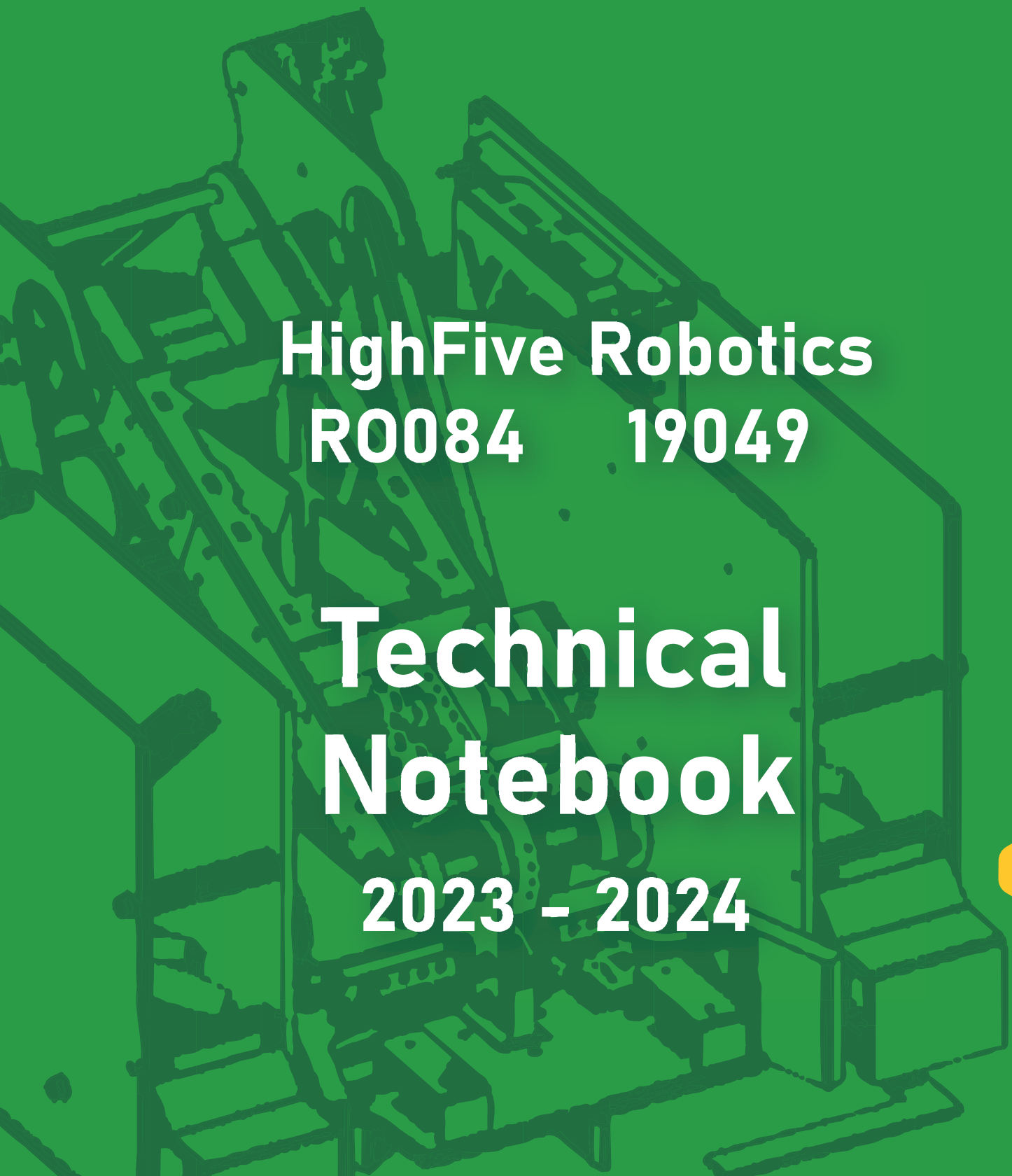
FIRST TECH CHALLENGE ROMANIA | SEASON#8 CENTERSTAGE PRESENTED BY RTX

ORGANIZATOR NAȚIE PRIN EDUCAȚIE | PARTENER FONDATOR BRD GROUPE SOCIETE GENERALE



HighFive Robotics RO084 19049

Technical Notebook

2023 - 2024

HighFive Robotics

CUPRINS

STRATEGIE

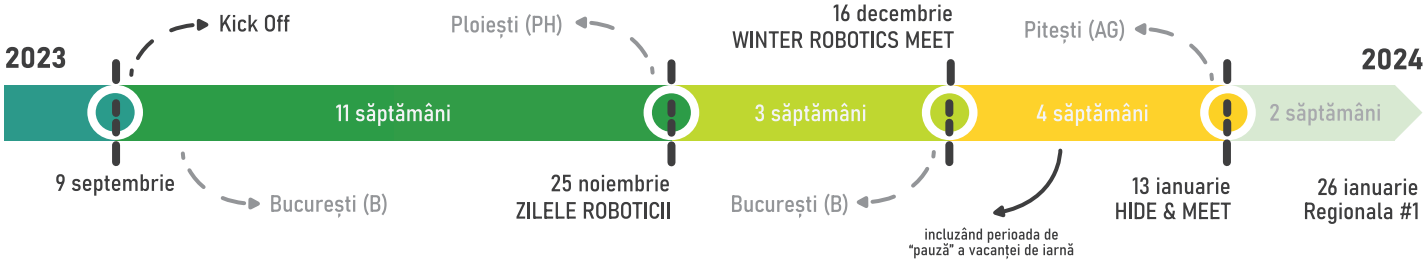
Aprilie 2023 - Ianuarie 2024

Strategie participare la League Meet-uri

25 noiembrie - 16 decembrie - 13 ianuarie

În ceea ce privește poziționarea pe axa temporală a League Meet-urilor, am optat să participăm la **primul** astfel de eveniment din regiunea Muntenia, urmat de celelalte două disponibile **echidistante** în timp față de acesta și etapa Regională. Astfel, putem itera eficient, remedia eventuale probleme și, totodată, performa cu o **creștere liniară**, susținută, având la dispoziție aproximativ o lună între competiții.

Ca localizare pe harta națională a roboticii, alegerile noastre au fost chiar principalele **epicentre ale roboticii** din regiune, și anume **Ploiești** (Prahova), **București** (București), respectiv orașul natal, **Pitești** (Argeș), pentru a putea interacționa cu cât mai multe echipe diverse, a forma alianțe variate, care să ne ajute la **adaptarea la strategii** de joc spontane, cu anumite aturi sau puncte slabe și a acumula idei și perspective variate.



Strategie de joc

30s Autonomie - 90s TeleOp - 30s End Game

Conform criteriilor de departajare în League Ranking, indispensabilă este **Autonomia**, urmată de **EndGame**, astfel încât perioada de **TeleOp** rămâne o resursă constantă de puncte, dar mai puțin valoroase pentru clasare.

Luând în considerare aceste aspecte, am creat o **strategie** de evoluție pe parcursul sezonului a roboților care să se focuseze pe obținerea constantă a unui scor cât mai **ridicat** în primele 30 de secunde ale meciului, apoi pe îndeplinirea misiunilor suplimentare din ultimele 30 de secunde, în cele din urmă exploatând și perioada controlată. Ne-am propus ***target*-uri detaliate** pentru fiecare competiție, respectând organizarea OKR și SMART:

OPR

OPR-ul (Offensive Power Rating) este un indicator care arată într-o manieră numerică contribuția medie individuală a fiecărei echipei din cadrul unei alianțe.

League Meet #1			
	Obiectiv individual	Rezultat mediu individual	Completare
Autonomie	50	39,29	78,58%
TeleOp	10	10,65	106,50%
Endgame	40	36,30	90,75%

Global OPR Ranking

45th	19049 *HighFive Robotics*	86.24	112.83

International OPR Ranking (în afara SUA)

2nd	19049 *HighFive Robotics*	86.24	112.83

League Meet #2			
	Obiectiv individual	Rezultat mediu individual	Completare
Autonomie	50	47,31	94,62%
TeleOp	35	38,34	109,54%
Endgame	40	49,46	123,65%

Global OPR Ranking

5th	19049 *HighFive Robotics*	135.12	177.17

International OPR Ranking (în afara SUA)

2nd	19049 *HighFive Robotics*	135.12	177.17

League Meet #3			
	Obiectiv individual	Rezultat mediu individual	Completare
Autonomie	55	56,33	102,42%
TeleOp	50	52,90	105,80%
Endgame	40	43,00	107,50%

Global OPR Ranking

4th	19049 *HighFive Robotics*	153.12	56.33

International OPR Ranking (în afara SUA)

1st	19049 *HighFive Robotics*	153.12	56.33

Meci (W-L-T)

Autonomie (TBP 1)
- Randomization tasks;
- Pixeli albi;

TeleOp
- Mozaic;
- Set Line;

End Game (TBP 2)
- Dronă;
- Cățărare;

Conform criteriilor de departajare în League Ranking, indispensabilă este **Autonomia**, urmată de **EndGame**, astfel încât perioada de **TeleOp** rămâne o resursă constantă de puncte, dar mai puțin valoroase pentru clasare.

Luând în considerare aceste aspecte, am creat o strategie de evoluție pe parcursul sezonului a roboților care să se focuseze pe obținerea constantă a unui scor cât mai ridicat în primele 30 de secunde ale meciului, apoi pe îndeplinirea misiunilor suplimentare din ultimele 30 de secunde, în cele din urmă exploatând și perioada controlată.

Autonomie:

După analiza modalităților de punctare în perioada de Autonomie, am constatat faptul că putem îmbina facil task-urile de **randomizare**, care au o importanță sporită (oferind în total maximum 45 de puncte), cu un task ușor realizabil, **parcarea**, ce aduce 5 puncte prin simpla deplasare a robotului în linie dreaptă, cele **50 de puncte** astfel obținute fiind critice.

Un robot capabil să puncteze complet **un *Stack*** și să obțină cele **50 de puncte** menționate atinge un scor total de **75 de puncte** (pentru un ecart de 25 puncte în plus, robotul trebuie să traverseze terenul de **cel puțin trei ori**, fapt ce poate consuma mult **timp**, adițional unui potențial grad ridicat de **risc**). Totuși, acesta este mai scăzut decât cel obținut de o alianță care **colaborează** armonios, fiecare echipă obținând **cele 50 de puncte** (totalul este de **100 de puncte** în Autonomie).

Prin urmare, am hotărât că este indispensabil să creăm o serie de **trasee** atent gândite, pentru a ne sincroniza cât mai eficient cu diverși parteneri de **echipă**.

Task-urile de randomizare de 50 de puncte și trasee colaborative.

TeleOp:

Datorită punctajului suplimentar relativ ridicat la **Mozaicurilor** și al ***Set Line*-urilor**, acestea sunt, evident, mult mai importante decât simpla punctare a pixelilor, fie ei albi sau colorați.

Pentru o strategie eficace, am prioritizat aceste două sarcini atât față de celelalte, cât și una față de cealaltă - Mozaicurile sunt mai **eficiente** decât *Set Line*-urile. Concret, am constatat că pentru a atinge ***Set Line*-ul 2** este nevoie de **12 pixeli**, ce pot forma **3 Mozaicuri**, triplând punctajul. Poziționarea strategică a acestor formațiuni pentru a minimiza numărul pixelilor necesari și a maximiza benzile albe de pe *Backdrop* atinse se pot obține cu **15 pixeli - 95 de puncte** (45 de puncte pixelii efectivi, 30 de puncte Mozaicurile și 20 de puncte *Set Line*-urile), iar cu **20 de pixeli** se pot obține **120 de puncte** (60 de puncte pixelii efectivi, 40 de puncte Mozaicurile și 20 de puncte Set Line-urile).

15 pixeli obțin 95 de puncte (cu 3 Mozaicuri & 2 Set Line-uri)

20 de pixeli obțin 120 de puncte (cu 4 Mozaicuri și 2 Set Line-uri)

End Game:

În End Game, misiunile extra, și anume **cățătratul și lansarea** corespunzătoare a **dronei**, au o valoare semnificativă (**de la 30 la 50 de puncte**, în funcție de *Landing Zone*). Ele devin proritare, atât la nivel de scor final, cât și de TBP2 .

Între 40 și 50 de puncte prin cățărare și dronă în Landing Zone 1 sau 2

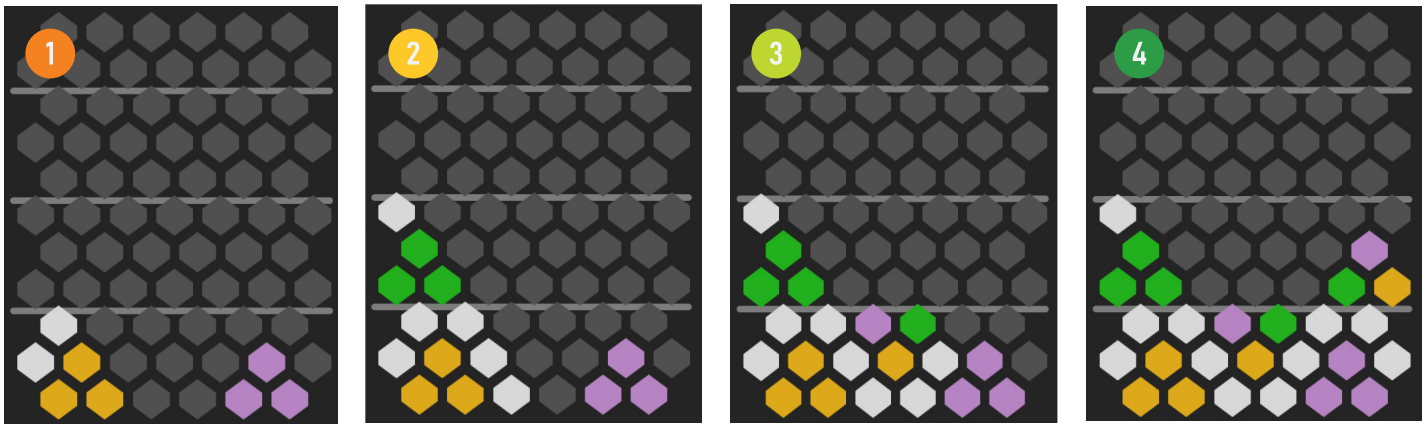
Autonomie				
Task	Points	Difficulty	Pts./Diff.	Obs.
Park	5	1	5,0	
Purple Pixel(White Pixel)	10	2	5,0	
Yellow Pixel(White Pixel)	10	3	3,3	
Purple Pixel(Team Prop)	20	2,5	8,0	
Yellow Pixel(Team Prop)	20	3,5	5,7	Mai aduce 5 pct. pt pixel pe backdrop
Pixel in Backstage	3	4	0,8	
Pixel on Backdrop	5	5	1,0	
TeleOp				
Task	Points	Difficulty	Pts./Diff.	Obs.
Pixel in Backstage	1	1	1,0	
Pixel on Backdrop	3	2	1,5	
Artist Bonus	10	2,5	4,0	
Set Bonus 1	10	2,5	4,0	
Set Bonus 2	10	4	2,5	
Set Bonus 3	10	5	2,0	
End Game				
Task	Points	Difficulty	Pts./Diff.	Obs.
Park	5	1	5,0	
Suspended	20	2,5	8,0	
Landing Zone 1	30	3	10,0	
Landing Zone 2	20	2	10,0	
Landing Zone 3	10	1	10,0	

Regionala #1			
	Obiectiv individual	Rezultat mediu individual	Completare
Autonomie	55	44,64	81,16%
TeleOp	50	42,28	84,56%
Endgame	40	45,00	112,50%

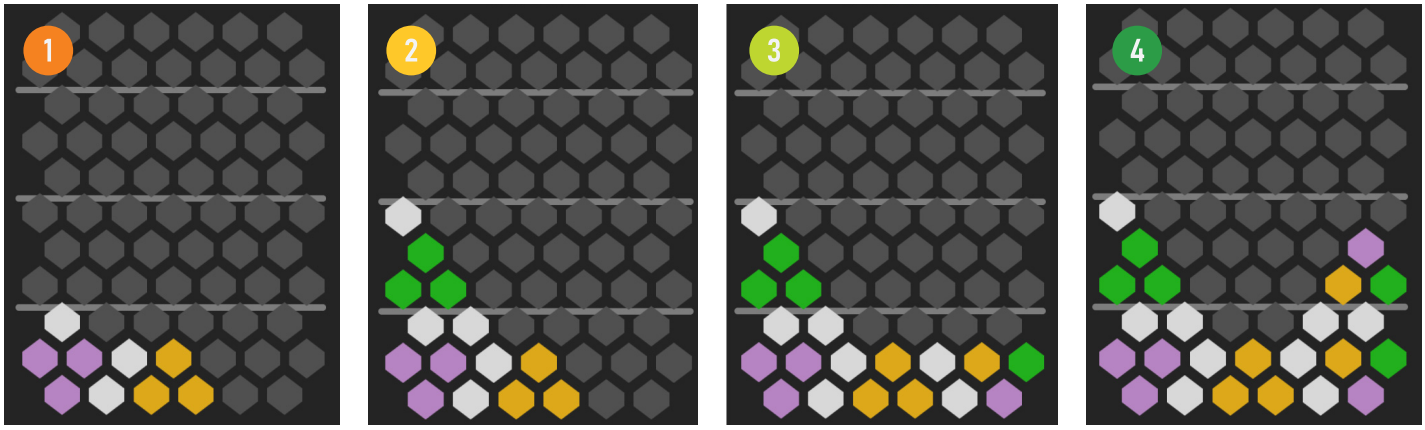
Îmbunătățiri Națională:

Pentru perioada de **TeleOp**, precum am procedat și în conceperea celor **96 de cazuri** diferite posibile, am definit, în funcție de cele **3 randomizări** posibile, modele care corespund strategiei definite (mozaicuri și *Set Line*-uri), utilizând un număr minim de pixeli.

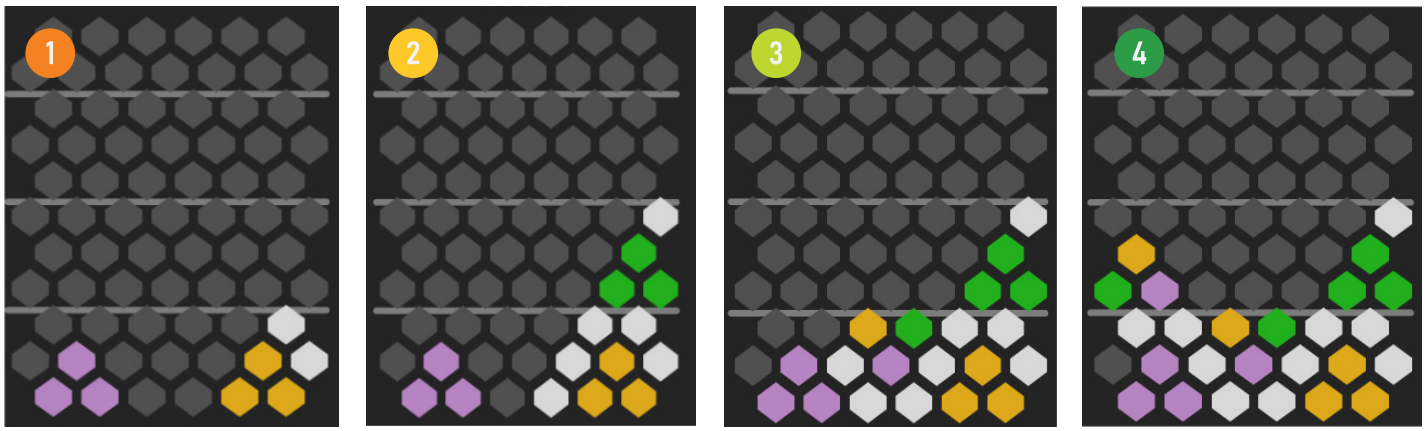
Plasarea pixelilor pe Backdrop pentru cazul ***Left***:



Plasarea pixelilor pe Backdrop pentru cazul ***Middle***:



Plasarea pixelilor pe Backdrop pentru cazul ***Right***:



Scouting

Noiembrie 2023 - Ianuarie 2024

Introducere:

Pentru o bună **strategie** generală (acțiuni în timpul meciurilor, în funcție de partenerul de alianță și de oponenți; anumite idei de piese, subansamble și algoritmi implementați pe anumiți roboți) și o evoluție catalizată a departamentului tehnic, ***Scouting***-ul este esențial.

Scouting roboți - Înainte de Regionala #1:

Pentru a ne planifica strategia descrisă privind aliențele am alcătuit un tabel care conține **informații** despre **funcționalitatea** fiecărui robot care participă la League Tournament #1, pentru a îi putea exploata atuurile, pentru meciuri cu performanțe cât mai ridicate. De la evoluția echipelor până la momentul etapei regionale, însemnând numărul de League Meet-uri, tipul de șasiu, intake sau outtake, mecanismele de lansare a dronei ori de cățărat, până la kit-urile și piesele custom, toate sunt aspecte esențiale pentru o strategie cât mai complexă și stabilă.

Team ID.	Team No.	Team Name	No. of Meets	Drivetrain	Format	Intake	Outtake	Drone	Climb	Build System us
RO_077	14270	Quantum Robotics	3	Mecan...	Pass-T...	Active	LinSlide		Arm	Custom + GoBilda
RO_014	14277	QUBE.	2	Mecan...	Pass-T...	Active	LinSlide			GoBilda
RO_139	15972	TehnoZ	3	Mecan...	Pass-T...	Active	LinSlide		LinSlide	Custom
RO_015	15989	RoboTitans	2	Mecan...	Conve...	Passive	Arm			GoBilda
RO_013	15991	GAMMA	2	Mecan...	Conve...	Passive	Arm			GoBilda
RO_140	15993	infO(1)Robotics	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...	Arm	Custom
RO_023	16166	WATT's UP	3	Mecan...	Pass-T...	Passive	LinSlide	Rubber...	Arm	Custom
RO_114	17624	IGNITE.	2	Mecan...	Pass-T...	Passive	LinSlide	Rubber...		Custom
RO_102	17869	ERCAST2020	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...		GoBilda
RO_120	17962	Ro2D2	3	Mecan...	Pass-T...	Active	LinSlide		Winch	Custom
RO_126	18160	4D-ROBOTICS	3	Mecan...	Pass-T...	Passive	LinSlide	Rubber...	LinSlide	GoBilda
RO_084	19049	HighFive Robotics	3	Mecan...	Pass-T...	Passive	LinSlide	Rubber...	LinSlide	Custom
RO_187	19054	NeuroBotix	3	Mecan...	Pass-T...	Active	LinSlide			GoBilda
RO_017	19055	TITANS	3	Mecan...	Pass-T...	Active	LinSlide		LinSlide	GoBilda
RO_145	19059	ArtRobotiX	3	Mecan...	Pass-T...	Active	LinSlide			GoBilda
RO_045	19062	Phantom Robotics	3	Mecan...	Pass-T...	Passive	LinSlide			Custom
RO_092	19063	UnderConstruction	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...	LinSlide	GoBilda
RO_039	19064	DRAGONIC-FORCE	2	Mecan...	Kooky	Passive	LinSlide			GoBilda
RO_116	19072	TehnoZ Junior	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...	LinSlide	Custom
RO_108	19075	Clockworks	3	Mecan...	Pass-T...	Active	LinSlide		LinSlide	GoBilda
RO_009	19082	RoboAS	1	Mecan...	Conve...	Passive	Arm			GoBilda
RO_053	19084	Zenith	3	Mecan...	Pass-T...	Active	LinSlide			GoBilda
RO_067	19090	TEHROCUZ	2	Mecan...						GoBilda
RO_111	19096	Danube Robotics	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...		GoBilda
RO_181	19098	EasternFoxes	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...		Custom
RO_106	19099	H-tech	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...		GoBilda
RO_008	19103	KronBot	3	Mecan...	Pass-T...	Passive	LinSlide			GoBilda
RO_112	19112	UNDEFINED	2	Mecan...	Pass-T...	Active	LinSlide		LinSlide	GoBilda
RO_147	19116	VV ROBOTS	3	Mecan...	Conve...	Passive	LinSlide	Rubber...	LinSlide	GoBilda
RO_016	19117	Robo-Sapiens	3	Mecan...	Pass-T...	Active	LinSlide		Winch	GoBilda
RO_155	19131	Evolution	3	Mecan...	Kooky	Passive	LinSlide		Arm	Custom
RO_041	19141	BraveBots	3	Mecan...	Kooky	Passive	LinSlide	Rubber...		Custom
RO_131	19151	TECHNO-MAISTER	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...		
RO_153	19176	Broboți	3	Mecan...	Conve...	Passive	LinSlide	Rubber...	LinSlide	GoBilda
RO_193	20691	Andromeda	3	Mecan...	Conve...	Passive	Arm			GoBilda
RO_046	20925	HyperCube	3	Mecan...	Conve...	Passive	LinSlide			GoBilda
RO_011	21050	MasterMinds	0							
RO_133	21071	SkyLine	3	Mecan...	Kooky	Passive	LinSlide		LinSlide	GoBilda
RO_070	21073	November Eleven	0							
RO_096	21169	CyberLions	1	Mecan...						Tetrix
RO_019	21476	ViCyber	1	Mecan...	Kooky	Passive	LinSlide			GoBilda
RO_043	22084	RNS	1	Mecan...	Pass-T...	Passive	LinSlide			Custom
RO_012	22226	Renaissance Robotics	2	Mecan...	Pass-T...	Passive	LinSlide			Custom
RO_021	22390	Black Feather	0							
RO_059	22552	ToTheStars	2	Mecan...	Pass-T...	Active	LinSlide	Rubber...		GoBilda
RO_128	23203	Light Bulb Robotics	2	Mecan...	Pass-T...	Active	LinSlide	Rubber...	LinSlide	Custom
RO_129	23220	Space Otters	3	Mecan...	Pass-T...	Active	LinSlide	Rubber...	LinSlide	Custom
RO_054	23473	Helios Robotics	1	Tank 4wd	Pass-T...	Passive	Arm			REV0
RO_018	24266	VIP	2	Mecan...	Pass-T...	Active	LinSlide		Arm	Custom
RO_055	24269	DEJA VU	2	Mecan...	Pass-T...	Active	LinSlide		Arm	Custom
RO_117	24345	SPARKTECH	0							
RO_185	24530	ElectroB0tics	2	Tank 4wd	Conve...	Passive	Arm			Rev
RO_200	24637	Black Phantoms	3	Mecan...	Pass-T...	Passive	LinSlide	Rubber...	LinSlide	GoBilda
RO_097	24964	OrionRobotix	2	Mecan...	Kooky	Passive	LinSlide			GoBilda

Exemplu pentru echipe din Regionala #1 după performanța din League Meet-uri

Scouting performanță în meciuri - Înainte de Regionala #1:

De asemenea am realizat și un tabel în care să analizăm **performanțele** echipelor din regiunea noastră, care ne-a confirmat faptul că cel mai important aspect al acesteia este obținerea unor **rezultate constante**: de la punctajele corespunzătoare fiecărei părți a meciului, până la cel final.

Rank	Team	Name	Avg RP	Max OPR	Avg OPR	OPR LM 1	OPR LM 2	OPR LM 3	Average TBP1	Average TBP2	High Score	W-L-T	Matches Played
1	19049	HighFive	2,00	153,12	124,83	86,24	135,12	153,12	85,90	62,00	312	10-0-0	18
2	16166	Watt'S Up	2,00	135,13	117,41	122,68	135,13	94,42	79,30	48,00	312	10-0-0	18
3	19176	Broboți	2,00	133,26	124,63	123,74	133,26	116,90	71,20	55,50	311	10-0-0	18
4	15972	TehnoZ	2,00	137,45	116,97	104,74	108,73	137,45	67,30	52,00	272	10-0-0	18
5	18160	4D	2,00	90,98	79,04	56,31	90,98	89,82	65,70	34,50	283	10-0-0	18
6	19072	TehnoZ Jr.	2,00	86,69	73,64	86,69	65,91	68,32	63,90	28,50	197	10-0-0	18
7	17962	Ro2D2	2,00	157,93	99,76	34,23	107,12	157,93	54,30	38,00	301	10-0-0	18
8	15993	inf0(1)Robotics	2,00	62,68	43,28	50,45	62,68	16,72	54,30	18,50	200	10-0-0	18
9	23203	Light Bulb	2,00	110,75	80,29	50,69	79,44	110,75	53,60	48,50	245	10-0-0	18
10	19099	H-tech	2,00	57,13	48,24	42,54	45,04	57,13	51,80	28,00	245	10-0-0	18
11	19098	Eastern Foxes	2,00	93,69	58,64	50,85	31,37	93,69	51,20	32,00	301	10-0-0	18
12	19141	BraveBots	2,00	74,75	53,87	17,46	69,41	74,75	50,40	34,00	179	10-0-0	18
13	19117	Robo-Sapiens	2,00	63,71	58,00	56,67	53,61	63,71	50,30	46,50	232	10-0-0	18
14	24269	DEJA VU	2,00	82,76	64,39	42,66	67,75	82,76	50,20	37,50	283	10-0-0	18
15	19055	TITANS	2,00	82,16	70,49	50,24	79,07	82,16	49,00	52,50	205	10-0-0	18
16	19063	UnderConstruction	2,00	75,12	49,87	24,06	50,42	75,12	41,50	40,00	221	10-0-0	18
17	24266	VIP	2,00	77,45	65,79	55,38	64,53	77,45	40,60	36,00	200	10-0-0	18
18	19067	BIONIC ROYALS	2,00	84,25	65,49	38,09	74,14	84,25	39,20	30,00	184	10-0-0	18
19	22084	RNS	2,00	35,71	27,83	21,46	26,33	35,71	38,10	15,00	157	10-0-0	18
20	21071	SkyLine	2,00	37,76	30,71	28,38	37,76	25,99	30,60	29,50	158	10-0-0	18
21	24637	Black Phantoms	2,00	0,00	0,00				30,40	46,50	245	10-0-0	18
22	19112	UNDEFINED	2,00	0,00	0,00				14,30	20,50	147	10-0-0	18
23	14270	Quantum Robotics	1,80	0,00	0,00				36,30	29,00	200	9-1-0	18
24	19096	DanubeRobotics	1,80	0,00	0,00				30,80	22,00	150	9-1-0	18
25	23473	Helios Robotics	1,80	0,00	0,00				30,00	33,50	181	9-1-0	18
26	22552	ToTheStars Robotics	1,80	0,00	0,00				29,20	25,00	194	9-1-0	18
27	19116	VV Robots	1,60	0,00	0,00				37,60	34,00	181	8-2-0	18
28	17869	ERcast2020	1,60	0,00	0,00				19,50	22,00	181	8-2-0	18
29	19084	Zenith	1,40	0,00	0,00				36,00	22,50	204	7-3-0	18
30	19054	NeuroBotix	1,40	0,00	0,00				34,50	29,00	168	7-3-0	24
31	19151	TECHNO-MAISTER	1,40	0,00	0,00				34,20	19,50	189	7-3-0	18
32	19131	Evolution	1,40	0,00	0,00				32,00	25,50	219	7-3-0	18
33	19059	ArtRobotix	1,40	0,00	0,00				22,60	16,00	192	7-3-0	18
34	19103	Kronbot	1,40	0,00	0,00				21,50	20,50	126	7-3-0	18
35	23220	Space Otters	1,20	79,35	60,80	67,69	35,37	79,35	59,00	32,00	212	6-4-0	18
36	20925	HYPERCUBE	1,20	0,00	0,00				29,10	23,00	167	6-4-0	18
37	20691	Andromeda	1,20	0,00	0,00				22,10	16,50	181	6-4-0	18
38	19090	TEHROCUZ RO 067	1,00	0,00	0,00				37,30	25,00	163	5-5-0	18
39	14277	QUBE.	1,00	0,00	0,00				30,80	24,00	171	5-5-0	12
40	19075	Clockworks	1,00	0,00	0,00				21,00	42,00	163	5-5-0	18
41	24530	ElectroB0tics	0,80	0,00	0,00				32,30	27,00	182	4-6-0	18
42	15989	RoboTitans	0,80	0,00	0,00				32,10	22,00	180	4-6-0	12
43	24964	ORION ROBOTIX	0,60	0,00	0,00				34,30	21,00	155	3-7-0	18
44	19062	Phantom Robotics	0,60	0,00	0,00				32,10	11,50	128	3-7-0	18
45	22226	Renaissance Robotics	0,60	0,00	0,00				24,70	10,00	119	3-7-0	18
46	15991	Gamma	0,40	0,00	0,00				18,80	19,50	160	2-8-0	12
47	21169	CyberLions	0,40	0,00	0,00				18,00	9,50	102	2-4-0	6
48	22390	Black Feather	0,40	0,00	0,00				4,80	2,00	42	2-4-0	6
49	21476	ViCyber	0,20	0,00	0,00				12,50	6,50	81	1-5-0	6
50	19064	Dragonic Force	0,20	0,00	0,00				12,00	11,00	154	1-5-0	6
51	17624	IGNITE.	0,20	0,00	0,00				10,00	8,50	118	1-5-0	6
52	24345	SPARKTECH	0,20	0,00	0,00				8,10	10,00	109	1-5-0	6
53	19128	RoboDac Dacia	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0
54	20131	The Inventors	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0
55	21050	MasterMinds	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0
56	21073	November Eleven	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0
57	22973	EXE-ROBOTICS1	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0
58	24155	BlueSpace	0,00	0,00	0,00				0,00	0,00	0	0-0-0	0

Scouting performanță în meciuri - Înainte de Națională:

Pentru a putea adopta o strategie cât mai bună, am realizat (cu ajutorul unui robot creat pe o platforma a **UiPath,** din conturile de învățare puse de aceștia la dispoziție, care să extragă și să prelucreze automat **date oficiale** de pe site-urile *FIRST*®) o **sinteză a performanței** tuturor echipelor participante la Națională.

1		League Tournment	Advanced	Team	Name	City	County	RP	TBP	np AVG	np OPR	np MAX
2	București	RONR1LT	4	16166	Watt'S Up	Câmpulung	Argeș	1,33	74,67	236,33	170,74	290
3	Iași	RONR4LT	2	21028	The Eagles RO143	Roman	Neamț	2,00	90,50	223,00	154,58	285
4	București	RONR1LT	8	17962	Ro2D2	Ploiești	Prahova	2,00	68,67	194,17	154,42	270
5	Cluj	RONR2LT	2	19091	The Resistance	Mediaș	Sibiu	2,00	79,67	218,17	148,63	259
6	Timișoara	RONR3LT	4	19047	RavenTech_HD	Hunedoara	Hunedoara	1,67	68,33	196,50	145,27	296
7	Timișoara	RONR3LT	5	17861	CSH	Timișoara	Timișoara	1,67	57,83	177,83	138,55	296
8	Iași	RONR4LT	3	19066	AiCitizens	Focșani	Vrancea	1,33	75,50	210,50	135,65	285
9	Cluj	RONR2LT	5	24033	Alphatronic	Cluj-Napoca	Cluj	1,33	55,33	173,33	135,01	233
10	București	RONR1LT	2	15972	TehnoZ	Pitești	Argeș	2,00	84,50	220,67	132,74	283
11	București	RONR1LT	14	19049	HighFive Robotics	Pitești	Argeș	1,67	59,67	168,17	131,93	208
12	București	RONR1LT	6	19067	BIONIC ROYALS	Râmnicu Vâlcea	Vâlcea	1,50	64,67	180,33	125,88	243
13	Cluj	RONR2LT	10	19068	ABSO-TECH	Gherla	Cluj	1,33	57,00	165,00	125,43	226
14	Iași	RONR4LT	4	19234	ByteForce	Galați	Galați	1,33	54,00	162,00	120,24	237
15	Timișoara	RONR3LT	2	15975	ROBOTX HUNEDOARA	Hunedoara	Hunedoara	2,00	60,50	170,50	118,93	217
16	București	RONR1LT	9	23203	Light Bulb Robotics	Pitești	Argeș	1,33	59,33	182,50	117,99	288
17	București	RONR1LT	12	19099	H-tech	București	București	1,33	47,67	148,33	111,60	187
18	Timișoara	RONR3LT	10	19121	Tea Borgs	Târgu Jiu	Gorj	1,33	66,50	158,00	107,32	225
19	Iași	RONR4LT	9	19043	CyLiis	Iași	Iași	1,67	63,67	155,33	105,76	259
20	Timișoara	RONR3LT	8	17713	Delta Force	Arad	Arad	1,67	68,83	166,50	105,10	244
21	București	RONR1LT	1	19141	BraveBots	Ploiești	Prahova	1,33	54,17	159,83	104,88	227
22	Iași	RONR4LT	6	19139	Snake-Tech40	Piatra Neamț	Neamț	1,67	45,50	161,00	102,00	197
23	Iași	RONR4LT	11	19044	Peppers	Iași	Iași	0,67	58,00	159,83	101,34	194
24	Cluj	RONR2LT	13	16000	Gear Maniacs	Sibiu	Sibiu	1,33	54,67	167,17	99,58	212
25	Iași	RONR4LT	8	17860	Helix	Brăila	Brăila	1,33	44,50	142,17	97,01	204
26	Iași	RONR4LT	12	24308	EchoPulse	Bacău	Bacău	1,00	54,67	158,17	92,92	259
27	Cluj	RONR2LT	1	19115	B-Robo	Satu Mare	Satu Mare	1,67	53,67	173,50	92,91	233
28	București	RONR1LT	13	19117	Robo-Sapiens	București	București	1,00	70,17	159,83	91,35	280
29	București	RONR1LT	11	23220	Space Otters	Pitești	Argeș	1,67	54,67	152,50	91,34	227
30	Cluj	RONR2LT	4	19079	PHOENIX	Cluj-Napoca	Cluj	0,67	55,83	134,67	90,88	193
31	Cluj	RONR2LT	6	19056	PrimeTech	Cluj-Napoca	Cluj	1,33	68,83	157,33	90,51	201
32	Cluj	RONR2LT	8	19101	Tech-X	Sighetu Marmației	Maramureș	1,33	59,17	162,00	90,34	259
33	Iași	RONR4LT	5	23161	CyberLIS76	Panciu	Vrancea	1,33	40,50	148,17	89,74	234
34	București	RONR1LT	10	18160	4D-Robotics	Pitești	Argeș	1,00	49,50	144,83	88,94	204
35	Cluj	RONR2LT	12	22586	CNapSys	Zalău	Sălaj	1,67	62,50	143,67	88,52	159
36	București	RONR1LT	5	15993	inf0(1)Robotics	Ploiești	Prahova	1,00	72,33	179,83	87,54	270
37	Iași	RONR4LT	16	21087	Velocity Red	Brăila	Brăila	1,00	45,83	147,50	86,81	210
38	Timișoara	RONR3LT	7	19109	RaSky	Craiova	Dolj	1,33	66,33	153,67	86,36	217
39	Cluj	RONR2LT	11	20043	SNGine	Mediaș	Sibiu	0,67	51,00	148,17	85,42	212
40	Cluj	RONR2LT	3	17844	RUBIX	Blaj	Alba	1,33	48,00	138,83	82,39	183
41	București	RONR1LT	7	19055	TITANS	București	București	1,67	57,67	160,00	77,20	257
42	Iași	RONR4LT	7	20265	Heart of RoBots	Buzău	Buzău	0,67	66,50	149,00	77,09	228
43	Cluj	RONR2LT	14	17875	Code Warriors	Mediaș	Sibiu	1,33	47,50	132,17	75,41	154
44	Timișoara	RONR3LT	3	19105	DecebalTech	Deva	Hunedoara	1,33	73,17	148,33	75,16	191
45	București	RONR1LT	16	19098	Eastern Foxes	Ploiești	Prahova	1,50	63,67	155,83	74,62	174
46	Timișoara	RONR3LT	1	12560	Soft Hoarders	Craiova	Dolj	1,33	65,50	136,17	72,03	177
47	Iași	RONR4LT	10	19083	North East Dynamics	Dorohoi	Botoșani	2,00	51,67	119,67	71,99	181
48	Iași	RONR4LT	14	19061	Boogeybots	Râmnicu Sărat	Buzău	1,00	46,00	125,50	70,07	244
49	Iași	RONR4LT	1	21455	RoSophia	Galați	Galați	1,33	38,33	139,50	69,73	197
50	Timișoara	RONR3LT	13	25225	Afton Robotics	Hațeg	Hunedoara	1,67	40,50	129,83	68,59	191
51	București	RONR1LT	3	14270	Quantum Robotics	București	București	1,33	52,00	136,67	63,43	202
52	Iași	RONR4LT	13	22998	CYB3RG0DS	Piatra Neamț	Neamț	1,33	45,50	122,33	61,05	160
53	Iași	RONR4LT	15	20972	VOLTA CIRCUITS	Suceava	Suceava	1,33	26,83	109,67	60,78	204
54	Timișoara	RONR3LT	11	20732	ATLAS_CNB_192	Timișoara	Timișoara	1,67	52,50	124,50	55,97	194
55	Timișoara	RONR3LT	9	24478	EngiNeerds	Slatina	Olt	1,67	35,50	108,33	52,98	158
56	București	RONR1LT	15	19131	Evolution	Slobozia	Ialomița	1,00	42,17	104,67	43,52	187
57	Timișoara	RONR3LT	12	19120	AlphaBit	Petroșani	Hunedoara	1,00	45,00	103,17	34,16	156
58	Timișoara	RONR3LT	14	19257	Unplugged	Timișoara	Timișoara	1,00	34,17	89,00	27,39	182
59	Cluj	RONR2LT	9	14278	Xeo	Alba Iulia	Alba	0,67	38,33	89,50	22,66	140
60	Timișoara	RONR3LT	6	19088	BROBOTS	Râmnicu Vâlcea	Vâlcea	1,33	28,83	99,00	21,96	197
61	Cluj	RONR2LT	7	15966	Esentza Revolution	Beclean	Bistrița-Năsăud	0,00	24,67	73,83	-2,98	147

Înființată în anul **2005** de cofondatori **români**, **UiPath** este una dintre cele mai mari companii la **nivel global** în ceea ce privește automatizarea prin roboți virtuali. Utilizând și inteligența artificială, firma realizează **softuri** care îmbunătățesc performanța task-urilor oamenilor din varii domenii.

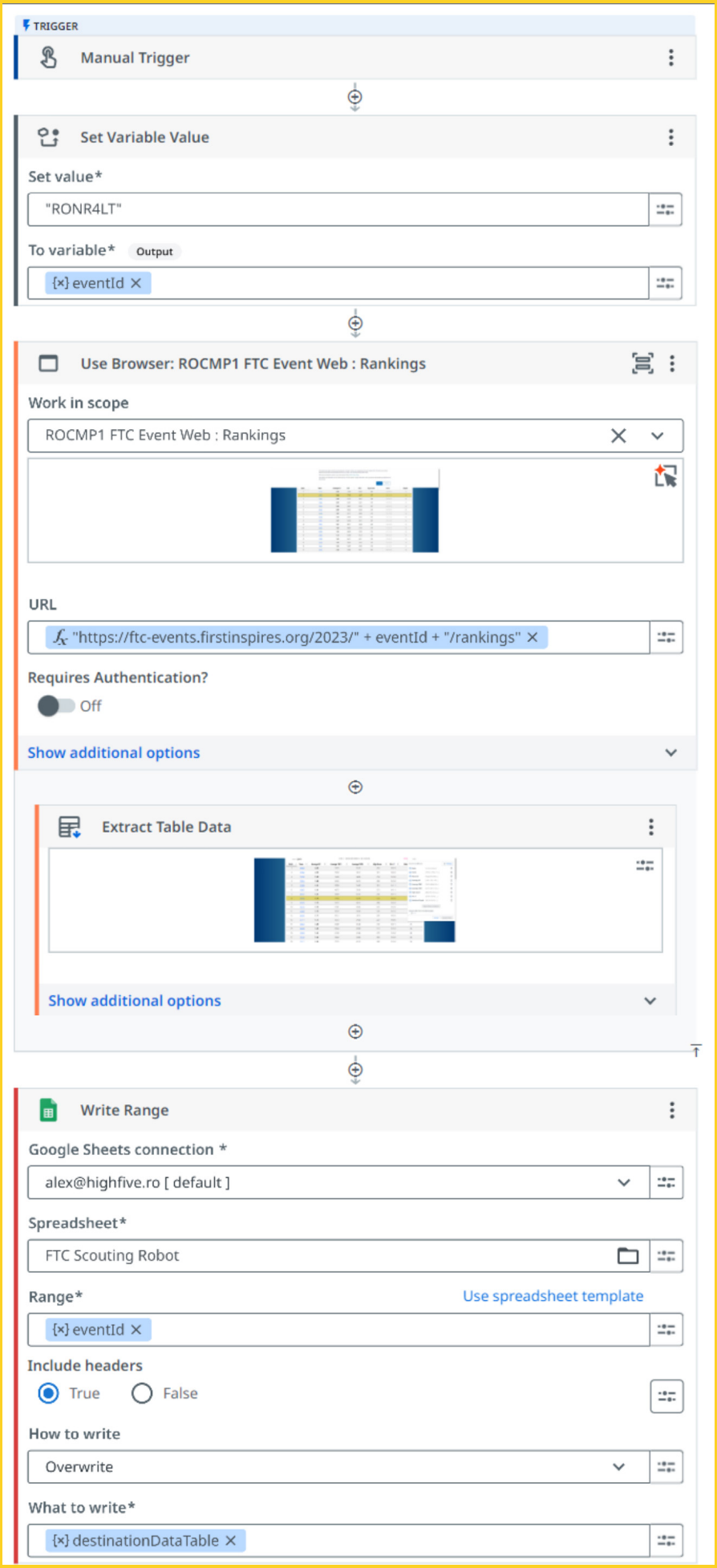
Tocmai pentru a putea utiliza în **mod optim** resuesele de care dispunem, dar și pentru a ne familiariza cu una dintre cele mai mari companii internaționale din domeniul tehnologiei și al prelucrării software, pentru etapa Națională am apelat la **modulele de studiu** oferite de aceștia.

Cu ajutorul **UiPath Studio Web** am realizat un robot software de bază, care extrage o serie de date bine definite existente în anumite surse din browser.

Folosind pași simpli, am ales să colectăm informațiile din majoritatea coloanelor relevante pentru **desfășurarea** în sine a meciurilor echipelor calificate.

Acestea sunt stocate într-un document amplu (cu diverse *Sheet*-uri și filtre) în **Google Drive,** pe care îl prelucrăm în funcție de anumite criterii.

Ne-am automatizat scoutingul cu un robot UiPath

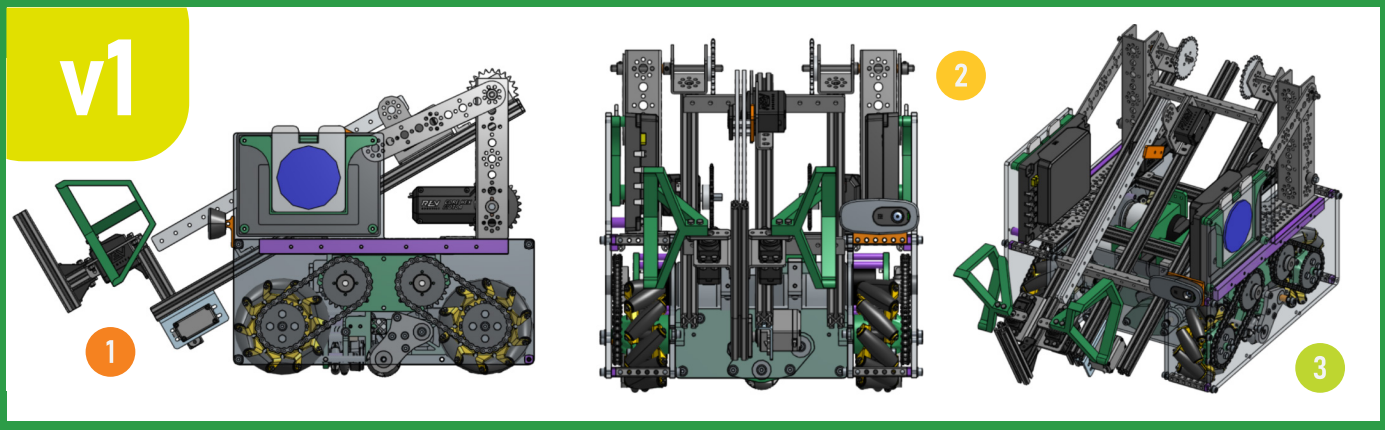


HARDWARE

Aprilie 2023 - Ianuarie 2024

Robot v1- KickAthon:

După lansarea misiunii sezonului CENTERSTAGE, am participat la **KickAthon-ul** organizat anual de echipa **Quantum Robotics #14270**, unde, pe parcursul a **16 ore**, participanții trebuie să se adapteze cât mai eficient la resursele pe care le au la dispoziție și la regulile de joc inedite, prin idei inovative și strategie adecvată.



Robot:

Capturi din OnShape cu robotul v1 - KickAthon:

- robotul complet din vedere laterală (1);
- robotul complet din vedere frontală (2);
- robotul complet din vedere izometrică (3).

Șasiu:

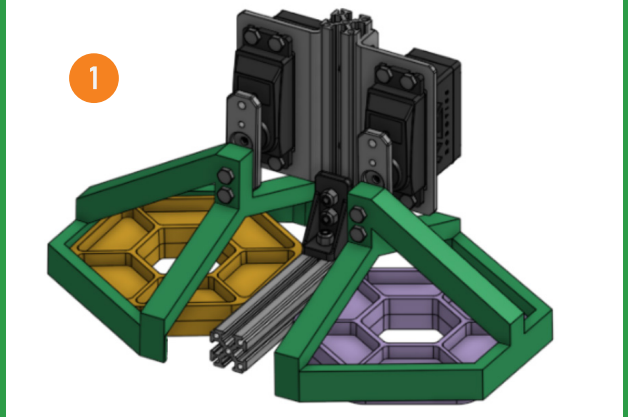
Pentru a reuși să ne încadrăm în timpul acordat și a ne focusa pe mecanismele specifice temei, am ales să folosim șasiul versatil la care am lucrat pe parcursul verii.

Gheară:

În urma analizării misiunilor din acest an, am conștientizat de la început avantajul semnificativ care ni-l aduce capacitatea de a colecta **2 pixeli** și de a îi plasa independent.

Având la dispoziție **16 ore** în contextul dat, am încercat să folosim cât mai multe piese aflate deja la îndemână. Componentele **proiectate 3D** și printate la fața locului, au trebuit să fie cât mai simple și inovatoare, mai ales datorită faptului că ne permit o **flexibilitatea** mai mare în gândire, însă acestea necesită mult timp de realizare, obstacol în fața începerii rapide a testelor.

Captură izometrică din OnShape cu gheara (1).



Avantaje:	Dezavantaje:
• Plasarea ușoară pe *Backdrop*; • Posibilitatea de a alege pixelul ce este plasat.	• Suprafață de colectare restrânsă; • Lipsa eficienței în situații de vizibilitate redusă; • Piese printate fragile.

Îmbunătățiri posibile:

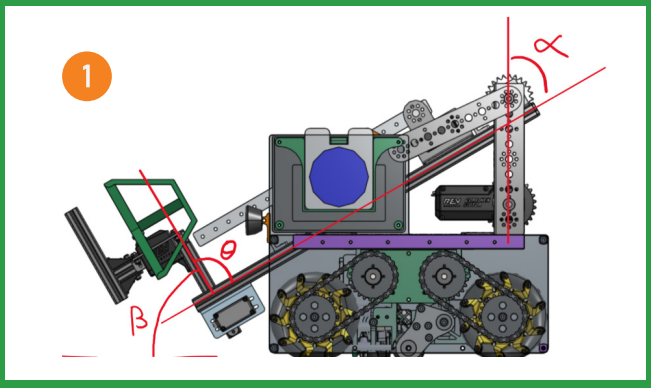
- Schimbarea poziției servourilor, de la o mișcare în planul vertical, la una în cel orizontal, pentru o suprafață mai mare de colectare;
- Reducerea numărului de piese necesare prin crearea unor piese personalizate;
- Realizarea mai robustă a pieselor printate.

Lift:

Pentru a reuși ridicarea pixelilor la înălțimile dorite am folosit 2 motoare **REV CORE HEX**, ce acționau brațul cu pinioane cu raport **27:24**. De asemenea, ne-am ajutat de un **GoBilda Torque Servo**, pentru a menține gheara paralela cu *Backdrop*-ul, respectiv cu solul.

Pentru a reduce forța ce se opune mișcării motorului, am contrabalansat brațul folosind **arcuri**.

Din cauza **timpului limitat**, nu am reușit să incorporăm rulmenți în mecanism, fiind nevoie de prinderi speciale pentru aceștia.



Captură din OnShape cu brațul și vizualizare calcule pentru menținere paralelă cu *Backdroop*-ul/solul (1).

Avantaje:	Dezavantaje:
• Ridicarea rapidă, datorită arcurilor; • Menținerea ghearei paralelă fie cu solul, fie cu *Backdropul*.	• Instabil - lipsa rulmenților; • Limitarea la plasarea pixelilor pe partea pe care au fost colectați; • Nu reușește să depășească nivelul primei benzi.

Îmbunătățiri posibile:

- Schimbarea motoarelor cu unele mai rapide și mai puternice;
- Modificarea design-ului pentru a putea ajungeșa întreaga înălțime a *Backdrop*-ului;
- Înlocuirea bucșelor cu rulmenți;
- Stabilizarea subansamblului.

- Poziția la care trebuie să ajunga servoul:

α = unghiul format de braț cu verticalca;

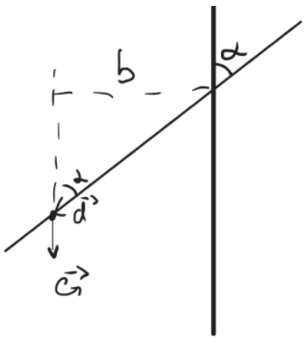
β = unghiul la care trebuie să ajungă gheara față de verticală;

$$\theta = \alpha + 90° - \beta$$

$$\alpha = \frac{EncoderCounts * 360°}{TicksPerRev} + Offset$$

- Contrabalansarea brațului folosind motoarele:

$$\tau = b \times G$$
$$\tau = d \sin\alpha \times m\, g$$
$$\tau = d\, m\, g \times \sin\alpha \,(d,\ m,\ g \text{ sunt constante})$$
$$\tau = k \times \sin\alpha$$

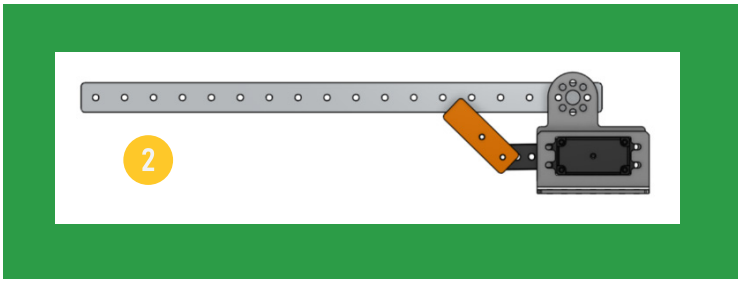


Lansator:

Pentru a lansa drona, am ales să folosim un elastic fixat cu un **servo**, care, după ce este eliberat, trage avionul printr-un **ghidaj**.

Pentru a putea **ajusta unghiul** sub care se face decolarea, am montat mecanismul pe sistemul de ridicare a pixelilor.

Avantaje:	Dezavantaje:
• Mecanism simplu, puține elemente ce se pot strica; • Lansare rapidă a dronei.	• Ghidajul are prea multă frecare cu drona; • Lipsa constanței din cauza prinderii pe sistemul de manipulat pixeli; • Mișcarea repetată a brațului făcea ca drona sa își schimbe poziția în ghidaj.



Capturi din OnShape cu mecanisme robot v1 - KickAthhon:
- lansatorul de drona din vedere izometrică (1);
- lansatorul de drona din vedere laterală (2).

Îmbunătățiri posibile:

- Schimbarea servoului cu unul mai mic, pentru a ușura ansamblul și a îl face mai compact;
- Modificarea ghidajului pentru a reduce frecarea și a îmbunătăți stabilitatea dronei înainte și în timpul lansării;
- Montarea subansamblului fix pe robot (dar să fie ajustabil înainte de meci).

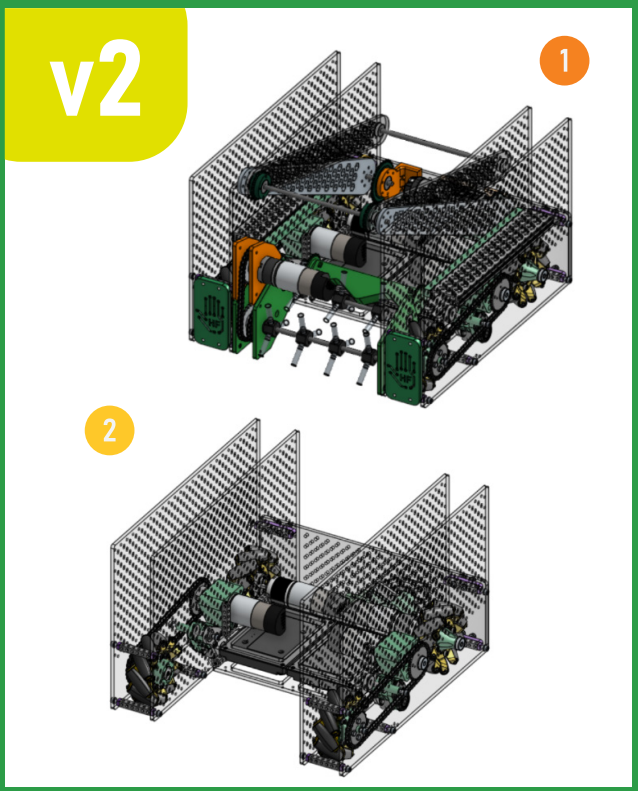
Prima echipă din lume: Zona 1

Robot Prototipări & Robot Didactic:

După începerea oficială a sezonului am constatat necesitatea a **doi roboți**: unul simplu și foarte ușor de înțeles de către public, cu **scop didactic** la prezentări și evenimente, și unul conceput pentru a putea **prototipa** succesiv, ca etapă predecesoare versiunilor de competiție.

Robot Prototipări:

Capturi din OnShape cu o variantă a robotului v2 și subansamble sale - Prototipări:
- una dintre versiunile robotului cu subansamble din vedere izometrică (1);
- șasiu din vedere izometrică (2);



Șasiu:

Realizând importanța prototipării rapide încă de la începutul sezonului, am realizat un șasiu care ne permite montarea și schimbarea foarte ușoară a diferitelor mecanisme. Acesta are un **model de găuri** compatibil cu cei mai comuni furnizori de componente folosite în *FIRST®* Tech Challenge.

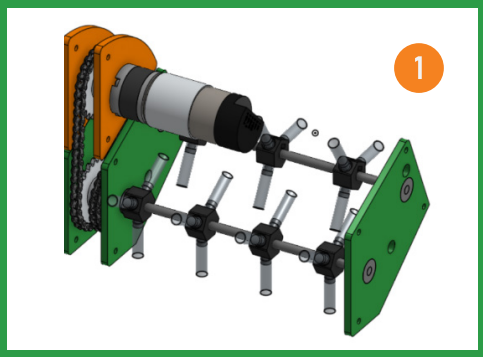
Calcul pentru viteza și viteza unghiulară maxime:

$$v_{max} = \omega_{max} \cdot d/2 \simeq 1.71 m/s$$
$$\omega_{max} = 340 RPM \simeq 35.60\, rad/s$$
$$d = 9.6\, cm = 0.096 m$$

Avantaje:	Dezavantaje:
• Model versatil pentru montarea diferitelor mecanisme; • Montarea elementelor electronice în partea inferioară pentru a elibera spațiu.	• Foarte mare și greu de manevrat.

Îmbunătățiri posibile:

- Reducerea dimensiunilor, pentru a obține manevrabilitate mai ridicată în teren (chiar dacă robotul a fost conceput în așa fel încât volumul total și structura să permită testarea, mai mult decât perfecționarea, mecanismelor);



Intake:

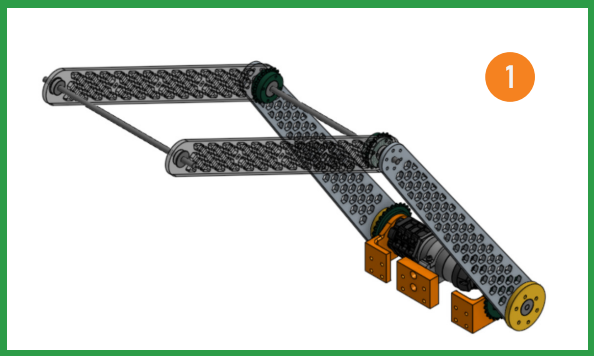
În urma mecanismului de colectare clasic, cu un model de gheară, am încercat pentru prima oară să punem în practică și un sistem bazat pe ***Surcical Tubing***, acționat de un motor **Andymark Neverest Orbital 3.7**. Primul ax de tuburi ridică pixelii, iar apoi cel de-al doilea îi introduce în interiorul unor sloturi realizate special în funcție de forma pixelilor, transmisia motor-ax cu tuburi făcându-se prin lanț, cu **raport 1:1**.

Capturi din OnShape cu Intake-ul pe Surgical Tubing (1).

Avantaje:	Dezavantaje:
• Nu necesită alinierea cu pixelii.	• Foarte voluminos; • Colectarea durează mult, din cauza rampei de colectare neoptimizate.

Îmbunătățiri:

- Înlocuirea vârfului rampei de pe sol cu o rolă care se învârtă în sens invers, pentru a ridica pixelul de pe sol;
- Înlocuirea lanțului cu curele mai subțiri pentru a reduce dimensiunile mecanismului de antrenare și pentru a îl ușura.



Braț:

Pornind de la ideea perfecționată anul trecut, cea de ***Double Reversed Virtual Four Bar***, am decis să realizăm și o iterație similară. Transmisia se face pe lanț, **raport 1:1**. Acesta este acționat de un motor **REV UltraPlanetary** cu **raport de 60:1**.

Capturi din OnShape cu brațul (1).

Avantaje:	Dezavantaje:
• Mecanism, eficient și rapid; • Ușor de programat.	• Instabil din pricina complexității.

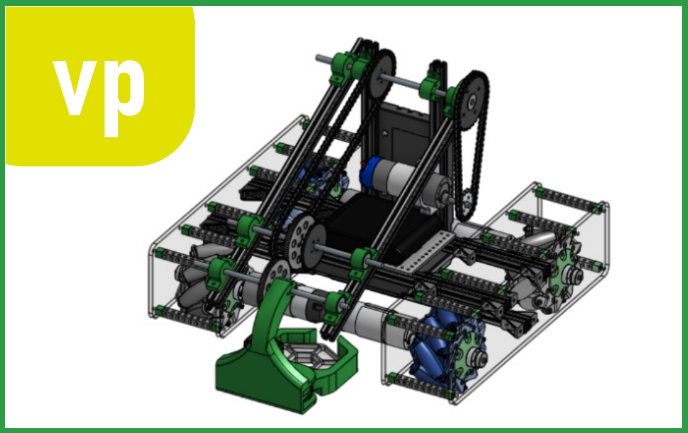
Îmbunătățiri:

- Rigidizarea ansamblului, pentru o acuratețe mai mare a brațului.

Robot Didactic:

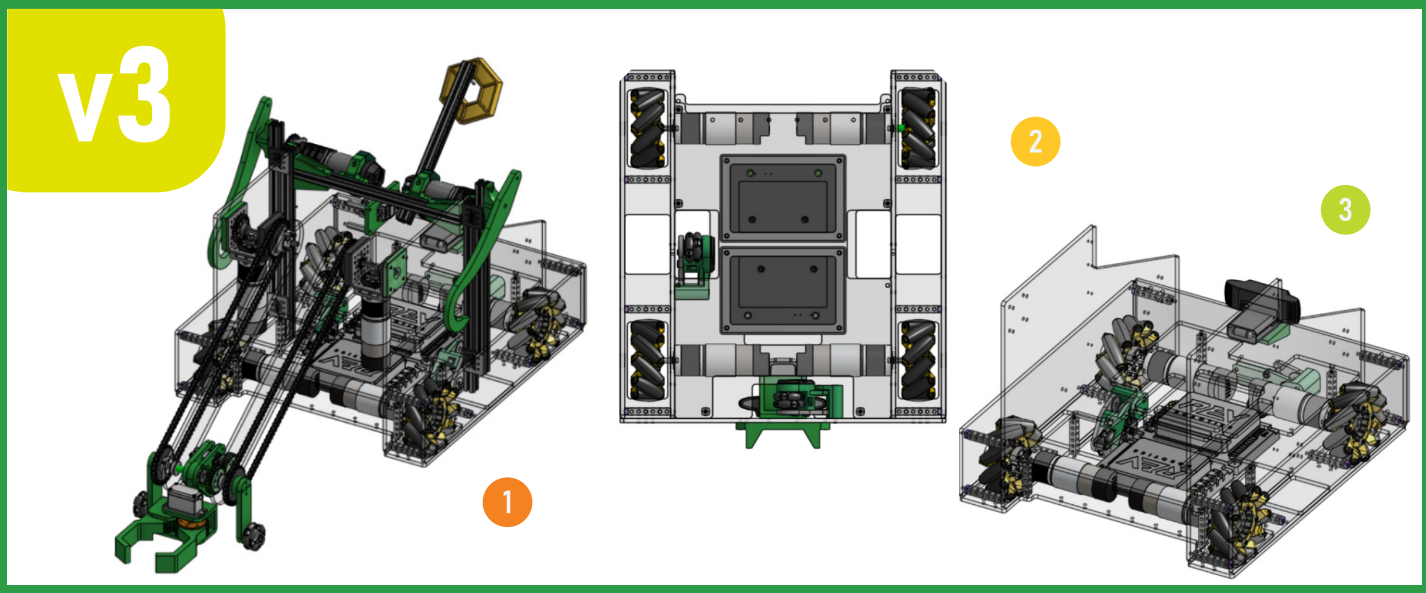
Capturi din OnShape cu robotul de prezentări complet, din vedere izometrică (1).

Robotul de prezentări a fost încă un proiect pentru **membrii nou-veniți**, întrucât am divizat, pentro o mai bună performanță pe termen lung, echipa tehnică (Hardware și Software) în două grupe, fiecare având **task-uri bine-definite** și interconectate, deși au un grad mare de independență, însă, o parte din ele au putut să fie realizate individual.



Robot v3 - League Meet #1:

Pentru primul **League Meet** (25 noiembrie - ZILELE ROBOTICII, la Ploiești, organizat de echipele **#15993 Inf0(1)Robotics și #17962 Ro2D2**) am hotărât să ne concentram pe misiunile ce contribuie cel mai mult la **Ranking** (Autonomie și End Game).



Robot:

Capturi din OnShape cu mecanismele și robotul v3 - League Meet #1:

- robotul complet din vedere izometrică (1);
- șasiul din vedere aeriană (2);
- șasiul din vedere izometrică (3);

Calcul viteza și viteza unghiulară maxime:

$$v_{max} = \omega_{max} \cdot d/2 \simeq 1.71 m/s$$
$$\omega_{max} = 340 RPM \simeq 35.60\, rad/s$$
$$d = 9.6\, cm = 0.096 m$$

Îmbunătățiri:

- Mai compact, pentru o mișcare ușoară pe teren.

Șasiu:

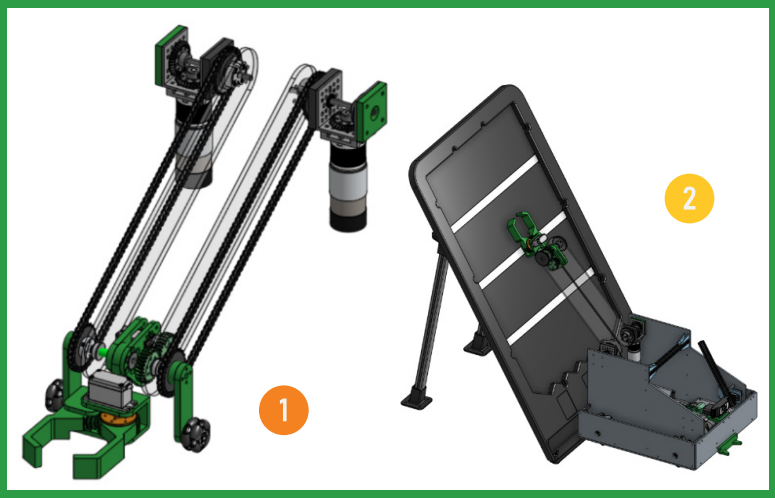
Șasiul trebuie să fie cât mai jos, pentru a acomoda celelalte sisteme, astfel încât să facem trecerea pe sub ***Stage Door***. Am optat să prindem roțile direct de motoare și elecronica necesara să fie între acestea, dimensiunile șasiului ajungând la **415mm x 380mm x 100mm** (L x l x h).

Avantaje:	Dezavantaje:
• Precizie foarte ridicată, datorită sistemului *Direct Drive*, ceea ce cauzează un joc al roților minim.	• Destul de mare, ceea ce îngreunează mișcarea pe teren.

Lift:

Observând potențialul brațului de la Kickathon, pentru prima competiție oficială am decis să folosim un sistem asemănător. Am utilizat, însă, motoare mult mai fiabile, rapide și puternice, **plăci de policarbonat** (ce ne-au permis sa atingem cel de-al doilea *Set Line*) și am înlocuit bucșele cu **rulmenți**, având, per total, un ansamblu mai stabil.

Acesta se bazează pe un ***Virtual Four Bar***, mecanism care permite ghearei să rămână **paralelă** cu solul și simplifică calculele necesare pentru poziția încheieturii.



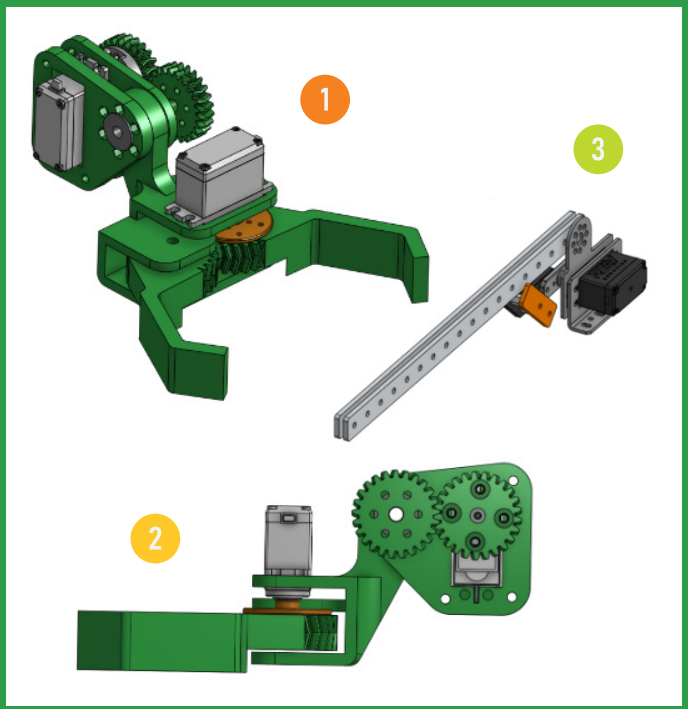
Avantaje:	Dezavantaje:
• Acționare rapidă (340 RPM); • Atingerea celui de-al doilea *Set Line*; • Ușor de programat/folosit; • Ansamblu stabil.	• Poziție imprecisă și PID instabil, cauzat de rezoluția redusă a encoderelor de pe motoare; • Dimensiuni foarte mari, ce reduc mobilitatea în teren.

Capturi din OnShape cu brațul robotului v3 - League Meet #1:
- brațul din vedere izometrică (1);
- detaliu cu brațul în fața *Backdrop*-ului (2).

Îmbunătățiri:

- Mai compact, pentru a favoriza manipularea și mișcarea pe teren.

Gheară:



Testele realizate pe Robotul v2 (Robot de Prototipări) ne-au arătat că un mecanism de tip **gheară** este mult mai precis și mai ușor de implementat decât unul ce colectează activ.

De aceea, am ales un design simplu, în care doi clești sunt acționați simultan de **un singur servo**, cel de-al doilea servo având rolul de a ajusta unghiul față de sol/*Backdrop*.

Avantaje:	Dezavantaje:
• Plasarea ușoară pe *Backdrop*; • Posibilitatea de a alege pixelul ce este plasat; • Suprafață de colectare ridicată; • Posibilitatea de a colecta doi pixeli în același timp.	• Sistemul de ajustare a unghiului este instabil, asta ducând la colectarea inconsistentă a pixelilor; • Colectatul a doi pixeli simultan este dificilă și necesită multe corecții din partea *driver*-ului.

Capturi din OnShape cu mecanismele v3 - League Meet #1:
- gheara din vedere izometrică (1);
- gheara din vedere laterală (2);
- lansatorul de dronă din vedere izometrică (3).

Îmbunătățiri:

- Să o modificăm astfel încât să colecteze doi pixeli așezați unul lângă altul față de unul deasupra celuilalt.

- Am analizat meciuri în care *Human Player*-ul plasează pixelii astfel și am constatat dificultatea suplimentară.

Lansator de dronă:

Pentru a lansa drona, am folosit un sistem asemănător cu cel de la KickAthon, care se bazează pe **elastice**. Acesta constă într-un **profil de aluminiu** pe post de ghidaj și un servo care eliberează elasticul. Lansatorul stă **fix** în interiorul robotului, la un unghi **45 de grade**, care să permită o boltă mai accentuată a traiectoriei și, astfel, **reduce glisarea** înainte a avionului după contactul cu solul, aterizând la un unghi mai apropriat de verticală. Cu acest sistem am reușit să punctăm drona, în mod constant, în zona 1, întrucât am remediat toate dezavantajele constatate anterior.

Cățărat:

Acest ansamblu folosește două motoare **REV Ultraplanetary** cu raport de **60:1**, care funcționează asemănător sistemului ***rack & pinion*** (cremalieră), în care pinionul de pe axul motor acționează lanțul fixat pe profil de aluminiu, astfel obținând mișcare liniară. Atașat de capătul profilului este un **cârlig** care pivotează liber. În timp ce motorul ridică mecanismul de cățărat, o **sfoară** atașată de cârlig și șasiu se tensionează și ridică cârligul în poziția optimă pentru cățărat.

Avantaje:	Dezavantaje:
• Mecanism sigur și eficient; • Ușor de programat.	

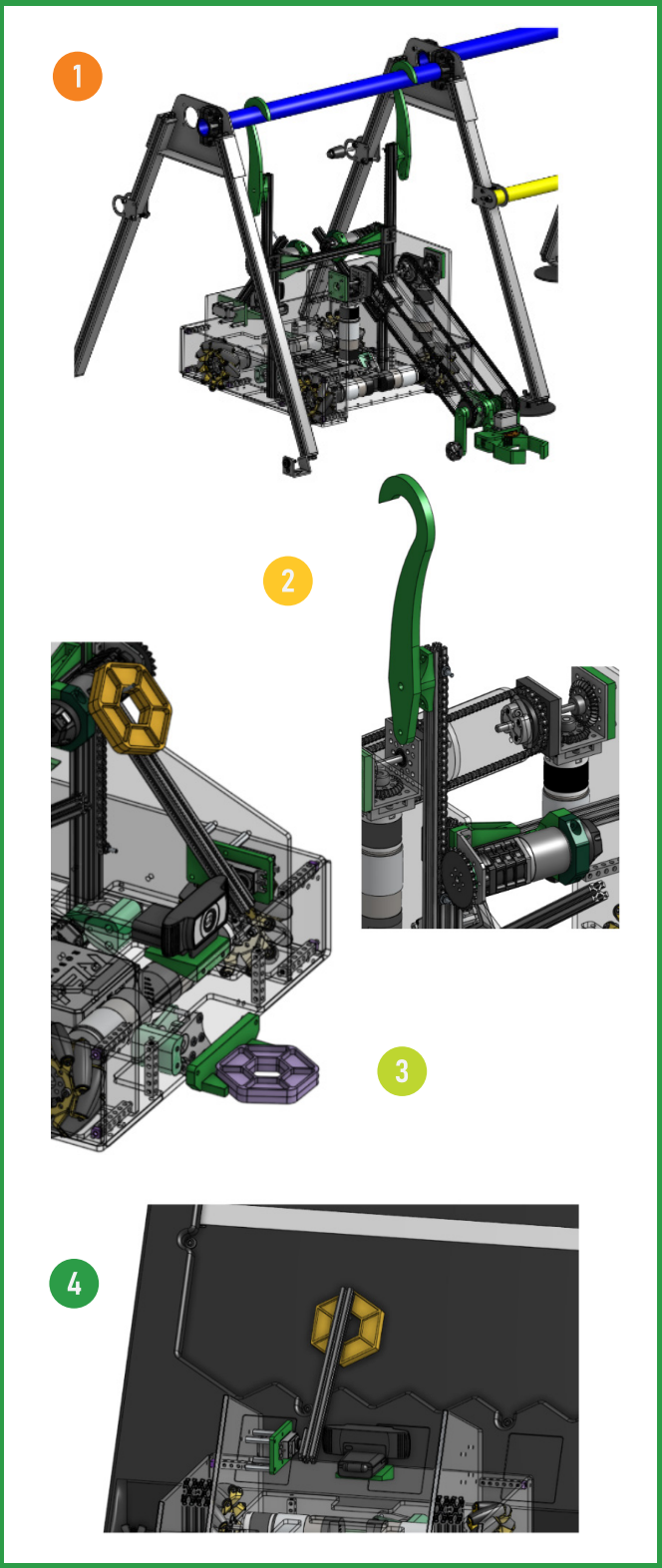
Îmbunătățiri:

- Optimizarea pozitiei sforii.

Mecanisme Autonomie:

Perioada de Autonomie este una dintre, dacă nu chiar cea mai importantă din meci. De aceea am optat să folosim sisteme extrem de simple și fiabile pentru a obține constant cele **50 de puncte** dorite în strategie.

"Ciocul" este fixat pe robot și are rolul de a împinge **pixelul mov** în poziție pe *Spike*. Pentru punctarea **pixelului galben**, am folosit un braț simplist montat de un servo care este acționat după ce robotul s-a aliniat pe *Backdrop*.



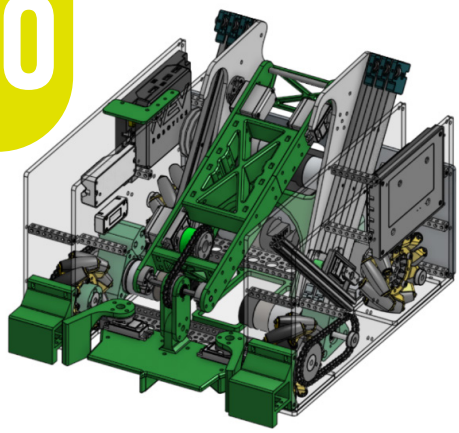
Capturi din OnShape cu mecanismele v3 - League Meet #1:

- robotul complet cățărat (1);
- cremaliera (2);
- ciocul și brațul simplu al pixelului galben (3);
- brațul punctând pe *Backdrop* (4).

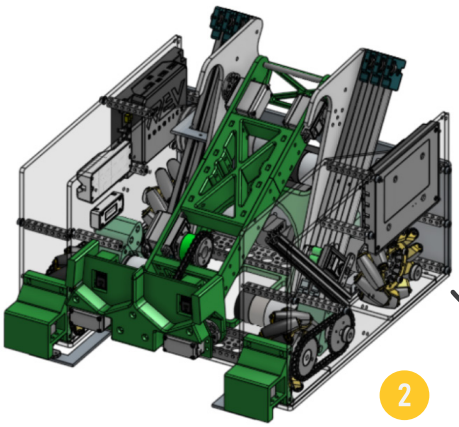
Robot v4 - LM #2/#3, Regionala #1 & Națională:

În urma primului League Meet am sesizat nevoia de a puncta mai consistent și în perioada de **TeleOp**. În concluzie, am regândit robotul astfel încât să aibă dimensiuni mai reduse și mecanisme mai **simple** și, totodată, **rapide**, care să permită o performanță mai stabilă mecanic și un potențail mai ridicat pentru programare.

v4.0



1



2

v4.1

Locul #7 în FTC
Top 25 January

Roboți:

Capturi din OnShape cu roboții v4:

- robotul v4.0 din vedere izometrică (1);
- robotul v4.1 din vedere izometrică (2).

Șasiu:

Am folosit același șasiu ca la versiunea anterioară, însă cu îmbunătățirile aferente. Pentru a elibera spațiul necesar mecanismelor și a reduce amprenta la sol, am fost nevoiți să angrenam roțile din față cu **lanț** și am utilizat doar două roți de odometrie, astfel scurtând lungimea șasiului cu 5,5 cm: **360mm x 380mm x 100mm** (L x l x h).

Avantaje:	Dezavantaje :
• Constituție compactă; • Majoritatea elementelor importante sunt protejate de laterale (de exemplu- Control hub, Expansion hub).	• Posibilitatea blocării *Truss*-ului între plăci.

Calcul viteza și viteza unghiulară maxime:

$$v_{max} = \omega_{max} \cdot d/2 \simeq 1.71 m/s$$
$$\omega_{max} = 340 RPM \simeq 35.60\, rad/s$$
$$d = 9.6\, cm = 0.096 m$$

Îmbunătățiri:

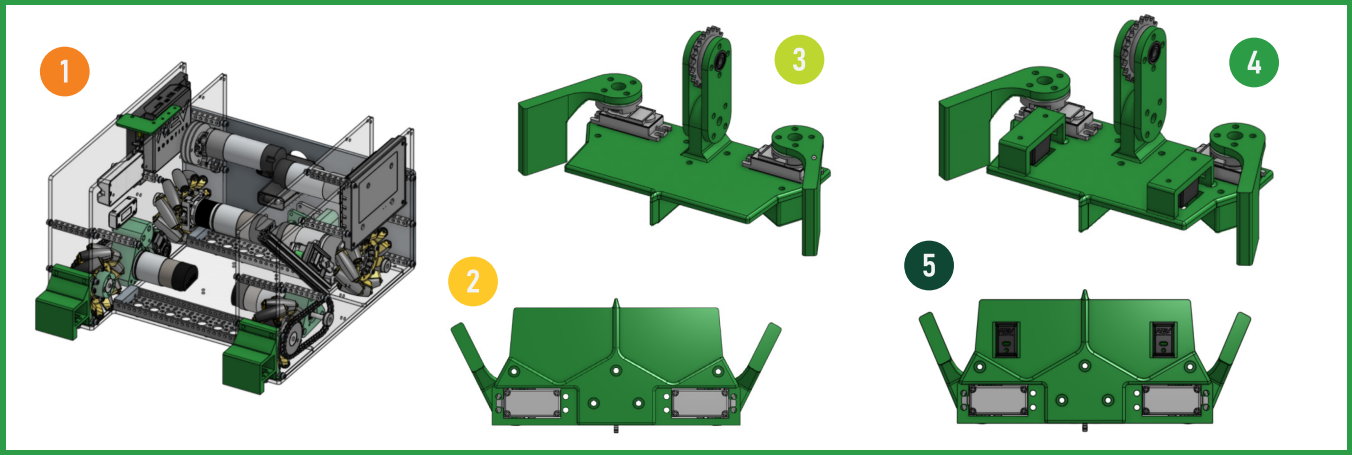
- Înlocuirea motoarelor cu unele mai rapide.

Gheară:

v4.0

În urma testelor am realizat că un mecanism de tip **clește** este mult mai simplu, fiabil și adecvat, timpul de iterație fiind foarte scurt. Față de primul League Meet, aceasta gheară poate să controleze colectarea și plasarea fiecărui pixel individual, fiind acționată de **2 servouri GoBilda Speed**. Pentru a se putea adapta la unghiul *Backdrop*-ului folosim un servo.

Avantaje:	Dezavantaje:
• Mai ușor de realizat Mozaicuri, putând plasa pixelii individual.	• Suprafață de colectare restransă reduce performanța în TeleOp.



Capturi din OnShape cu roboții v4 și mecanismele lor:
- șasiul din vedere izometrică (1);
- gheara v4.0 din vedere de jos (2);
- gheara v4.0 din vedere izometrică (3);
- gheara v4.1 din vedere izometrică (4);
- gheara v4.1 din vedere de jos (5).

Îmbunătățiri: Adăugarea senzorilor pentru a putea colecta automat.

v4.1 Pentru a eficientiza colectarea în TeleOp am adăugat **senzori de culoare**. Astfel gheara se închide automat când detectează prezența unui pixel, iar driverii sunt anunțați prin **vibrația controllerelor**.

Braț:

Gheara trebuie să ajungă din fața robotului, în spatele lui. Pentru a îl acționa am folosit **3 servouri GoBilda Torque**: două pentru articulația de tip umăr și al treilea la încheietură. Această **dublă articulație** ne permite să plasăm pixeli direct pe primul rând al *Backdrop*-ului, pentru a nu risca să ricoșeze pe altă poziție. Totodată, acesta poate să așeze pixelii încât să atingă primul *Set Line* fără a utiliza glisierele.

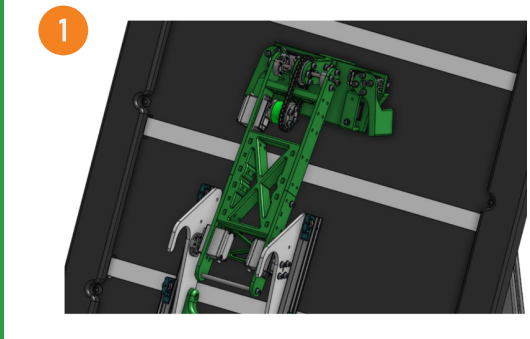
Avantaje:	Dezavantaje:
• Utilizarea unui intake care să fie și outtake scade șansele de fail; • Brațul se ajustează în funcție de unghi, astfel încât putem plasa pixeli inclusiv pe cea mai de jos linie, fără să riscăm să sară pe alte sloturi.	• Folosește multe servouri. • Lungimea ne limitează viteza cu care îl putem folosi.

Îmbunătățiri:

- Înlocuirea segmentelor de plăci printate 3D cu plăci din policarbonat, astfel reducând numărul de șuruburi utilizate și ușurând brațul;
- Înlocuirea servourilor GoBilda Torque cu unele mai rapide și mai puternice, pentru a optimiza timpul de punctare.

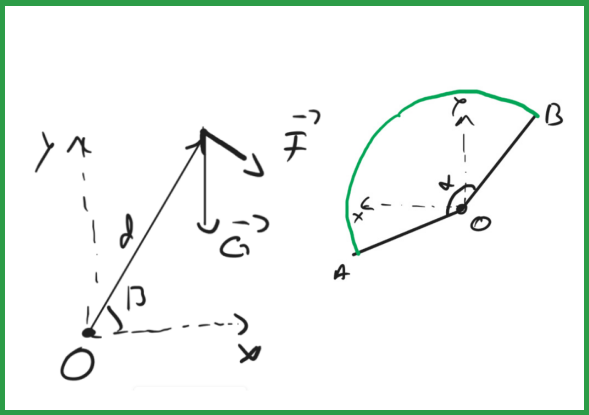
În timpul testelor am întâmpinat o problema: servo-urile ce acționau brațul s-au rupt. Analizând uzura am dedus că apărea o problemă la finalul mișcării, și anume **suprasarcina mecanismului** când acesta se oprește din cauza frânării bruște. Pentru a **remedia** eroarea, am optat să implementăm un *Motion Profiler* care să **oprească lin brațul.** Astfel, am crescut timpul de frânare cu 0.13s, noua valoare a cuplului fiind:

$$\tau' = 1.22Nm < 3.36Nm = 2\tau_{GoBilda}$$



Capturi din OnShape cu brațul robotului v4.1 pe *Backdrop* (1)

$$F = \frac{\Delta p}{\Delta t_s} = \frac{m \times v}{\Delta t_s} = \frac{m \times d \times \omega}{\Delta t_s} = \frac{m \times d}{\Delta t_s} \times \frac{\angle AB}{\Delta t_r} =$$
$$= \frac{m \times d \times 2\pi \times d \times (\alpha/360°)}{\Delta t_s \times \Delta t_r} = \frac{2\pi \times m \times d^2 \times \alpha}{\Delta t_s \times \Delta t_r \times 360°}$$
$$F = 3.77N(\Delta t_s = 0,2s; \Delta t_r = 1s)$$
$$\vec{\tau} = \vec{M}_G + \vec{M}_F = \vec{G} \times \vec{d} + \vec{F} \times \vec{d}$$
$$\tau = G \times d \times sin\angle(\vec{G}, \vec{d}) + F \times d \times sin\angle(\vec{F}, \vec{d})$$
$$\tau = m \times g \times d \times cos\beta + F \times d$$
$$\tau = d(m \times g \times cos\beta + F)$$
$$\tau = 3.67Nm > 3.36Nm = 2\tau_{GoBilda}$$



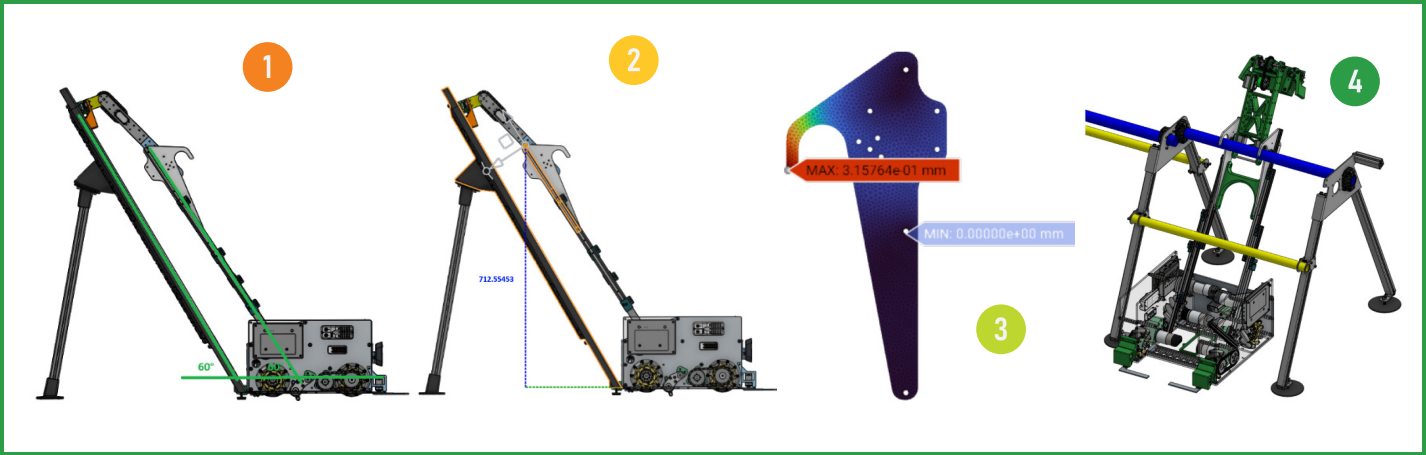
Glisiere:

Pentru a simplifica plasarea pixelilor pe *Backdrop* am decis să montăm brațul pe **glisiere paralele cu *Backdrop*-ul**, astfel, reducând șansa de a depuncta sau a strica mozaicurile. Pentru acest mecanism am folosit **două seturi de trei glisiere MISUMI SAR 230**, având o extensie maximă de **540mm**.

Avantaje:	Dezavantaje:
▪ Putem să punctăm oriunde pe *Backdrop* fără să deplasăm brațul.	▪ Viteza nu este optimizată, căci cele mai rapide motoare care încap între glisiere au raport de 40:1.

Îmbunătățiri:

- Înlocuirea motoarelor cu unele mai rapide, pentru eficiență sporită în teren;
- Mărirea diametrului mosorului pentru a mișca mai rapid glisierele.



Capturi din OnShape cu robotul v4.1 și mecanismele lui:

- robotul din vedere laterală, cu glisierele extinse total, paralele cu *Backdrop*-ul (1);
- robotul din vedere laterală, cu glisierele extinse total: 54cm (2);
- analiză cu element finit a deformației cărligului pentru cățărat în nTop (3);
- robotul cățărat din vedere izometrică (4).

Cățărare:

Deoarece un aspect important în conceperea acestui robot a fost să **reducem complexitatea mecanică** și să fie cât mai compact, am decis să ne folosim de glisiere. Această modificare necesită motoare puternice, pentru a putea ridica robotul de pe sol, iar motoarele pe care le folosim (AndyMark Neverest 40), datorită limitării spațiului, sunt ideale.

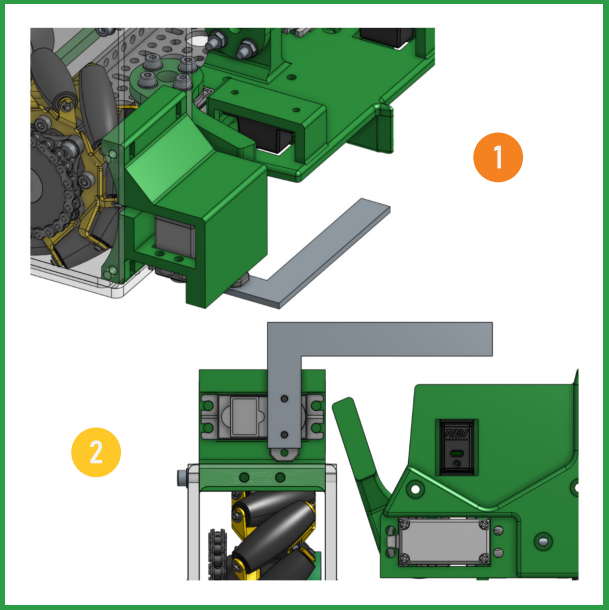
Avantaje:	Dezavantaje:
• Cățărare rapidă în doar două secunde; • Poziție foarte stabilă cât timp stă cățărat.	• Stă atârnat doar cât timp are curent în motoare, devenind un inconvenient la terminarea meciului.

Mecanisme autonomie:

Pentru a putea lua pixeli din *Stack*, gradele de libertate ale ansamblului de manipulat pixelii fac ca poziția ghearei să fie suficient de imprecisă încât să nu putem colecta. Astfel, am simțit nevoia implementării unui **sistem anex** care să ne ajute în Autonomie.

Acest sistem constă în două palete subțiri ce sunt acționate de câte un servo. Momentul în care robotul împinge două *Stack*-uri în perete poate să extragă pixelul de la baza fiecăruia. Acest mecanism ne-a ajutat să îmbunătățim fiabilitatea colectării din *Stack* **de 4.5 ori**.

Avantaje:	Dezavantaje:
• Colectează constant pixeli din *Stack*; • Nu cauzează penalizări, chiar dacă nu colectează.	• Dacă robotul deviază de la traseu prea mult este posibil să nu colecteze.

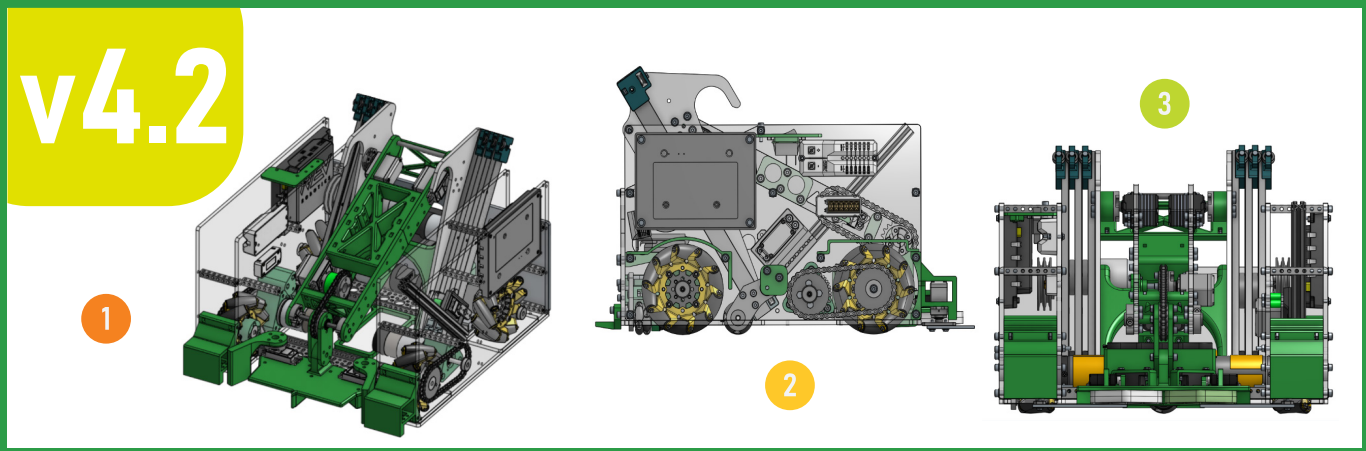


Capturi din OnShape cu „gheruțele" din Autonomie, vedere izometrică (1) și de jos (2).

Lansator de dronă:

Mecanismul îmbunătățit la League Meet-ul #1 s-a dovedit a fi unul foarte **simplu** și, totodată, extrem de **eficient**, corespunzând necesităților robotului nostru, atât la nivel de fiabilitate și execuție, cât și ca posibilitate de **reglare înainte de meci**.

Pentru etapa Națională am ales să lucrăm și la varianta îmbunătățită, v4.2, a robotului pentru Regionala #1.



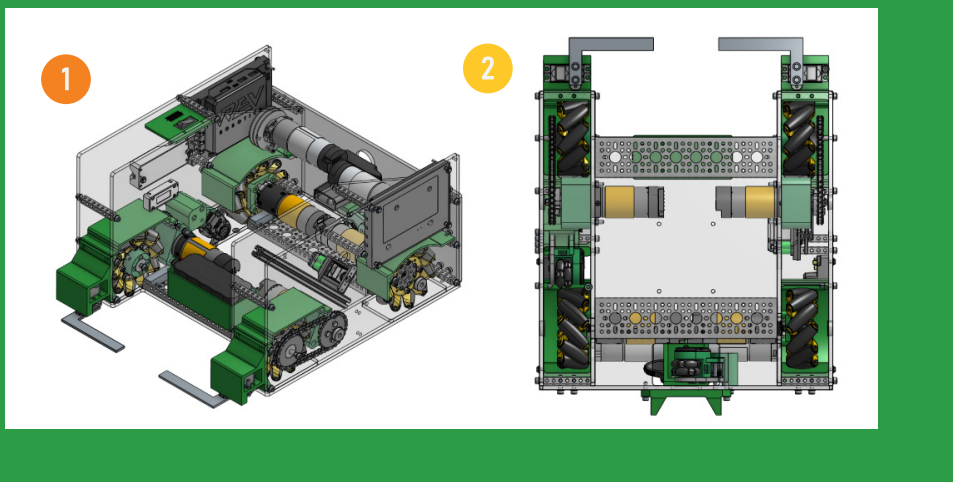
Robot:

Capturi din OnShape cu robotul v4.2:
- robotul v4.2 din vedere izometrică (1);
- robotul v4.2 din vedere laterală (2);
- robotul v4.2 din vedere frontală (3).

Șasiu:

Luând în considerare problemele pe care le-am întâmpinat în timpul Regionalei #1, după cele mai multe meciuri consecutive în fomat competițional, am adus următoarele modificări șasiului:

- Am înlocuit motoarele cu raport de 20:1 cu unele cu raport de **13.7:1**, ce au crescut viteza cu 21%;
- Am adăugat **protecții** pentru a evita agățarea cablurilor în roțile mecanum;
- Am mutat bateria în partea din față a robotului, scopul fiind acela de a muta **centrul de greutate** cât de mult posibil spre față, pentru a ne putea deplasa mai ușor cu brațul extins;
- Pentru a ne asigura că evităm erorile și că începem meciul cu bateria încărcată, am adăugat un **voltmetru**;
- Am schimbat prinderea pentru odometrie cu una mai robustă și mai stabilă, pentru o deplasare mai precisă în timpul perioadei de Autonomie.

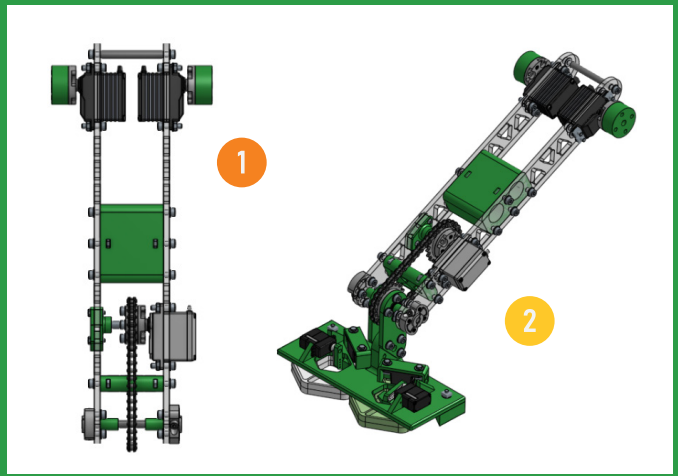


Capturi din OnShape cu robotul v4.2 și mecanismele lui:
- șasiul din vedere izometrică (1);
- șasiul din vedere de sus (2).

Calcul pentru viteza și viteza unghiulară maxime față de cele de la versiunea anterioară:

$$v_{max} = \omega_{max} \cdot d/2 \simeq 1.71 m/s$$
$$\omega_{max} = 340 RPM \simeq 35.60\, rad/s$$
$$d = 9.6\, cm = 0.096 m$$

Avantaje:	Dezavantaje:
• Este compact; • Elementele critice sunt protejate; • Viteza crescută cu 21%.	• Deplasarea (în principal accelerarea) consumă mult mai mult curent, ceea ce înseamnă că trebuie să avem bateria încărcată înainte de fiecare meci;



Braț:

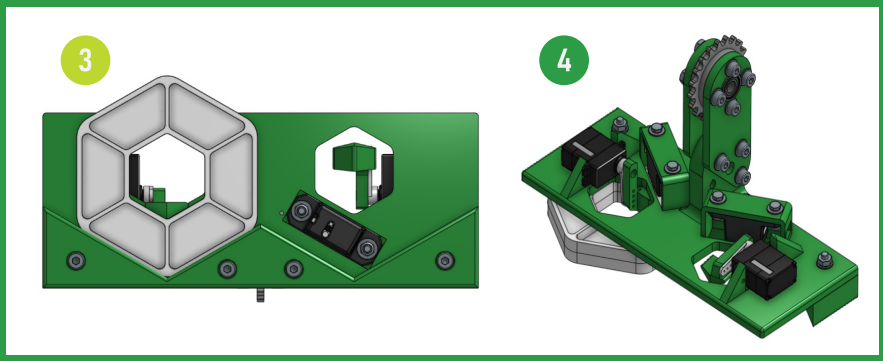
Am schimbat structura brațului pentru a **reduce greutatea** și am diminuat tendința robotului de a se răsturna când glisierele sunt ridicate complet. Apoi am înlocuit servourile GoBilda Torque cu Axon MAX+, îmbunătățind viteza cu **44%**. Am păstrat mecanismul de încheietură, întrucât acesta ușurează acțiunea de a plasa pixelii pe *Backdrop*.

Avantaje:	Dezavantaje:
• Destul de ușor de utilizat și fiabil.	• Utilizează multe servouri.

Gheară:

Pentru a plasa pixelii mai ușor pe *Backdrop* am fost nevoiți să schimbăm mecanismul de prindere. Astfel am realizat colectarea pixelilor din **interiorul** acestora. Am păstrat **senzorii de culoare** de la varianta precedentă, pentru a ne asigura colectarea pixelilor, deoarece nu avem suficientă vizibilitate asupra acestora. În plus, am înlocuit servourile GoBilda Speed cu Axon Micro+, pentru a face ansamblul cât **mai mic** și mai ușor posibil.

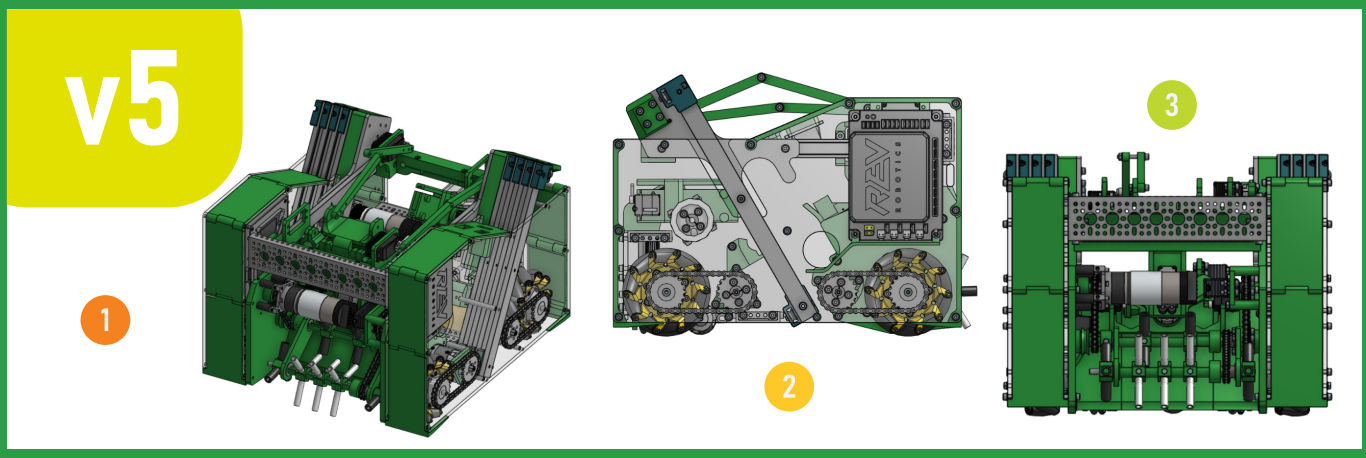
Capturi din OnShape cu robotul v4.2 și mecanismele lui:
- brațul din vedere frontală (1);
- brațul din vedere izometrică (2);
- gheara din vedere de jos (3);
- gheara din vedere izometrică(4).



Avantaje:	Dezavantaje:
• Ansamblu mult mai ușor, cu 148g mai puțin; • Colectare mai rapidă cu aproximativ 30%, datorită servourilor.	• Necesită precizie pentru a colecta.

Robot v5 - Națională:

După etapa Regională am constatat nevoia unui robot **mai rapid**, care să ajute *Drive Team*-ul în ceea ce privește precizia coletării și a punctării, dar și care diminuează riscul problemelor neprevăzute apărute în contactul agresiv cu ceilalți roboți, elementele de joc sau elementele de pe teren.



Robot:

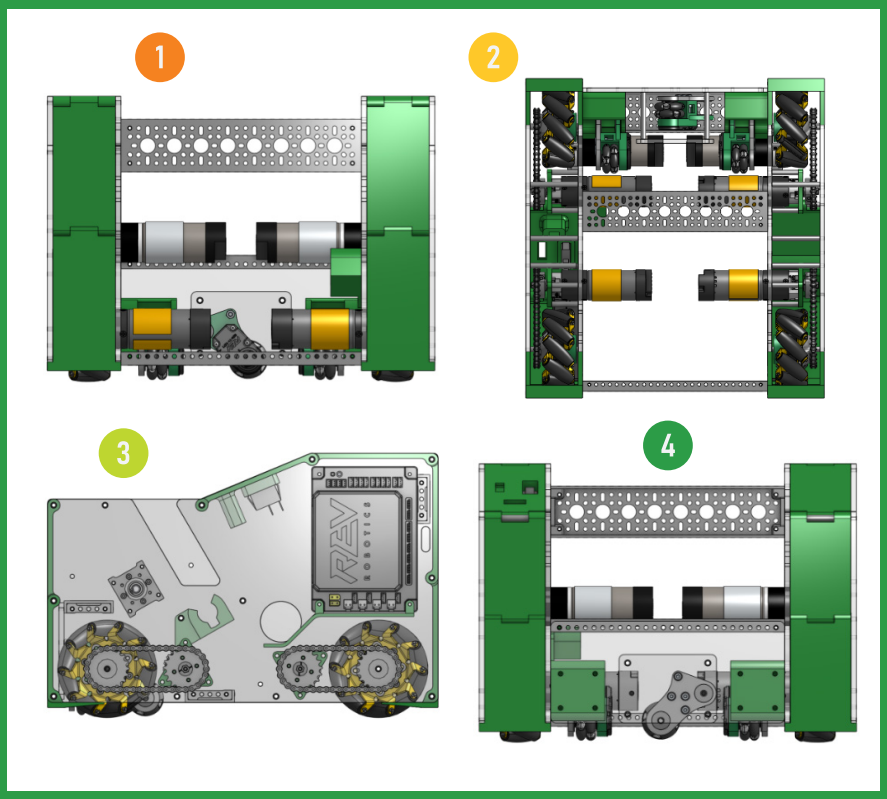
Capturi din OnShape cu robotul v4.2:

- robotul v5 din vedere izometrică (1);
- robotul v5 din vedere laterală (2);
- robotul v5 din vedere frontală (3).

Șasiu:

În urma etapei Regionale ne-am propus să rezolvăm problemele pe care le-am întâmpinat și am adus următoarele modificări șasiului:

- Am folosit motoare noi, cu **raport 13.7:1**, ce au crescut viteza cu 21%;
- Am făcut **protecții** pentru a nu ne agăța de Truss sau alți roboți;
- Am realizat un panou care conține toate elementele la care avem nevoie de acces facil la robot: Power Button, voltmetru și un USB pentru a programa robotul.



Capturi din OnShape cu robotul v5 și mecanismele lui:
- șasiul din vedere frontală (1);
- șasiul din vedere de jos (2);
- șasiul din vedere laterală (3);
- șasiul din vedere de jos (4).

Avantaje:	Dezavantaje:
• Este compact; • Elementele sunt protejate, fiind acoperite din toate direcțiile; • Este cu 21% mai rapid.	• Accelerarea consumă mult mai mult curent, raportul motoarelor fiind mai mic;

Calcul pentru viteza și viteza unghiulară maxime:

$$v_{max} = \omega_{max} \cdot d/2 \simeq 1.71 m/s$$
$$\omega_{max} = 340 RPM \simeq$$
$$\simeq 35.60\, rad/s$$
$$d = 9.6\, cm = 0.096 m$$

Glisiere:

Am optimizat timpul de ridicare al glisierelor prin înlocuirea **gearboxului** motoarelor. Am folosit **20:1** pentru a ne **dubla viteza** de punctare și de cățărare și, de asemenea, am utilizat două seturi de 4 glisiere MISUMI SAR 230, având o extensie maximă de **720mm**.

Avantaje:	Dezavantaje:
• Poate atinge Setline 3, doar din mișcarea liniară.	• Motoarele trebuie să stea în tensiune pentru a mentine poziția, raportul fiind mai mic.

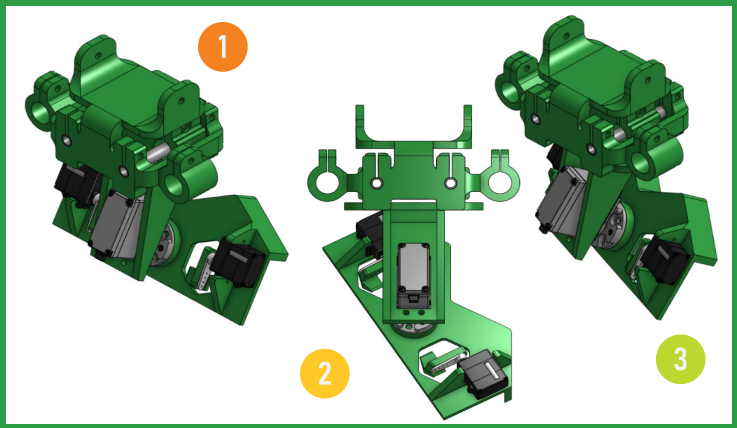
Capturi din OnShape cu robotul v5 și mecanismele lui:
- robotul cu glisierele extinse din vedere laterală (1);
- robotul cu glisierele extinse din vedere frontală (2).

Îmbunătățiri: Realizarea unui mecanism care să le blocheze în poziția de cățărare.

Outtake - End Effector:

Pentru a prinde și plasa pixelii am ales un mecanism de tip **clește**, care îi apucă din interior, și care ne permite să realizăm un pivot, deoarece **nu obstrucționează** eliberarea laterală a pixelului.

Un alt lucru important este **suspensia**, ce ne ajută să diminuăm până la **6 cm și 20 de grade de eroare**, atât în perioada de Autonomie, cât și în TeleOp pentru a nu avea nevoie de foarte multă precizie.



Capturi din OnShape cu robotul v5 și mecanismele lui:
- outtake-ul din vedere izometrică (1);
- outtake-ul din vedere frontală (2);
- outtake-ul din vedere izometrică rotit (3).

Îmbunătățiri:

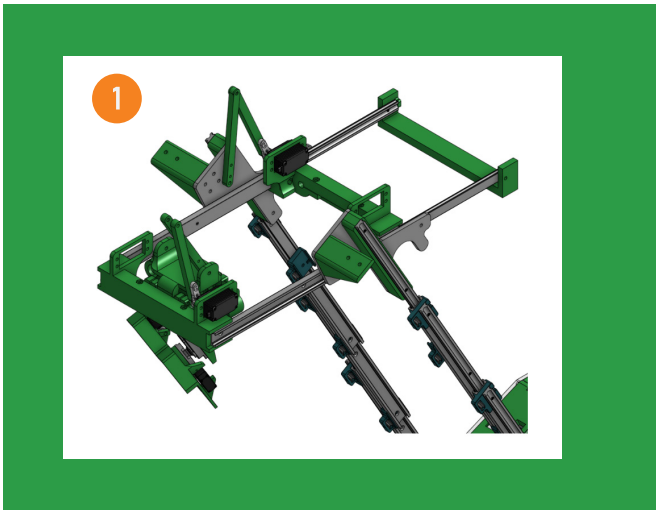
- Optimizarea traseului cablurilor pentru a putea roti complet pivotul;
- Creșterea distanței care poate fi amortizată.

Avantaje:	Dezavantaje:
• Elimină necesitatea alinierii exacte pe Backdrop; • Permite așezarea mai precisă a pixelilor datorită pivotului.	• Pivotul nu se poate roti la 180 de grade, lucru care ne-ar permite să salvăm timp.

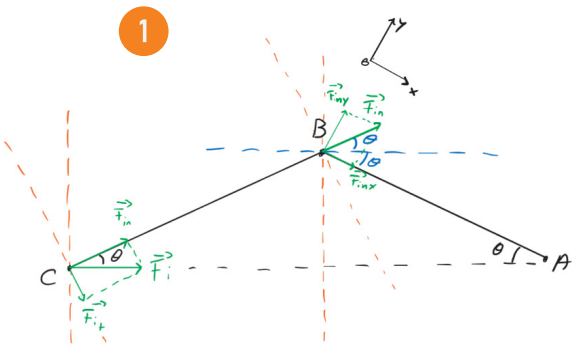
Outtake - Horizontal Extension:

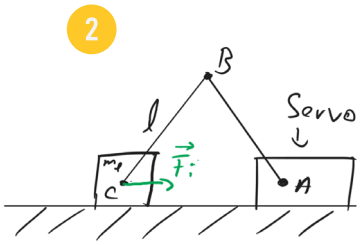
În urma folosirii pentru o lungă perioadă de timp a unui braț pentru a transfera pixelii dintr-o parte în alta a robotului, am realizat că acesta poate fi înlocuit cu o **extensie orizontală**, care micșorează forțele suportate de servouri.

Avantaje:	Dezavantaje:
• Reduce riscul de a agăța poarta; • Aduce rapid pixelii în poziția de punctare.	• Nu poate puncta fără acțiunea glisierelor.



Captură din OnShape cu Outtake-ul - Horizontal Extention din vedere izometrică (1).





3

$$F_{in} = F_i \cos\theta$$

$$F_{iny} = F_{in} \sin 2\theta = F_i \cos\theta \sin 2\theta = 2F_i \sin\theta \cos^2\theta$$

$$M = lF_{iny} = 2lF_i \sin\theta \cos^2\theta$$

Ca M să fie maxim =>

$$(2sin\theta cos^2\theta)' = 0 <=> 2cos^3\theta - 4cos\theta sin^2\theta = 0 <=> 2cos\theta(cos^2\theta - 2sin^2\theta) = 0$$

$$=> 1)\ cos\theta = 0 => M = 2lF_i sin\theta \times 0 = 0 \neq \text{maxim}$$

$$2)\ cos^2\theta - 2sin^2\theta = 0 => ctg^2\theta = 2 => \theta = \pm\sqrt{2} => 2sin\theta cos^2\theta \simeq 0.77$$

$$M_{max} = 2lF_i sin\theta cos^2\theta = 2l \times m \times a \times sin\theta cos^2\theta = 0.12 \times 0.3 \times 1.58 \times 0.77 \simeq 0.043Nm$$

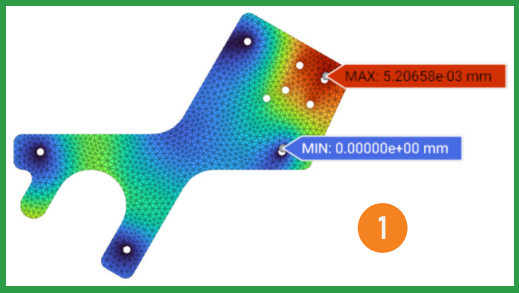
4

- Reprezentarea și descompunerea forțelor ce acționează asupra linkage (1);
- Model simplificat al linkageul (2);
- Calculele corespunzătoare graficelor (3), (4);

Cățărare:

Pentru a obține punctele cățăratului am **păstrat** cârligelele pe glisiere, astfel încât să nu avem nevoie de un mecanism independent.

Avantaje:	Dezavantaje:
• Foarte simplu și rapid; • Poziție stabilă.	• Consumă foarte mult curent.

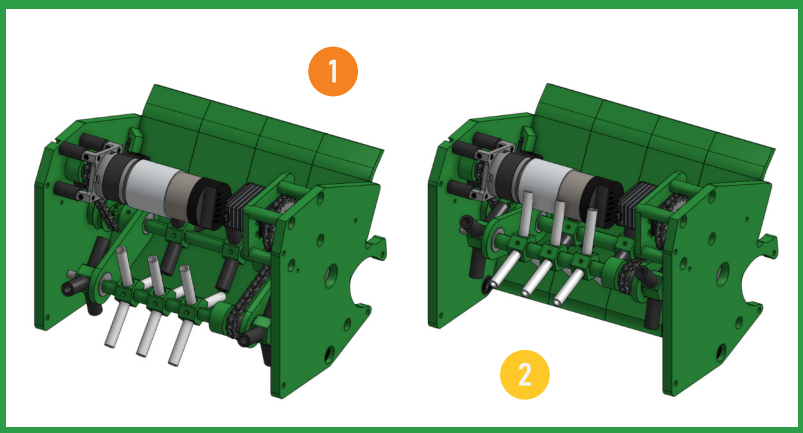


Captură de ecran a analizei cu element finit a deformației cârligului pentru cățărat în nTop (1).

Intake:

Una dintre problemele întâmpinate la Regionala #1 a fost **colectatul**, deoarece necesită precizie prea mare și scapă pixelii care se agață în *tile*-uri. Astfel, am considerat că una din cele mai bune variante este trecerea la **intake activ**.

Pentru a ajunge la această versiune am trecut prin alte **6 iterații**, cea actuală constând în 2 seturi de tuburi și un *bottom roller* care se învârte în sens opus acestora.



Rampa este plasată la un unghi de **60 de grade față de sol** și astfel eliminăm nevoia de a articula outtake-ul. Primul set de role (cel fix) are rolul de a ajuta pixelii să **urce pe rampă** și să intre în outtake fără a fi dați afară din robot. Al doilea set de tuburi (cel mobil) are ca scop **posibilitatea de a colecta** atât din *Stack*, cât și de pe *tile*-uri fără dificultăți, adaptând înălțimea în funcție de nevoie. *Bottom roller*-ul ajută la ridicarea de pe sol a pixelilor.

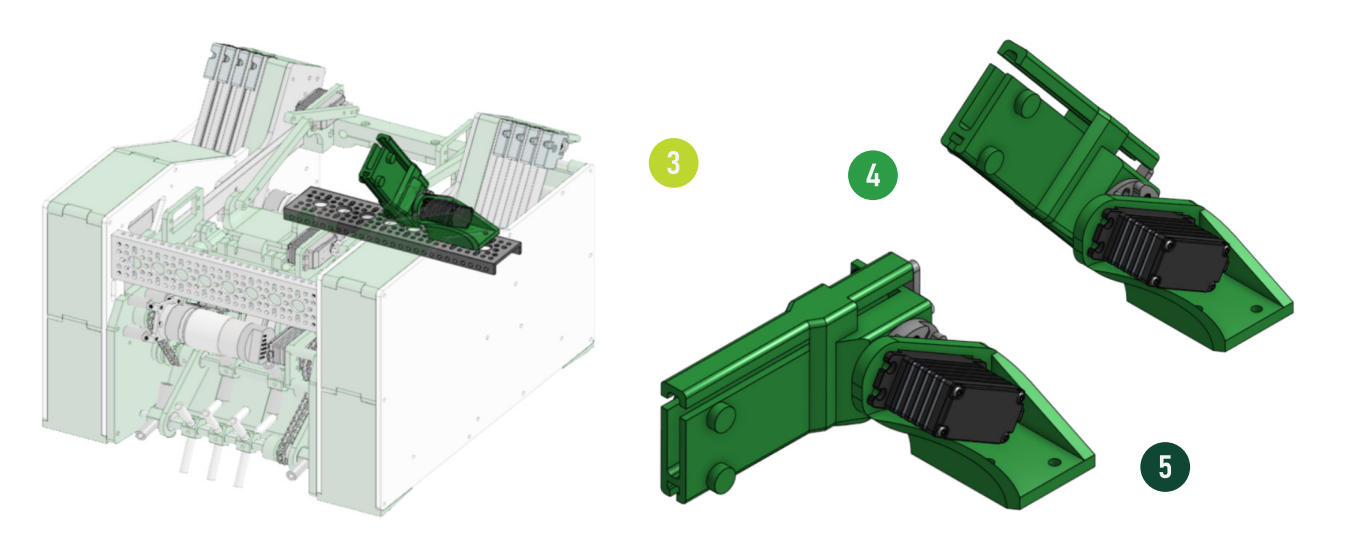
Avantaje:	Dezavantaje:
• O suprafață mare de colectare a pixelilor; • Colectare rapidă și eficientă; • Ansamblu modular, care poate fi înlocuit ca o singură bucată, fără sa fie dezasamblat.	• Pivotul nu se poate roti la 180 de grade, lucru care ne-ar permite să salvăm timp.

Lansator de dronă:

Din considerenta spațiului disponibil în noul robot, am fost nevoiti să **regândim mecanismul** de lansare, acest lucru ducând atât, la **schimbarea dronei, cât și a lansatorului**. Pentru a reduce dimensiunile am ales să ținem drona plată, urmând ca aceasta să își extindă aripile pe parcursul zborului.

Am micșorat lansatorul prin înlocuirea servoului standard de la lansare, cu unul micro. Pe lânga acesta, **unghiul de lansare este reglat** de un alt servo, astfel poate să fie pliat pentru a păstra o poziție compactă. Acest reglaj ne permite și ajustarea poziției de tragere pentru a regla punctul de aterizare.

Avantaje:	Dezavantaje:
• Ocupă mai puțin spațiu; • Unghiul de lansare este reglabil, utilizând un servo.	• Întrucât avionul este încărcat în lansator cu aripile pliate, nu putem garanta o traiectorie dreaptă.



Capturi din OnShape cu robotul v5 și mecanismele lui:
- intake-ul din vedere izometrică (1);
- intake-ul cu rola de jos ridicată din vedere izometrică (2);
- robotul cu lansatorul de dronă din vedere izometrică (3)
- lansatorul de dronă în poziție orizontală din vedere izometrică (4);
- lansatorul de dronă în poziție de lansare (sub unghi) din vedere izometrică (5).

Off Season:

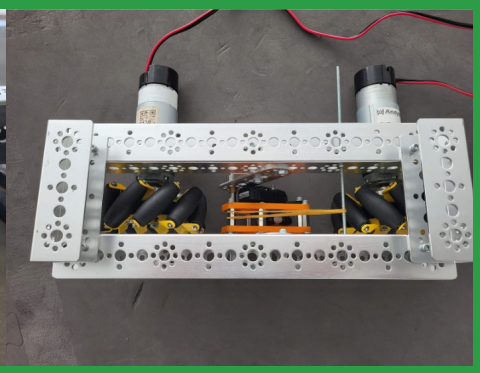
Odopod:

„Odopod-ul" este un **encoder** acționat de o roată omni ce se rotește datorită frecării cu suprafața terenului. Cu ajutorul acestora putem calcula constant poziția robotului în teren. Acest proiect a trebuit amânat, deoarece am avut nevoie de anumite piese (roți omni de dimensiuni reduse), pe care le-am putut găsi numai la producătorul din Australia, iar transportul a durat peste o lună.

v1

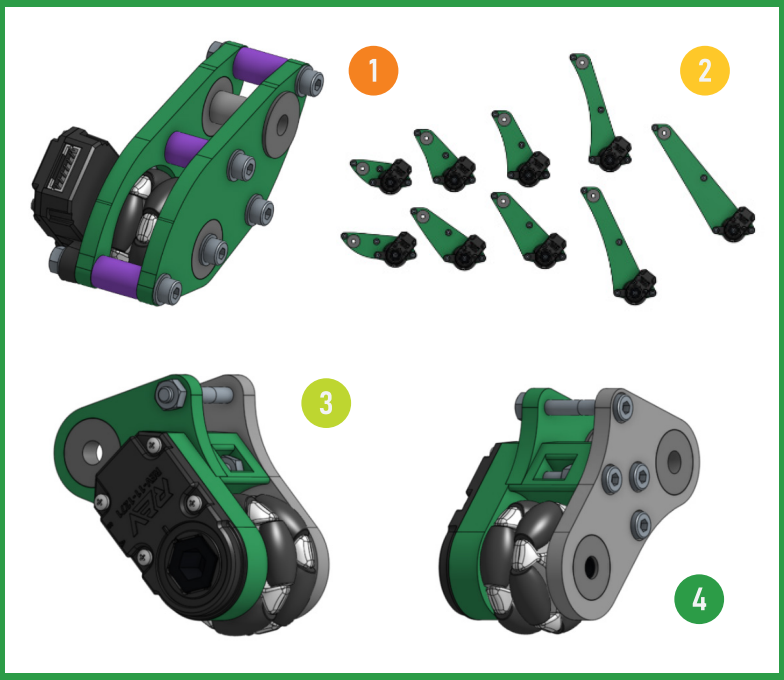
Pentru a capătă experiență cu ceea ce înseamnă odometria, am decis să folosim **OpenOdometry**, un proiect ***open source***, care ne-a permis să învățăm cum funcționează mecanismul în sine.





Avantaje:	Dezavantaje:
• Simplu și rapid de experimentat cu el.	• Dimensiuni fixe, ce constrâng opțiunile pentru montat; • Multe componente (se pot desface de la vibrații și să cauzeze erori).

v2 În urma necesității de a **varia forma** și dimensiunile odopod-ului, am creat propriul ansamblu care poate să fie generat după nevoi, prin modificarea a doar **doi parametri** (Height și Offest).



Avantaje:	Dezavantaje :
• Ușor de adaptat pentru a încăpea în șasiu.	• Multe componente (se pot desface de la vibrații).

Capturi din OnShape cu prototipurile de odopod-uri:
- odopod v2 (1);
- odopoduri în funcție de parametri (2);
- odopod v3 din vedere izometrică din două părți opuse (3), (4).

v2 Pe parcursul testelor am constatat că prezența unui număr mare de componente, în total **20** (9 șuruburi, 1 piuliță, 4 standoff-uri, 3 rulmenți, 2 plăci, 1 encoder) reduce **fiabilitatea** ansamblului. Păstrând doar componentele strict necesare și combinând carcasa encoder-ului cu una dintre plăci și cu *standoff*-urile, am redus numărul la **13** (4 șuruburi, 4 piulite, 3 rulmenți, 1 placă,1 encoder), ceea ce înseamnă o **optimizare cu 35%**.

Avantaje:	Dezavantaje:
• Fiabilitate ridicată datorită numărului redus de piese.	

Testele ne-au arătat că **tensionarea odometriei** prin intermediul elasticelor nu este o soluție optimă, deoarece proprietățile acestora variază mult în funcție de temperatură și își pierd caracterisicile elastice dacă sunt lăsate la soare ori întinse pentru o perioada îndelungată, pe când arcurile nu prezintă aceste dezavantaje.

	Avantaje:	Dezavantaje:
Arcuri:	• Constanta de elasticitate nu este influențată de factorii externi.	• Necesită căutarea unor arcuri cu caracteristicile dorite.
Elastice:	• Ușor de găsit în comerț.	• Pierderea proprietăților elastice; • Variația constantei elastice între două elastice din același lot; • Fluctuația constantei elastice cauzată de factori externi.

Șasiu:

v1 Pornind de la șasiul sezonului precedent, am realizat un șasiu de **300x300 mm**, cât mai versatil, care utilizează **odometria**, pregătindu-ne pentru începutul noului an competițional. Acesta este acționat de 4 motoare AndyMark Neverest 20 cu transmisie cu lanț (raport **24:23**), iar pentru odometrie am folosit **3 odopod-uri**: 2 paralele și 1 perpendicular.

Avantaje:	Dezavantaje:
• Mobilitate foarte bună datorată roților mecanum și a dimensiunilor mici; • Precizie mare mulțumită odometriei.	• Prezența frecărilor la mișcare cauzată de anumite toleranțe defectuoase.

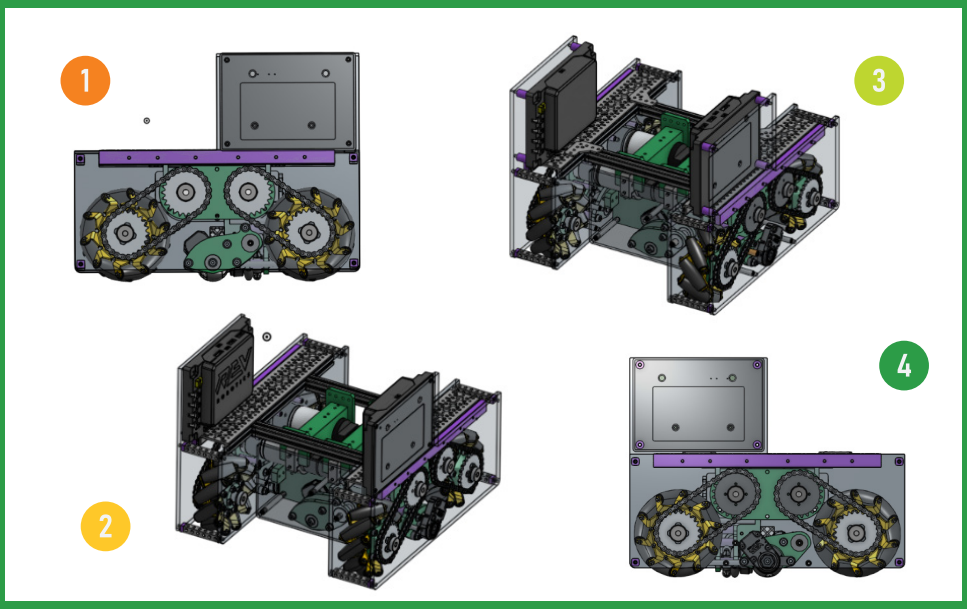
v2 La cea de-a doua versiune am schimbat raportul lanțului în 1:1 și am modificat toleranțele acestuia.

Avantaje:

- Mobilitate foarte bună;
- Precizie ridicată.

Capturi din OnShape cu versiunile de șasiu:

- șasiu v1 din vedere laterală (1);
- șasiu v1 din vedere izometrică (2);
- șasiu v2 din vedere laterală (3);
- șasiu v2 din vedere izometrică (4).



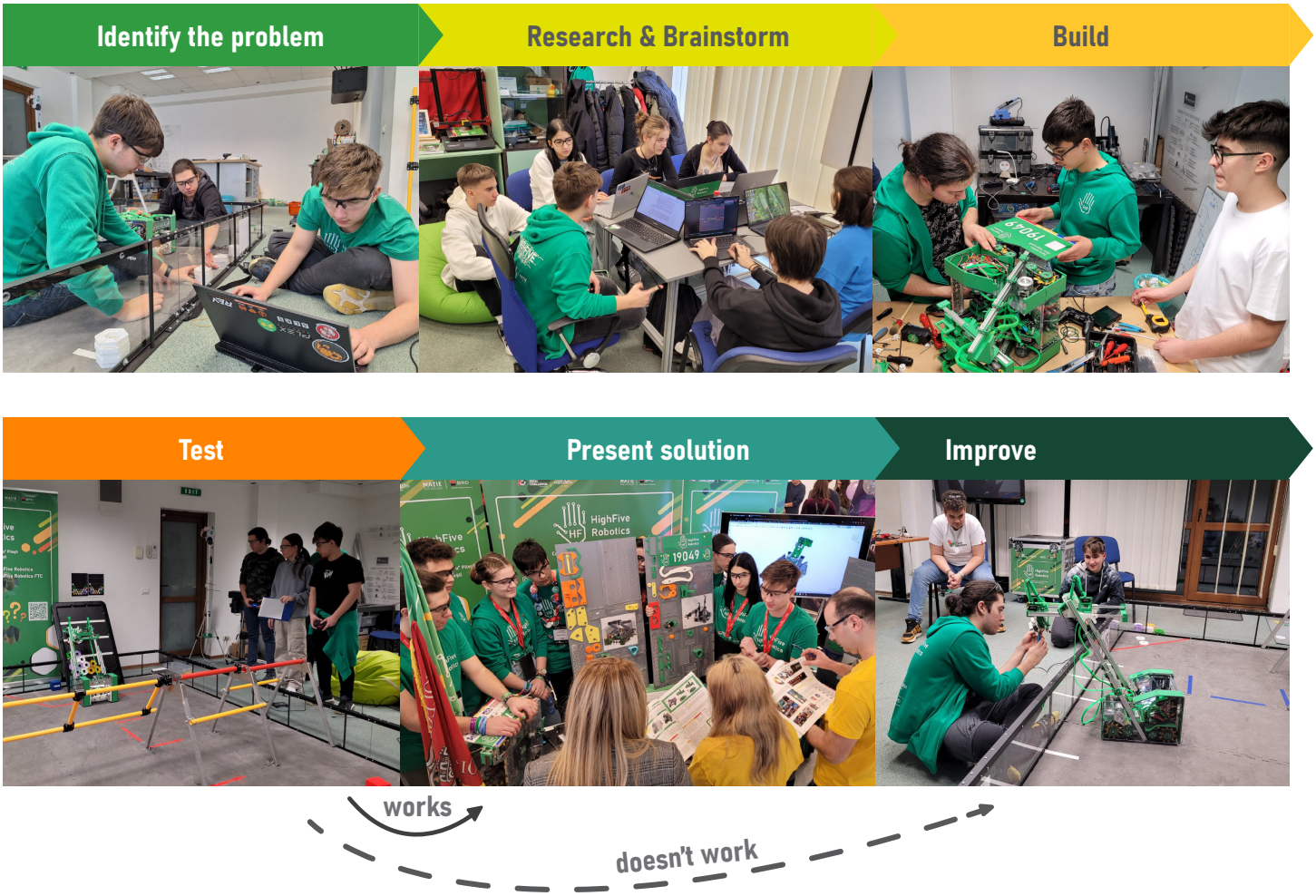
EVOLUTIE ROBOT'

Septembrie 2023 - Martie 2024

Engineering Design Process:

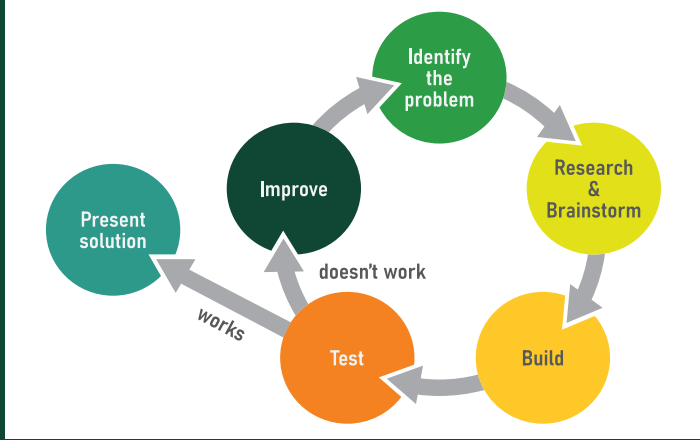
Acest proces reprezintă un **concept** prin intermediul căruia ne putem **organiza** mult mai ușor, formându-ne un algoritm, o serie precisă de pași ce trebuie parcurși după crearea fiecărui nou prototip. Primul pas este **identificarea problemei.** O considerăm esențială, deoarece ne ajută să identificăm atât posibile impedimente pe care ne-am dori să le evităm, cât și probleme actuale ce trebuie corectate.

După ce am stabilit ce dorim să realizăm, începe etapa de ***brainstorming*** pentru a putea găsi cât mai multe soluții inovative. De asemenea, **ne informăm** privind îmbunatățirile posibile cu scopul de a descoperi variate idei. **Construim, testăm**. Apoi repetăm procesul, parcurgând toți pașii până când suntem mulțumiți de soluția la care am ajuns.



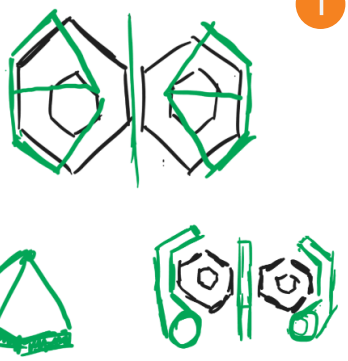
Totodată, ne-am împărțit întreaga activitate pe **subechipe**, astfel eficientizând dezvoltarea robotului prin sesiuni de lucru în paralel, întrucât fiecare grup să aibă un *task* diferit, ce poate fi realizat independent. Am descoperit că această structură se mulează cel mai bine pe modul nostru de organizare, dovadă fiind cele **5 versiuni de robot** prin care am trecut de la începutul lunii septembrie și până acum.

Acest mecanism ne-a ajutat să **iterăm** cât mai mult, să facem **transferul de cunoștințe** și responsabilitate, colaborând în proiecte variate.



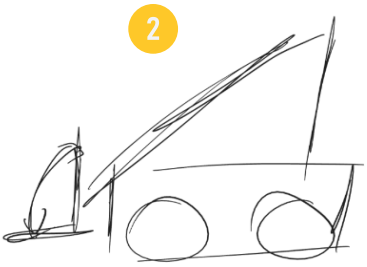
Robot v1- KickAthon:

1



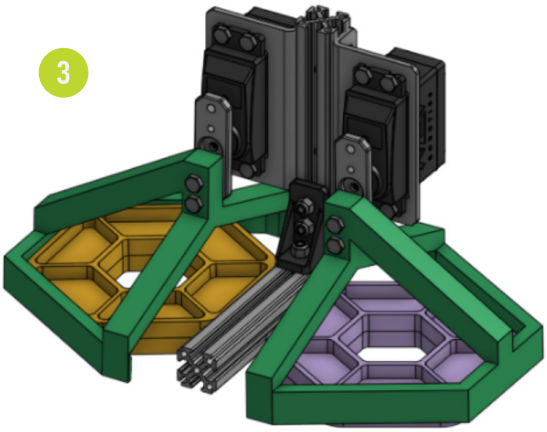
Schițele inițiale cu gheara dublă

2

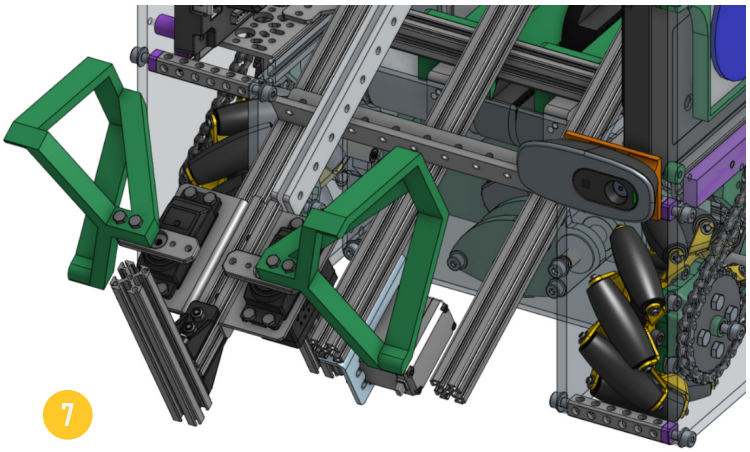


Schițele inițiale cu robotul

3

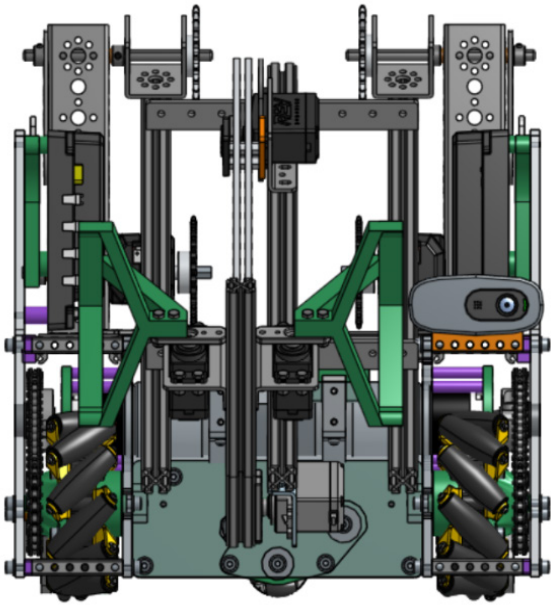


Captură din OnShape cu gheara
- vedere izometrică -



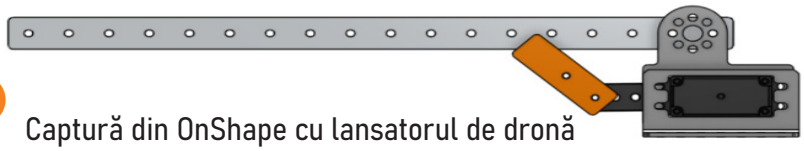
7

Detaliu din OnShape cu sistemul de colectare
- vedere izometrică -



Captură din OnShape cu robotul
- vedere din față -

8

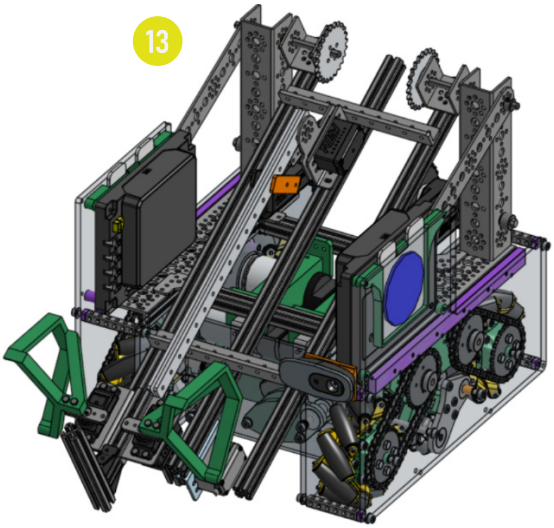


11

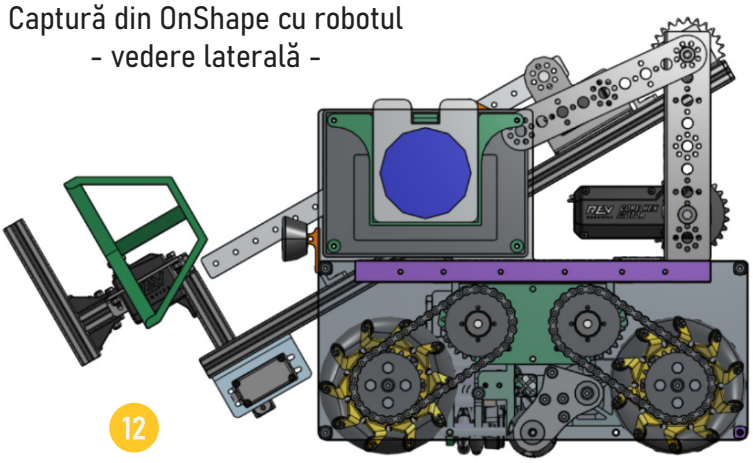
Captură din OnShape cu lansatorul de dronă
- vedere laterală -

Captură din OnShape cu robotul
- vedere izometrică -

13

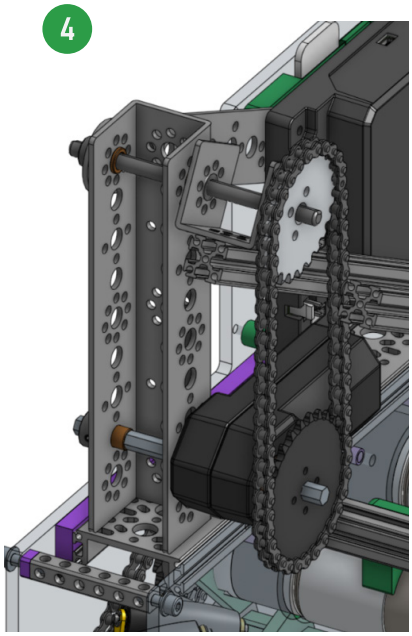


Captură din OnShape cu robotul
- vedere laterală -



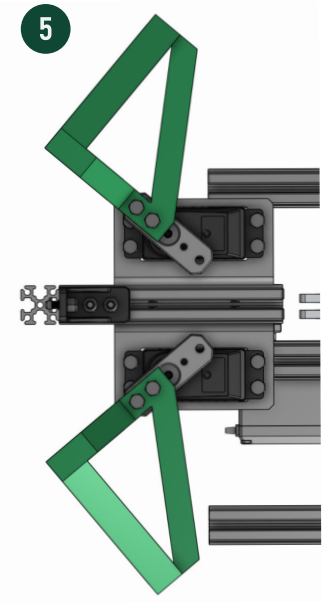
12

4



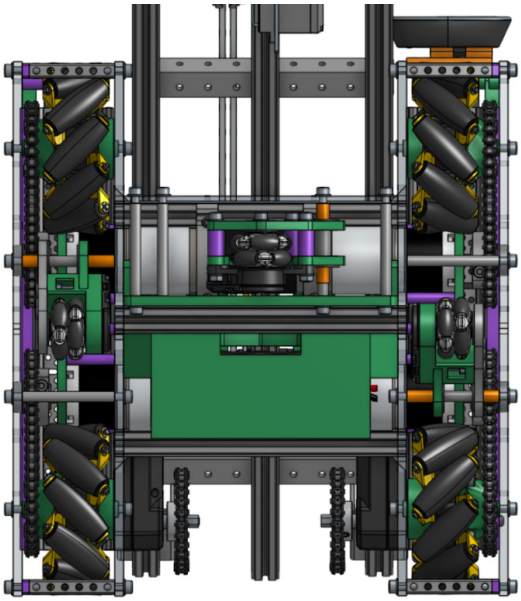
Detaliu din OnShape cu brațul
- vedere izometrică -

5



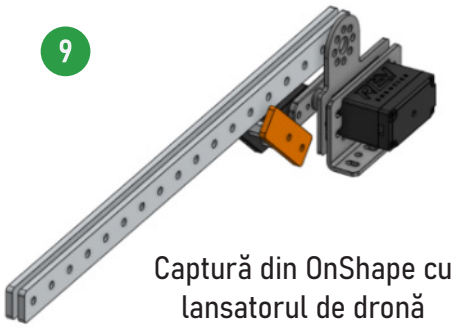
Detaliu din OnShape cu gheara
- vedere din față (rotit la 90°) -

6



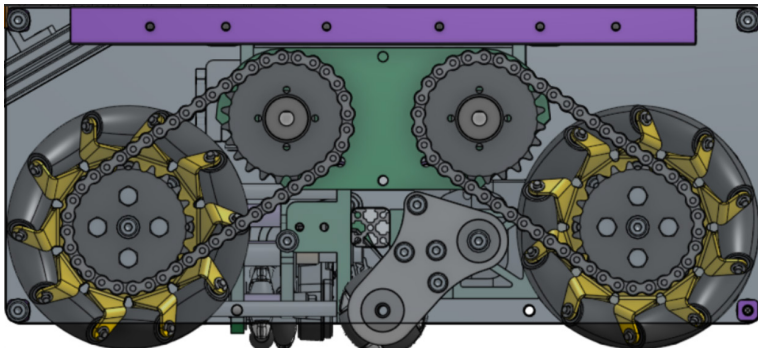
Captură din OnShape cu robotul
- vedere de dedesubt -

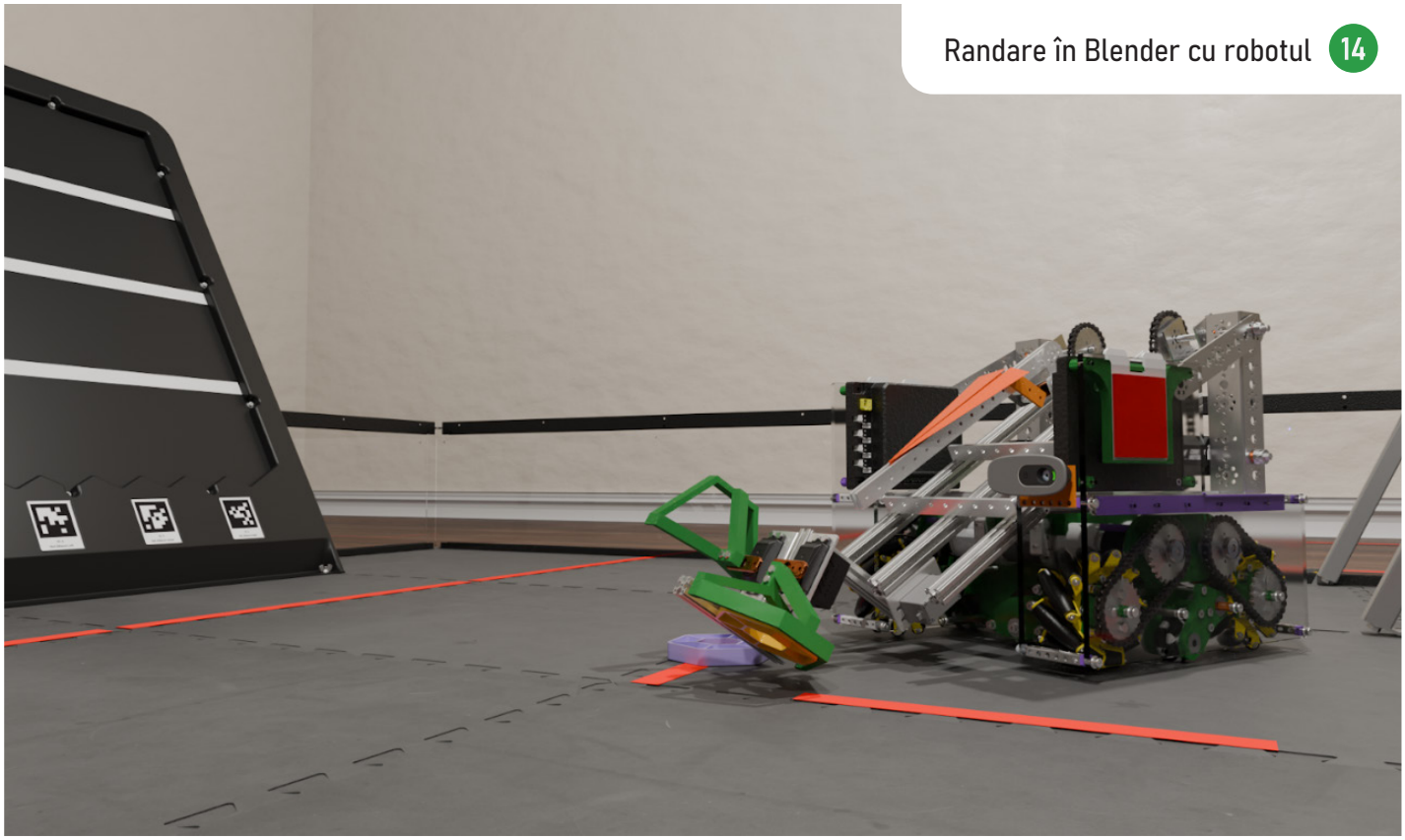
9



Captură din OnShape cu
lansatorul de dronă
- vedere izometrică -

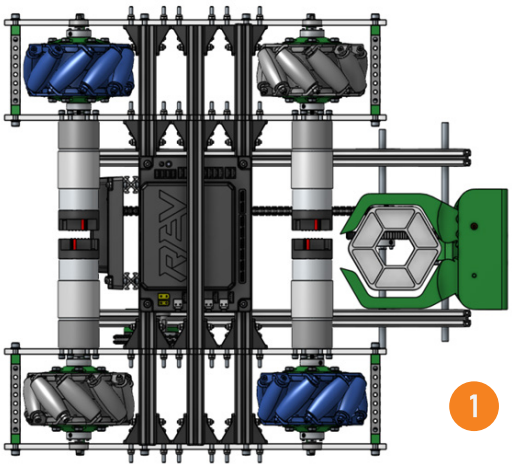
10





Randare în Blender cu robotul 14

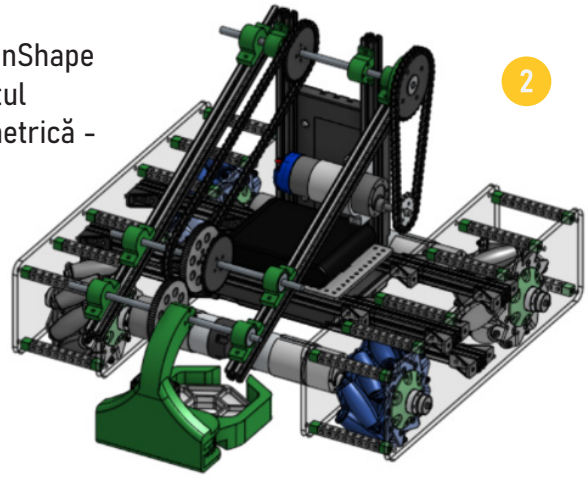
Robot v2 (prototipări) & Robot Prezentări:



1

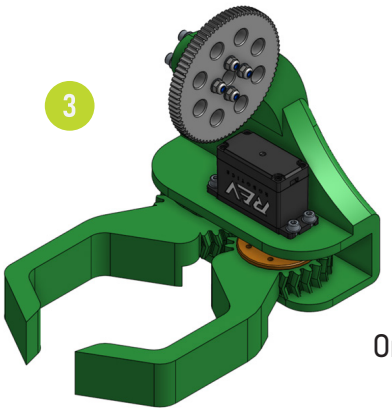
Captură din OnShape
cu robotul
- vedere izometrică -

2



Captură din OnShape cu robotul
- vedere de jos -

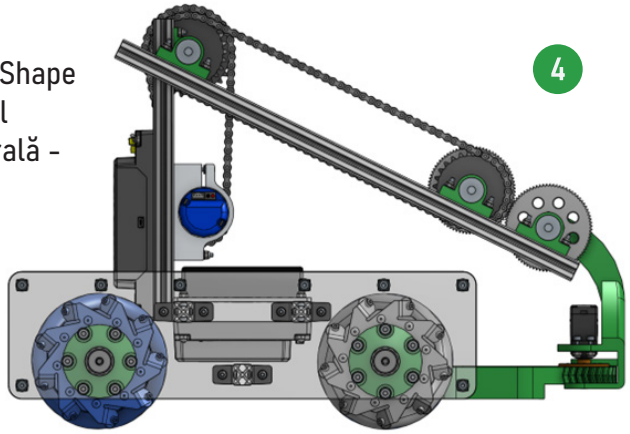
3



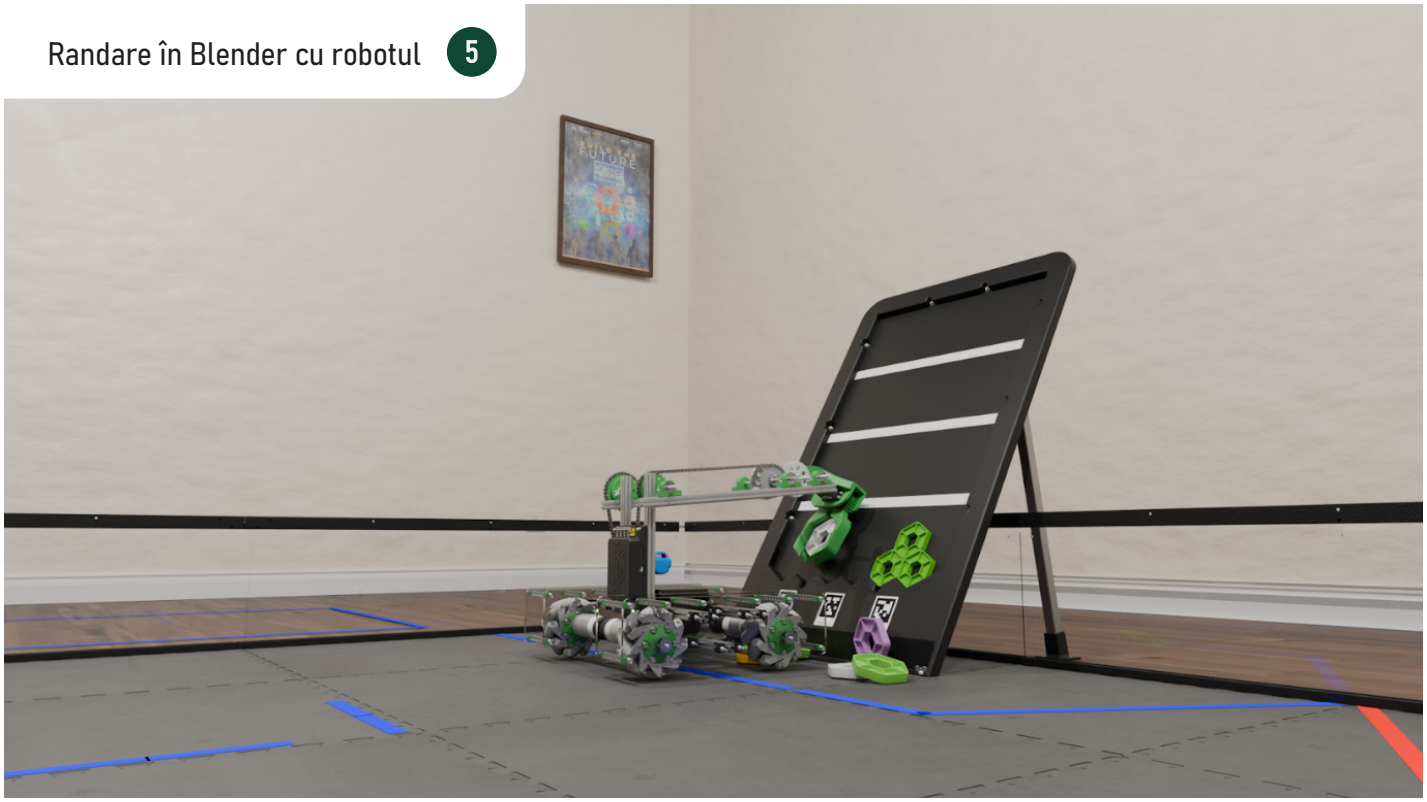
Detaliu din
OnShape cu gheara

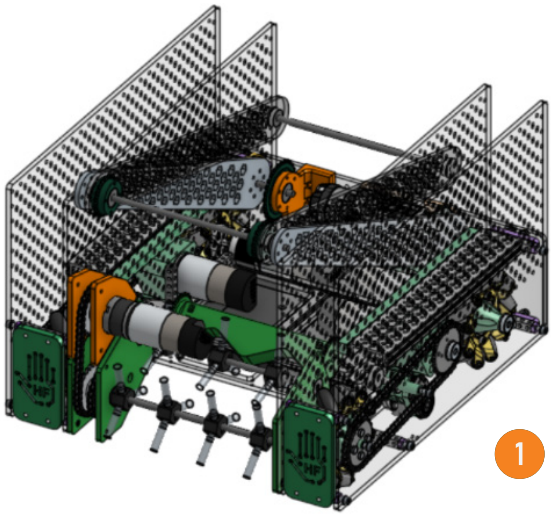
Captură din OnShape
cu robotul
- vedere laterală -

4



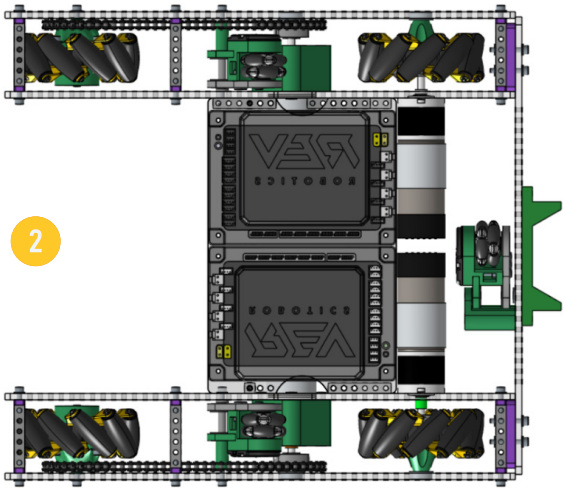
Randare în Blender cu robotul 5





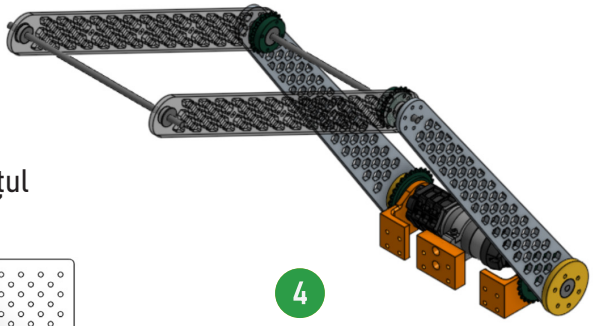
1

Captură din OnShape cu robotul
- vedere izometrică -



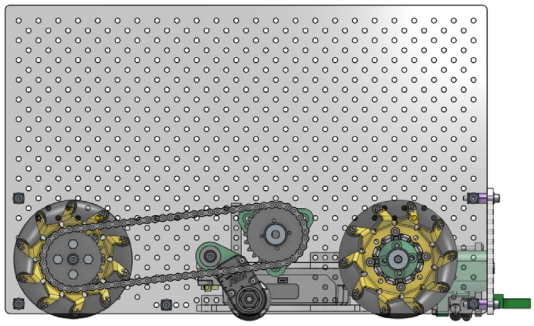
2

Captură din OnShape cu robotul
- vedere de jos -



3

Detaliu din OnShape cu brațul
- vedere izometrică -

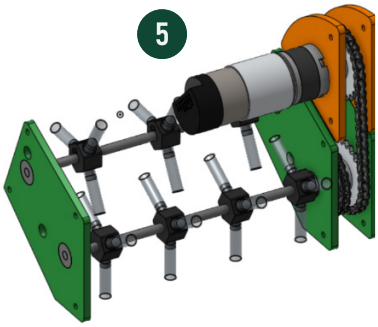


4

Detaliu din OnShape cu brațul
- vedere izometrică -

Detaliu din OnShape cu intake
- vedere izometrică -

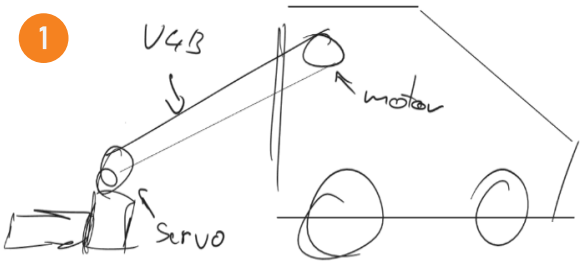
5



Randare în Blender cu robotul 6

Robot v3 - League Meet #1:

1

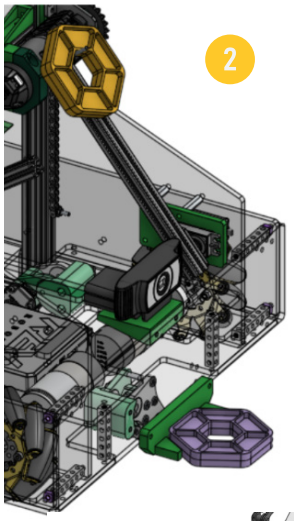


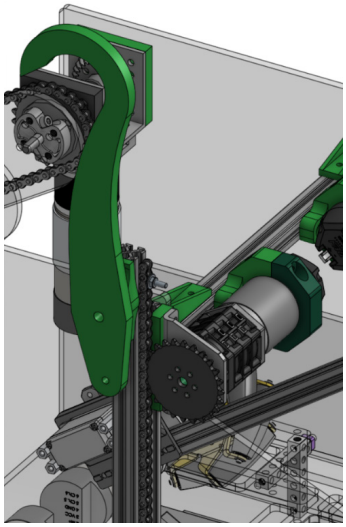
Schițele inițiale cu robotul

3

Detaliu din OnShape cu sistemele din autonomie
- vedere izometrică -

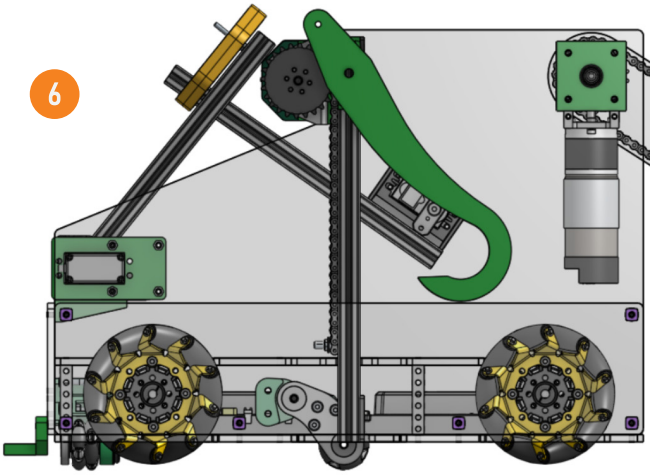
2





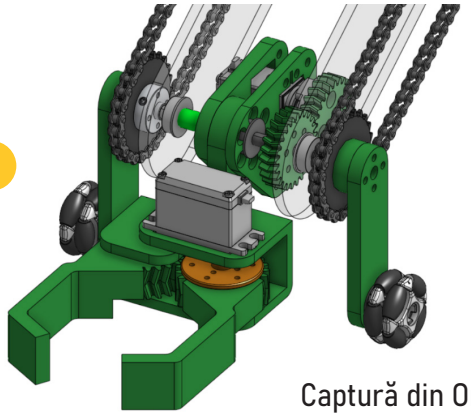
Detaliu din OnShape cu brațul
- vedere izometrică -

6



Detaliu din OnShape cu robotul
- vedere laterală -

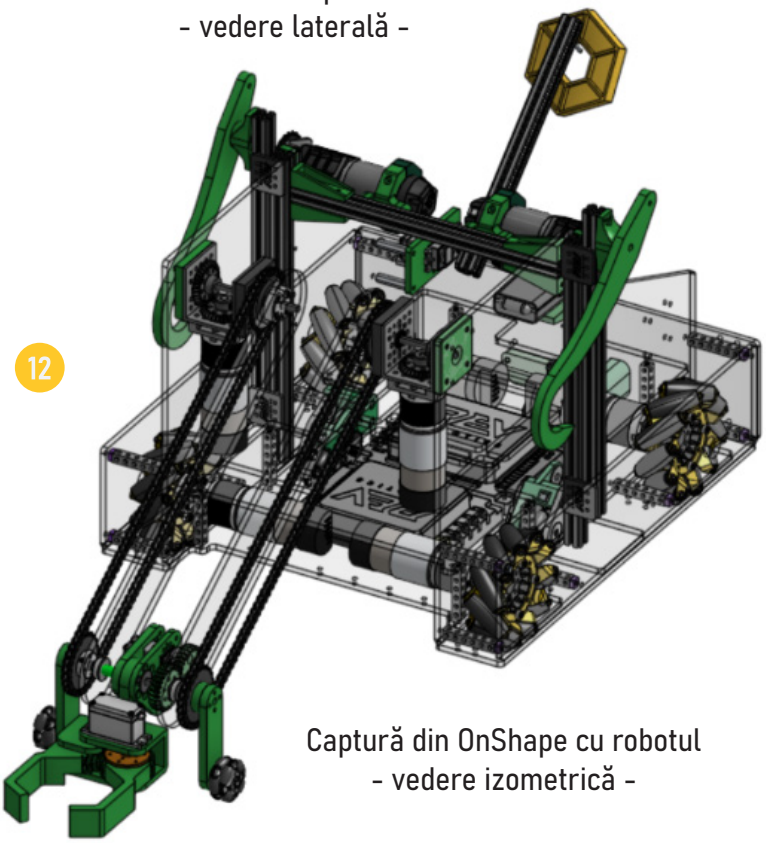
7



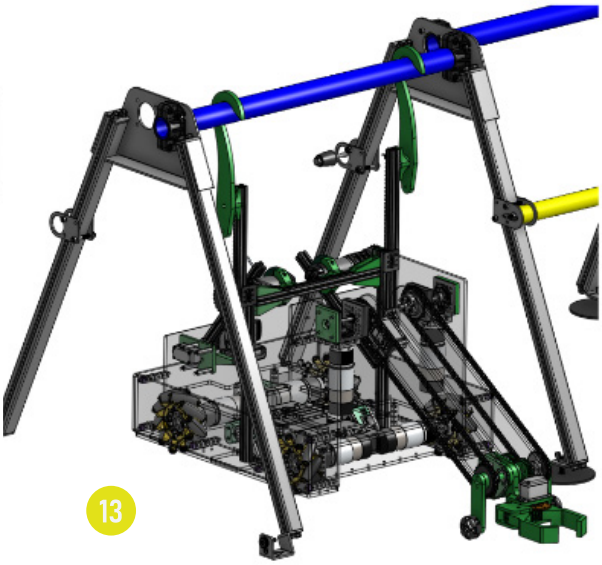
Captură din OnShape
cu gheara
- vedere izometrică -

Captură din OnShape cu robotul cățărat
- vedere izometrică -

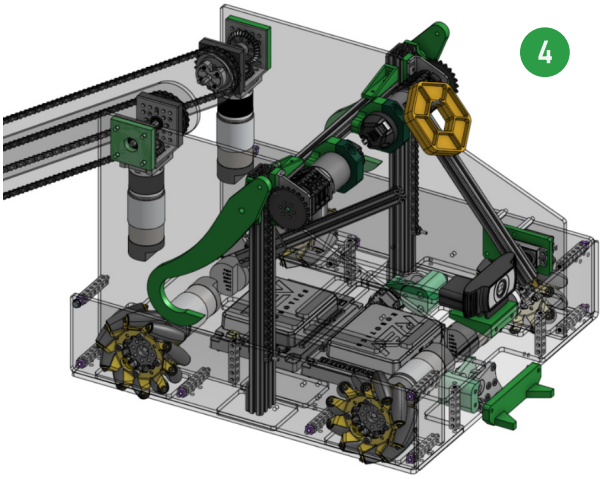
12



Captură din OnShape cu robotul
- vedere izometrică -

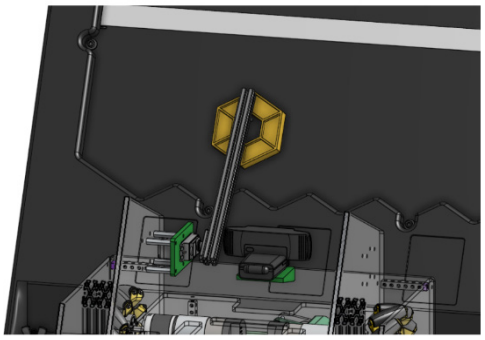


13



4

Detaliu din OnShape cu robotul
- vedere izometrică -



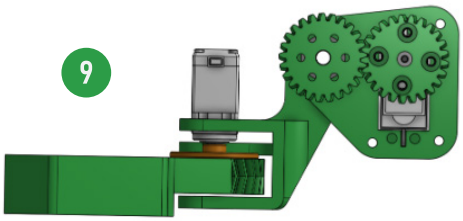
Detaliu din OnShape cu sistemele din autonomie
- vedere izometrică -

5

10

Captură din OnShape cu lansatorul de dronă
- vedere izometrică -

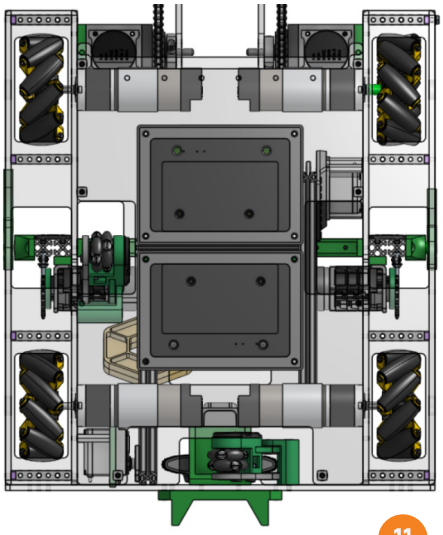
Captură din OnShape cu gheara
- vedere laterală -



9

8

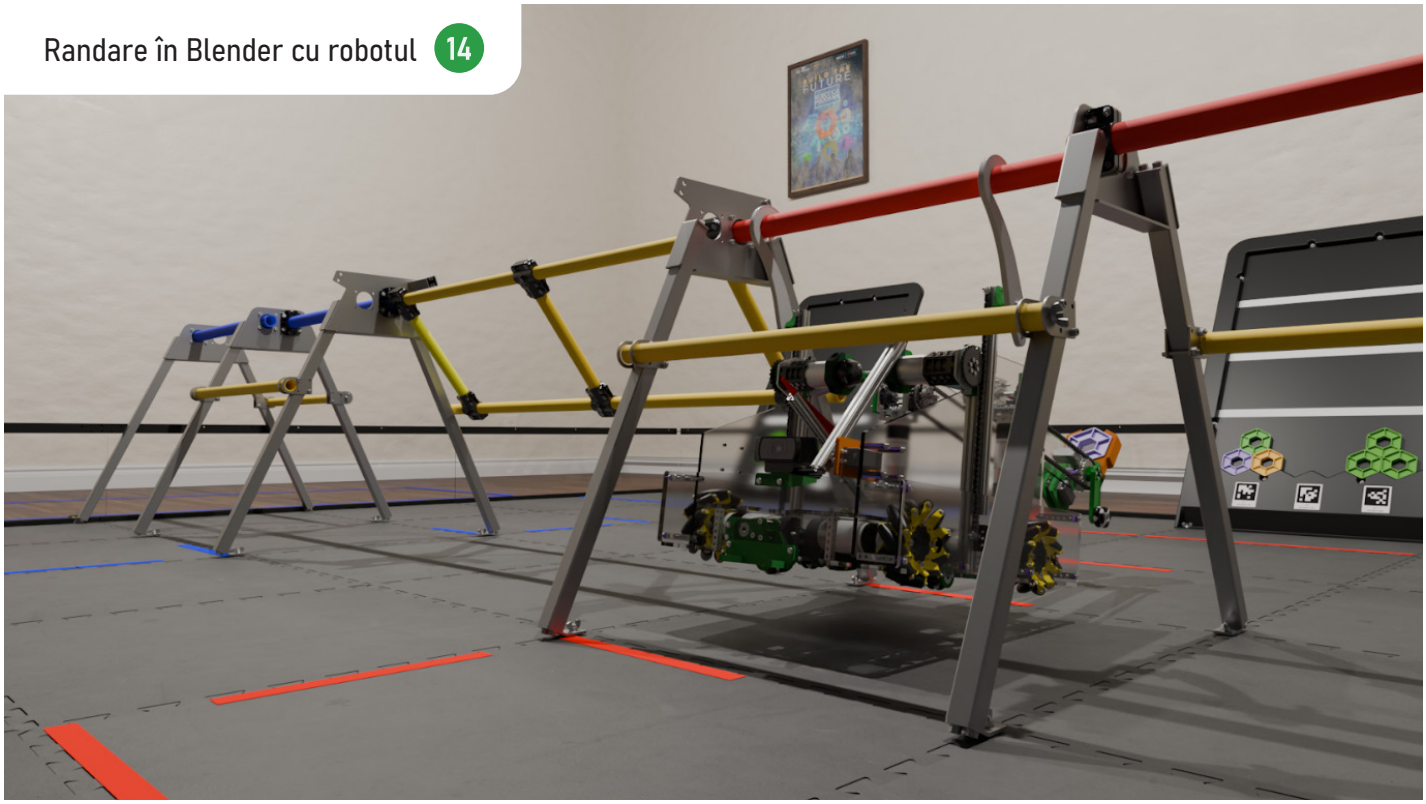
Captură din OnShape cu robotul cu brațul ridicat
- vedere de izometrică -



Captură din OnShape cu robotul
- vedere de dedesubt -

11

Randare în Blender cu robotul 14

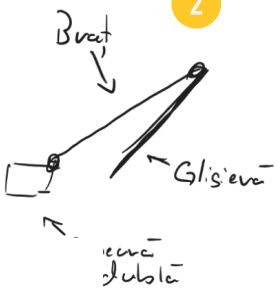


Robot v4 - League Meet #2/#3 & Regionala #1 și Națională:

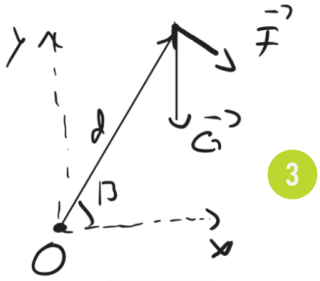
1

Schițele inițiale cu gheara dublă

2

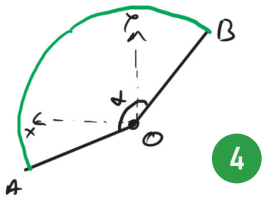


Schițele inițiale cu brațul și glisierele



3

Grafice pentru formule și calcule

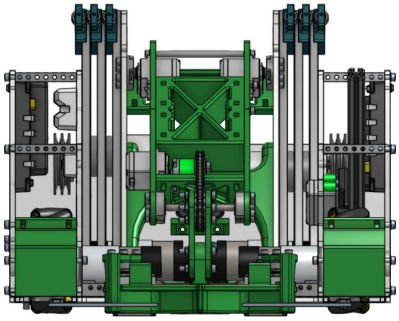


4

Grafice pentru formule și calcule

Captură din OnShape cu extinderea maximă a brațului

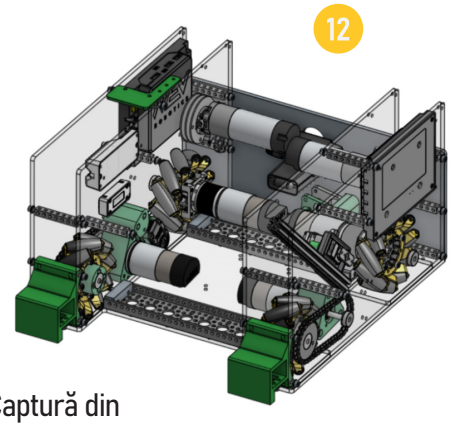
10



Captură din OnShape cu robotul - vedere frontală -

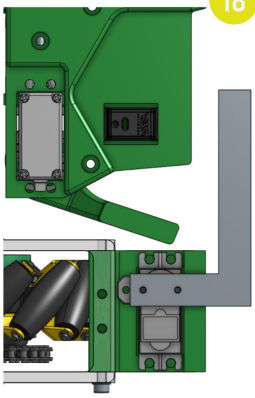
11

12

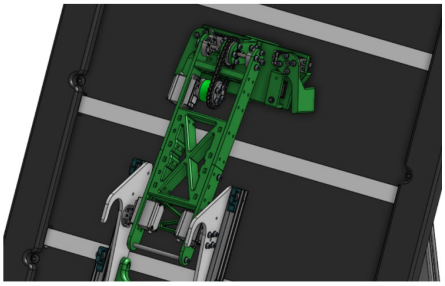


Captură din OnShape cu șasiul - vedere izometrică -

18



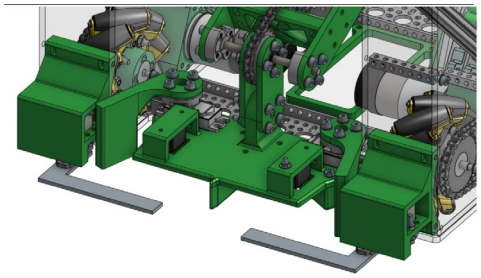
Detaliu din OnShape cu mecanismul de colectare din stack - vedere de sus - - vedere izometrică -

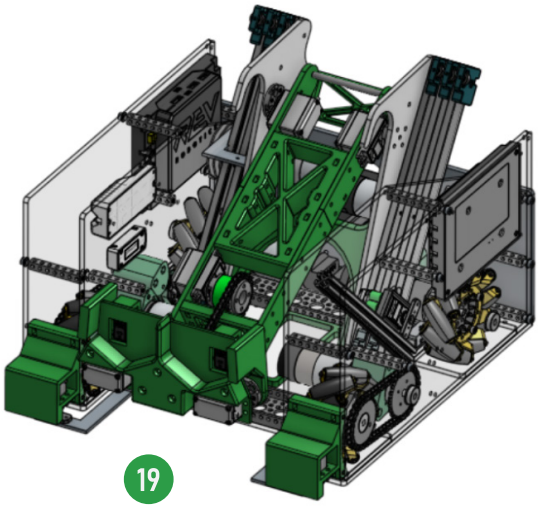


Detaliu din OnShape cu brațul

16

17

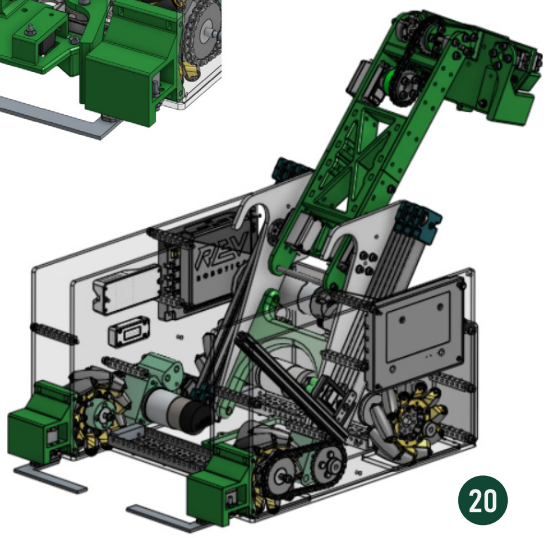




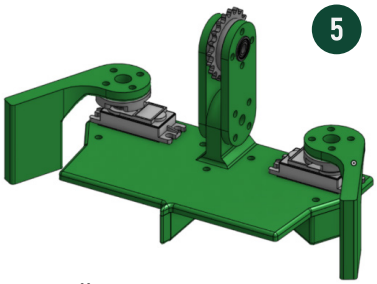
19

Captură din OnShape cu robot - vedere izometrică -

Captură din OnShape cu robotul cu brațul ridicat - vedere sub un unghi -

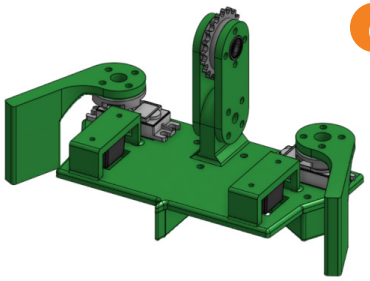


20



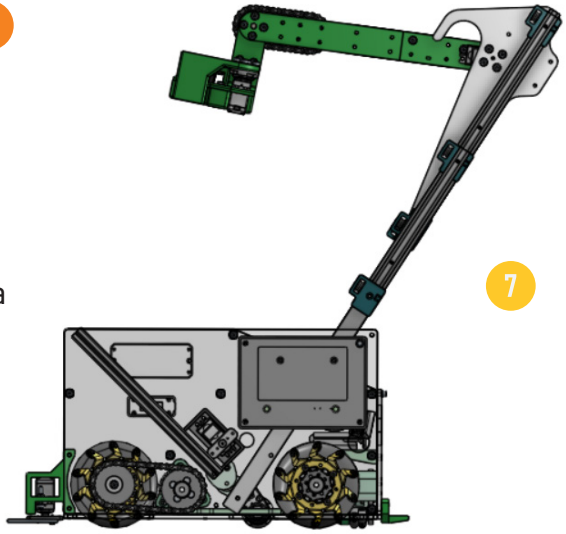
5

Captură din OnShape cu gheara
- vedere izometrică v0 -



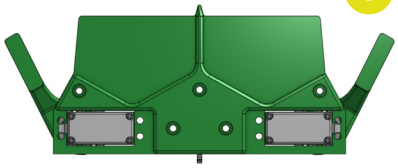
6

Captură din OnShape cu gheara
- vedere izometrică v1 -



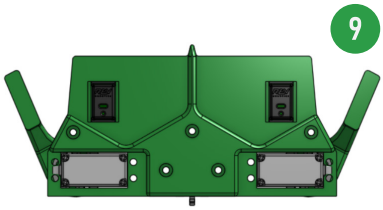
7

Captură din OnShape cu robotul
- vedere de din lateral -



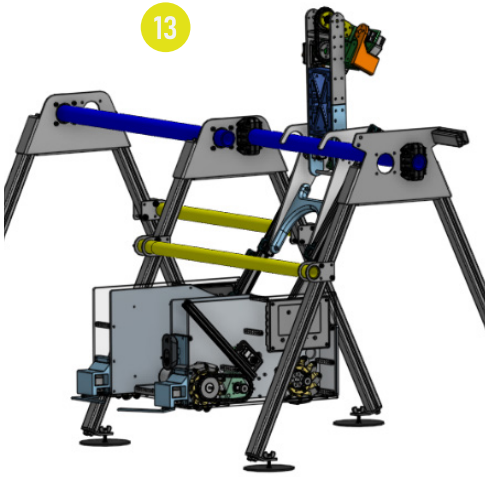
8

Captură din OnShape cu gheara
- vedere de jos v0 -

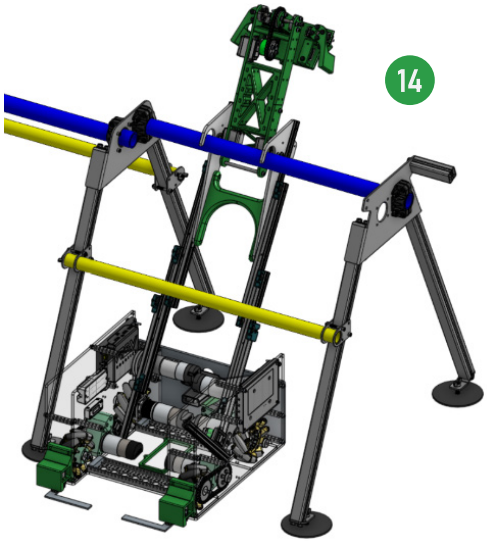


9

Captură din OnShape cu gheara
- vedere de jos v1 -



13

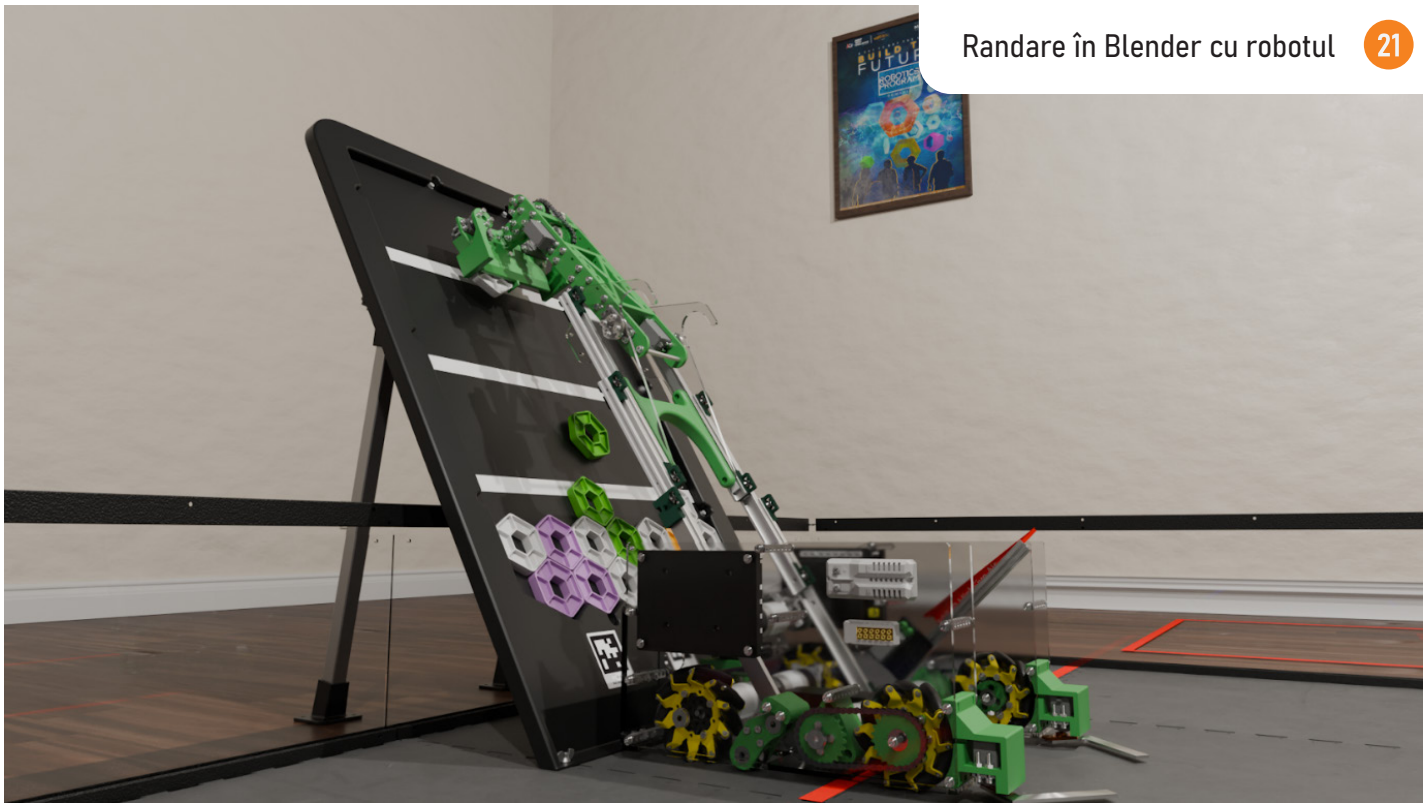


14

Capturi din OnShape cu
robotul cățărat

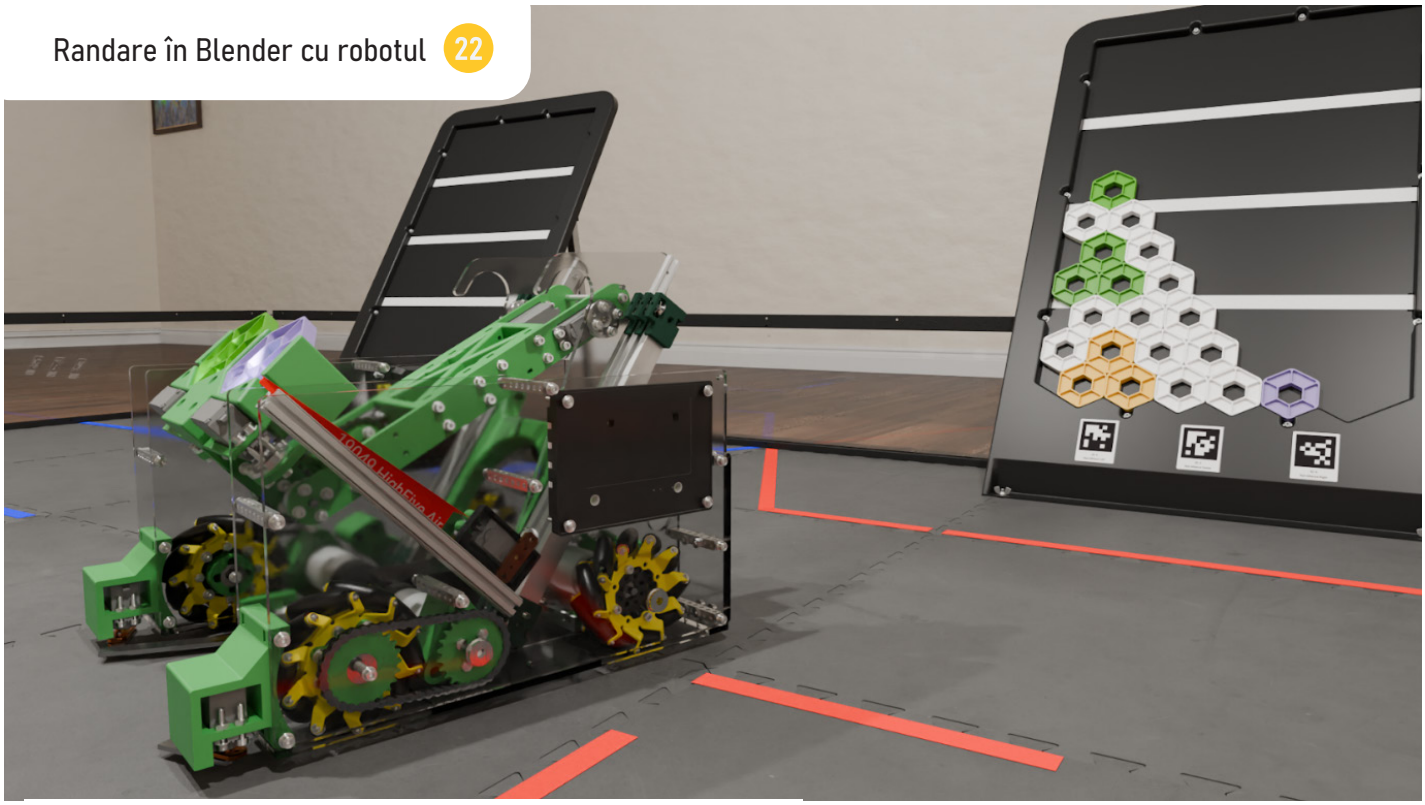
15

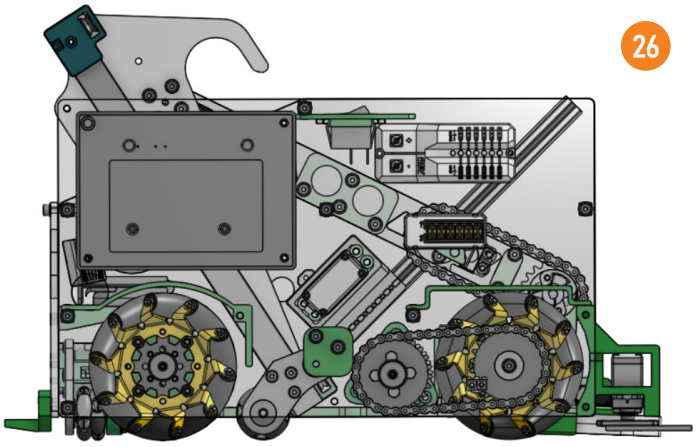
Captură din OnShape cu
lansatorul de dronă
- vedere laterală -



Randare în Blender cu robotul 21

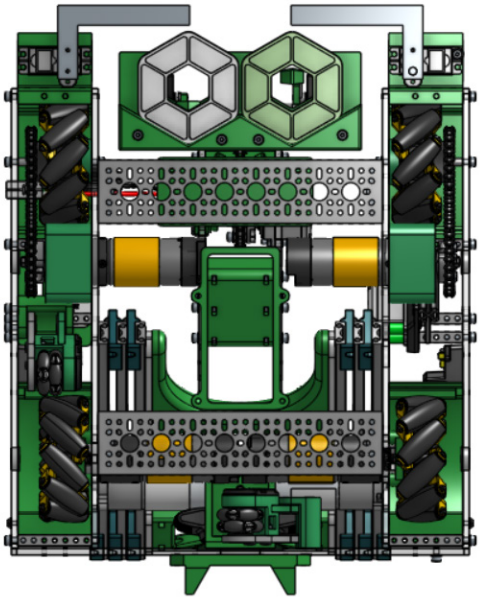
Randare în Blender cu robotul 22





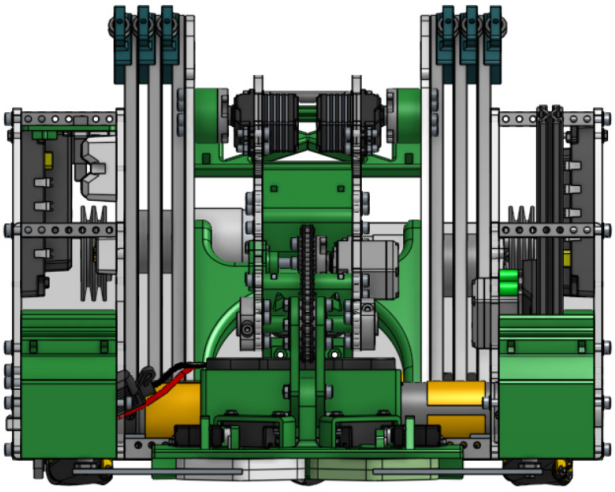
26

Captură din OnShape cu robotul
- vedere din lateral -



27

Captură din OnShape cu robotul
- vedere de jos -

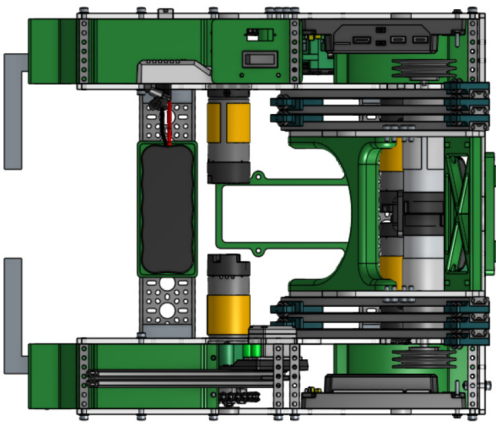


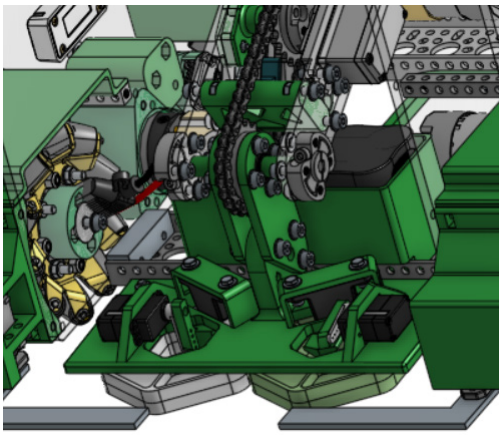
31

Captură din OnShape cu robotul
- vedere din față -

32

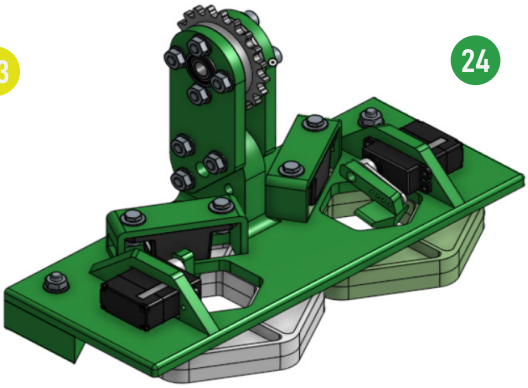
Captură din OnShape
cu șasiul
- vedere de sus -



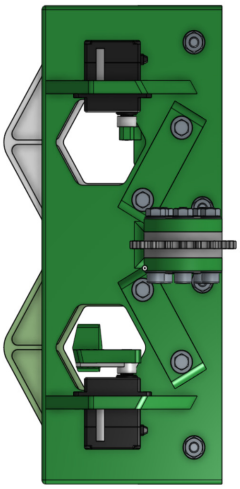


23

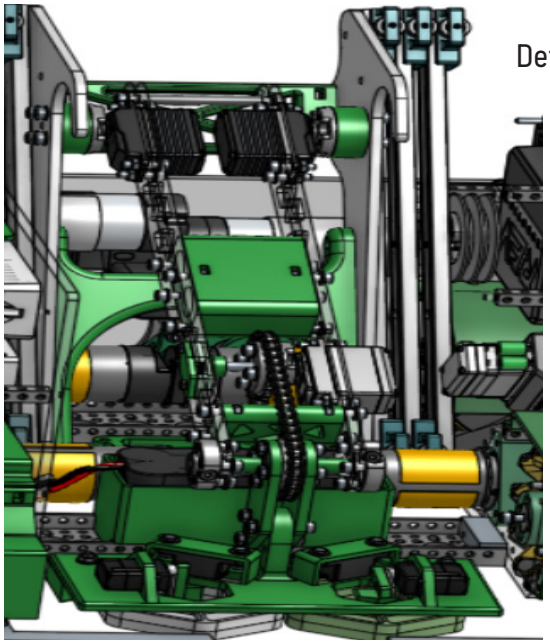
Detaliu din OnShape cu gheara



24



25

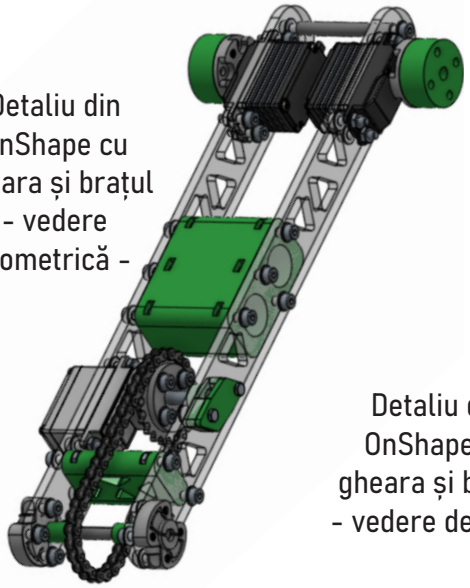


Detaliu din OnShape cu brațul

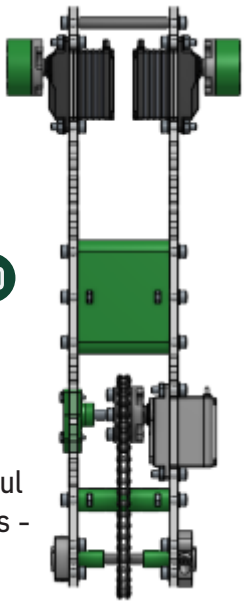
29

28

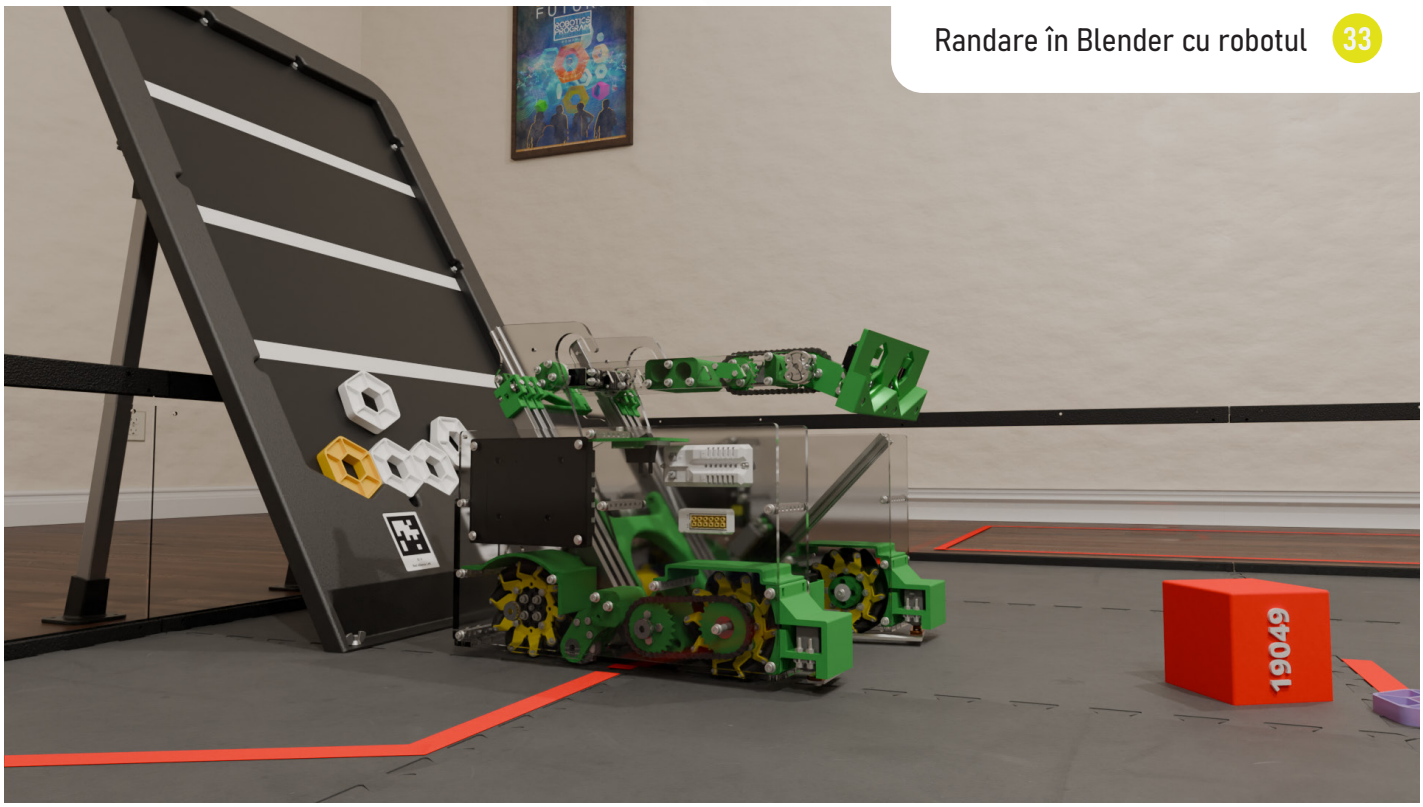
Detaliu din OnShape cu gheara și brațul - vedere izometrică -



30



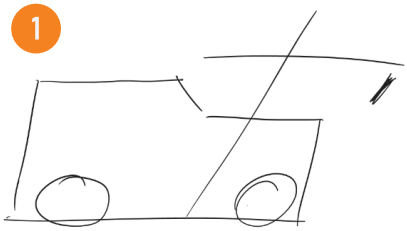
Detaliu din OnShape cu gheara și brațul - vedere de sus -



Randare în Blender cu robotul 33

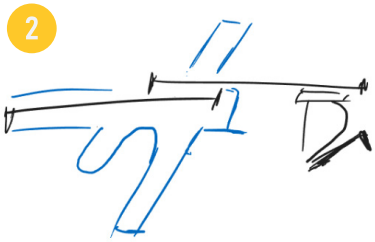
Robot v5 - Națională:

1



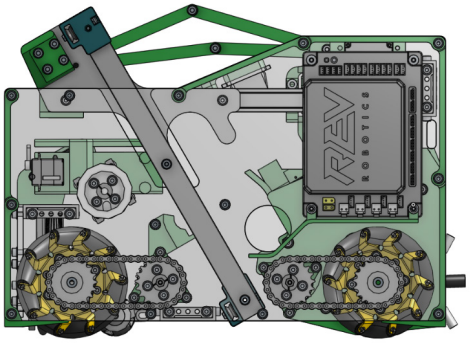
Schițele inițiale cu robotul

2



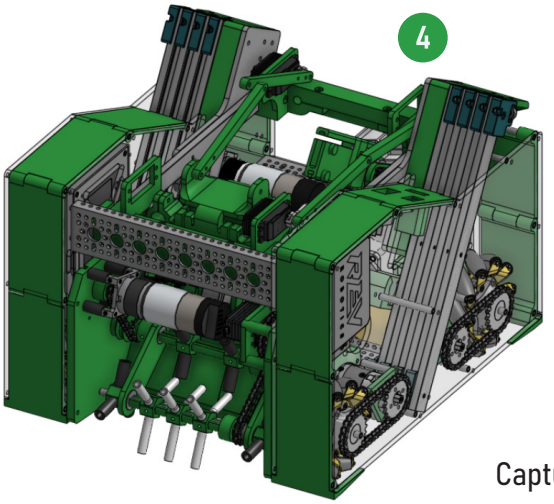
Schițele inițiale pentru outtake și cățărat

3



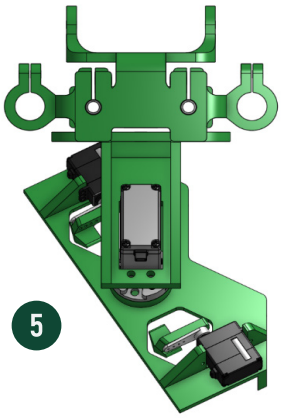
Captură din OnShape cu robotul - vedere laterală -

4



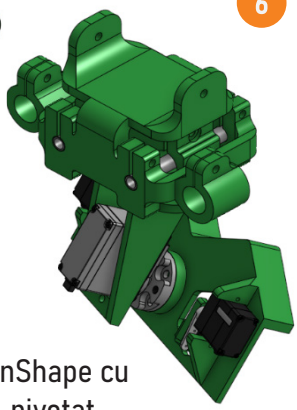
Captură din OnShape cu robotul - vedere frontală -

Captură din OnShape cu robotul - vedere izometrică -



5

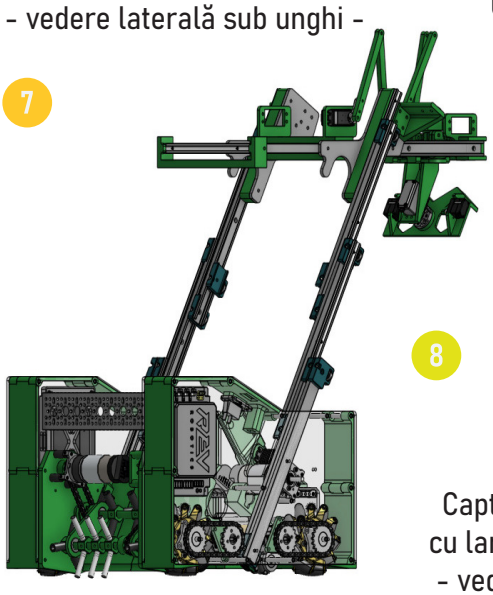
6



Capturi din OnShape cu outtake-ul pivotat - vedere din spate - - vedere izometrică -

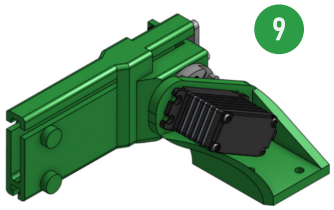
Captură din OnShape cu robotul semiextins - vedere laterală sub unghi -

7



8

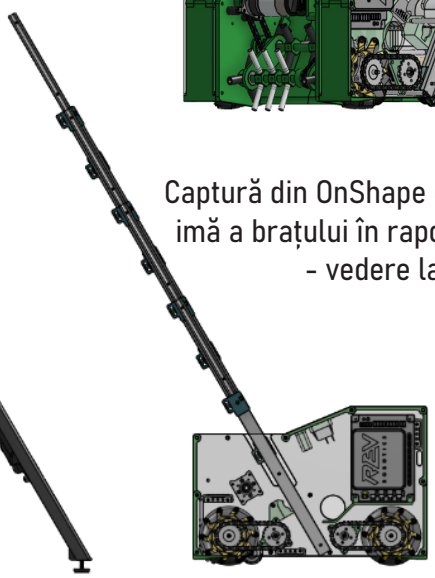
9



10

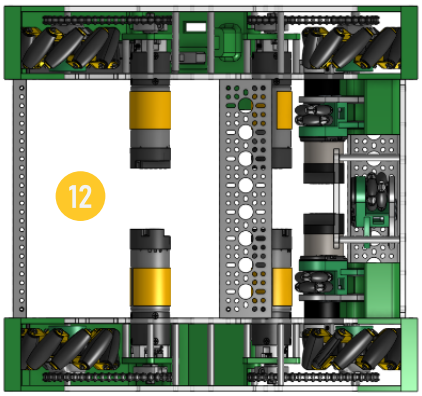
Capturi din OnShape cu lansatorul de dronă - vedere izometrică -

11

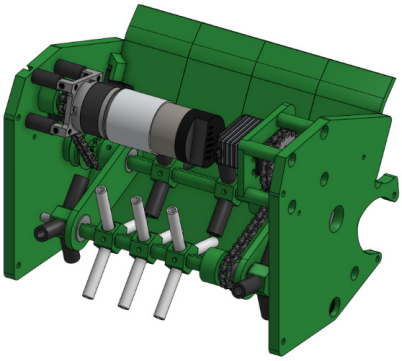


Captură din OnShape cu extinderea maximă a brațului în raport cu BackDrop-ul - vedere laterală -

12



Captură din OnShape cu șasiul - vedere de dedesubt -

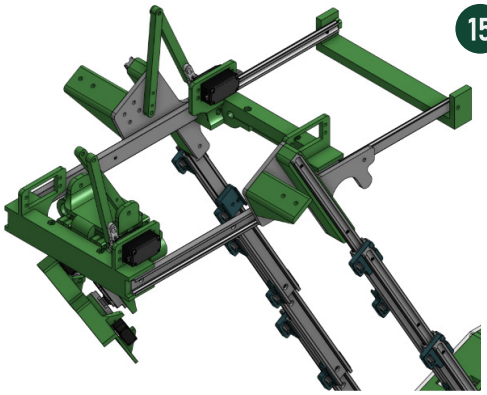


13

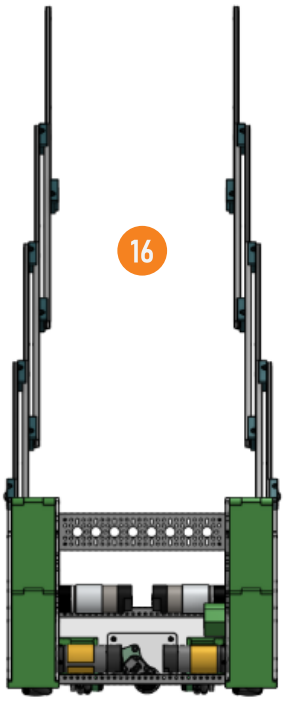
Captură din OnShape cu intake-ul - vedere izometrică -

Captură din OnShape cu outtake-ul - vedere izometrică -

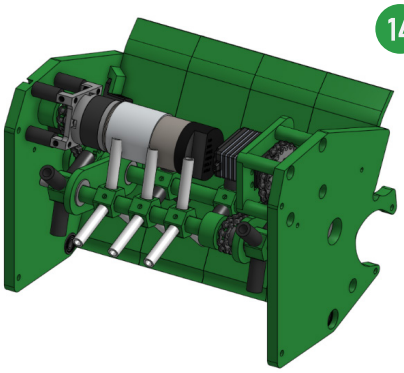
15



16



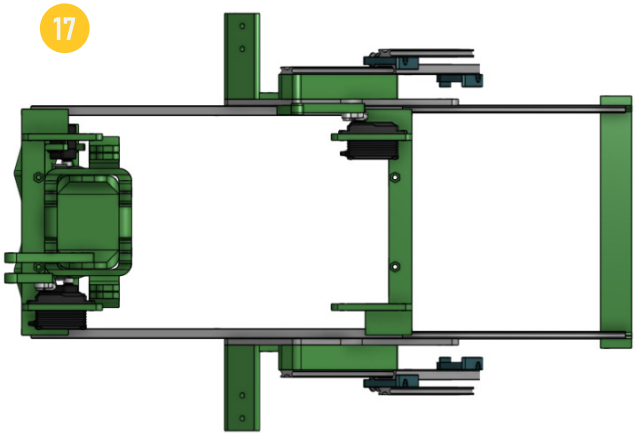
14



Captură din OnShape cu intake-ul cu rola de jos ridicată - vedere izometrică -

Captură din OnShape cu extinderea maximă a brațului

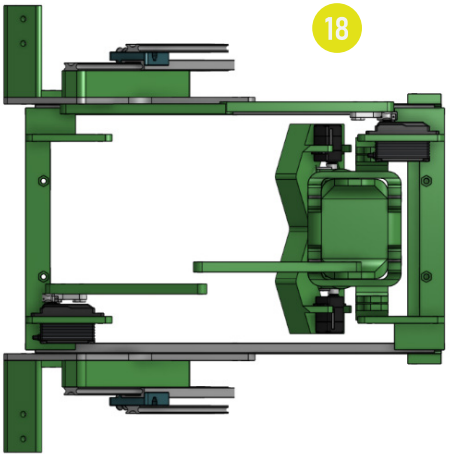
17



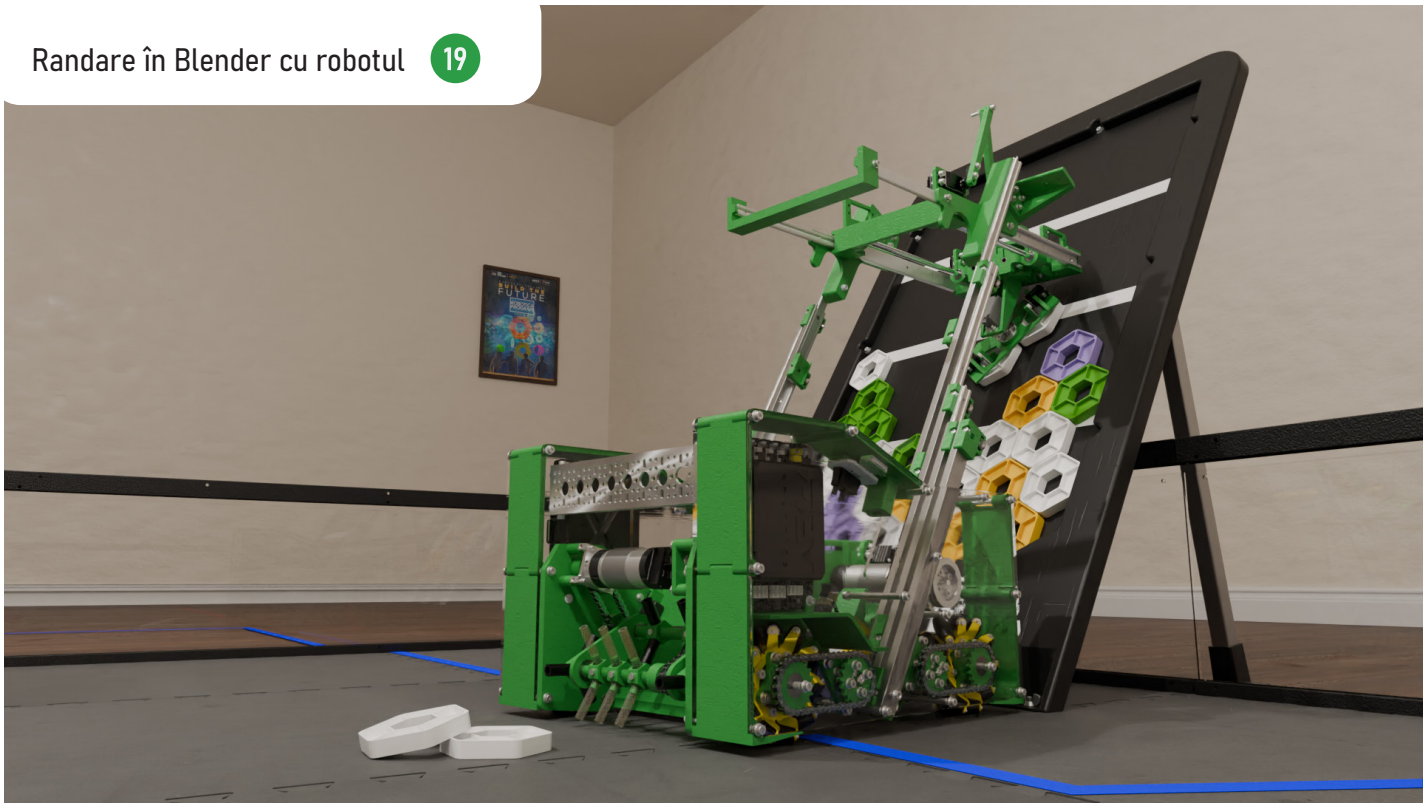
Detaliu din OnShape cu outtake-ul extins - vedere de sus -

Detaliu din OnShape cu outtake-ul restrâns - vedere de sus -

18



Randare în Blender cu robotul 19



SOFTWARE

Aprilie 2023 - Ianuarie 2024

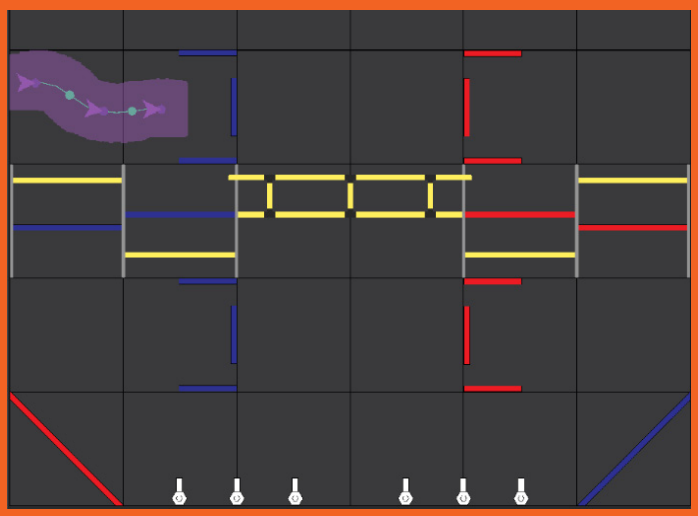
Robot v1- KickAthon:

Autonomie:

Considerând **Autonomia o parte esențială** în câștigarea meciului am realizat un program care citește cazul randomizat și, în funcție de aceasta, plasează pixelul mov pe *Spike*-ul corespunzător. Am ales o misiune valoroasă și, totodată realizbilă în cele **16 ore** puse la dispoziție pentru KickAthon, organizat de echipa **Quantum Robotics #14270**.

Implementări - Pixelul mov:

Pentru a plasa pixelul mov am folosit **o singură traiectorie** pentru a merge în față, la finalul acesteia robotul rotindu-se în funcție de cazul citit și deschizând gheara pentru a lăsa pixelul.



Schiță cu terenul și prima traiectorie folosită

Concluzii:

- Autonomia ne-a adus câștigarea tuturor meciurilor, fiind consecventă;
- Am primit confirmarea că o Autonomie bună poate face diferența între un meci câștigat și unul pierdut.

TeleOp:

În perioada de **TeleOp**, numărul automatizărilor a fost redus, din cazua structurii compacte a evenimentului.

Control Components:

- Gheara își păstrează constant poziția față de sol/*Backdrop*:

α = unghiul format de braț cu verticalca;

β = unghiul la care trebuie să ajungă gheara față de verticală;

$$\theta = \alpha + 90° - \beta$$

$$\alpha = \frac{EncoderCounts * 360°}{TicksPerRev} + Offset$$

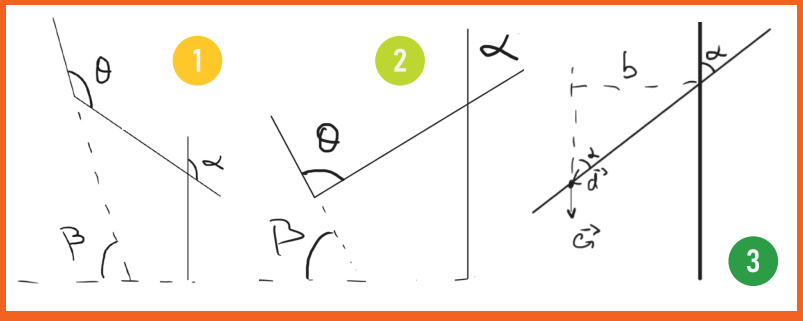
- Menținerea poziției brațului, folosind PIDF. Pentru FeedForward avem formulele:

$$\tau = b \times G$$
$$\tau = d \sin\alpha \times m\, g$$
$$\tau = d\, m\, g \times \sin\alpha \ (d,\ m,\ g \text{ sunt constante})$$
$$\tau = k \times \sin\alpha$$

Concluzii:

- Automatizările din perioada de TeleOp sunt extrem de importante, întrucât ajută *Driverii* să efectueze **mai puține mișcări** manuale ale subansamblelor, să se coordoneze și să acționeze mai rapid, ducând la **eficiență** sporită în timpul meciului.

Grafice pentru formule: Servo gheară (1), (2) și PIDF Feed Forward (3)



Robot v3 - League Meet #1:

Autonomie:

La scurt timp după KickAthon am realizat că perioada de Autonomie este complexă anul acesta și oferă o multitudine de posibilități de a puncta. Astfel, obiectivul nostru pentru **primul League Meet** (ZILELE ROBOTICII #2, organizat în data de **25 noiembrie** de echipele Inf0(1)Robotics #15993 și Ro2D2 #17962) a fost autonomia de **50 de puncte**, cuprinzând plasarea pixelului mov, a pixelului galben și parcarea.

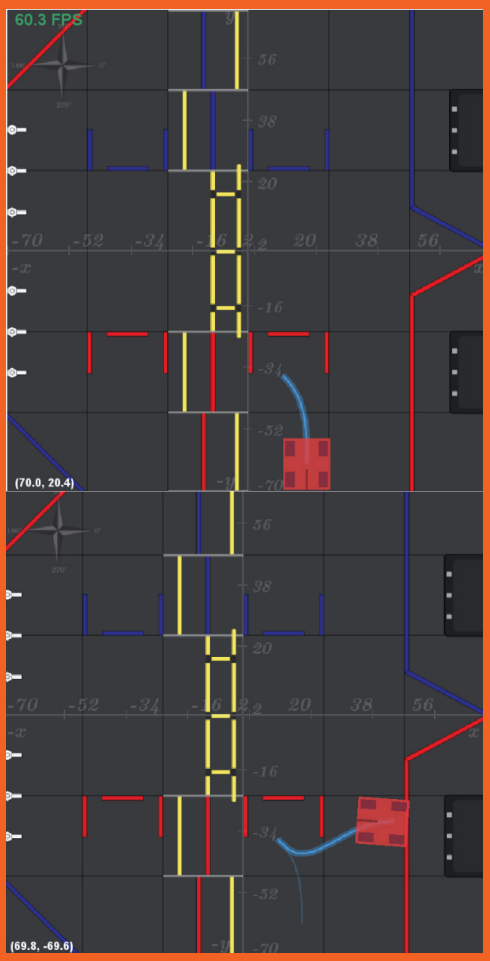
Implementări:

Pixelul mov:

Față de strategia pe care am gândit-o la KickAthon privind pixelul mov, am făcut o singură schimbare: în loc să plasăm pixelul cu ajutorul brațului, acum îl împingem cu **un ghidaj**. Pentru a aduce pixelul în poziție am folosit o singură traiectorie de tip **splineTo**, având grijă ca unghiul *Spline*-ului să nu fie prea agresiv pentru a nu scăpa pixelul din ghidaj.

Pixelul galben:

Pentru a afla poziția exactă în care trebuie plasat pixelul, este necesară **detectarea *Team Prop*-ului**, adică a locației acestuia într-o variabilă de tip integer (1-stânga, 2-mijloc, 3-dreapta). Acest proces trebuie realizat la începutul Autonomiei, în funcție de culoarea alianței. După plasarea pixelului mov, ne așezăm în fața *Backdrop*-ului și pornim **procesatorul de detecție**, ce returnează o listă de taguri. În funcție de primul citit începem **alinierea**, dând puteri motoarelor din obiectul *drive*. Când poziția tagului căutat pe axa Ox se încadrează în **toleranțele** stabilite, având în vederea poziția dorită a pixelului galben (stânga/dreapta), înaintăm și dăm servoului o poziție specifică pentru *Backdrop*. Apoi, acesta se retrage cu un delay de 0.5s și rămâne în poziție pe parcursul perioadei de TeleOp.



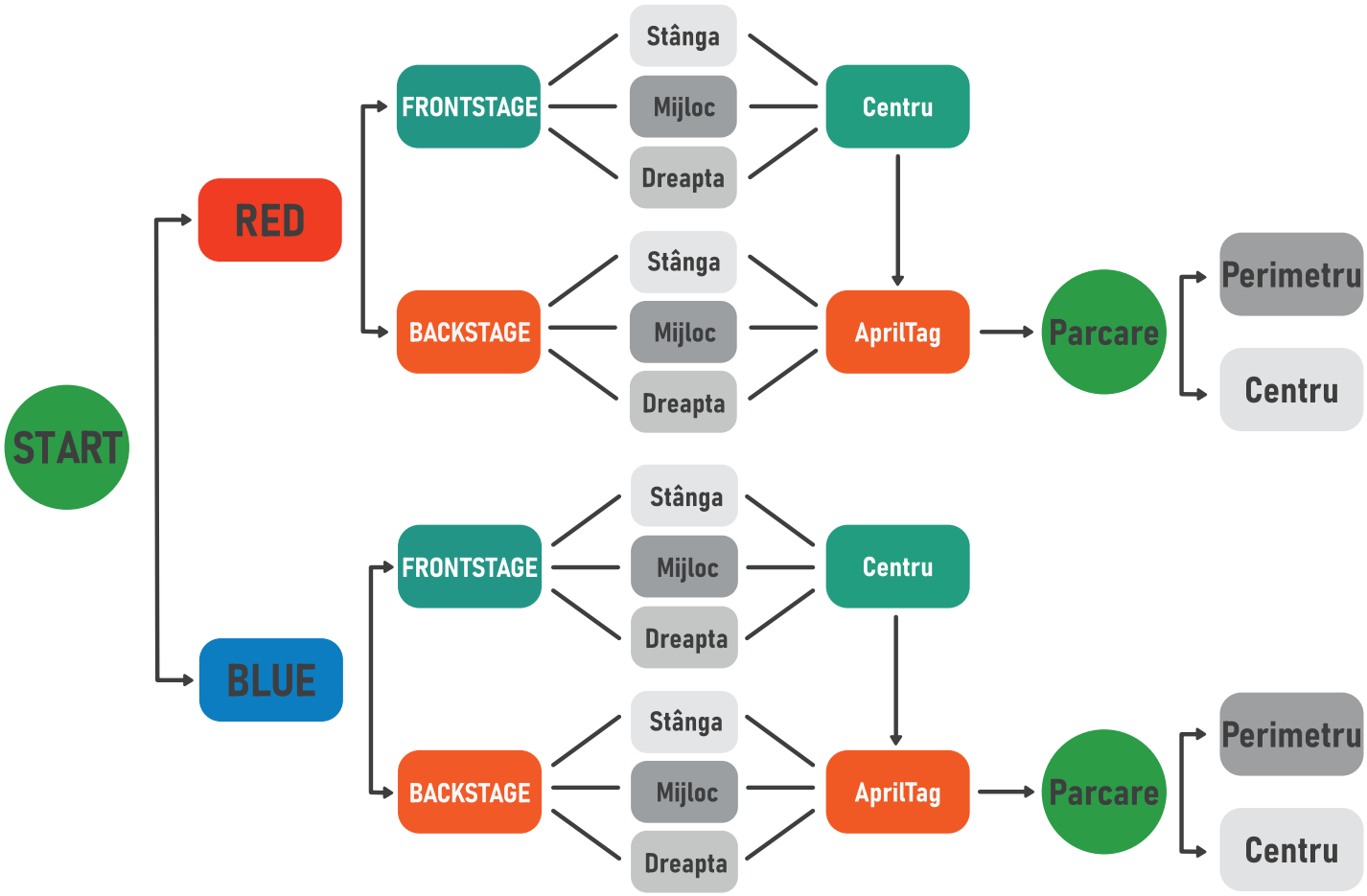
Reprezentare a traiectoriilor din Autonomie în terenul de joc

96 de cazuri:

În urma Autonomiei cu doi pixeli (galben și mov) am constatat anumite dificultăți, cea mai importantă fiind posibilitatea ciocnirii cu robotul din alianță. Astfel, am stabilit **6 criterii** de cazuri pentru a ne putea adapta în orice caz:

- Primul criteriu este cazul randomizat: stânga, dreapta sau centru (3 cazuri);
- Al doilea depinde de culoarea alianței, albastru sau roșu (2 cazuri);
- Al treilea reprezintă poziția de început pe teren, mai aproape (Backstage) sau mai departe (Frontstage) de *Backdrop* (2 cazuri);
- Al patrulea este poziția pixelului galben pe *Backdrop*, stânga sau dreapta (2 cazuri);
- Al cincilea depinde de traiectoria aleasă prin teren, pe lângă perimetru sau prin centru (2 cazuri);
- Ultimul criteriu reprezintă parcarea finală, în apropierea perimetrului sau a centrului (2 cazuri);

96 de cazuri: 2 alianțe x 2 poziții x 3 randomizări x 2 traiectorii x 2 poziții de pixeli x2 parcări



Parcare:

Am constatat că este necesară **parcarea separată** pentru a elibera spațiul eventualilor roboți din alianță care vor să facă misiunea cu pixelul galben, chiar dacă la finalul autonomiei suntem considerați parcați.

Finite State Machine:

Am considerat necesară această metodă de programare, deoarece aveam nevoia de a trece la stări anterior definite astfel, scăzand numărul de linii de cod.

Concluzii:

- RoadRunner 0.5 nu este exact;
- Eroare Index Out Of Bounds;
- Aliniere pe AprilTag ineficientă.

TeleOp:

Pentru primul League Meet am perfecționat Autonomia și Endgame-ul, din punct de vedere **strategic**, și abia după aceea focusul nostru s-a mutat asupra TeleOp-ului, pe care le am îmbunătățit după fiecare etapă.

Control Components:

- **Braț**: am folosit un PID și un Feedforward separat pentru a controla brațul. Pe lângă acestea, am avut și două servouri pentru gheară și încheietură;
- **Cățărat**: am utilizat câte un PID și un Feedforward pentru fiecare cârlig;
- **Șasiu**: am utilizat un sistem de conducere *field-centric*. De asemenea, folosim localizarea prin intermediul a 3 Dead Wheels.

Concluzii:

- Din cauza faptului că ne bazăm pe encoderele motoarelor, brațul nu avea precizie, oscilând.
- Odată ce ridicăm mecanismul de cățărat, acesta nu mai putea să fie retras pentru a încăpea sub truss. În cazul în care acesta este acționat accidental, rămânem blocați pe partea respectivă a terenului.

Robot v4.0 - League Meet #2:

Autonomie:

Pentru **al doilea League Meet** (organizat în data de **16 decembrie**, la București, de echipele Clockworks #19075 și Robo-Sapiens #19117) ne-am propus să punctăm atât **pixeli albi, cât și pixelii preîncărcați** în Autonomie, neavând suficient timp să realizăm aceste aspecte înainte de primul eveniment oficial. Astfel, am crescut fiabilitatea celor **50 de puncte** din Autonomie făcut anterior și, în plus, am adaugat și **colectarea pixelilor albi**.

Implementări:

Pixelul galben:

Cum pentru al doilea League Meet am avut o gheară cu o formă optimizată și am schimbat metoda de aliniere, am început să folosim **brațul pentru a plasa pixelul galben**.

Concluzii:

Poziția în care ajungem la *Stack* este variabilă (Nu ajungem mereu în poziția dorită la *Stack*).

Pixeli albi:

- Pentru prima variantă am avut idea de a ne duce la *Stack* cu ajutorul ***RoadRunner***-ului, ținând brațul suspendat, și de a apuca pixeli din **vârful *Stack*-ului**, gheara permițându-ne să colectăm între unul și doi pixeli într-un singur slot. Această metodă s-a dovedit foarte inconsistentă, existând posibilitatea de a lua 3 pixeli cât timp pixelul galben se află în partea cealaltă. Altă problemă întâmpinată a fost riscul de a dărâma *Stack*-ul. Ultima dificultate a fost faptul că ne este foarte greu să ajungem în poziția respectivă, din cauza erorilor de la odometrie.
- Pentru cea de-a doua iterație am avut idea de a colecta din **două *Stack*-uri simultan**, cu ajutorul a două cîrlige ce împing pixelul de jos, restul *Stack*-ului fiind menținut de frecarea dintre robot și peretele din spate.

TeleOp:

Pentru al doilea League Meet, ne-am concentrat și asupra **optimizării TeleOp-ului**, prin automatizări și programarea unui tip diferit de braț, cel cu **glisiere**, față de restul subansamblelor care au păstrat elemente comune între versiunile de robot.

Control Components:

- **Braț & Cățărat:** Am folosit **două controlere PIDF** pentru poziționarea glisierelor. Pe lângă acestea am avut și **5 servouri** pentru braț, gheară și încheietură;
- **Șasiu:** *Robot-centric drive*, transformă mișcarea robotului din **sistemul relativ de coordonate** al acestuia, în **sistemul absolut al terenului**. Acest mod de control este preferat de unii driveri. Astfel, nu mai este necesar ca aceștia să ia în considerare **orientarea robotului față de teren**;
- **Automatizări:**

Când *Driver*-ul ridică liftul, brațul se pune automat în poziția de *Backdrop*, și când îl coboară se duce în poziția de colectare, indiferent de nivelul la care a fost ridicat liftul;

Driver-ul poate **alege nivelul liftului** fără ca acesta să se ridice în acel moment, iar la următoarea apăsare a butonului de mutare a brațului în poziția de punctare, glisierele ajung automat la *poziția selectată anterior*;

Control simultan al ambelor paletele de la gheară sau individual pentru mai multă precizie.

Concluzii:

- Am realizat necesitatea unui ***motion profile***, care să elimine forțele excesive care apar în urma opririi bruște;
- Nu știm dacă avem sau nu pixeli în gheară, irosind mult timp să ne întoarcem după cei lăsați în urmă.

Robot v4.1 - League Meet #3 și Regionala #1:

Autonomie:

Partea de plasare a pixelilor a pixelului mov și a celui galben din Autonomie a funcționat la League Meet-ul anterior, așadar am decis să o păstrăm și pentru cel în care suntem coorganizatori alături de echipele 4D-ROBOTICS #18160, Broboți #19176 și LighBulb Robotics #23203 (HIDE & MEET, în data de 13 ianuarie 2024, în Pitești). Modificarea adusă este **alinierea pe AprilTag**, introducând și o corecție folosind *unghiul*. Pentru partea de colectare a pixelilor albi, am remediat majoritatea problemelor pe care le aveam (colectarea inconsistentă cauztă de poziționarea imprecisă și dărâmarea sau deplasarea ocazională a *Stack*-urilor).

Implementări:

Motion Profile:

Accelerare pe servourile de la palete pentru a prinde mai bine din *Stack*-uri.

Senzori de culoare:

Folosim senzorii de culoare din gheară pentru a determina automat în ce gheară este pixelul galben la începutul meciului. Totodată, cu ei se verifică dacă pixelul alb a fost prins de gheară după ce a fost scos din *Stack*.

Probleme:	Soluții:
• Uneori pixelii erau prinși incorect din *stack* și scapă înainte să ajungă la *Backdrop*; • După ce folosim paletele la *Stack*, dacă nu sunt închise, cablurile de la gheara se pot prinde în ele, iar dacă strângem paletele înainte să ridicăm wrist-ul, acestea se pot bloca definitiv.	• Verificăm cu senzorul de culoare dacă un pixel este prins incorect. În caz afirmativ, robotul le dă drumul și încearcă să îi mai prindă o dată; • Facem o poziție auxiliară de tranzit la palete care să nu încurce la ridicarea wrist-ului și strângem paletele la final de Autonomie.

Concluzii: Riscăm să nu punctăm pixelul galben dacă coechipierul nu respectă slotul stabilit inițial. !

Robot v5 - Națională:

Autonomie:

După finalizarea etapei Regionale, am realizat o **analiză retrospectivă** a robotului v4.1, ajungând la concluzia că îmbunătățirile posibile nu corespund cu performanța dorită și astfel am apelat la un nou robot, cu o autonomie regândită.

Implementări:

Pixelul mov:

Pentru această etapă am renunțat la **ghidajul extern** de pixel mov, transformându-l într-un sistem intern. În zona de *Backstage* am restructurat traseul, plasând mai întâi pixelul galben. Astfel, folosim Intake-ul pentru a **împinge pixelul mov** și utilizăm *Team Prop*-ul atât cu scopul de a controla poziția în care ajunge, cât și de a ne asigura că nu îl împingem prea departe. În zona de *Frontstage* folosim ghidajul intern.

Pixeli albi:

Pentru etapa Națională ne-am propus să plasăm pe *Backdrop* **3 pixeli albi** din zona de *Frontstage*, și **4** din *Backstage*, acest lucru fiind permis de motoarele mai rapide ale șasiului și de **intake-ul activ**. Pentru colectarea din *stack*, inatke-ul are o poziție care controlează exact numărul necesar de pixeli. Pe parcursul testelor, am observat posibilitatea ca ei să nu intre în sistemul de transfer și am decis că cea mai bună soluție pentru colectarea constantă este să **coborâm intake-ul pe sol** după ce dărâmăm pixelii din *stack*. În cazul puțin probabil în care colectăm mai mulți pixeli, când plecăm de la *stack* pornim intake-ul în **sens invers**, pentru a îi scoate pe cei în plus, gheara fiind concepută încât să nu poată prelua mai mult de doi.

Concluzii:

- Dacă ajungem prea aproape de *stack* există riscul ca pixelii să intre **suprapuși**, blocându-se intake-ul;
- Cum brațul are părți mobile, riscăm ca acesta să nu se închidă complet dacă avem **voltajul scăzut**;
- Cu ajutorului intake-ului activ și al suspensiei de pe braț, avem o mult mai mare **toleranță** la unghiul de la care colectăm și plasăm pe *Backdrop*.

TeleOp:

Pentru etapa Națională ne-am concentrat și asupra **eficientizării** perioadei de TeleOp, cu scopul de a obține punctaje maximizate, astfel introducând o serie de noi automatizări.

Control Components:

- **Automatizări:**

Photonul, pentru îmbunătățirea *loop time*-ului (informații mai detaliate sunt la capitolul **Noțiuni Generale**).

Prin apăsarea unui buton, brațul se poziționează la **niveluri prestabilite**, acestea putându-se ajusta pentru a crește precizia plasării pixelilor pe *Backdrop*;

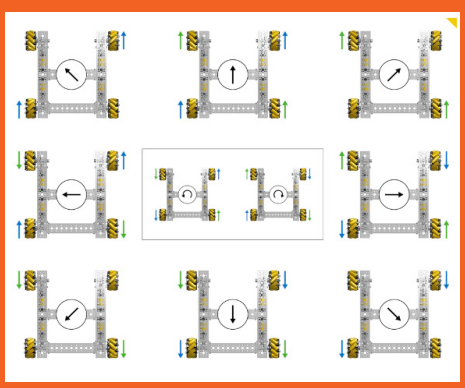
Motoarele care acționează brațul sunt foarte rapide, dar mișcarea devine astfel prea bruscă. În consecință, am decis că este necesară încetinirea acestora, pentru a putea **controla cățărarea**;

Pentru a ne asigura că nu controlăm mai mulți pixeli din greșeală, imediat după ce colectează doi și senzorii din intake închid ghearele automat, acesta se **rotește în sensul opus**;

Deoarece avionul are 2 servo-uri care trebuie acționate pentru a funcționa, am făcut ca la apăsarea unui singur buton, acestea să funcționeze **secvențial**;

Pozițiile pivotului sunt calculate pe baza poziției *stick*-ului de la *Driver*-ul 2.

Noțiuni generale:



Mișcare:

Cinematica roților Mecanum:

Cinematica este **aplicarea geometriei** la controlul diferitelor navigații ale robotului. Ecuațiile cinematicii sunt utilizate pentru a controla mecanismele, furnizând date specifice pentru a obține un rezultat dorit.

Cinematica Directă vs. Cinematica Inversă:

Mecanisme pot avea seturi diferite de ecuații pentru cinematica lor directă și cinematica lor inversă. **Cinematica directă** constă în ecuațiile

utilizate pentru **determinarea stării** unui sistem dat utilizând datele furnizate de componentele sale. **Cinematica inversă** determină setul de date necesare pentru ca un sistem **să ajungă în starea dorită**. De exemplu, într-un sistem de mișcare, cinematica directă ar determina viteza robotului pe baza vitezelor individuale ale roților, iar cinematica inversă ar determina vitezele necesare ale roților pentru o viteză dorită a corpului.

Formulele aferente pentru Cinematica Directă și Cinematica Inversă

$$V_f = \frac{V_{fr} + V_{fl} + V_{br} + V_{bl}}{4}$$

$$V_s = \frac{V_{bl} + V_{fr} - V_{fl} - V_{br}}{4}$$

$$\omega = \frac{V_{br} + V_{fr} - V_{fl} - V_{bl}}{4 * 2R_b}$$

unde:
- V_f reprezintă viteza de înaintare a robotului, în raport cu el însuși;
- V_s reprezintă viteza laterală a robotului, în raport cu el însuși;
- ω reprezintă viteza de rotație a robotului în radiani/secundă;
- R_b reprezintă distanța dintre roată și centrul robotului (jumătate din distanța dintre roți).

$$V_{fl} = V_f - V_s - (2R_b \times \omega)$$

$$V_{bl} = V_f + V_s - (2R_b \times \omega)$$

$$V_{br} = V_f - V_s + (2R_b \times \omega)$$

$$V_{fr} = V_f + V_s + (2R_b \times \omega)$$

- V_{fr} reprezintă viteza liniară a roții din față dreapta;
- V_{br} reprezintă viteza liniară a roții din spate dreapta;
- V_{fl} reprezintă viteza liniară a roții din față stânga;
- V_{bl} reprezintă viteza liniară a roții din spate stânga.

PID & FeedForward:

PIDF Controller este o buclă de control care acționează în funcție de valorile furnizate de sistem, aplicând o corecție bazată pe **termeni proporționali, integrali și derivați**, de unde și numele. Acestora li se adaugă un termen corespunzător sistemului, în general, *feedforward*. Pe scurt, acesta aplică o corecție precisă, corespunzătoare erorii.

Parametri:

- Setpoint (SP) - valoarea dorită;
- Process Variable (PV) - valoarea returnată de sistem;
- KP - coeficientul proporțional;
- KI - coeficientul integralei;
- KD - coeficientul derivatei;
- KF- coeficientul FeedForward.

Formula generală

$$corecție = KP \times eroare + KI \times sumEroare + \\ + KD \times (eroare - ultEroare) + KF \times valSist$$

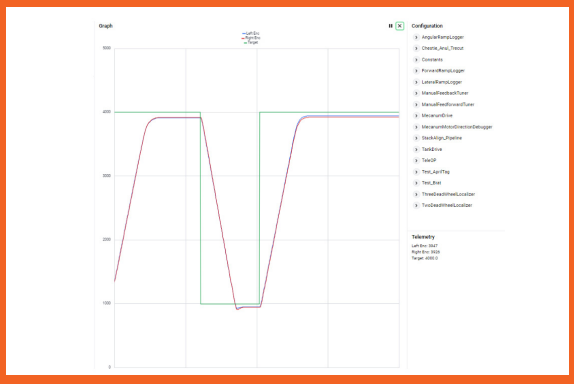
unde: eroare - diferența dintre valoarea citită și cea dorită;
sumErori - suma tuturor erorilor;
ultEroare - eroarea precedentă;
valSist - variabilă specifică sistemului.

Obs.1

În funcție de sistemul pentru care este utilizată, această buclă de control folosește doar anumiți termeni. În industrie, este foarte des întâlnit un PD Controller, iar pe versiunea actuală de robot folosim un PF Controller.

Obs.2

Ajustarea parametrilor PIDF poate implica testarea și reglarea manuală sau utilizarea unor algoritmi de optimizare.



Motion Profiles:

Un ***motion profile*** este o funcție folosită pentru a **schimba gradual viteza unui sistem**, într-un mod controlat și consecvent, prin incrementarea respectiv decrementarea treptată a acesteia până să ajungă la cea dorită. Spre exemplu, dacă vrem ca robotul, care stă pe loc, să meargă în față la viteză maximă, în general, dacă am da putere maximă la toate motoarele, există o șansă foarte mare ca robotul să facă mișcări nedorite și incontrolabile pentru că, deși dăm puterea la toate motoarele simultan, robotul nu va ajunge instant la viteza respectivă. *Motion Profiling*-ul rezolvă această problemă.

Avantaje:	Dezavantaje :
• Mișcări mai controlate și mai predictibile; • Reduce schimbarea rapidă a tensiunii aplicate motorului.	• Folosește mișcări mai încete, deci timpul de executare efectivă este mai ridicat.

Există două tipuri de *motion profile*: profiluri de **tip trapezoidal**, respectiv profiluri de **tip S-Curve**. În acest sezon am ales să folosim un *motion profile* de tip trapezoidal, deoarece permite control optim asupra mișcărilor în schimbul a puțin timp în plus față de cel S-Curve și este mai usor de implementat.

Obs.1 Numele acestui tip de profile vine de la forma graficului vitezei în funcție de timp.

Formule pentru graficele funcțiilor:

$$v_c = v_0 + a \times t$$

$$x = x_0 + v_0 \times t + \frac{a \times t}{2}$$

unde v_c viteza actuală;
v_0 viteza de început (de obicei 0);
a acceleratia actuală;
t timpul actual;
x pozitia actuală;
x_0 poziția de start.

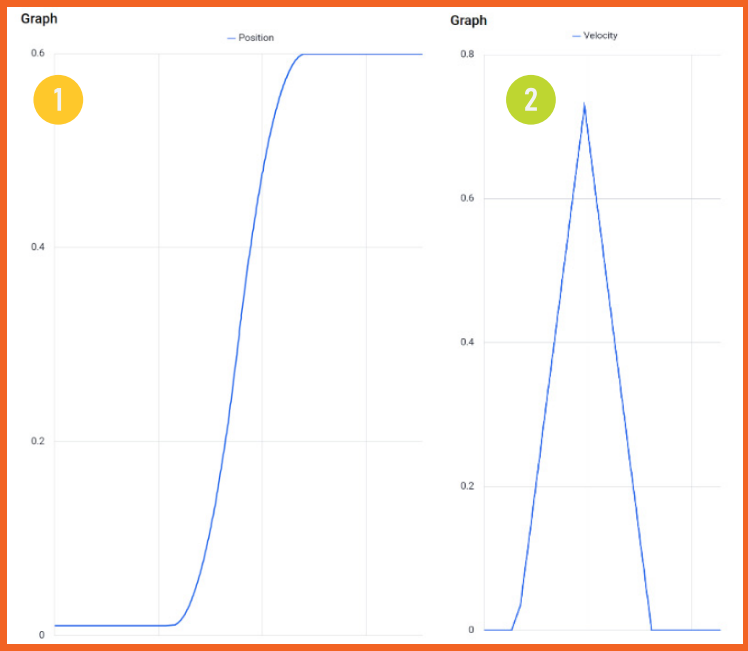
Formule pentru graficele funcțiilor:

$$t_a = \frac{v_{max}}{a}$$

$$d_a = \frac{a}{2} \times t_a$$

Obs.

- Dacă $d_a > \frac{d}{2}$, atunci $t_a = \sqrt{\frac{d}{a}}$ și $v_{max} = a * t_a$
- Accelerăm la fel de mult pe cât decelerăm (doar pentru acest exemplu): $t_d = t_a$, $t_{total} = t_a + t_{cruise} + t_d$, $t_{cruise} = \frac{d_{cruise}}{v_{max}}$, $d_{cruise} = d - 2 \times d_a$



Graficul poziției unui servo de la braț dintre două mișcări (1)
Graficul vitezei unui servo de la braț dintre două mișcări (2)

Obs.2 Acest *motion profile* (2) a fost gândit fără o perioada de "cruising" în care el ar avea o viteză constantă pentru a avea mai multă accelerare și frânare, care sunt mai importante pentru acest sistem.

- În funcție de momentul la care suntem la una din patru etape:
  - Dacă $t > t_{total}$, nu mai suntem în *profile*, deci poziția este defapt distanța finală;
  - Dacă $t < t_a$, suntem în perioada de accelerare și $x = \frac{a}{2} \times t$;
  - Dacă $t < t_d$, suntem în perioada de cruising și $x = \frac{a}{2} \times t_a + v_{max} \times (t - t_a)$;
  - Altfel, suntem în perioada de decelerare și $x = \frac{a}{2} \times t_a + d_{cruise} + (v_{max} - \frac{a}{2})(t - t_d)$, unde d este distanța pe care vrem să implementăm *Motion Profiler*-ul.

Mișcarea din TeleOp:

Mișcarea cu Robot-Centric:

Robot-Centric Drive păstreaza mișcarea robotului în cadrul **sistemul relativ de coordonate al acestuia**.

$$axial = -1 \times (mișcarea\ Oy\ a\ gamepad\text{-}ului\ din\ stânga)$$
$$strafe = mișcarea\ Ox\ a\ gamepad\text{-}ului\ din\ stânga$$
$$turn = mișcarea\ Ox\ a\ gamepadu\text{-}lui$$

$$power\_left\_back = axial - strafe + turn$$
$$power\_left\_front = axial + strafe + turn$$
$$power\_right\_back = axial + strafe - turn$$
$$power\_right\_front = axial - strafe - turn$$

Mișcarea cu Field-Centric:

Cu un sistem de conducere *Field-Centric*, joystick-ul controlează direcția robotului în raport cu **sistemul de coordonate relativ al terenului**.

Acest lucru este preferat de unii Driveri și facilitează anumite acțiuni evazive, deoarece se poate roti mai ușor în timp ce se deplasează într-o anumită direcție. Pentru a realiza acest lucru, componentele x/y ale joystick-urilor sunt rotite în sens opus față de unghiul robotului, furnizat de IMU.

Formule pentru determinarea puterilor care trebuie distribuite la roți în mișcarea cu Field-Centric:

$$L \times cos(\alpha + \beta) = L \times cos\alpha \times cos\beta - L \times sin\alpha \times sin\beta$$
$$L \times sin(\alpha + \beta) = L \times sin\alpha \times sin\beta + L \times cos\alpha \times cos\beta$$

$$L \times cos(\alpha + \beta) = x_1 \times cos\beta - y_1 \times sin\beta \Rightarrow x_2 = x_1 \times cos\beta + y_1 \times sin\beta$$
$$L \times sin(\alpha + \beta) = y_1 \times sin\beta + x_1 \times sin\beta \Rightarrow y_2 = y_1 \times sin\beta + x_1 \times cos\beta$$

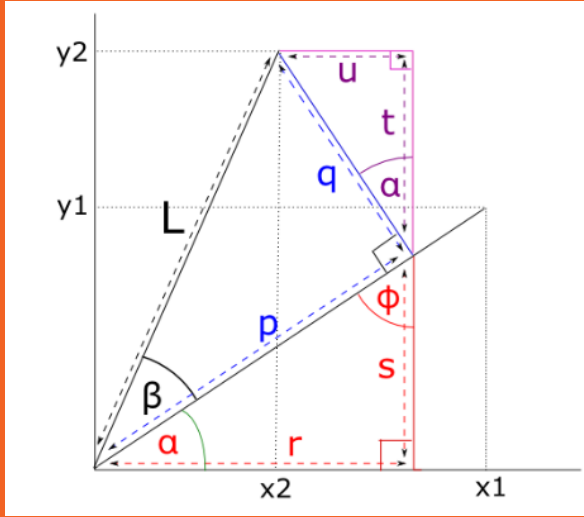
$$x_1 = L \times cos\alpha$$
$$y_1 = L \times sin\alpha$$
$$x_2 = L \times cos(\alpha + \beta)$$
$$y_2 = L \times sin(\alpha + \beta)$$

$$rotX = strafe \times cos(-angle) - forward \times \times sin(-angle)$$
$$rotY = strafe \times sin(-angle) + forward \times \times cos(-angle)$$

$$power\_left\_back = rotY - rotX + turn$$
$$power\_left\_front = rotY + rotX + turn$$
$$power\_rightt\_back = rotY + rotX - turn$$
$$power\_right\_front = rotY - rotX - turn$$

unde am folosit formulele de calcul trigonometric:

$$cos(\alpha + \beta) = cos\alpha \times cos\beta - sin\alpha \times sin\beta$$
$$sin(\alpha + \beta) = sin\alpha \times sin\beta + cos\alpha \times cos\beta$$



RoadRunner 0.5.x:

Road Runner este o bibliotecă de planificare a mișcării. Aceasta este concepută în principal pentru **mișcarea autonomă**, permițând urmărirea și generarea de **traiectorii complexe**. Cu ajutorul acesteia reușim să gândim autonomii cât mai eficiente din punct de vedere al mișcării pe teren.

Drive Constants:

Constantele de amplificare feedforward utilizate pentru modelarea motoarelor de șasiu au următoarele unități de măsură:

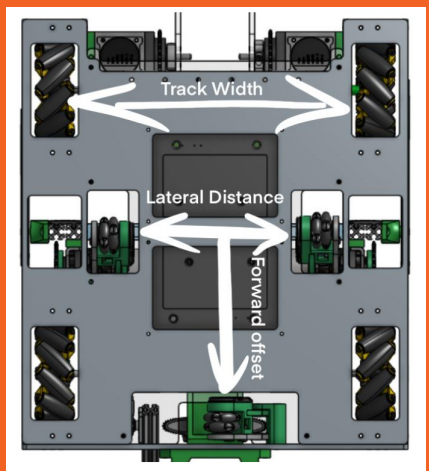
$$[kV] = \frac{V \times s}{m}$$
$$[kA] = \frac{V \times s^2}{m}$$
$$[kStatic] = V$$

$$v_{max} = \frac{MAX\_RPM}{60} \times \times GEAR\_RATIO \times \times WHEEL\_RADIUS \times 2\pi$$

Formulă pentru calculul vitezei unghiulare:

$$\omega_{max} = \frac{v_{max}}{TRACK\_WIDTH}$$

Variabilele utilizate au nume sugestive, precum WHEEL_RADIUS este raza roții, MAX_RPM este numărul rotațiilor pe minut efectuate cu tensiunea maximă recomandată sau GEAR_RATIO este raportul de transmisie.

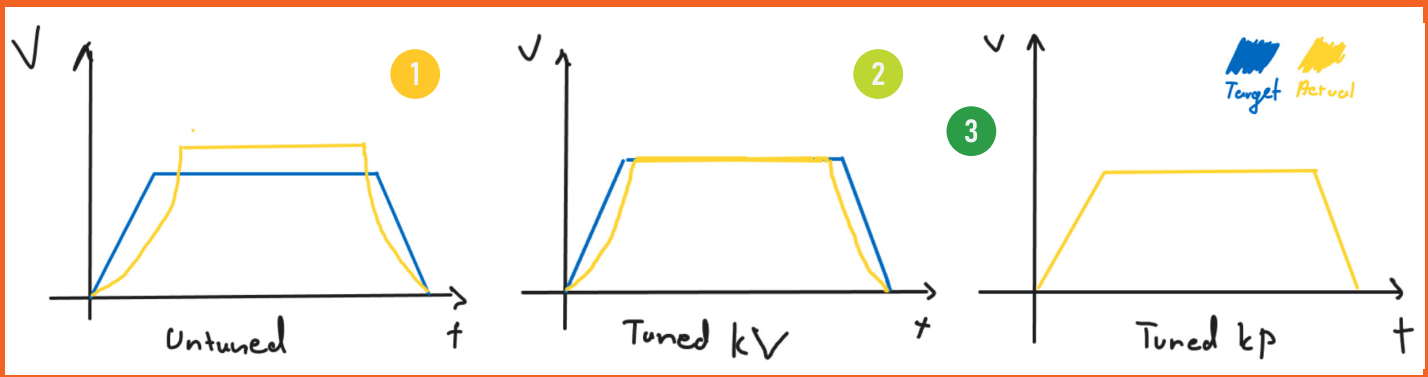


Reprezentarea grafică pe robot a variabilelor TRACK_WIDTH, LATERAL_DISTANCE, respectiv FORWARD_OFFSET.

Calibrare:

FeedForward:

Pentru o urmărire **precisă** a traiectoriilor este necesară ajustarea controller-ului FeedForward, în caz contrar, erorile apărute nu vor fi corectate precis și vor începe să se adune.



O serie de trei grafice care evidențiază evoluția pentru valorile Target și Actual (1), (2) și (3).

Acest proces include două moduri diferite de calibrare: BackAndForth și FollowerPIDTuner. Utilizăm BackAndForth atunci când vrem să obținem un set de parametri **PID** aproximativi, după care utilizămw FollowerPIDTuner pentru o ajustare mai precisă. În modul BackAndForth, robotul se deplasează în linie dreaptă înainte și înapoi. În modul FollowerPIDTuner, robotul urmărește o serie de traiectorii care formează un pătrat.

RoadRunner 1.0.x:

Aceasta este ultima versiune de RoadRunner care are același scop, dar implementarea este mai accesibilă fiind scris în **Java**, dar și mai fiabilă.

Actualizări:

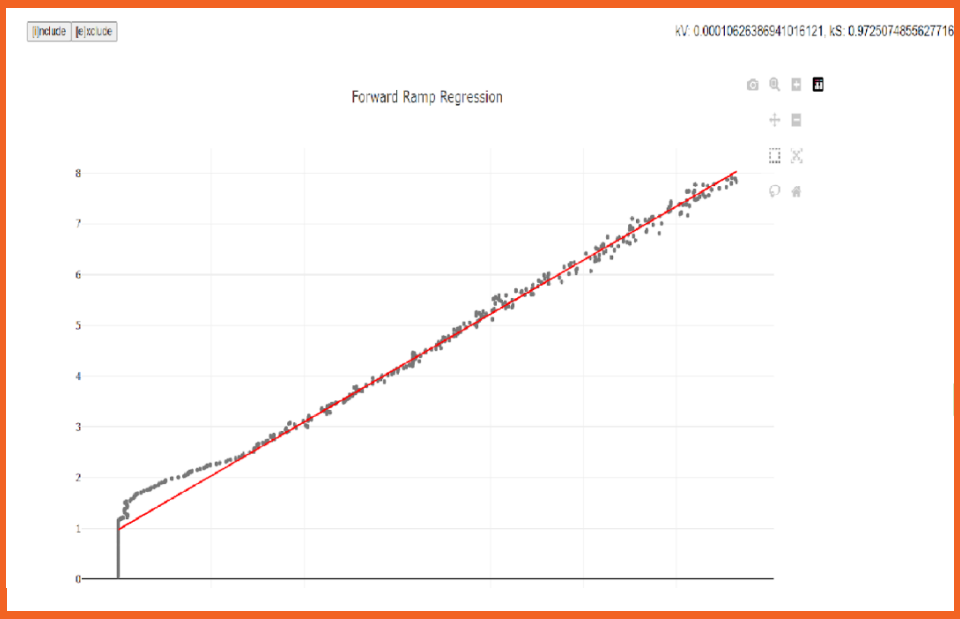
Cea mai mare schimbare constă într-un nou **quickstart** pentru a oferi rezultate mai bune de calibrare într-un timp mai scurt. De asemenea, există schimbări semnificative în API-ul bibliotecii de bază:

- În caz că apare o problema de continuitate la traseu, atunci traiectoria este împărțită în altele mai mici;
- Se pot specifica **limite de accelerare și decelerare** separate în traiectorii, în funcție de necesitate. ProfileAccelerationConstraint acceptă două valori, o accelerație minimă (negativă) și o accelerație maximă (pozitivă);
- În cazul în care dorim oprirea robotului înainte ca acesta să fii ajuns la poziția finală, ultima versiune de RoadRunner ne permite să facem acest lucru. În loc de a se opri instant, robotul folosește **decelerarea forțată** (“frâna de urgență”).

Calibrare:

Fiecare robot este diferit, iar Road Runner-ul trebuie calibrat individual. Procesul constă în rularea mai **multor teste** și ajustarea parametrilor manual în funcție de răspuns sau adaptarea automată a acestora pe baza datelor colectate.

Calibrarile manuale au fost înlocuite cu citiri ce pot să fie interpretate sub **formă de grafic**. Acest lucru ne ajută să observăm citirile eronate mai ușor, ajungând la o valoare mai apropiată de cea reală.



Obs.

Calibrarea ar trebui să fie repetată periodic și mai ales atunci când există modificări semnificative, la nivel mecanic, ale robotului.

Acțiuni:

Acțiunile ne ajută să definim mișcări **mai puțin complexe**, ușor de îmbinat. Prin delimitarea programelor, le facem mai simplu de înțeles și de modificat. Acțiunile pot fi reutilizate în programe autonome noi, astfel niciodată nu începem de la zero.

Acțiunile personalizate:

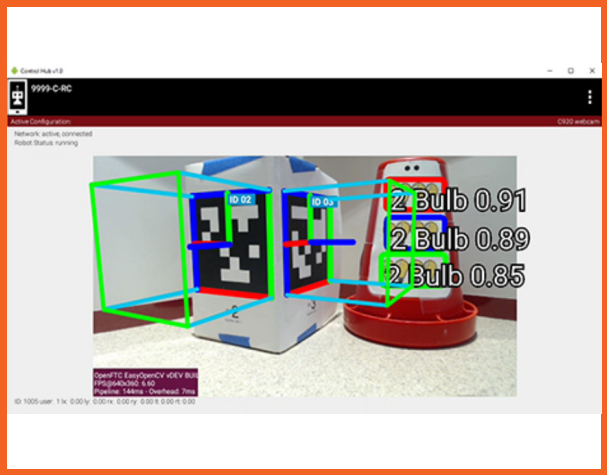
La bază, acțiunile sunt segmente de cod care rulează în mulți pași mici. Această proprietate ne permite să rulăm două acțiuni A și B **în paralel**, fără a utiliza mai multe Threaduri de execuție. Prin executarea “Pasul A”, ”Pasul B”,

"Pasul A", ... într-un mod alternant, acțiunile A și B par să progreseze simultan. Dar iluzia poate fi ușor distrusă dacă "Pasul A" durează mult timp și îl împiedică pe B să funcționeze.

Pentru a crea o acțiune personalizată, realizăm o clasă care implementează interfața Action:
- public boolean run (TelemetryPacket packet): Codul rulează repetitiv în timp ce metoda returnează true.
- Apelurile la run() ar trebui să se completeze rapid. Întârzierile mai lungi de 100ms vor începe să afecteze semnificativ alte acțiuni.

MeepMeep:

MeepMeep este un **vizualizator de traiectorie** proiectat special pentru noul API de Road Runner, introdus în mai 2021. Acesta ne permite crearea, testarea și îmbunătățirea traiectoriilor în lipsa unui robot, îmbunătățind major **productivitatea** noastra ca echipă. Desigur MeepMeep nu poate lua în considerare toți factorii externi, acesta fiind util în special pentru prima iterație, testarea pe robot fiind esențială.



Camera:
Vision Portal:

VisionPortal reprezintă o nouă clasă care îmbină funcțiile fundamentale din EasyOpenCv și TensorFlow Object Detection(TFOD). Acestea pot fi folosite simultan. La un VisionPortal putem adăuga unul sau mai multe procesatoare pentru a **prelucra imaginea** furnizată de cameră în moduri diferite.

Pentru partea Autonomă, am decis să folosim o singură cameră web cu interfață de VisionPortal la care am adăugat 2 procesatoare: unul pentru detecția Team Propului și unul pentru recunoașterea unui AprilTag.

Detecția Team Prop-ului:

Pentru acest procesor, am creat o clasă separată în care prelucrăm imaginea astfel:
- Transformă imaginea din RGB în HSV;
- Definim 2 zone de căutare, a unei anumite culori (roșu sau albastru), în stânga și în mijlocul ecranului;
- Returnăm zona în care culoarea căutată este cea mai proeminentă.

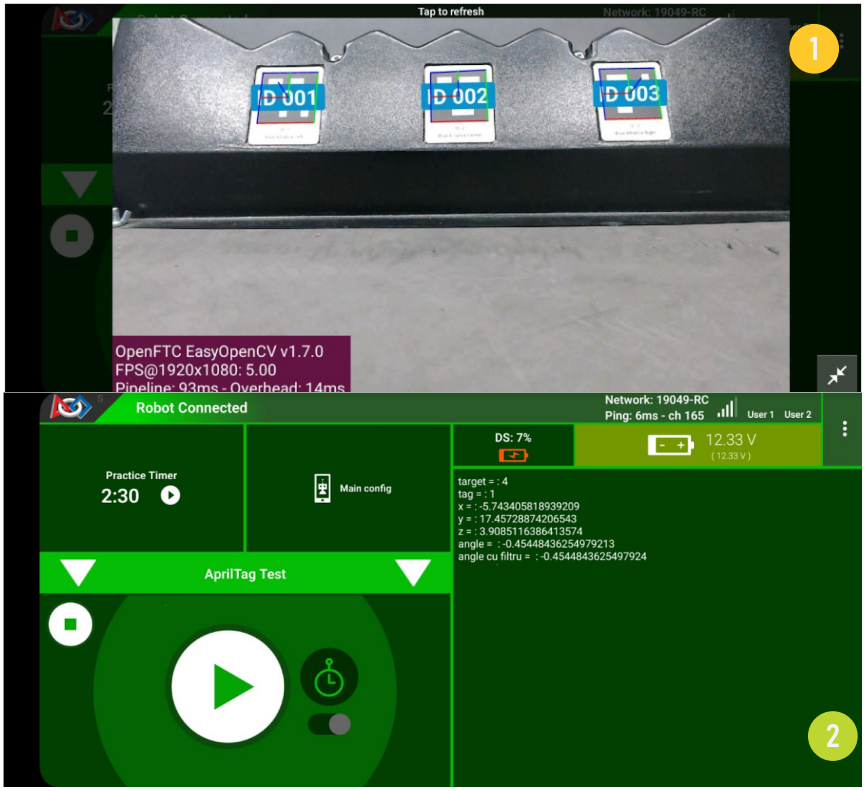
Dacă ne interesează culoarea roșu transformă imaginea din BGR in HSV. **Obs.1**

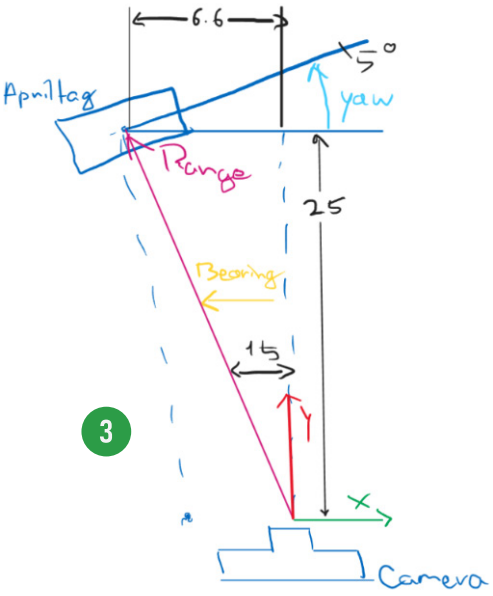
Dacă Team Prop-ul nu este nici în zona stângă și nici în zona de mijloc, atunci este automat în aria dreaptă. **Obs.2**

Identificarea cazului la început de Autonomie realizată cu camera

Recunoașterea unui AprilTag:

Recunoașterea unui AprilTag începe prin **procesarea imaginii**, urmând citirea contururilor sau a marginilor. AprilTag-urile au un contur specific care le face ușor de identificat. După detectarea contururilor, urmează etapa de identificare a modelului AprilTag. Fiecare tip de AprilTag are un model specific cu coduri binare, prin intermediul cărora îl putem identifica. După identificare, se decodează datele stocate în AprilTag. Aceste date includ un identificator unic pentru tag și alte informații relevante, cum ar fi poziția și orientarea lui.



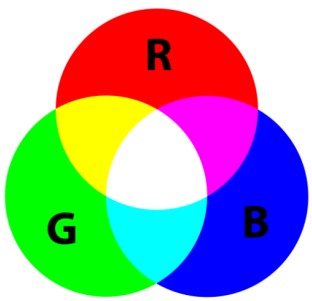


Inagini cu:
- Recunoașterea AprilTag-urilor de pe BackDrop (1);
- Citirea distanțelor din centrul camerei la Tag-uri(este afișată doar distanța pana la Tag-ul 1) (2);
- Citirea poziției camerei fata de un AprilTag (3).

Computer Vision:

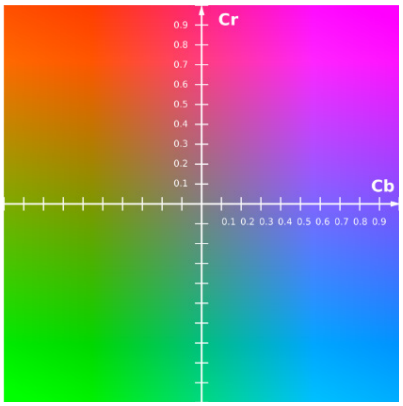
Pentru a prelucra imagini, acestea trebuie **interpretate numeric**. Culoarea este redată cu ajutorul a **3 matrice**, fiecare exprimând o caracteristică a culorii. Astfel de reprezentări sunt **modele cromatice** și printre ele se numără:

RGB: Modelul standard pentru o cameră web, având 3 tipuri de receptori de culoare: **roșu**, **verde** și **albastru**. Valorile înregistrate semnifică **intensitatea luminii** captate de fiecare receptor.



Avantaje:	Dezavantaje:
• Nu necesită schimbarea modelului cromatic,salvând un pas.	• Pentru a descrie o culoare este se modifică toate cele 3 valori.

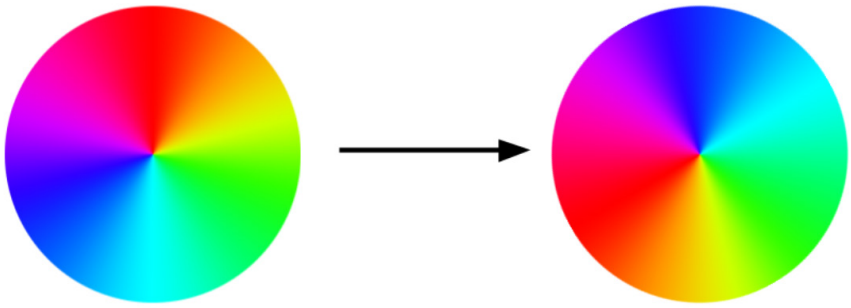
YCbCr/YUV: **Y (luma)** este luminozitatea echivalentului **acromatic** (nuanță de gri) al culorii; **Cb/U (A - Y)** redă componența cromatică reprezentată de **diferența** dintre **albastru** și luminozitatea acromatică; **Cr/V (R - Y)** redă componența cromatică reprezentată de **diferența** dintre **roșu** și luminozitatea acromatică;



Avantaje:	Dezavantaje:
· Nuanțele sunt ușor de identificat, mai ales roșu și albastru.	· Anumite culori sunt influențate de luminozitate mai puternic.

HSV/HSB: **H (hue)** simbolizează culoarea, așezată pe un spectru ce începe de la roșu și se termină tot la roșu; **S (saturation)** reprezintă saturația culorii. Aceasta este intensitatea culorii; **V (value)/B (brightness)** este luminozitatea. Ea desemnează luminarea sau întunecarea relativă a culorii.

Avantaje:	Dezavantaje:
· Culoarea este reprezentată de o singura valoare; · Diferența de lumină este foarte ușor de sesizat.	· Culoarea roșu se afla la ambele capete ale modelului.



RGB -> HSV

BGR -> HSV

Pentru a rezolva problema culorii roșu, ce se află simultan la ambele capete, în loc să transformăm din RGB în HSV, putem să ne prefacem că transformăm din BGR în HSV, punând de fapt valorile RGB. Astfel, obținem un alt spectru, ce este inversat, roșu cu albastru făcând schimb de locuri. Folosind acest truc rezolvăm și problema culorii roșu.

Detecția de obiecte :

Imaginile obținute de camera web pot fi folosite pentru a detecta anumite obiecte, precum elementele de joc.

Color Thresholding:

Pentru a detecta un obiect, căutăm toți pixelii din imagine ce se află într-un interval de valori. Apoi creăm un Mesh binar în care, pentru fiecare pixel ce se află în intervalul bun, alocăm un pixel alb, iar pentru restul păstrăm negru.

Avantaje:	Dezavantaje:
· Este ușor de implementat	· Trebuie calibrat pentru fiecare mediu nou în care e folosită.

Detecția adâncimii folosind două camere:

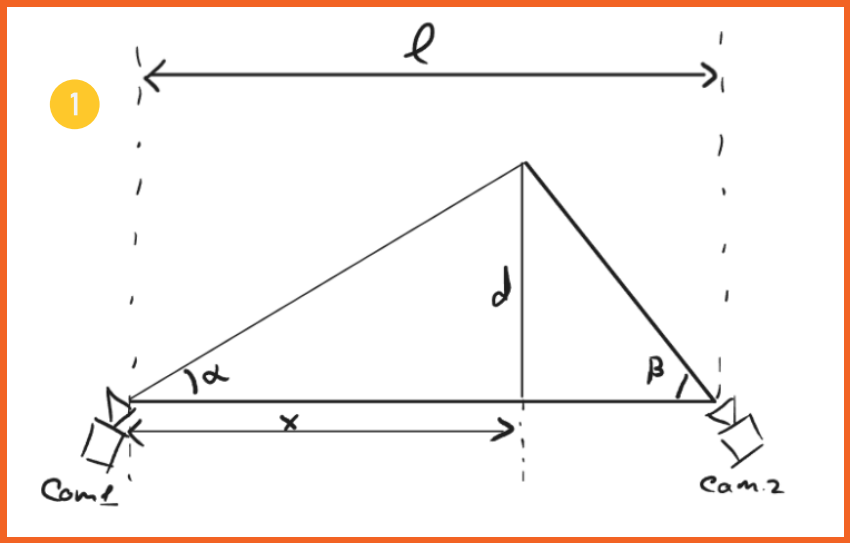
Pentru a determina distanța la care se afla un obiect față de robot am plecat de la abilitatea omului de a percepe distanță cu ajutorul ochilor. Folosind două camere plasate la o distanță cunoscută între ele, putem recreea acest lucru.

$$\alpha = \frac{poziție\ pixeli \cdot FOV}{rezolutie}$$
$$x = \frac{d}{tan(\alpha)}$$
$$l - x = \frac{d}{tan(\beta)}$$

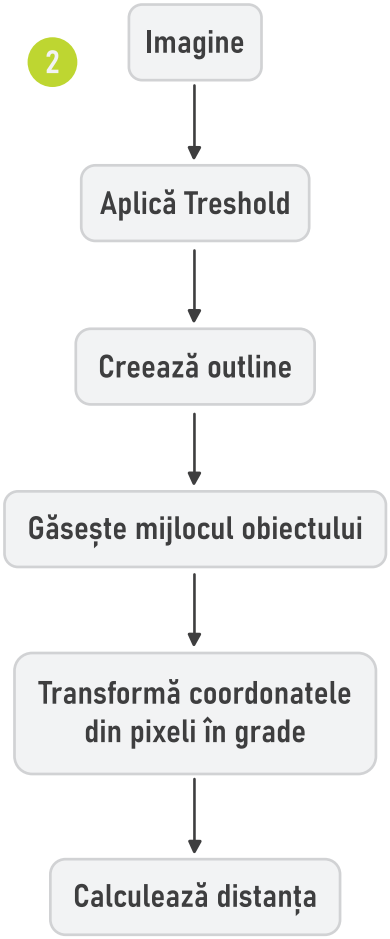
$$l = \frac{d}{tan(\alpha)} + \frac{d}{tan(\beta)}$$
$$l = d\left(\frac{1}{tan(\alpha)} + \frac{1}{tan(\beta)}\right)$$
$$d = d\frac{tan(\alpha)tan(\beta)}{tan(\alpha)+tan(\beta)}$$

Calcule folosite pentru detecția adâncimii folosind două camere

1



2



Schemă cu calcule (1) și etape de aplicare (2) pentru detecția adâncimii folosind două camere

Aliniere pe baza unui AprilTag:

În Autonomie am observat că punctarea pixelului galben doar cu ajutorul odometriei și algoritmului de navigație RoadRunner nu este atât de consistentă pe cât ne-am dori. Pentru a rezolva această problemă, am decis că cea mai bună soluție ar fi să ne **aliniem** pe unul dintre cele 3 AprilTag-uri de pe Backdrop, corespunzătoare cazurilor randomizate la începutul meciului. Această abordare ne aduce o creștere semnificativă în rata de plasare a pixelului galben în zona indicată de Team Prop la începutul meciului. Astfel am dezvoltat mai multe versiuni:

v1.1 Această versiune a fost făcută înainte de primul League Meet și constă în folosirea navigației până aproape de AprilTag-ul dorit și, ulterior, citirea poziției la care se afla camera față de acesta.

Probleme:	Soluții:
• Există șansa ca odometria să aibă o eroare atât de mare, încât robotul să nu ajungă suficient de aproape de AprilTag-ul corect; • Dacă camera nu vede niciun AprilTag, programul se blochează și dă eroare.	• Căutăm orice AprilTag și în funcție de acesta ne dăm seama în ce direcție trebuie să căutăm în continuare; • În loc să încercăm să citim distanța direct de la un AprilTag, mai întâi verificăm dacă vedem vreunul. În caz contrar, facem strafe dreapta pentru roșu sau stânga pentru albastru până găsim unul.

v1.2 Această versiune a fost făcută pentru primul meet și constă într-o abordare similară cu varianta anterio-

ară, diferența făcând-o faptul că robotul, indiferent de poziția în care ajungea, căuta mai întâi un AprilTag, iar după aceea făcea strafe stânga sau dreapta trecând prin ele până când îl găsea pe cel dorit.

Probleme:	Soluții:
• De multe ori când robotul făcea strafe, acesta nu găsea distanța corectă la care trebuia să fie față de AprilTag și rămânea blocat într-un loop infinit, făcând continuu stânga-dreapta; • Dacă bateria avea un voltaj de sub 12.5V, robotul aproape nu se mișca; • Aceasta aliniere era făcută doar stânga-dreapta, eroarea de la navigație fiind, de obicei, și față -spate; • Putem vedea maxim 2 AprilTag-uri.	• Reducem viteza de strafe a șasiului, astfel încât să facă mișcări mai încete, dar mai precise; • Începem Autonomia cu o baterie încărcată în fiecare meci; • Indiferent de eroarea față-spate, după ce suntem aliniați stanga-dreapta, robotul merge o distanța prestabilită în față; • Folosim o cameră cu un FOV (Field of view) mai mare.

v2 Aceasta versiune a fost făcută pentru al doilea League Meet și constă în poziționarea robotului astfel încât camera poate vedea cât mai multe AprilTag-uri. Algoritmul le parcurge și se deplasează către cel care îl interesează, folosind localizarea din odometrie ca reper.

Probleme:	Soluții:
• Robotul ajungea strâmb și există o șansă ca pixeli să cadă în slotul greșit.	• Realizăm o corecție pe unghi, astfel ajungând paralel cu Backdrop-ul.

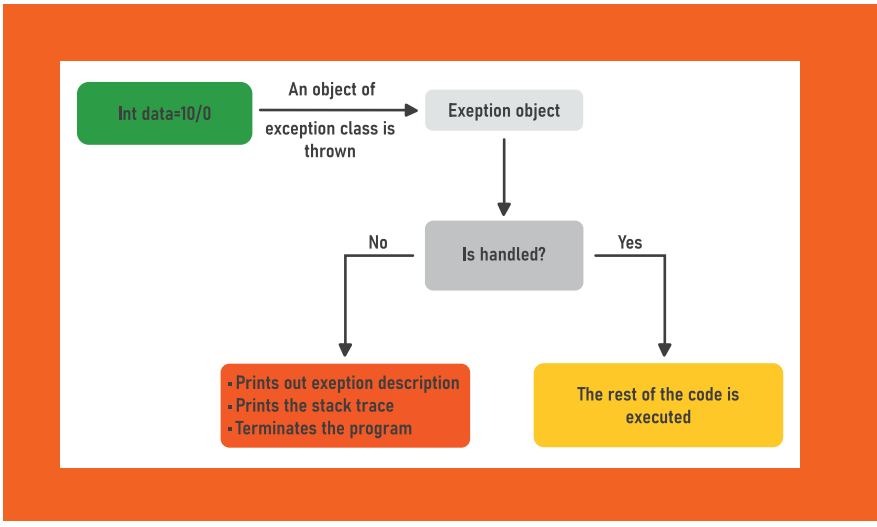
Obs. Folosim camera Logitech C920, deoarece are un FOV de 78°, spre deosebire de camera folosită anterior (C270) care avea un FOV de 55°.

FailSafe Camera:

În urma primului League Meet am observat că există posibilitatea unei erori a camerei în timpul competiției. Astfel, am decis să folosim blocul din Java, „ try-catch" pentru **tratarea excepțiilor**, permițându-ne să încercăm o secvență de cod, iar în cazul unei erori să executăm altă secvență sau să trecem mai departe în cod, fără ca acesta să returneze o excepție și să se deconecteze robotul.

Implementare:

Am folosit try-catch pentru eroarea **„index out of bounds"**. Aceasta apare în cazul în care încercăm să citim o poziție mai mare decât dimensiunea listei, astfel, dacă programul returnează o excepție vom folosi un set de coordonate prestabilite pentru fiecare poziție în funcție de caz. Acestea sunt foarte imprecise, dar ne permit **o a doua** șansă și previn codul din a se opri, astfel asigurând continuitatea Autonomiei fără probleme.



Schematizarea etapelor FailSafe-ului

Senzori:

Low Pass Filter:

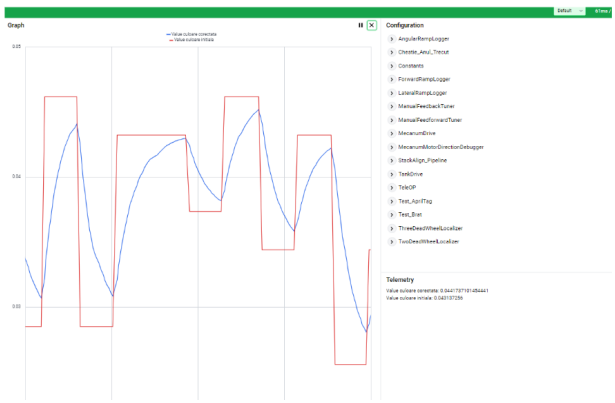
Măsurătorile senzorilor nu sunt mereu precise și, de multe ori, **diferența** dintre citiri este destul de evidentă. Aceasta poate duce la discrepanțe foarte mari între două citiri. Astfel, pentru a ne asigura că nu avem **inconsistențe** semnificative, am implementat un Low Pass Filter (LPF). Acesta are următoarea formă:

- $current\_estimate$ = valoarea filtrata a citirii (valoarea reală);
- $previous\_estimate$ = valoarea trecută filtrată;
- $current\_reading$ = valoarea citită de senzor;
- P = reprezintă cât de mult valorează citirea curentă la valoarea reală.

Obs. 0 < P < 1 deoarece P este un procent, P este o constantă și acest filtru se reglează în funcție de valoarea dată lui P.

În acest sezon folosim un LPF pentru a îmbunătăți acuratețea unui senzor de distanță. Exemplu de implementare:

Avantaje:	Dezavantaje :
• Valorile filtrate sunt foarte apropiate de distanța reală • Valorile variază mult mai puțin	• Filtrarea funcționează pe baza mai multor citiri, așadar se pierde puțin timp până la terminarea citilor.



Grafic pentru Value-ul citit de un senzor de culoare pentru un pixel alb

```
2 usages
DistanceSensor distance_sensor;
3 usages
Telemetry telemetry = FtcDashboard.getInstance().getTelemetry();
2 usages
double previous_estimate = 0, current_estimate = 0;
2 usages
double P = 0.8;

Alex5597 *
@Override
public void runOpMode() throws InterruptedException {
    distance_sensor = hardwareMap.get(DistanceSensor.class, deviceName: "DS");

    waitForStart();

    while (opModeIsActive()) {
        double current_reading = distance_sensor.getDistance(DistanceUnit.CM);
        current_estimate = (P * previous_estimate) + (1 - P) * current_reading;
        telemetry.addData( caption: "Distanta in cm", current_reading);
        telemetry.addData( caption: "Distanta in cm corectata", current_estimate)
        telemetry.update();
        previous_estimate = current_estimate;
    }
}
```

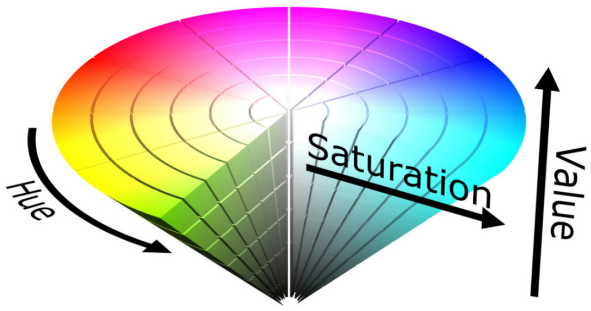
Cod folosit pentru calibrarea constantei P

Senzori de culoare:

Pentru a îmbunătăți timpul de colectare a pixelilor, am introdus **2 senzori de culoare**, astfel **automatizând** procesul.

Implementare:

În perioada de TeleOp, după ce intake-ul a fost activat și a tras pixeli, senzorii din acesta **detectează** momentul în care ei ajung într-o poziție favorabilă pentru a fi colectați. Simultan, atât gheara, cât și ambele palete se închid, urmând ca senzorii să se dezactiveze, **fără a afecta *loop-time*-ul**, intake-ul pornind în sens invers pentru a scoate orice **pixel suplimentar** și evitând penalizarile.



Navigarea:

În urma implementării roților de odometrie, am decis ca o metodă de navigare pe care o putem adapta pentru perioada de Autonomie este crearea unui **algoritm propriu**. Pentru acesta ar trebui să știm în fiecare moment **unde se află** robotul pe teren, prin intermediul unui set de coordonate dintr-un sistem prestabilit, dar care sa fie și o metodă eficientă de a **crea traiectorii** și a le urmării cu eroare cât mai mică.

Implementare:

Algoritmul de navigare are 3 etape: **Odometria robotului pe teren** (prin două etape, și anume Odometria cu ajutorul encodere-lor de la roți, respectiv Odometria cu ajutorul roților moarte (dead wheels)); **Crearea unei traiectorii**; **Urmărirea traiectoriei create anterior** (prin GoToPoint, respectiv GuidingVectorField).

Odometria robotului pe teren:

Odometria este o formă de localizare a robotului pe teren cu ajutorul **encoderelor** sau a altor **senzori externi**, din care se poate deriva o poziție relativă a acestuia față de un punct de start. Odometria este folositoare mai ales în timpul perioadei de Autonomie, deoarece cunoașterea poziției relative a robotului este necesară pentru navigarea optimă în teren.

O poziție are 3 componente:
- Poziția pe Ox a robotului (x);
- Poziția pe Oy a robotului (y);
- Orientarea robotului (θ).

Obs. Poziția se notează sub forma (x,y,θ) și începe de la (0,0,0).

▪ Odometria cu ajutorul encodere-lor de la motoare:

Obs.1 Fiecare valoare citită de la Encoder este transformată din tick-uri în cm cu ajutorul formulei: $\pi \times WheelDiameter \times ticks \div MotorTicksRev$, unde $WheelDiameter$ reprezintă diametrul roții Mecanum, $ticks$ reprezintă valoarea citită de la encoder, iar $MotorTicksRev$ este o valoare specifică fiecărui tip de motor și reprezintă numărul de tick-uri dintr-o rotație completă.

$$\Delta FL = FL\_current\_pos - FL\_prev\_pos$$
$$\Delta FR = FR\_current\_pos - FR\_prev\_pos$$
$$\Delta BL = BL\_current\_pos - BL\_prev\_pos$$
$$\Delta FL = FL\_current\_pos - FL\_prev\_pos$$

$$\Delta Left = \frac{\Delta FL + \Delta BL}{2}$$
$$\Delta Right = \frac{\Delta FR + \Delta BR}{2}$$
$$\Delta diag_1 = \frac{\Delta FL + \Delta BR}{2}$$
$$\Delta diag_2 = \frac{\Delta FR + \Delta BL}{2}$$

$$\Delta x = (\Delta Left + \Delta Right) / 2$$
$$\Delta y = (\Delta diag_1 - \Delta diag_2) / 2$$

(1)

Obs.2 Δx și Δy sunt coordonatele locale ale robotului față de poziția de start, dar pe noi ne interesează un set de coordonate globale, așadar ne vom folosi de unghiul citit din imu pentru a le calcula pe cele globale.

$$POS_X = cos(angle) \times \Delta x - sin(angle) \times \Delta y$$
$$POS_Y = cos(angle) \times \Delta y - sin(angle) \times \Delta x$$

(2)

Formule generale utilizate pentru Odometria cu encodere (1) și formule pentru cea de-a doua observație (2):

Avantaje:	Dezavantaje:
▪ Este mai accesibil din punct de vedere al prețului și al timpului de livrare, fiind necesare doar motoarele;	▪ Exista o șansa mare ca roțile mecanum să alunece pe teren, rezultând în pozițiile encodere-lor să rămână pe loc, în timp ce robotul încă merge;

Avantaje:	Dezavantaje:
• Este necesar doar șasiul fără alte modificări pentru a face asta.	• Dacă robotul se lovește de orice (inclusiv alt robot) localizarea se pierde complet.

▪ Odometria cu ajutorul roților moarte (Dead Wheels):

Pentru a calcula diferența de poziție între două citiri folosim următoarele formule:

$$\Delta left\_encoder = current\_left\_encode - previous\_left\_encoder$$
$$\Delta right\_encoder = current\_right\_encoder - previous\_right\_encoder$$
$$\Delta middle\_encoder = current\_midle\_encoder - previous\_middle\_encoder$$

Unde valorile care au "current" reprezintă poziția citită de encoder în momentul de față, iar cele care au "previous" reprezintă poziția anterior citită.

Obs.1

Deoarece vrem să lucrăm în centimetri, iar encodere-le citesc valori în tick-uri, vom folosi aceeași formulă de transformare ca mai sus.

E_1 = valoare citită de encoderul din dreapta;
E_2 = valoare citită de encoderul din stânga;
E_3 = valoare citită de encoderul din mijloc;

Notăm distanța dintre E1 si E2 cu Lateral_distance și distanța dintre E3 și centrul de rotație al robotului și Forward_distance.

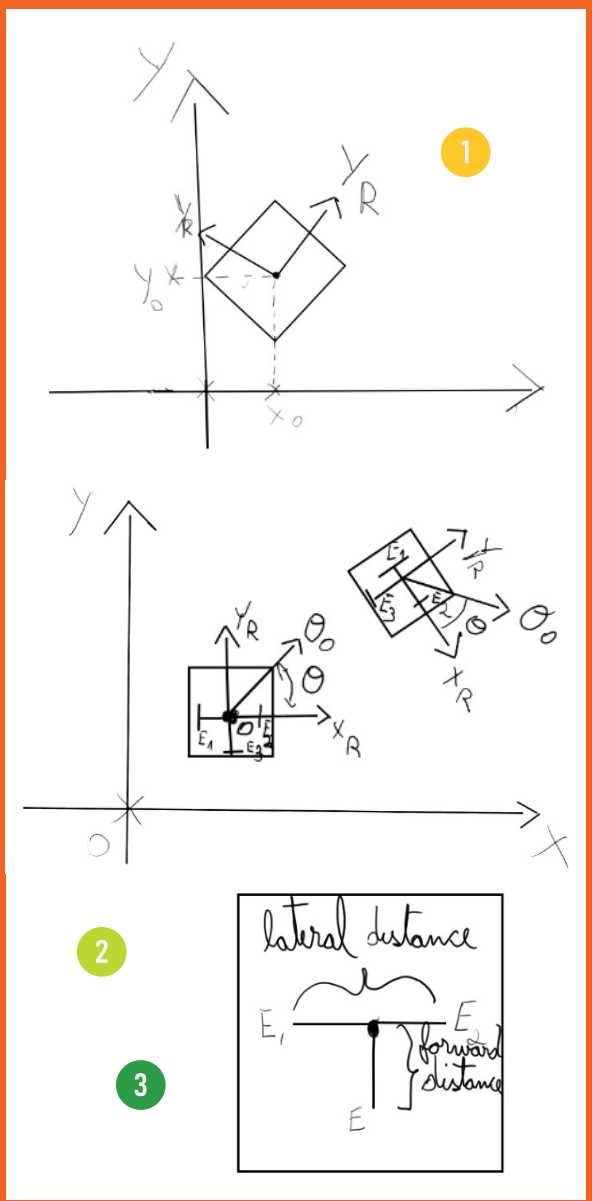
Pentru a calcula schimbarea unghiului dintre 2 citiri folosim:

$$\Delta Q = \frac{\Delta RightEncoder - \Delta LeftEncoder}{ForwardDistance}$$

Pentru a calcula schimbările pe O'xr și O'yr folosim:

$$\Delta X = \frac{\Delta Right\ Encoder + \Delta Left\ Encoder}{2}$$
$$\Delta Y = \Delta Right\ Encoder - \Delta\theta \times Forward\ Distance$$

Pentru *ΔX* este evident că este suficient să facem media aritmetică celor 2 encodere paralele. Pentru *ΔY* nu este suficient *ΔMiddle_encoder*, deoarece în momentul în care robotul se rotește, valoarea s-ar schimba și acest lucru ar fi un impedment pentru calculul distanței parcurse mergând lateral (strafing). De aceea, trebuie să scădem „schimbarea" unghiului ($\Delta\theta$) înmulțit cu *Forward_distance*.



Schițe care evidențiază formulele utilizate pentru odometria cu ajutorul Dead Wheels (1), (2), respectiv (3)

Obs.2 Pentru a calcula poziția robotului pe teren nu este suficient să adunăm valorile obținute pentru fiecare citire. Dacă robotul este la 90 de grade față de start și merge în față, în loc să crească ΔY, va crește tot ΔX (am calculat coordonatele robotului față de o poziție de start a lui, nu față de teren).

Pentru a transforma din coordonate față de start în coordonate față de teren vom folosi următoarele formule:

$$Poziție_{OX} = Poziție_{OX} + \Delta X \times cos(angle) - \Delta Y \times sin(angle)$$
$$Poziție_{OY} = Poziție_{OY} + \Delta X \times sin(angle) + \Delta Y \times cos(angle)$$

unde: $Poziție_{OX}$ și $Poziție_{OY}$ reprezintă poziția robotului pe teren de la început, iar $angle$ este unghiul la care se află robotul în momentul de față și se calculează: $angle = angle + \Delta\theta$

Obs.3 Adunăm de fiecare dată vechea valoare, deoarece ne interesează coordonatele de la început.

Obs.4 Diferența în timp dintre poziția actuală și poziția trecută trebuie să fie cât mai mică pentru a îmbunătăți aproximările, așadar, poziția trebuie actualizată în fiecare ciclu al unui loop. De aceea vom folosi diferența (delta) între valoarea citită actual și valoarea de la ultima citire.

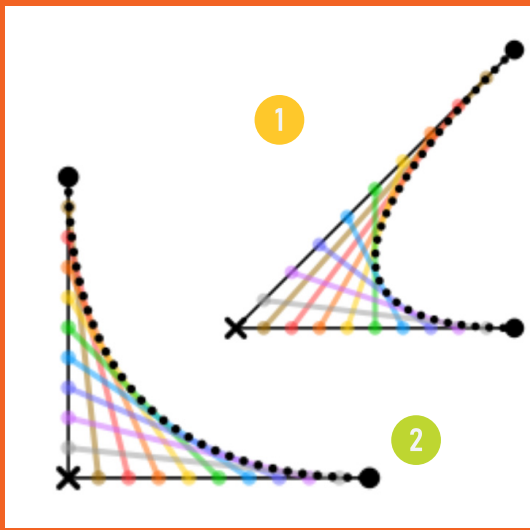
Pentru calcularea poziției vom folosi 2 roți paralele și una perpendiculară pe acestea. Cu ajutorul roților paralele vom calcula schimbarea unghiului și schimbarea pe Oy a poziției (cât a parcurs în față între două citiri consecutive), iar roata perpendiculară și schimbarea unghiului pentru distanța parcursa pe Ox.

Obs.5 Dacă adunăm schimbările de poziție la o variabilă, fără să luăm în considerare unghiul la care suntem, vom obține coordonate pe baza robotului, care nu ne ajuta pentru algoritmul de navigare, fiind nevoiți să aplicăm funcții trigonometrice pentru calculul unui set de coordonate pe baza terenului.

▪ Crearea unei traiectorii:

Pentru crearea traiectoriilor am ales să folosim **Bézier curves** din mai multe motive:

- **Simplitate**: Bézier curve-urile sunt relativ simple de înțeles și de utilizat. Acestea sunt definite de doar câteva puncte de control, ceea ce le face accesibile în orice situație;
- **Control precis**: B zier curve-urile oferă un control precis asupra traiectoriei. Poziția și tangențele punctelor de control se pot ajusta ușor pentru a obține exact traiectoria dorită;
- **Unire ușoară:** Bézier curve-urile sunt ușor de unit și formează Bézier splines care sunt traiectorii complexe, mai lungi și ușor de modelat.



Exemple vizuale generale pentru Bézier Curves (1), (2)

Tipuri de Bézier Curves:

O curbă Bézier este definită de un set de puncte de control P_0 până la P_n, unde n se numește ordinul curbei . Primul și ultimul punct de control sunt întotdeauna punctele finale ale curbei; cu toate acestea, punctele intermediare de control nu se află în general pe curbă. Pe baza ordinului curbei există mai multe tipuri de Bézier curves:

▪ Liniare:

Având în vedere punctele distincte P_0 și P_1, o curbă liniară Bézier este pur și simplu o linie între aceste două puncte. Ecuațiile pentru un punct P(X,Y) de pe curba liniară sunt de forma:

$$X = P_{x_0} + t(P_{x_1} - P_{x_0}), \text{ unde } 0 \le t \le 1;$$
$$Y = P_{y_0} + t(P_{x_1} - P_{y_0}), \text{ unde } 0 \le t \le 1;$$

unde P_0(x, y) și P_1 (x, y) reprezintă coordonatele x și y ale punctului de început respectiv de final.

▪ Pătratice

O curbă Bézier pătratică este calea trasată de funcția B(t), punctele date P_0, P_1 și P_2 și care poate fi interpretată ca o ecuație liniară a punctelor corespondente de pe curbele liniare Bézier de la P_0 la P_1 și, respectiv, de la P_1 la P_2. Ecuațiile pentru un punct P(x, y) de pe curba pătratică sunt de forma:

$$X = (1-t)^2 P_{x_0} + 2(1-t)tP_{x_1} + t^2 P_{x_2}, \text{ unde } 0 \le t \le 1;$$
$$Y = (1-t)^2 P_{y_0} + 2(1-t)tP_{y_1} + t^2 P_{y_2}, \text{ unde } 0 \le t \le 1;$$

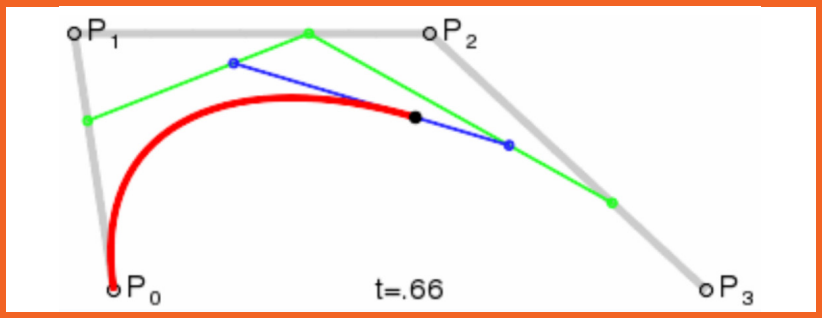
▪ Cubice

Patru puncte P_0,P_1, P_2 și P_3 în plan sau în spațiul de dimensiuni superioare definesc o curbă Bézier cubică. Curba începe de la P_0, mergând spre P_1, și ajunge la P_3, venind din direcția P_2. De obicei, nu va trece prin P_1 sau P_2; aceste puncte sunt acolo doar pentru a oferi informații direcționale. Distanța dintre P_1 și P_2 determină „cât de departe" și „cât de repede" curba se deplasează spre P_1 înainte de a se întoarce spre P_2. Ecuațiile pentru un punct P(x,y) de pe curba cubică sunt de forma:

$$X = (1-t)^3 P_{x_0} + 3(1-t)^2 tP_{x_1} + 3(1-t)\, t^2 P_{x_2} + t^3 P_{x_3}, \text{ unde } 0 \le t \le 1;$$
$$Y = (1-t)^3 P_{y_0} + 3(1-t)^2 tP_{y_1} + 3(1-t)\, t^2 P_{y_2} + t^3 P_{y_3}, \text{ unde } 0 \le t \le 1;$$

Obs. Apoi putem calcula derivatele funcțiilor (ajungem la x' și y'), iar, aplicând atan(y',x') obținem heading-ul pe care robotul trebuie să îl aibă în acest punct.

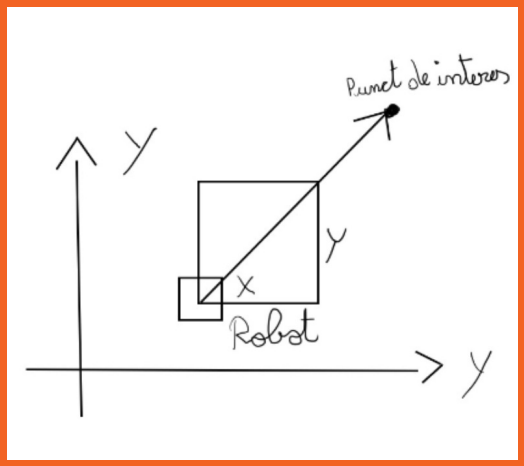
Exemplu de grafic pentru Bézier Curves de tipul cubic



- **Urmărirea traiectoriei:**

GoToPoint:

Pentru acest algoritm am folosit 3 PIDF-uri, câte unul pentru fiecare coordonată: unul pentru Ox, unul pentru Oy și unul pentru heading. Pentru a calcula FeedForward-ul am folosit următoarele formule:



$$relativeAngleToTarget = target_{angle} - current_{angle}$$

$$distanceToTarget = \sqrt{(target_x - current_x)^2 + (target_y - current_y)^2}$$

$$rel_X = cos(relativeAngleToTarget) \times distanceToTarget$$

$$rel_Y = sin(relativeAngleToTarget) \times distanceToTarget$$

$$nominator = abs(rel_X) + abs(rel_Y)$$

Schiță pentru algoritmul de GoToPoint

Obs.1

În caz că rel_X și rel_Y sunt 0, atunci $nominator$ = 1, pentru a nu împărți la 0. Formule pentru algoritmul de GoToPoint

$$F_X = rel_X \times movementSpeed / nominator$$
$$F_Y = rel_Y \times movementSpeed / nominator$$

unde: $relativeAngleToTarget$ este eroarea pentru unghi;
$distanceToTarget$ este distanța până la target din poziția curentă;
rel_X este distanța pe coordonata x până la target din poziția curentă;
rel_Y este distanța pe coordonata y până la target din poziția curentă;
$movementSpeed$ este viteza dorită la care să meargă robotul;
F_X este viteza pe coordonata x a robotului;
F_Y este viteza pe coordonată y a robotului;

Obs.2

Adunăm F_X și F_Y (vitezele pe coordonatele celor două axe din sistemul xOy) la viteza de corecție, apoi, cu ajutorul cinematicii inverse a roților mecanum (Mecanum Inverse Kinematics) pentru a distribui puterea la motoare.

Guiding Vector Field:

Ideea din spate este să definească direcțiile și intensitățile mișcării pe baza unei traiectorii predefinite, permițând obiectelor să se deplaseze în mod controlat și să răspundă în mod inteligent la condiții specifice din mediu.

Implementare:

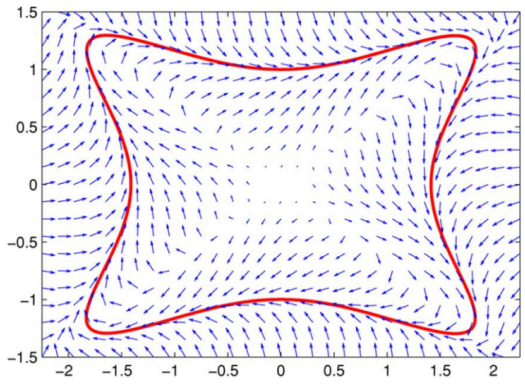
Primul pas este să găsim cel mai apropiat punct de pe traiectorie de poziția curentă a robotului;

- *calculateMinimizationFunction* reprezintă funcția care se ocupă de calculul distanței dintre punctul actual (point)

și un punct de pe traiectorie dat;
- *findClosestPoint* reprezintă funcția care găsește punctul de pe traiectorie care este cel mai aproape de robot.

Apoi, după ce știm care e cel mai apropiat punct calculam un factor de corecție cu un PID și un factor de mișcare (un fel de FeedForward) cu ajutorul distanței calculate anterior.

1



2

```
1 usage
public double findClosestPoint(CubicBezierCurve curve, Vector2D point)
{
    double minT = -1;
    double minDist = Double.POSITIVE_INFINITY;
    int numOfPoints = 150;
    for (int i = 0; i ⩽ numOfPoints; i++) {
        double t = (double) (i / (double) numOfPoints);
        double dist = calculateMinimizationFunction(curve, t, point);
        if (dist < minDist) {
            minDist = dist;
            minT = t;
        }
    }
    if (minT == -1)
        minT = 1;
    return minT;
}

1 usage
private double calculateMinimizationFunction(CubicBezierCurve curve, double t, Vector2D point)
{
    return curve.calculate(t).subtract(point).getMagnitude();
}
```

Grafic pentru Guiding Vector Field (1) și metodă de implementare în codul propriu-zis (2)

$$correctionFactor = Math.min(0.6, nextPoint.getMagnitude() \times \times kP\_GVFHeading);$$
$$movementDirection = hlerp(curve.heading(closestT), nextPoint.getHeading(), correctionFactor)$$

Obs.1

Am ales să folosim constant un factor de corecție de minim 0.6 pentru a ne corecta pe traiectorie cât mai precis.

Obs.2

Hlerp - heading lerp (interpolare a direcției): Interpolează între a și b, luând cea mai scurtă cale în intervalul [-pi, pi], presupunând că intervalul de intrare este continuu în acest interval. Să luăm a = -0.9pi, b = 0.9pi. O interpolare liniară tradițională ar roti în sens invers acelor de ceasornic, trecând prin 0 la t = 0.5. Hlerp va roti în sensul acelor de ceasornic, trecând prin +/- pi la t = 0.5. Hlerp este folosit pentru a evita un caz limită în care direcția mișcării trece prin +/- pi pe măsură ce trece de la corectarea căii la ghidarea pe cale.

La pasul al treilea, calculăm vectorul de mișcare cu formulele:

$$movmentVector_X = cos(movementDirection)$$
$$movmentVector_Y = sin(movementDirection)$$

Pe baza celor doi factori găsiți anterior, calculăm viteza pe care trebuie să o aibă robotul în funcție de distanța pe care o mai are de parcurs până la final.

Obs.

Funcția lerp implementează o operație de interpolare liniară între două valori, a și b, utilizând un parametru t care se află în intervalul [0, 1]. Interpolarea liniară (lerping) produce o valoare intermediară între a și b în funcție de parametrul t (t funcționează ca un procent).

Înmulțim cele două variabile de tipul *movmentVector* cu *speed* și, astfel, obținem viteza robotului.

```
double speed = 0.8;
double distanceToEnd = robotToEnd.getMagnitude();
if (distanceToEnd < 200) {
    if (distanceToEnd ⩽ 3)
        speed = 0;
    if (distanceToEnd ⩽ 20)
        speed = lerp( a: 0.2,  b: 0.4,  t: robotToEnd.getMagnitude() / 20);
    else
        speed = lerp( a: 0.6, speed,  t: robotToEnd.getMagnitude() / 200);
}
```

Metodă de implementare în codul propriu-zis

Obs.

Exact ca la tipul anterior de urmărire a traiectoriei, vitezele din movmentVector trebuie transformate cu ajutorul cinematicii inverse a roților Mecanum (Mecanum Inverse Kinematics) pentru a distribui puterea la motoare.

Photon:

Photon este o bibliotecă prin intermediul căreia putem îmbunătăți loop time-ul unui program (loops/second) din variate aspecte. Un exemplu concludent este reprezentat de senzorii de tip I2C care au cele mai lente citiri. Fiecare dintre acestea poate dura pană la 5-7 ms. Utilizând Photon, timpul pentru fiecare citire este redus până la 4 ms.

Două exemple de senzori de tip I2C sunt cei de culoare și cei de distanță, pe care îi avem montați în intake. Aceștia necesită citiri frecvente pentru a ști când pixelii se află în poziția de transfer. Inițial, cei doi senzori scădeau frecvența loop-ului până la 30Hz. După ce am început să utilizăm Photon, acesta a crescut-o până la 200Hz.

În cazul în care conexiunea dintre Control Hub și Expansion Hub este făcută de un cablu de tipul RS485, Photon nu poate îmbunătății senzorii conectați în Expansion Hub. Prin intermediul unui cablu de tip USB, transferul datelor este cu adevărat bidirecțional, astfel Photon poate reduce timpul de citire a senzorilor din Expansion Hub.

Photon ne oferă posibilitatea să creăm anumite obiecte ce funcționează ca niște tabele. Acestea au locurile rezervate pentru anumite date existente atât în prezent, cât și în viitor, care urmează să fie citite. Odată ce locul devine completat, se returnează valoarea din spațiul respectiv. În timp ce programul continuă, numărul de spații crește constant, astfel repetându-se procesul.