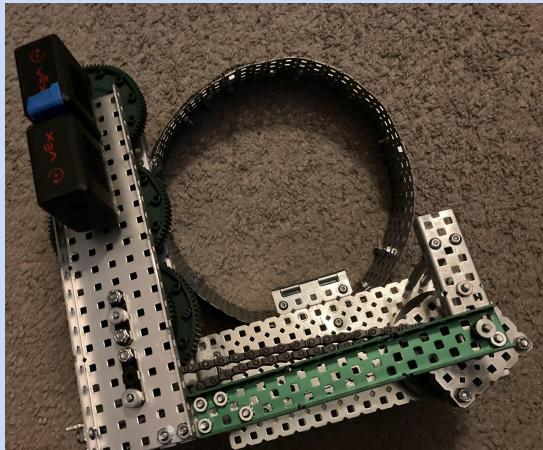
Name: Aidan	Meeting Tasks: Build the Flywheel	Tasks Completion: 80% Time Worked: 3 hours	Next Tasks:
Name: Ethan	Meeting Tasks: 1, Build the Flywheel 2, Work on the intake	Tasks Completion: 65% 42% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Build the Flywheel	Tasks Completion: 65% Time Worked: 2 hours	Next Tasks:
Name: JOSH ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #51 Details

Name: Aidan, and Tyler	Task: Build the Flywheel	Details: We were putting the finishing touches on the Flywheel so it can fire.	
Name: Ethan	Task: Work on the intake	Details: Almost complete the intake so it reaches the turret.	

Design Process Page 11

Flywheel V5



Change: Finished new Flywheel with less speed but more mass for more force

After attaching the gearbox and using our old weighted flywheel and ramp the flywheel is finally complete. So far I haven't been able to launch shooting tests but it has spun up to half speed. Its max speed is 1960rpm, and the flywheels mass is 19.1 oz. I measured the weight specifically using a food scale because regular scales aren't meant to measure in oz rather in pounds.

Problem: The Flywheel Doesn't fully spin up for one motor.

This problem may be apart of a larger set of problems but hopefully it doesn't impact us too much in the future. I'm thinking the problem is that 1 of the motors create too much fiction when idle.

Solution: When Testing use both motors by using the flywheel code.

This is a temporary solution for now but if problems get worse another redesign may be needed. When Testing since the turret isn't fully ready yet I'm going to be unplugging the chassis motors and plugging one motor to port 2 and one motor to port 3 or Bottom left and top right. One of the flywheel motors needs to be reversed and the right side of the chassis is reversed.

Pre-Meeting #52

Name: Aidan	Previous Task: Build the Flywheel	Current Task: Fix the flywheel	Task Deadline 1/7/22	Secondary Task: redo odometry wheel
Name: Ethan	Previous Task: Work on the intake	Current Task Redesign Intake	Task Deadline 1/7/22	Secondary Task: Finalize turret
Name: Josh ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task
Name: Tyler ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #52

Name: Aidan	Meeting Tasks: Fix the flywheel	Tasks Completion: 60% Time Worked: 5 hours	Next Tasks: Finish the flywheel
Name: Ethan	Meeting Tasks: Redesign Intake	Tasks Completion: 100% Time Worked: 4 hours	Next Tasks: Create the Indexer
Name: JOSH ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:
Name: Tyler ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #52 Details

Name: Aidan

Task: Fix the Flywheel

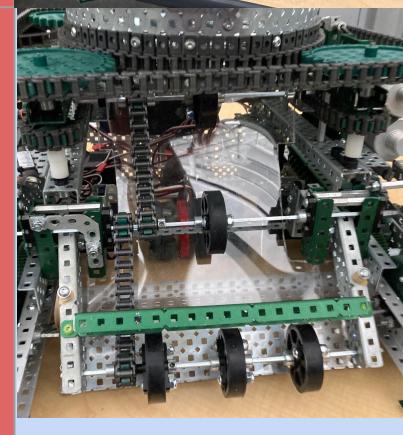
Details: There was some issues with the flywheel like the friction being too high and the chain was getting hit constantly



Name: Ethan

Task: Redesign Intake

Details: Redesigned the intake to use a polycarb ramp which is simpler than our old intake system.



Design Process Page 12

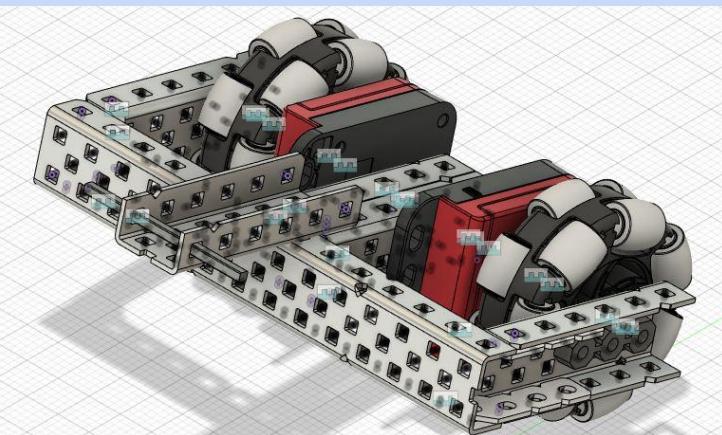
Odometry Wheels 2.3

Problem: The Odometry wheels didn't stay in one place when mounted

The Odometry wheels wiggle around too much and when they wiggle they stay in that bad position so it would yield inconsistent results when testing with.

Solution: Add more support for the wheel This is a temporary solution for now but if problems get worse another redesign may be needed. When Testing since the turret isn't fully ready yet I'm going to be unplugging the chassis motors and plugging one motor to port 2 and one motor to port 3 or Bottom left and top right. One of the flywheel motors needs to be reversed and the right side of the chassis is reversed.

Odometry Wheels 2.3 CAD



Change: Created a New Design for the Odometry wheels for structure stability

Instead of using angle gussets to connect the odom wheels together I replace them with channels. Both wheels here are connected by a 14 hole c-channel which is then connected to the robot using a shaft joint with rubber band suspension

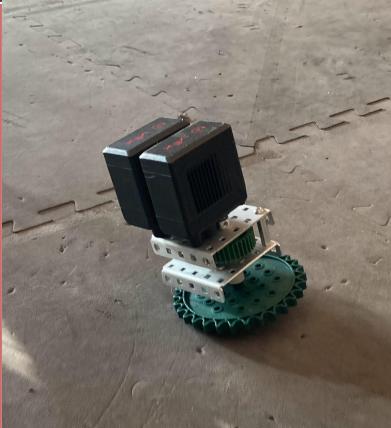
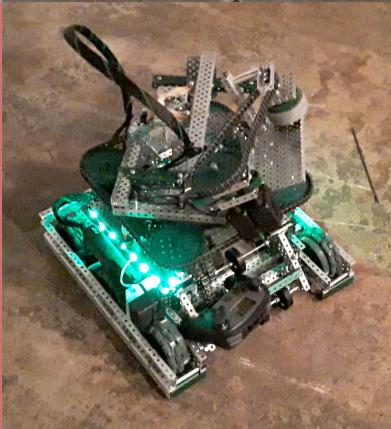
Pre-Meeting #53

Name: Aidan	Previous Task: Fix the flywheel	Current Task: Redesign the Flywheel	Task Deadline 1/7/22	Secondary Task: Begin building the new Odometry wheels
Name: Ethan	Previous Task: Redesign Intake	Current Task Create the indexer	Task Deadline 1/7/22	Secondary Task: Maintenance the robot
Name: Josh ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task
Name: Tyler ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #53

Name: Aidan	Meeting Tasks: Redesign Flywheel	Tasks Completion: 85% Time Worked: 3 hours	Next Tasks: Finish the flywheel
Name: Ethan	Meeting Tasks: Create Indexer, Maintenance the Robot	Tasks Completion: 100% 55% Time Worked: 4 hours	Next Tasks: Create the Odometry wheels
Name: JOSH ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:
Name: Tyler ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #53 Details

Name: Aidan	Task: Redesign Flywheel	Details: Redesigned the flywheel gearbox to be more compact and faster.	
Name: Ethan	Task: Maintenance the robot.	Details: Fix some things around the bot, Mainly removing unnecessary parts or replacing broken motors	
Name: Ethan	Task: Create the indexer	Details: Created a workable design for the indexer	Images

Design Process Page 13

Problem: Flywheel Was slow, big, and had too much friction

The flywheel was too slow to launch a disc to the goal

Problem: Flywheel had too much friction

The flywheel had too much friction due to the amount of connection points and its weight

Problem: Flywheel was too Big

The flywheel was really big for its size and almost went outside the 18" by 18". It also was really really heavy

Solution:

Create a more compact design which utilizes speed motors rather than Normal motors, Also instead of gearing the flywheel then connecting it to the chain drive I just use the chain drive to gear the flywheel.

Flywheel V6



Change: Created a new design to be compact, lightweight, and faster

I removed 3 connection points by instead of using a big bulky flywheel I used 2 speed motors geared together than gearing 6x using the sprocket gear.

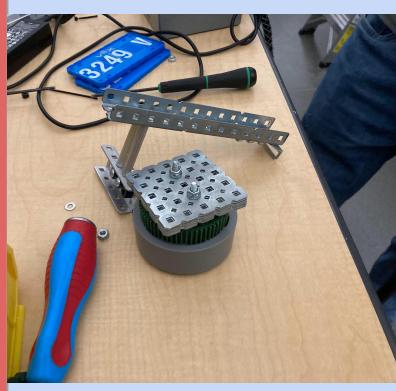
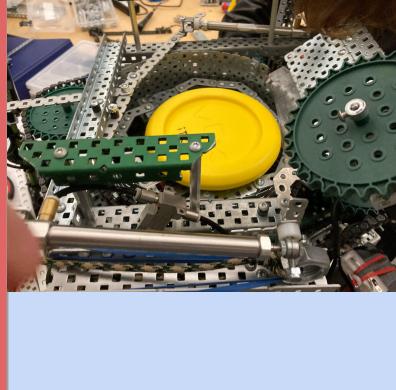
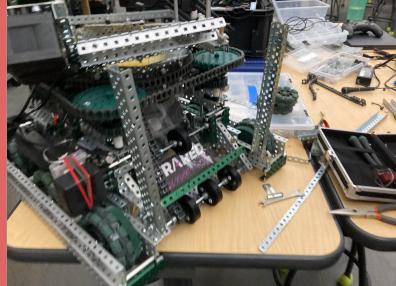
Pre-Meeting #54

Name:	Ethan	Previous Task: Create turret head	Current Task Build indexer:	Task Deadline right now	Secondary Task
Name:	Aidan	Previous Task: code flywheel and auto aim	Current Task Redesign flywheel	Task Deadline 1/6/22	Secondary Task
Name:	Tyler	Previous Task: Build the Flywheel	Current Task Roller mech	Task Deadline 1/6/22	Secondary Task
Name:	Josh ABSENT	Previous Task: expansion	Current Task	Task Deadline	Secondary Task

Post Meeting #54

Name: Aidan	Meeting Tasks: Redesign Flywheel	Tasks Completion: 48% Time Worked: 2 hours	Next Tasks:
Name: Ethan	Meeting Tasks: 1, Build Indexer 2, Fix Intake	Tasks Completion: 1, 65% 2, 100% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Build Roller Mech	Tasks Completion: 35% Time Worked: 2 hours	Next Tasks:
Name: Josh ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #54 Details

Name: Aidan	Task: Redesign Flywheel	Details: The Flywheels Flywheel portion was misaligned to the ramp. The redesign should fix these issues.	
Name:Ethan	Task: Build Indexer	Details: The indexer is made using 1by1's which push a small plate of polycarbonate.	
Name: Ethan	Task: Fix Intake	Details: fixed an issue with the intake where a screw was making it uneven preventing it from cleanly getting discs	
Name: Tyler	Task: Create Roller Mech	Details: Began creating the base of the roller mech. Then an Omniwheels will be attached to roll the rollers	

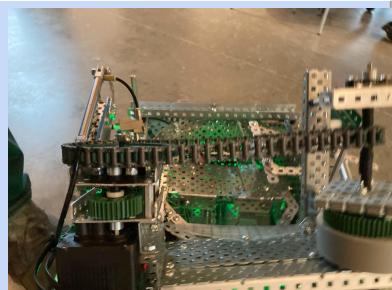
Pre-Meeting #55

Name:Aidan	Previous Task: Redesign Flywheel	Current Task Finish flywheel redesign	Task Deadline 1/5/22	Secondary Task

Post Meeting #55

Name: Aidan	Meeting Tasks: Redesign Flywheel	Tasks Completion: 78% Time Worked: 4 hours	Next Tasks: Finish flywheel

Meeting #55 Details

Name: Aidan	Task: Redesign Flywheel	Details: The flywheel is almost ready to shoot. The gearbox just needs to be nylocked to keep Alignment	

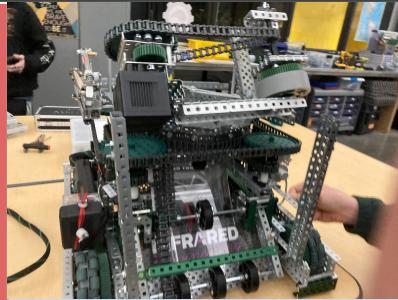
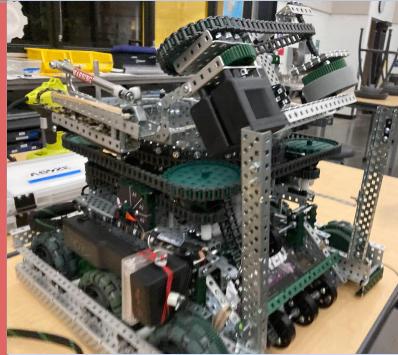
Pre-Meeting #56

Name:Ethan	Previous Task: Work on indexer	Current Task :finish indexer	Task Deadline Right Now!!!	Secondary Task Wire Management
Name: Tyler	Previous Task: roller mech	Current Task Finish roller mech	Task Deadline before tournament	Secondary Task driving practice (maybe)
Name:Aidan	Previous Task: develop flywheel	Current Task Finalize flywheel and mount for testing	Task Deadline Before thursday	Secondary Task code aimbot
Name: josh ABSENT	Previous Task: Work on Expansion	Current Task	Task Deadline	Secondary Task

Post Meeting #56

Name: Aidan	Meeting Tasks: Finalize flywheel and mount for testing	Tasks Completion: 90% Time Worked:	Next Tasks:
Name: Ethan	Meeting Tasks: finish indexer	Tasks Completion: 65% Time Worked:	Next Tasks:
Name: Tyler	Meeting Tasks: Finish roller mech	Tasks Completion: 60% Time Worked:	Next Tasks:
Name: Josh	Meeting Tasks: ABSENT	Tasks Completion: Time Worked:	Next Tasks:

Meeting #56 Details

Name: Aidan	Task: Finalize Flywheel	Details: I fixed the alignment of the flywheel by better securing the top part of the flywheel itself	
Name:Ethan	Task: Finish Indexer	Details:Added a second 1x1 to make the movement more linear instead of the end moving around	
Name: Tyler	Task: Finish Roller mech	Details: Added shaft collars then when testing with size constraints the roller mech didn't fit	

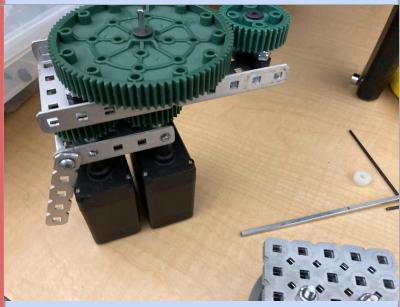
Pre-Meeting #57

Name: Ethan	Previous Task: finish indexer	Current Task: Create rake for wires, fix intake, fix flywheel	Task Deadline Right now!!!	Secondary Task Lubricate robot
Name: Aidan	Previous Task: Fix flywheel	Current Task: Work on code	Task Deadline 1/6/22	Secondary Task Work on flywheel
Name: Tyler	Previous Task: Work on Roller	Current Task: Work on Roller	Task Deadline Right Now!!!	Secondary Task
Josh has Left the Team due to Inactivity				

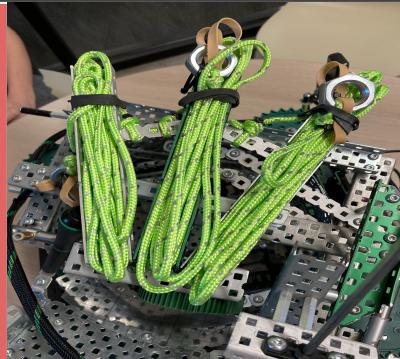
Post Meeting #57

Name: Aidan	Meeting Tasks: 1, Code the Robot 2, work on flywheel	Tasks Completion: 1, N/A 2, 40% Time Worked: 2 hours	Next Tasks: Run Code tests on the Flywheel
Name: Ethan	Meeting Tasks: 1, Grind down Intake plate 2, Finalize the indexer 3, Work on Expansion	Tasks Completion: 1, 100% 2, 100% 3, 85% Time Worked:	Next Tasks: Put finishing touches on the robot
Name: Tyler	Meeting Tasks: Finish Roller Mechanism	Tasks Completion: 100% Time Worked: 2 Hours	Next Tasks: Drive the robot at the competition

Meeting #57 Details

Name: Aidan	Task: Code the Robot	Details: overviewed the code and made some last minute changes. I was originally going to test but we ran out of time.	N/A
Name:Aidan	Task: Work on flywheel	Details: The current flywheel has a lot of friction so from a suggestion I received from Members 7121B and 4154V on discord.	
Name: Ethan	Task: Grind down Intake plate	Details: Dremoled down a metal plate which supports the bottom part of the polycarb for the intake so it doesn't catch on the disc.	No Image
Name: Ethan	Task: Finalize the indexer	Details: Securing the 1by1 so it doesn't rattle around when being used	No Image

Meeting #57 Details

Name:Ethan	Task: Work on Expansion	Details: Created a new expansion to cover more tiles for the competition	
Name: Tyler	Task: Finish Roller Mech	Details: Finished the roller mech but wasn't able to use it due to it colliding with the flywheel	No Image

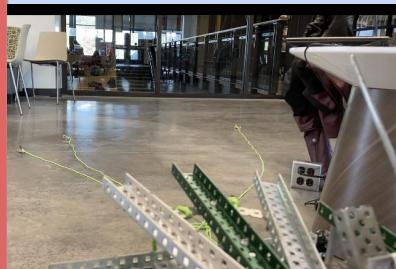
Pre-Meeting #58

Name: Aidan	Previous Task: Code the Robot	Current Task: Fix the Flywheel	Task Deadline Right now!!!	Secondary Task Code Testing
Name: Ethan	Previous Task: Work on Expansion	Current Task: Work on expansion	Task Deadline 1/6/22	Secondary Task Work on flywheel
Name: Tyler ABSENT	Previous Task: Work on Roller	Current Task:	Task Deadline	Secondary Task

Post Meeting #58

Name: Aidan	Meeting Tasks: 1, Work on Flywheel 2, Code the robot	Tasks Completion: 1, 100% 2, 30% Time Worked: 3 hours	Next Tasks:
Name: Ethan	Meeting Tasks: 1, Run Expansion Tests	Tasks Completion: 50% Time Worked: 30 minutes	Next Tasks:
Name: Tyler ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #58 Details

Name: Aidan	Task: Code the Robot	Details: Ran odom code and PID code with both failing so a new code will be needed after the comp	N/A
Name:Aidan	Task: Work on flywheel	Helped fix an alignment issue where the flywheel wasn't fully aligned with the flywheel ramp.	
Name: Ethan	Task: Run Expansion Tests	Ran stationary expansion tests to get the robot to consistently fire. Didn't get consistent within time	

December Monthly Update

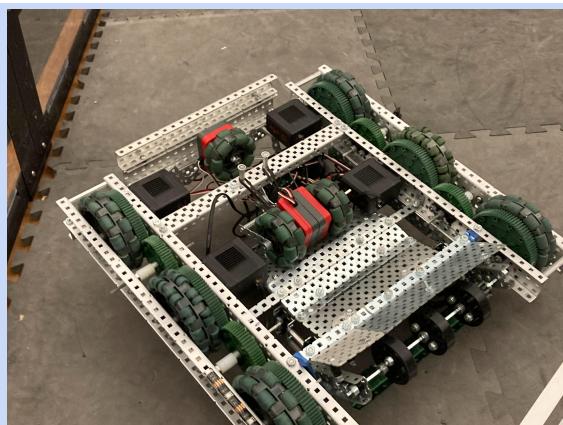
Overview

Basic Summary of the Month

The month of December has been mostly a recovery month. We fixed a majority of our issues we encountered at the November 19th tournament. Overall I feel like we didn't do much but we did only have half the time. The bot by the end of month and heading into new years could've been a lot better.

This month has also been the most rough for the notebook. There isn't any notebook award at the comp so we have been far more focused on the code and building without much leftover time for Notebook. During the month of January we will improve the notebooks overall quality and transparency.

Beginning Of the Month



Beginning of the Month Robot: Rehab Robot

This month was a big recovery bot. At the end of November we removed everything except the chassis and began with a almost clean slate. The biggest addition is the odometry wheels integrated directly into the chassis.

Beginning of the Month Code: Idled Progress

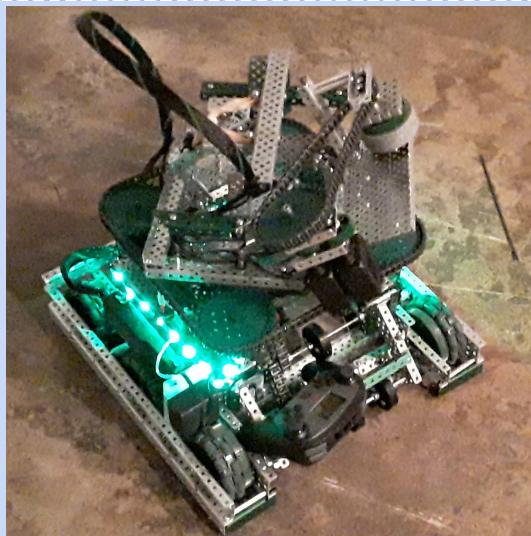
The code at this point remained relatively the same as throughout November. We focused more on the building portion so code wasn't the focus.

December Monthly Update Continued

Beginning of the Month Notebook: Good Recovery

The notebook is the biggest thing that was improved with the new format and style. It did leave some blank space but I feel like that was expected. It's fine to have blank space as long as you're being honest with time frames.

End of Month



End of Month Robot: Almost there

Other than a couple issues the robot was almost ready to compete. If we didn't have winter break we would've had a completely finished robot.

End of Month Code: Continued Stagnation

Not much testing at all since we simply didn't have the time for it. I really hope to actually test the code next month

End of Month Notebook: Really Messy

I will be honest the notebook ended of really messy this month. Like I said we really needed to build so the notebook wasn't a huge priority. I really hope to do more notebook next month for our next Tournament on February.

Pre-Meeting #59

Name: Aidan	Previous Task: Code the robot	Current Task: Do last minute coding	Task Deadline: Before our first round	Secondary Task Scout out teams
Name: Ethan	Previous Task: Work on Expansion	Current Task: Do last minute changes and repairs	Task Deadline: Before our First round	Secondary Task Get turret to work
Name: Tyler	Previous Task: Work on Roller mech	Current Task: Drive the robot during the comp	Task Deadline: During the Comp	Secondary Task Help ethan with build issues

Post Meeting #59

Name: Aidan	Meeting Tasks: 1,Do last minute coding 2,Run last minute odom tests 3,Get the flywheel working 4,Code the Flywheel and Intake manually 5, Scouting Other Teams 6. Match Repairs at Comp 7.Team Discussion	Tasks Completion: 1, N/A 2, N/A 3, 100% 4, 100% 5, 100% 6. 100% 7, 100% Time Worked: 10 hours	Next Tasks: Get Odometry Working
Name: Ethan	Meeting Tasks: 1,Last minute changes 2,Take expansion off 3,Made indexer worked 4,Control Intake and Flywheel 5, Maintenance 6,Team Discussion	Tasks Completion: 1, 75% 2, 100% 3, 100% 4, 100% 5, 100% 6, 100% Time Worked: 10 hours	Next Tasks: Get turret Working
Name: Tyler	Meeting Tasks: 1, Drive the Robot 2, Help with Repairs 3, Attach Roller Mech 4, Maintenance 5, Scouting 6, Team Discussion	Tasks Completion: 1, 100% 2, 75% 3, 0% 4, 100% 5, 100% 6, 100% Time Worked: 10 hours	Next Tasks: Redesign Robot

Meeting #59 Details

Name: Aidan	Task: Do last Minute Coding	Details: On the car ride to the tournament I wanted to try and get the code working to see if it works. The odometry wheels were messed up so it didn't matter	No images for this page
Name: Aidan	Task: Run Last minute Odometry Tests	Details: Due to a wiring issue Ethan plugged the odometry wheels into the extender which has a delay compared to being wired to the brain which caused an inaccurate reading. This issue was only solved after the tournament	
Name: Aidan	Task: Get the Flywheel Working	Details: The flywheel was ready but a bit unfinished. It was loud but I got it working and we were able to launch discs a solid 3-5 feet	
Name: Aidan	Task: Code the Flywheel and intake Manually	Details: After getting the flywheel working I had to give ethan a controller to use it because we had no odometry. Luckily enough its range was short enough so it was easy to know where to fit and the possibility of overfiring was impossible	

Meeting #59 Details

Name: Ethan	Task: Last minute changes	Details: We had to make some last minute tweaks to get it all working. Such as moving the wiring so as to not leave expansion	No image
Name: Ethan	Task: Take Expansion Off	Details: In our first two rounds the string kept falling off and got us an early expansion so we had to take it off to avoid getting Disqualified	
Name: Ethan	Task: Get Indexer working	Details: As we were getting the flywheel ready we needed to adjust the indexer and finalize it to get it working for the comp.	No Image
Name: Ethan	Task: Control intake and flywheel	Details: When the flywheel and intake was readied it was decided that we cannot overload tyler with controlling so I had to help him with the upper parts of the robot.	

Meeting #59 Details

Name: Tyler	Task: Drive Robot	Details: As we are at a tournament we have to drive the robot. We had no rounds as our robot had multiple difficulties.	Images
Name: Tyler	Task: Help with Repairs	Details: As Aidan was scouting. Tyler helped Ethan with last minute repairs that Ethan didn't have the time to do.	No image
Name: Tyler	Task: Attach Roller Mech	Details: The roller mech was going to be attached but we were unable to do so as we forgot the roller mech at our school before leaving for the tournament.	No image
Name: Ethan and Tyler	Task: Run Maintenance	Details: Between each match we had to run maintenance such as fill pneumatics, change batteries, and charge controllers.	N/A

Meeting #59 Details

Name: Aidan and Tyler	Task: Scout other Teams	Details: Scouting other teams was important to getting us an alliance due to how bad we did. We got 1460M to alliance which was the 10th seed	N/A
Name: Aidan	Task: Match Repairs	Details: Right before our last round on the top left of the chassis a C-channel came loose and we had to fix it or else it was going to be caught in the drive train and cause us to not be able to move properly.	N/A
Name: Aidan, Ethan and Tyler	Task: Team Discussion	Details: After our rounds we had a team discussion essentially pointing out what went wrong and how we need to address this.	N/A

Team Reflection

The Tournament was a hard loss so we reflected on why we did bad.

This tournament was our worst loss of 0-7 on the record. In general we did really bad and after our matches we decided to have a group discussion on what we did wrong and where we could improve for our next tournament. We don't have much time so we need to make reforms now better than later.

We Failed due to a Lack of Testing

For our first 2 rounds we were stationary due to an early expansion. For our next 3 rounds we were unable to move forward due our intake caving in so it wedged into the field. Which even DQed us from one of our rounds. Our last qualification round we lost due to a disc jam in the intake. These problems would have all been solved if we actually tested the bot.

Why we progress so far at a tournament.

At tournaments we test the robot which helps us make great improvements to the bot. We also do a reflection afterwards so we learn from our mistakes and know how to make our next move. We are deciding to do this every week from now on when all teammates are present on Thursdays at the last 30 minutes of practice.

Planning for the Future

Now we are at 3 people which is 2 less than what we started with. We need to use our time more wisely. From now until our next tournament we aren't going to take apart everything that worked which includes the Chassis, and Intake. If needed we would fix issues that may harm performance in the game. The focus now is getting Rollers, Expansion, and High goal scoring completed then work on coding.

Pre-Meeting #60

Name: Aidan	Previous Task: Code the robot	Current Task: Fix the Odometry wheels	Task Deadline 1/20/22	Secondary Task
Name: Ethan	Previous Task: Work on Expansion	Current Task: Disassemble bot	Task Deadline: N/A	Secondary Task
Name: Tyler ABSENT	Previous Task: Drive the robot during the comp	Current Task:	Task Deadline:	Secondary Task

Post Meeting #60

Name: Aidan	Meeting Tasks: Fix odometry wheels	Tasks Completion: 50% Time Worked: 1 hour	Next Tasks:
Name: Ethan	Meeting Tasks: Last minute repairs Take expansion off	Tasks Completion: 100% Time Worked: 30 minutes	Next Tasks:
Name: Tyler ABSENT	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #60 Details

Name: Aidan	Task: Work on Odometry Wheels	Details: Took apart and began redesigning current odometry wheels to fix slanting issue where the odometry wheels weren't aligned due to a screw causing one to be angled	NO IMAGE
Name: Ethan	Task: Disassemble Robot	Details: Disassembled the Robot in order to fix the Odometry Wheels	NO IMAGE

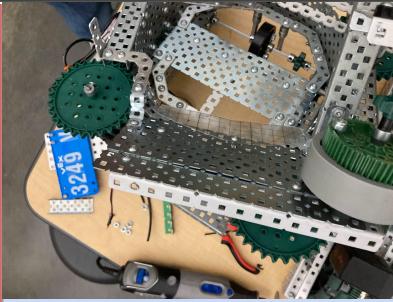
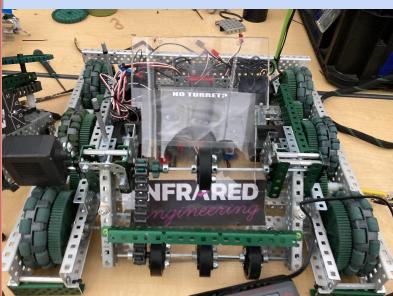
Pre-Meeting #61

Name:Ethan	Previous Task: Disassemble Robot	Current Task Modify motor cartridge, get turret working, clean up pneumatics, wiring solutions, remount turret with c chans, fix intake ramp	Task Deadline: before next tourney	Help rewire robot
Name:Aiden	Previous Task: Fix Odometry Wheels	Current Task Make odometry wheels work + wiring solutions, begin code on aimbot	Task Deadline	Secondary Task Test odom wheels
Name:Tyler	Previous Task:N/A	Current Task Research and create passive roller mech + get turret working	Task Deadline	Secondary Task

Post Meeting #61

Name: Aidan	Meeting Tasks: Uninstall Flywheel Create Direct Motor	Tasks Completion: 1, 100% 2, 100% Time Worked: 2 hours	Next Tasks:
Name: Ethan	Meeting Tasks: Work on Expansion	Tasks Completion: 75% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Work on Odometry Wheels	Tasks Completion: 100% Time Worked: 2 hours	Next Tasks:

Meeting #61 Details

Name: Aidan	Task: Uninstall Flywheel	Details: I uninstalled the new flywheel as we are getting new parts soon and it will make the current flywheel absolute with the old parts	
Name: Aidan	Task: Create Cartridgeless Motor	Details: As we are getting new parts I have a new design which utilizes the raw RPM of the motor. The design uses a insert (which is vex legal) to bypass adding a cartridge.	
Name: Ethan	Task: Work on Expansion	Details: Started working on a design which prevented the expansion from falling out mid match	
Name: Tyler	Task: Work on odometry wheels	Details: Made a new odometry wheels so they could be properly aligned.	

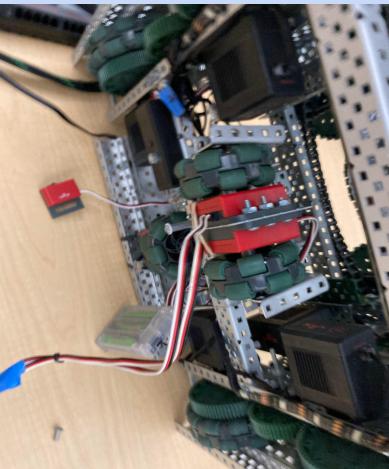
Pre-Meeting #62

Name:Aiden	Previous Task: Create Cartridgless motor	Current Task Create flywheel	Task Deadline 2/4/22	Secondary Task Test odom wheels
Name:Ethan	Previous Task: N/A	Current Task Work on Expansion	Task Deadline: 2/4/22	Help rewire robot
Name:Tyler	Previous Task:N/A	Current Task Work On odometry wheels	Task Deadline 2/4/22	Secondary Task Work on Roller Mech

Post Meeting #62

Name: Aidan	Meeting Tasks: Create Flywheel	Tasks Completion: 20% Time Worked: 2 hours	Next Tasks:
Name: Ethan	Meeting Tasks: Work on Expansion Work on Direct Motor	Tasks Completion: 80% 100% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Worked odometry wheels	Tasks Completion: 100% Time Worked: 2 hours	Next Tasks:

Meeting #62 Details

Name: Aidan	Task: Create Flywheel	Details: I began creating the framework of our new flywheel. The new flywheel will include new parts which we get tomorrow.	
Name: Ethan	Task: Work on Expansion	Details: Ran some tests to see how we can quickly coil the expansion. Also ran some tests with it	
Name: Ethan	Task: Work on Direct Motor	Details: made a second direct motor so incase we are having back to back matches we can quickly take off one motor and put another one on	N/A
Name: Tyler	Task: Work on Odometry wheels	Details: Fixed an alignment issue with the odometry wheels which caused one to be crooked due to a misplaced screw where the encoders are connected	

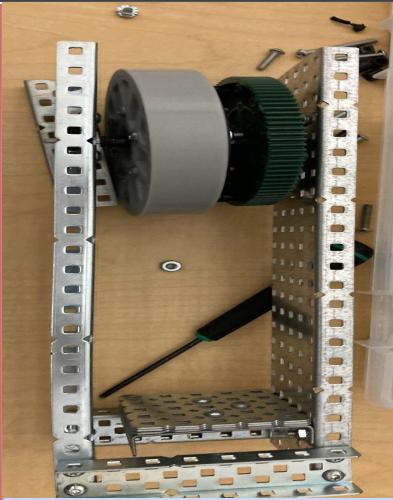
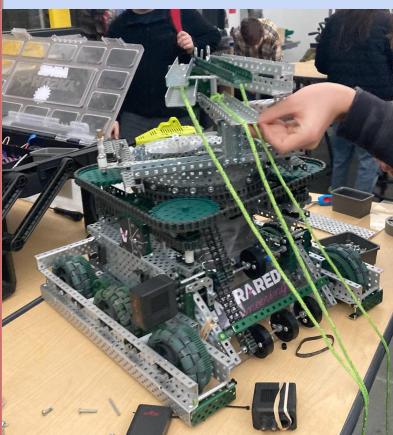
Pre-Meeting #63

Name:Aidan	Previous Task: Create flywheel	Current Task Continue Creating Flywheel	Task Deadline 1/20/22	Secondary Task: Work on Code
Name:Ethan	Previous Task: Work on Expansion	Current Task Work on Expansion	Task Deadline: 2/4/22	Secondary Task: Work on Indexer
Name:Tyler ABSENT	Previous Task:	Current Task	Task Deadline	Secondary Task

Post Meeting #63

Name: Aidan	Meeting Tasks: Create Flywheel	Tasks Completion: 75% Time Worked: 2 hours	Next Tasks: Finish Flywheel
Name: Ethan	Meeting Tasks: Work on Expansion Work on Indexer	Tasks Completion: 1, 90% 2, 100% Time Worked: 2 hours	Next Tasks: Finish Expansion
Name: Tyler Absent	Meeting Tasks:	Tasks Completion: Time Worked:	Next Tasks:

Meeting #63 Details

Name: Aidan	Task: Continue building the Flywheel	Details: Today our club got new parts for flywheels such as ball bearings, Flex Wheels, and flywheel weights, I used these new parts in tandem with the frame I made yesterday.	 A photograph showing a grey flywheel attached to a green motor, mounted on a metal frame. The frame is part of a larger structure, likely the flywheel assembly.
Name: Ethan	Task: Work on Expansion	Details: the expansion has been framed and fitted to it's pneumatic cylinder. 95% complete	 A photograph of a complex metal frame structure being worked on. It appears to be a frame for a pneumatic cylinder, with various holes and a green support arm visible.
Name: Ethan	Task: Work on Indexer	Details: indexer has been tuned 100% complete	 A photograph of a completed indexer mechanism. It is a green and black assembly with a circular component labeled "INDEXER". A hand is visible pointing at the mechanism.

Coding Page 5

Code Update

I lost the code :/

The USB I have been coding on stopped worked so I no longer have the code. The last backup I made was in the beginning of december so I didn't lose too much but I need to start testing tomorrow for my robot in order for it to be ready for January 20th which my club will running a 20+ team scrimmage with 3 other schools.

I decided to Switch the Code to PROS

The last backup of the code was on January 5th which isn't that far back at all. But I've wanted to switch to PROS for a while now. I can back the code up on Github and I can input libraries without much hassle. The biggest issue now is learning how to use PROS in such a short amount of time. As of now all code previously put into the notebook will no longer be used except for the calculations.

New Flywheel RPM Conversion

$$w_{final} = \frac{v \cdot 60}{2 \cdot \pi \cdot r_{wheel} \cdot \sqrt{\frac{I_{object}}{I_{wheel} + I_{wheelDelta}}}}$$

New Flywheel Velocity to RPM conversion made based off of the Torque Formula.

I got a new formula which is far more accurate than the previous formula used. I decided to change the formula when trying to calculate the amount of RPM needed to launch a disc

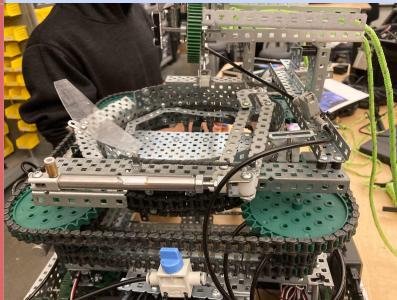
Pre-Meeting #64

Name:Aiden	Previous Task: Create flywheel	Current Task Continue Creating Flywheel	Task Deadline 1/20/22	Secondary Task: Work on Code
Name:Ethan	Previous Task: Work on Expansion	Current Task Create Wire Mangement	Task Deadline: 2/4/22	Secondary Task: Work on Expansion
Name:Tyler	Previous Task: Work On odometry wheels	Current Task Work on Roller Mech	Task Deadline 1/20/22	Secondary Task

Post Meeting #64

Name: Aidan	Meeting Tasks: Create Flywheel	Tasks Completion: 85% Time Worked: 2 hours	Next Tasks: Finish Flywheel
Name: Ethan	Meeting Tasks: Work on Wiring	Tasks Completion: 1, 40% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Work on Roller mech	Tasks Completion: 25% Time Worked:	Next Tasks:

Meeting #64 Details

Name: Aidan	Task: Continue building the Flywheel	Details: Today I spent the entire day Aligning the flywheel and gettin the spacing for the shaft right by using washers and spacers. As well as putting the motor on the Flywheel	
Name: Ethan	Task: Work on Wiring	Details: started by adding the framework of the new wiring setup to allow for a larger turn radius on the turret	
Name: Tyler	Task: Begin working on new roller mechanism	Details: Began creating a new roller design thats on the turret due to our previous one not fitting with out turret	

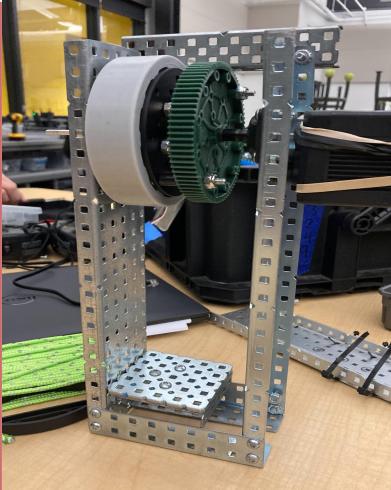
Pre-Meeting #65

Name:	Previous Task:	Current Task:	Task Deadline	Secondary Task
Aidan	Work on Flywheel	Continue working on flywheel		
Ethan	Work on Wiring	Wire the Robot		
Tyler	Work on roller mech	Work on Roller Mech		

Post Meeting #65

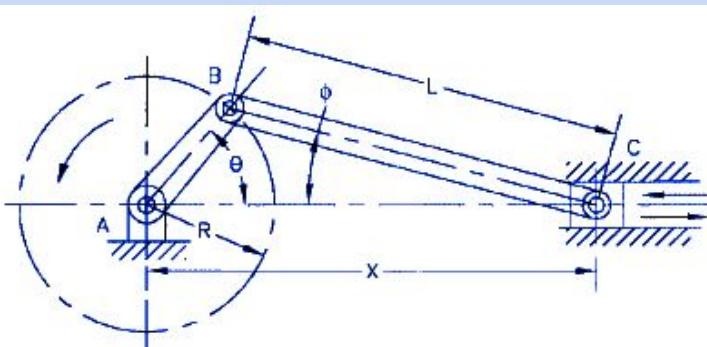
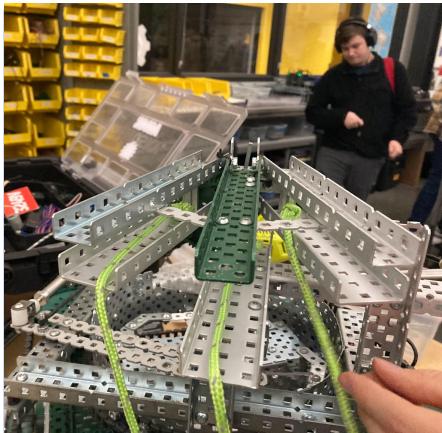
Name: Aidan	Meeting Tasks: Work on the Flywheel	Tasks Completion: 95% Time Worked: 2 hours	Next Tasks:
Name: Ethan	Meeting Tasks: Work on Wiring	Tasks Completion: 80% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Work on Roller Mechanism	Tasks Completion: 45% Time Worked: 1 hours	Next Tasks:

Meeting #65 Details

Name: Aidan	Task: Work on the Flywheel	Details: Fixed the spacing with the flywheel and fixed the alignment of the flywheel where the C-channels connecting both 5-channels were angled at 5 degrees.	
Name: Ethan	Task: Work on Wiring	Details: Worked on wiring the turret and started creating wiring areas for the turret.	
Name: Tyler	Task: Work on Roller Mechanism	Details: Mainly focused on making a mounting bracket and created a new design for the roller	

Design Process Page 14

Expansion Version 2



Changes

Complete reboot of original string mech

Problem: the old expansion was unreliable and covered 3 tiles at most

Solution: design a new 3 way expansion

Specifications

We decided to reboot the original expansion mech, as we were only able to score 3 tiles at most. We are using a pin system to launch 3 pneumatic bolts under load, followed by 3 8 feet strings. On a good launch we can score up to 76 points, on a fail, we get an average of 26 to 33 points. The mechanism itself uses a lever attached to a pneumatic to rotate a gear and pull out a pin which keeps all the bolts loaded. The bolts travel down a track, then across the field.

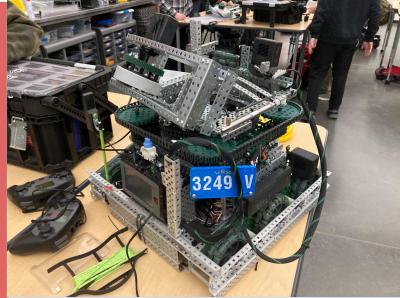
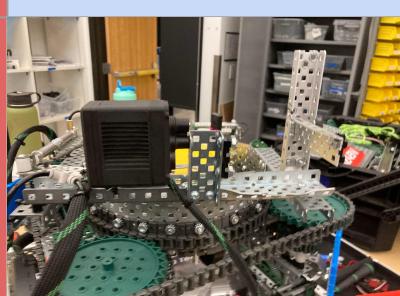
Pre-Meeting #66

Name:Aidan	Previous Task:Work on Flywheel	Current Task:Finalize Flywheel	Task Deadline 1/19/23	Secondary Task Work on Code
Name:Ethan	Previous Task:wire robot	Current Task:Install flywheel ramp + wire turret motors	Task Deadline 1/19/23	Secondary Task clean up chassis wires
Name: Tyler	Previous Task: Work on roller mech	Current Task: Work on roller mech	Task Deadline 1/20/22	Secondary Task

Post Meeting #66

Name: Aidan	Meeting Tasks: Finalize Flywheel	Tasks Completion: 100% Time Worked: 2 Hours	Next Tasks:
Name: Ethan	Meeting Tasks: 1, Work on Wiring 1, Work on Expansion	Tasks Completion: 100% 100% Time Worked: 2 Hours	Next Tasks:
Name: Tyler	Meeting Tasks: Work on Roller Mech	Tasks Completion: 55% Time Worked: 2 Hours	Next Tasks:

Meeting #66 Details

Name: Aidan	Task: Finalize Flywheel Mechanism	Details: After a lot of work the flywheel is now ready to launch discs. I got the compression right it and it launches far around 10-16ft	
Name: Ethan	Task: Worked on Wiring	Details: Was able to set the wiring up to the point where we can move the turret slightly while having functionality there.	
Name: Ethan	Task: Worked on Expansion	Details: Finished the expansion and was ready to mount for the Scrimages tomorrow.	
Name: Tyler	Task: Work on Roller Mech	Details: Installed motor and mounting area for Roller Mechanism. Bent C-channel to allow for wheelspace	

Pre-Meeting #67

Name:Aidan	Previous Task:Work on Flywheel	Current Task:Finalize Flywheel	Task Deadline 1/19/23	Secondary Task Work on Code
------------	--------------------------------	--------------------------------	-----------------------	-----------------------------

Post Meeting #67

Name: Aidan	Meeting Tasks: Mount the flywheel	Tasks Completion: 65% Time Worked: 3 hours	Next Tasks:
-------------	--------------------------------------	---	-------------

Meeting Details #67

Name: Aidan	Task: Mount the Flywheel	Details: Finally mounting the flywheel to the robot. I was only able to come up with a left side mounting method for the time constraint. I put a angle gusset then I a c-channel to further secure it all	
-------------	--------------------------	---	---

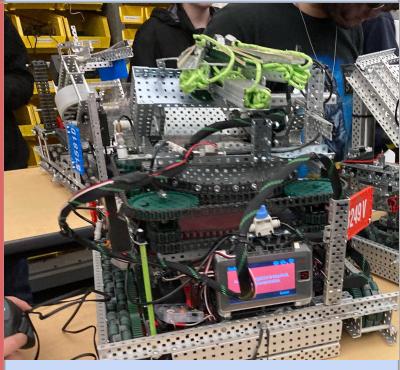
Pre-Meeting #68

Name:Aidan	Previous Task: Finalize Flywheel	Current Task: Code the bot	Task Deadline 1/20/23	Secondary Task Work on Odometry code
Name:Ethan	Previous Task: Wire the robot	Current Task: Attach the Flywheel	Task Deadline 1/19/23	Secondary Task clean up chassis wires
Name: Tyler	Previous Task: Work on roller mech	Current Task: Run Drive tests	Task Deadline 1/20/22	Secondary Task

Post Meeting #68

Name: Aidan	Meeting Tasks: Code the Robot	Tasks Completion: N/A Time Worked: 4 hours	Next Tasks:
Name: Ethan	Meeting Tasks: 1,Finish mounting the flywheel 2, Test Expansion 3, Drive the Robot	Tasks Completion: 1, 100% 2, 100% 3, 100% Time Worked: 4 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Drive the Robot	Tasks Completion: 100% Time Worked: 4 hours	Next Tasks:

Meeting #68 Details

Name: Aidan	Task: Code the Robot	Details: Due to just getting the turret working I modified the competition code to allow for turret control with the second controller	N/A
Name: Ethan	Task: Finish Mounting Flywheel	Details: Finishing Aidans job my mounting the left side of the flywheel	
Name: Ethan	Task: Test Expansion	Details: For one of our rounds our solenoid for the expansion broke so we just switched the indexer solenoid with the expansion for a round to get a 13 tile expansion	
Name: Ethan and Tyler	Task: Drive the Robot	Details: As this is a scrimmage event we ran dual controller for the bot when driving. We won 2 rounds and lost 1	

Engineering Notebook

3249V

Team Number

Infrared

Team Name

iTech Preparatory

School

1/23/23

Start Date

00/00/0000

End Date

2

of 2

Book #

NOTEBOOK #1

v1.0.8.29.22



Resources

students.vex.com

Engineering Resources, Information on Notebooks, Videos, VEX Library, Teams Resources, and Scholarships



mentors.vex.com

Team and Mentor Resources, Mentor Professional Development, VEX Mentor Community and more



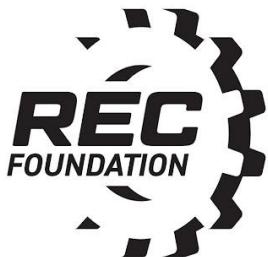
teams.vex.com

A Collection of Resources for Teams Provided by the REC Foundation



library.vex.com

Information on Building, Documentation, Troubleshooting, Coding, and other Educational Resources



About the REC Foundation

The REC Foundation's global mission is to provide educators with hands-on, student-led competition programs and educational resources to prepare future innovators for a diverse and inclusive STEM workforce. We see a future where all students design and innovate as part of a team, experience failure, persevere, and emerge confident in their ability to meet global challenges.

engineering.vex.com
notebooking.vex.com
coding.vex.com

Judging Rubric for Notebooks

vex.com
roboticseducation.org
[VRC 2022-2023 Game - Rules & Game Video](https://www.vex.com/vrc-2022-2023-game-rules-and-game-video)

Send suggestions and comments about these Digital Notebooks and Digital Parts to notebooks@vex.com



Table of Contents

Page	Linked Project Slides	Date
1	Notebook Statement	1/23/23
2	Notebook Statement Continued	1/23/23
3	Notebook Statement Continued +	1/23/23
4	Notebook Update	1/23/23
5	Pre-Meeting #69	1/24/23
6	Post Meeting #69	1/24/23
7	Meeting #69 Details	1/24/23
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		
29		
30		
31		
32		
33		
34		
35		
36		
37		
38		
39		
40		

Notebook Statement

Virtual Notebook Statement

In the Virtual Notebook we will take measures to keep it as traditional as possible to a physical notebook. Of course due to the nature of the genre some things will differ from the physical notebook. Some changes made out of convenience and some made out of necessity. Some changes are just clarification.. The list of changes are below

We will not be striking out minor mistakes when writing the notebook

Spelling mistakes in all formats are quite common. All of us have some form of auto correct which will correct for us so going back and correcting every single spelling mistake will be redundant and give us many headaches.

We will not be Crosshatching Empty Space

Cross hatching takes a lot of time and is extremely redundant. Cross Hatching on a virtual notebook isn't effective as person can delete cross hatches and put something there. If a page is left mostly empty then that means the page is done for that date and had to finish business on a previous page or the pages topic was short to complete.

We will not have a Person Sign off another person's work on each Page

This may be a big infraction to the rules, but this was done out of the fact that the format given simply didn't include the ability to do so

Some pages will be considered "Live" Pages will not conform to Regular dating, rather conform to individual dating for each entry.

Live pages will usually be pages about strategies. The date that will be on the page will be when the page was first created. To keep it traditional to what an engineering notebook stands for. Nothing will be deleted and every time something is added It will be dated. I have some of these pages on the previous notebook for strategy but I plan not to use them as much.

Notebook Statement Continued

Some Pages labeled with (OD) are not up to date with the current robot. Rather they're used to Notebook on old designs which haven't been notebook for some reason.

I will be making a lot of these pages really soon due to us needing to catch up. The designs shown won't be up to date but will contain dates to show when the mechanisms were originally created. A lot of us don't have much time to notebook especially me (Aidan) due to having other things such as college and schoolwork to worry about. So we cannot update the notebook about everything we are doing every single day or we wouldn't be asleep till 4 am. We're already stressed enough for being a 3 person team building a complicated turret so I am sorry if we aren't up to date on everything.

The Notebook will be explained in the 3rd person view for meeting notes and everything that includes the whole team, and in the 1st person view for pages with a singular author.

This isn't a change with any notebook I know of but this is something I wanted to clarify for going forward. It's much easier to explain how you came up for a design in the first person rather in the third person but it makes more sense when talking in the 3rd person on Multi person pages such as meeting notes.

On Multi person pages (Meeting notes, Team Reflections, etc) the Author will be the person who creates the page rather than everyone involved.

Like the previous statement this is something I just want to clarify for going forward.

The "Project" Area will be used to indicate what Project is being used. Not the title of the Page

This is something I wanted to clarify for the future not a change. This change is mostly so I can easily scan where the previous page was if I don't have a header.

Notebook Statement Continued +

Why I made the Virtual Statement now Rather than in the Previous notebook

Originally we didn't have a virtual notebook rather had a physical notebook. Around when we starting build a bot we realized it was simply more efficient to have a virtual notebook seeing 3249U at around 60-80 pages while we were at 20. As well it was harder for us to print images which we use on a daily basis in the notebook due to not having school printer access. So we switched to the virtual notebook. This switch happened over the course of 2 weeks then we were fully virtual.

At the time I never considered a notebook statement until after my tournament where I tried to win design and lost. So I decided to ask 3249U's Joseph for help. He mentioned doing a statement where he stated what will he will be doing differently between the physical notebook and his digital notebook. He also mentioned color coding which is why I made the new format.

After making the new format I didn't have a good place to put the statement. If I put it in the middle of the notebook it would be lost among the new formats or confused by the new format pages. If I put it near the end it would be strange to see a notebook statement so long into the notebook. When we decided to make a second virtual notebook, I decided it would be a good idea to start fresh with the statement right at the start.

Why we made a Second Virtual Notebook.

There's multiple reasons behind this. Mainly because it would take too long to go through the previous one and it was causing performance issues due to the size. Going onwards my virtual notebook will be limited at 350 Slides. I have to specifically say slides and not pages because table of contents adds some pages to the overall notebook which aren't counted for actual pages.

Notebook Update

Notebook two will be a bit different to start out with

At the start of notebook two will be a lot of OD pages so we can quickly catch up to the current design. This is just so we can maintain iterative design while still not editing old notebook pages. I will also not be readdressing old projects such as Design process, coding, etc.

Team Introductions

The team introductions for all current team members are on notebook pages 2-4 in notebook 1.

The Design Process

Design Process is the engineering process we use to create our robot. Full description on pages 12-13 in Notebook 1.

Our Goals

Our goal is to get the excellence award for iTech. The full page of our goals is on page #22

Page Formatting

The current iteration of the notebook format is on pages 222-224, 226, 228-229. We are using the second iteration of our Formatting for all future pages going forward unless a new format is necessary.

Going Forward

We will continue off of page 339 of the previous notebook. All projects will continue the numbering and will not reset back to #1 as this is meant to be a continuation of the previous notebook.

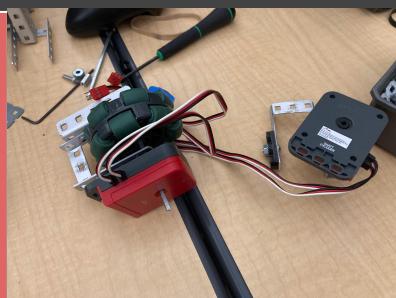
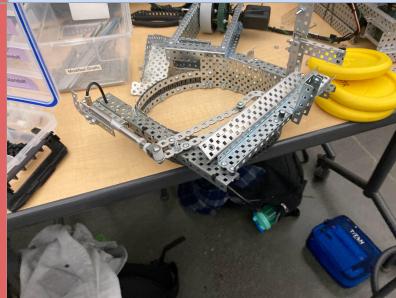
Pre-Meeting #69

Name: Aidan	Previous Task: Code the Robot	Current Task: Create New Odometry wheels with new parts	Task Deadline 1/24/23	Secondary Task Work on Flywheel
Name: Ethan	Previous Task: Drive the Robot	Current Task Fix Turret Wheel	Task Deadline 1/24/23	Secondary Task
Name: Tyler	Previous Task: Drive the Robot	Current Task Work on Roller mech	Task Deadline 1/24/23	Secondary Task

Post Meeting #69

Name: Aidan	Meeting Tasks: Create Odom Wheels	Tasks Completion: 30% Time Worked: 1 hour 30 minutes	Next Tasks:
Name: Ethan	Meeting Tasks: Fix turret wheel	Tasks Completion: 100% Time Worked: 1 hour 30 minutes	Next Tasks:
Name: Tyler	Meeting Tasks: Work on Notebook	Tasks Completion: N/A Time Worked: 1 hour 30 minutes	Next Tasks:

Meeting #69 Details

Name: Aidan	Task: Create Odom Wheels	Details: After Ethan cut the inners of a 3-wide c-channel Aidan used them as wheel casing for odometry wheels	
Name: Ethan	Task: Fix turret wheel	Details: Ethan took the turret off along with the flywheel and expansion to take off the bent metal near the flywheel ramp, replace it with 1 bys to allow for a better way of angling the flywheel so it doesn't conflict with the flywheel.	
Name: Tyler	Task: Work on notebook	Details: As we are trying to get all the old mechanisms in the notebook tyler worked on notebooking the old roller mechanisms	N/A

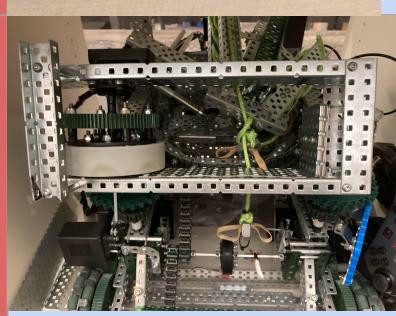
Pre-Meeting #70

Name: Aidan	Previous Task: Code the Robot	Current Task: Work on Odometry wheels	Task Deadline 2/4/23	Secondary Task Work on Flywheel
Name: Ethan	Previous Task: Drive the Robot	Current Task Mount flywheel	Task Deadline 2/4/23	Secondary Task
Name: Tyler	Previous Task: Drive the Robot	Current Task Test the robot	Task Deadline 2/4/23	Secondary Task

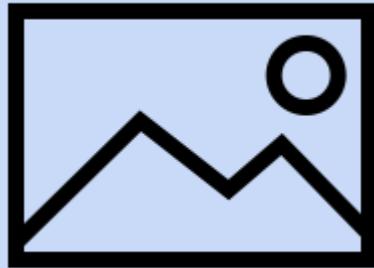
Post Meeting #70

Name: Aidan	Meeting Tasks: Build Odometry wheels	Tasks Completion: 50% Time Worked: 1 hour 30 minutes	Next Tasks: Finish Odometry wheels
Name: Ethan	Meeting Tasks: Mount Flywheel	Tasks Completion: 100% Time Worked: 1 hour 30 minutes	Next Tasks: Work on Wiring
Name: Tyler	Meeting Tasks: Test the robot	Tasks Completion: N/A Time Worked: 1 hour 30 minutes	Next Tasks: Build roller mech

Meeting #70 Details

Name: Aidan	Task: Build Odometry wheels	Details: Figured out the spacing and replicating the design a second but mirrored for a left wheel.	
Name: Ethan	Task: Remount Flywheel	Details: Fixed the flywheels and made it more secure than what it was previously	
Name: Tyler	Task: Tested the robot	Details: Ran some minor tests on the turret shooting to the highgoal.	N/A

Design Process - Robot testing



What to look for:

*The turret wheel and indexing ramp was recently redone, we are looking for smooth, seamless indexing

* Solved indexer ramp problem

* flywheel shoots full court at speed 100; whole idea of turret works.

* flywheel doesn't cause discs to wobble - should have been fixed by indexer change

Successes:

- Turret spins and indexes

*

Changelog:

*need to adjust intake

*

Pre-Meeting #71

Name: Aidan	Previous Task: Work on Odometry wheels	Current Task: Finish Odometry wheels	Task Deadline 2/4/23	Secondary Task Work on Code
Name: Ethan	Previous Task: Mount flywheel	Current Task Fix Intake	Task Deadline 2/4/23	Secondary Task
Name: Tyler	Previous Task: Drive the Robot	Current Task Test the robot	Task Deadline	Secondary Task

Post Meeting #71

Name: Aidan	Meeting Tasks: 1,Finish Odometry Wheels 2, Work on code	Tasks Completion: 1,100% 2,35% Time Worked: 3 hours	Next Tasks:
Name: Ethan	Meeting Tasks: Wire the Robot Fix Intake Ramp	Tasks Completion: 1, 2,100% Time Worked: 2 hours	Next Tasks:
Name: Tyler	Meeting Tasks: Test the Robot	Tasks Completion: N/A Time Worked: 2 hours	Next Tasks:

Meeting #71 Details

Name:	Task:	Details:	Images
Name: Aidan			
Name: Ethan			

Pre-Meeting #72

Name: Aidan	Previous Task:Work on Odometry wheels	Current Task: Create new flywheel	Task Deadline 2/4/23	Secondary Task Replace Intake Motor with Speed Motor
----------------	--	---	----------------------------	---

Post Meeting #72

Name: Aidan	Meeting Tasks: 1, Cut out holes for the flywheel 2, Create new flywheel with High Strength Shafts 3, Go back to original flywheel design. 4, Replace intake motor with speed motor	Tasks Completion: 1,100% 2,100% 3,100% 4,100% Time Worked: 9 hours	Next Tasks:
-------------	---	---	-------------

Meeting #72 Details

Name: Aidan	Task: Cut out holes for new flywheel	Details: Now switching to HS shafts after system overloads (51581s) suggestion. I needed to cut holes in the current flywheel framing to allow for spacing of HS shafts which should be more stable	Images
Name: Aidan	Task: Create new flywheel with HS shafts	Details: Now with the new framing I began with reputting the flywheel ball bearing in with the side with the motor having nylon spacers to allow for a motor screw to go underneath it. Then put another ball bearing on the other side directly touching the metal. I found that using washers to slightly offset the bearing so it doesn't touch the metal is a bad idea due to the spacers compressing and screwing with the alignment to the metal. Then redo all the spacing with HS spacings and HS shaft collar. This time due to the shear size of the shaft collars I need to put one in the inner casing rather than under the motor. After all that I make a new flywheel which has 2 weights now one on each side for extra inertia.	Images

Meeting #72 Details

Name: Aidan	Task: Go back to original Flywheel design	Details: After putting the motor on after making the hole system I realized that the alignment is off to the point where the motor doesn't fit. Right now I don't have the resources to fix the alignment issue. Another issue there is that its way to heavy for the motor to spin the system up past 1800RPM which is roughly half speed. It also didn't launch a disc very far. So I switched the flywheel back to the old LS shaft system which took 3 hours to completely remake. The final product is now able to spin up to around 3000~ from a single initial test.	Images
Name: Aidan	Task: Replace Intake Motor with Speed motor (then go back)	Details: A modification which I wanted to see if was plausible was to see if I can exchange the 200rpm intake motor with a 600rpm speed motor to speed up the intake. The motor wasn't strong enough to overcome the friction and failed so I had to switch back to the normal motor	N/A

Design Process Page 15

Expansion V1 (OD) (12/31/22)

Expansion V1 Created

After months of work from josh he made a single shooter expansion. The expansion works by pulling the standoff connected to the pneumatic nut back

Problem:

Problem details

Solution:

Solution details

Design Process Page Formatting

Roller Mech Version 4

Image Placeholder

Changes

Details

Problem:

If we wanted the roller to spin with the intake, but it either got in the way of the turret, or it was too short, and we couldn't attach it to the turret because we need the 8 motors for other robot functions

Solution:

Instead of using 2 motors on the flywheel, we modified a motor so that we only had to use 1, meaning I got access to a motor specifically for the rollers, so I made a basic roller mech using 4-inch omni wheels that could attach to the turret.]

Design Process - Indexer

Indexer Version 3

Image Placeholder

Changes

Polycarb + double bar lever

Problem:

The old Indexer would not push the discs correctly, and got in the way of the intake

Solution:

We are using a double bar mechanism to push discs forward from the back of the turret. The double bar flap stays at the back of the turret, and rotates 360 degrees around the disc platform, and promptly returns to the back after pushing forward so another disc can be indexed

Design Process - intake stabilizing

Intake Version 4

Image Placeholder

Changes

Added brackets for stabilization

Problem:

Intake ramp cuts into field, needs to be fixed immediately

Solution:

I will be spanning a c channel across the back of the polycarb intake to ensure no movement occurs during testing (1/26/23)