Team LEBOB's

Robot Design

2025 Unearthed

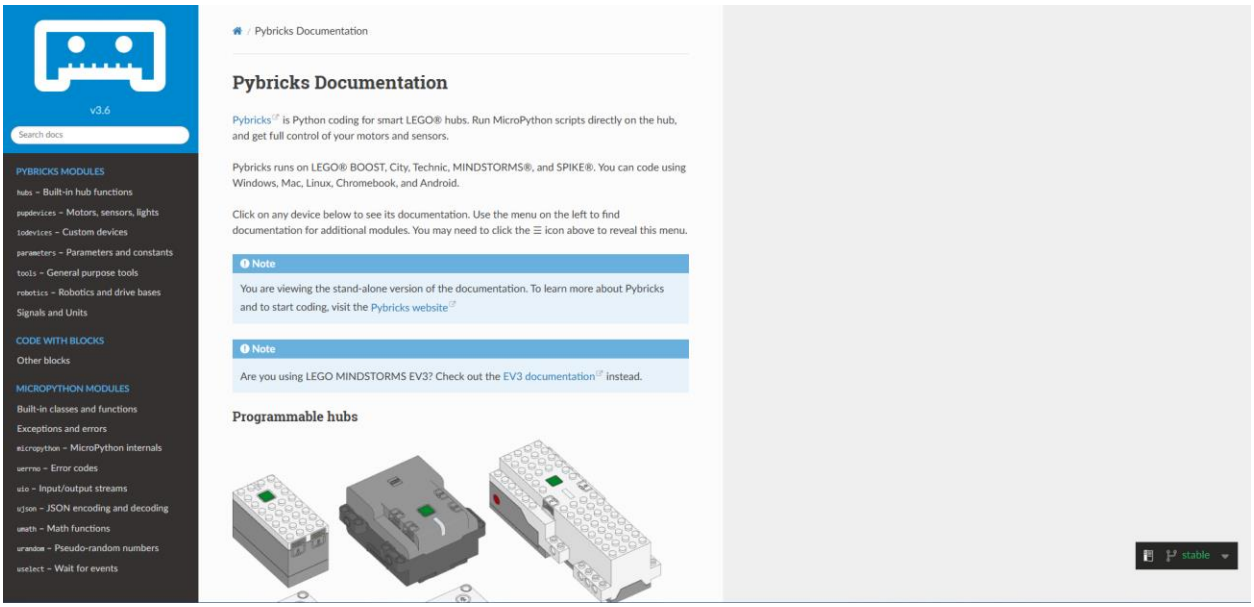
Members:
Kingsley Wong
Sean Chan
Oliver Liu
Leven Shi
Subesh Sukumaran
Yuxin (Chris) Wang
Aaron Zhang
Andre Nijman

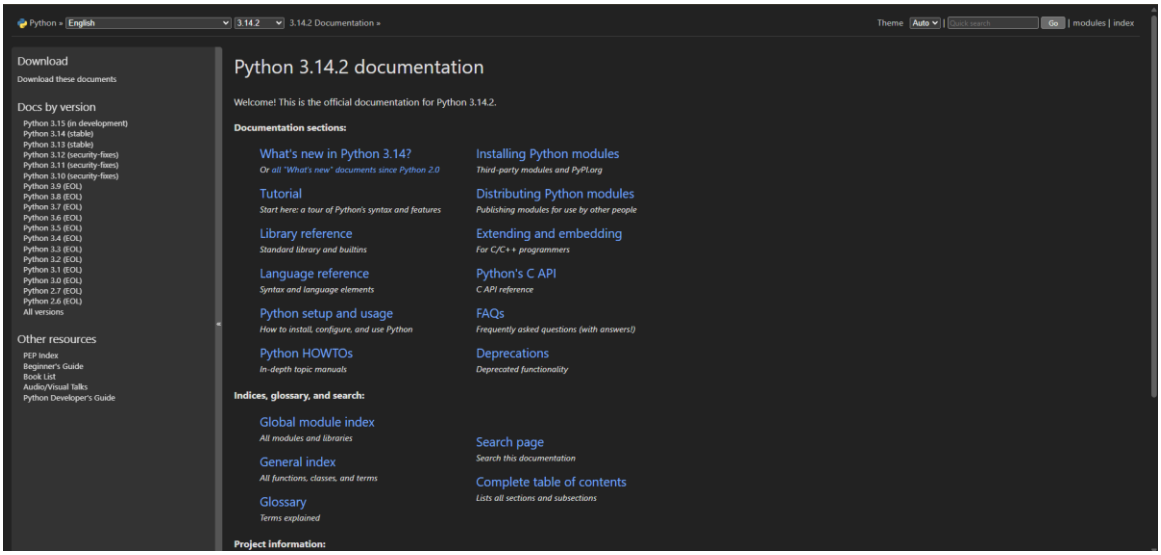
IDENTIFY

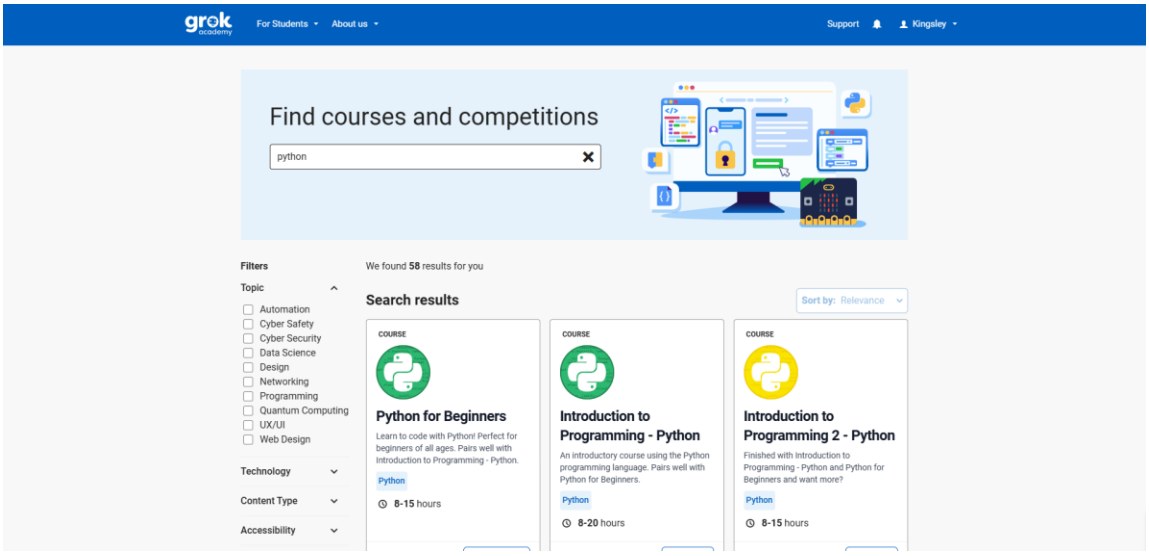
Clear use of building or coding resources to support their mission strategy:

Coding resources that we used to help our team code and learn how to code and do new things include PyBricks documentation and python documentation as well as a website called grok that allows each team member to learn python from whatever skill level, beginner to advanced.

- **Pybricks documentation** allowed us to explore the potential of using pybricks. This helped us to understand what pybricks was and showed how pybricks had much more potential in controlling the robot including PID and allowed us to directly control the circuitry of the robot which helped us add more power and load into motors which allowed us to reach faster speeds of motors. Benefits of pybricks over bricks:
    - Runs directly on the hub itself rather than relying on a tablet or computer for more reliability and speed because it doesnt need to rely on external sources like computers
    - Provides advanced models like drivebase and gyro drive base which delivers extremely accurate navigation
    - It supports python and block code which helps us learn programming and software engineering at a young age
    - There are many team around the world that use pybricks for FLL and they have all been successful such as the Monongahela Cryptid Cooperative (Team 45775) and Noddin Robotmakers (Team 60215)
- **Python documentation** allowed us to understand, explore, and effectively use python by providing detailed explanations of its syntax, functions, libraries, and modules along with examples and guidelines. It serves as a reference and a learning resource, enabling us to look up how specific features work and troubleshoot issues. Whether you're a beginner learning the basics or an experienced programmer working with advanced tools, the documentation empowers you to write efficient reliable code with confidence
- **Grok Learning** helps people learn coding by providing interactive, curriculum-aligned courses, competitions, and resources that make programming accessible, engaging, and practical for all ages and skill levels.


v3.6
Search docs
PYBRICKS MODULES
hubs – Built-in hub functions
pupdevices – Motors, sensors, lights
iodevices – Custom devices
parameters – Parameters and constants
tools – General purpose tools
robotics – Robotics and drive bases
Signals and Units
CODE WITH BLOCKS
Other blocks
MICROPYTHON MODULES
Built-in classes and functions
Exceptions and errors
micropython – MicroPython internals
uerrno – Error codes
uio – Input/output streams
ujson – JSON encoding and decoding
umath – Math functions
urandom – Pseudo-random numbers
uselect – Wait for events
Pybricks Documentation
Pybricks Documentation
Pybricks is Python coding for smart LEGO® hubs. Run MicroPython scripts directly on the hub, and get full control of your motors and sensors.
Pybricks runs on LEGO® BOOST, City, Technic, MINDSTORMS®, and SPIKE®. You can code using Windows, Mac, Linux, Chromebook, and Android.
Click on any device below to see its documentation. Use the menu on the left to find documentation for additional modules. You may need to click the ☰ icon above to reveal this menu.
Note
You are viewing the stand-alone version of the documentation. To learn more about Pybricks and to start coding, visit the Pybricks website
Note
Are you using LEGO MINDSTORMS EV3? Check out the EV3 documentation instead.
Programmable hubs
stable


Python » English 3.14.2 3.14.2 Documentation »
Theme Auto Go | modules | index
Download
Download these documents
Docs by version
Python 3.15 (in development)
Python 3.14 (stable)
Python 3.13 (stable)
Python 3.12 (security-fixes)
Python 3.11 (security-fixes)
Python 3.10 (security-fixes)
Python 3.9 (EOL)
Python 3.8 (EOL)
Python 3.7 (EOL)
Python 3.6 (EOL)
Python 3.5 (EOL)
Python 3.4 (EOL)
Python 3.3 (EOL)
Python 3.2 (EOL)
Python 3.1 (EOL)
Python 3.0 (EOL)
Python 2.7 (EOL)
Python 2.6 (EOL)
All versions
Other resources
PEP Index
Beginner's Guide
Book List
Audio/Visual Talks
Python Developer's Guide
Python 3.14.2 documentation
Welcome! This is the official documentation for Python 3.14.2.
Documentation sections:
What's new in Python 3.14?
Or all "What's new" documents since Python 2.0
Tutorial
Start here: a tour of Python's syntax and features
Library reference
Standard library and builtins
Language reference
Syntax and language elements
Python setup and usage
How to install, configure, and use Python
Python HOWTOs
In-depth topic manuals
Installing Python modules
Third-party modules and PyPI.org
Distributing Python modules
Publishing modules for use by other people
Extending and embedding
For C/C++ programmers
Python's C API
C API reference
FAQs
Frequently asked questions (with answers!)
Deprecations
Deprecated functionality
Indices, glossary, and search:
Global module index
All modules and libraries
General index
All functions, classes, and terms
Glossary
Terms explained
Search page
Search this documentation
Complete table of contents
Lists all sections and subsections
Project information:


grok academy
For Students
About us
Support
Kingsley
Find courses and competitions
python
Filters
Topic
Automation
Cyber Safety
Cyber Security
Data Science
Design
Networking
Programming
Quantum Computing
UX/UI
Web Design
Technology
Content Type
Accessibility
We found 58 results for you
Search results
Sort by: Relevance
COURSE
Python for Beginners
Learn to code with Python! Perfect for beginners of all ages. Pairs well with Introduction to Programming - Python.
Python
8-15 hours
COURSE
Introduction to Programming - Python
An introductory course using the Python programming language. Pairs well with Python for Beginners.
Python
8-20 hours
COURSE
Introduction to Programming 2 - Python
Finished with Introduction to Programming - Python and Python for Beginners and want more?
Python
8-15 hours

Engineering resources that we used to help us build our robot include watching YouTube videos on the missions so we could construct the routes for our mission plan. This helped us a lot because it helped us decide on which plan we chose as we had many choices to decide from and we had to make these runs as fast and as efficient as possible. We wanted this to be fast and efficient and this was useful as it allowed us to start coding without having to worry about any changes in routes after we had chosen the one we wanted.

Clear evidence of mission strategy:

Mission No. and Name		*Distance*	*Programming*	*Mechanical*
1 Surface Brushing	30 (10+10+10)	1	1	3 (for left and right) 2 (for removing the stick)
2 Map Reveal	30 (10+10+10)	3	4 (for all three) 3 (for 2 only) 2 (for only the rotate one)	2
3 Mineshaft Explorer	40 (30+10(technically))	5	4	2
4 Careful Recovery	40 (30+10)	5	5 (with standing) 3 (without)	5 (with standing) 3 (without)
5 Who Lived Here?	30	3	2	1
6 Forge	30 (10+10+10)	3	3	1
7 Heavy Lifting	30	5	3	3
8 Silo	30 (10+10+10)	1	4	2
9 What's on Sale? (First only. Second is a maybe)	30 (20+10)	1.5	2	3
10 Tip the Scales	30 (20+10)	2	4	2.5
11 Angler Artefacts	30 (20+10)	2	4 (time wise)	3
12 Salvage Operation	30 (20+10)	1	3	2
13 Statue Rebuild	30	3	4	2
14 Forum (Might do some only)	35 (5+5+5+5+5+5+5)	10	5	5
15 Site Marking (Won't unless on the way)	30 (10+10+10)	7	5	3

MAX RUN LINK: https://www.youtube.com/watch?v=c62qryVwEQw

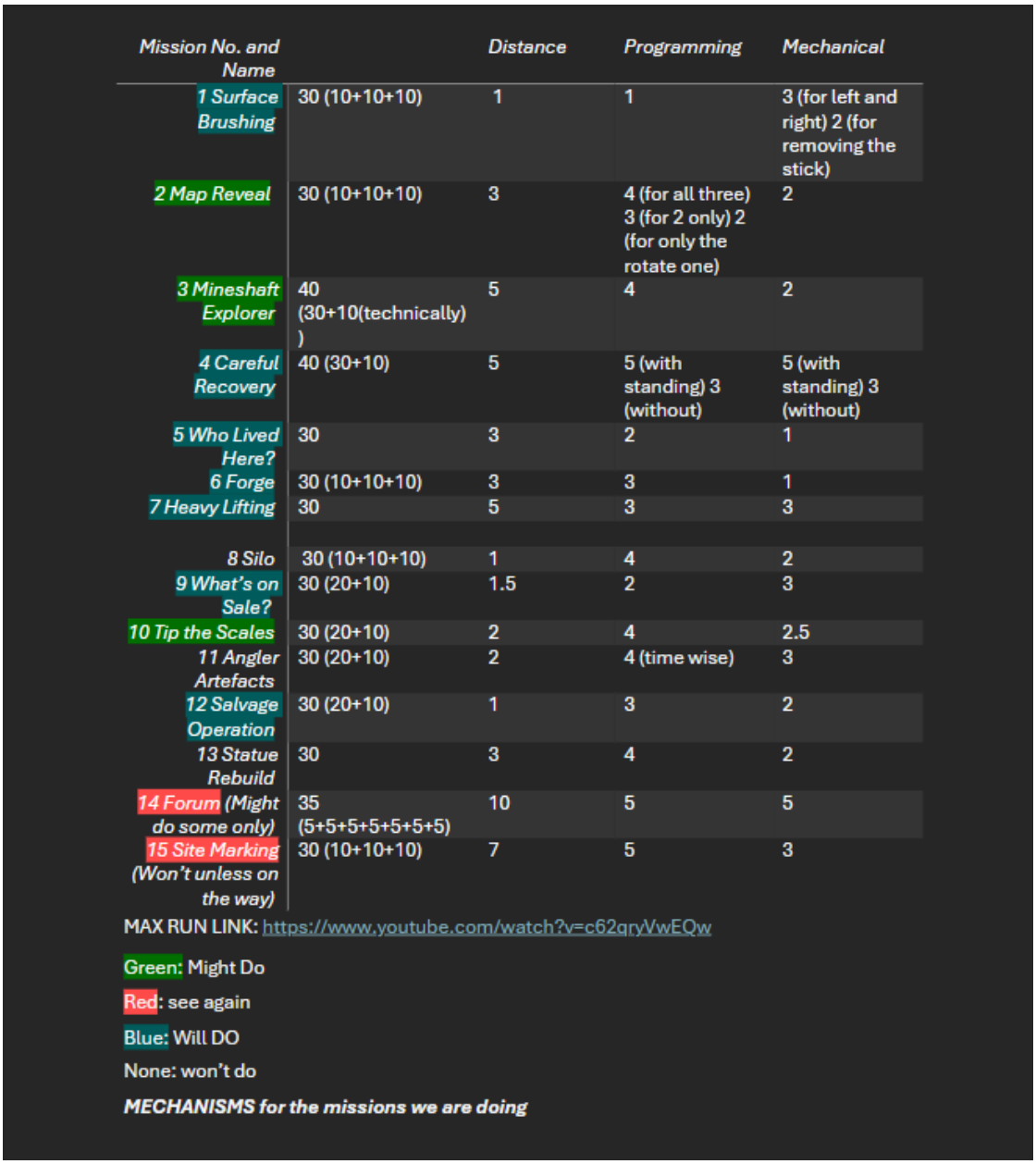
Green: Might Do

Red: see again

Blue: Will DO

None: won't do

(regionals)



Mission No. and Name		Distance	Programming	Mechanical
1 Surface Brushing	30 (10+10+10)	1	1	3 (for left and right) 2 (for removing the stick)
2 Map Reveal	30 (10+10+10)	3	4 (for all three) 3 (for 2 only) 2 (for only the rotate one)	2
3 Mineshaft Explorer	40 (30+10(technically))	5	4	2
4 Careful Recovery	40 (30+10)	5	5 (with standing) 3 (without)	5 (with standing) 3 (without)
5 Who Lived Here?	30	3	2	1
6 Forge	30 (10+10+10)	3	3	1
7 Heavy Lifting	30	5	3	3
8 Silo	30 (10+10+10)	1	4	2
9 What's on Sale?	30 (20+10)	1.5	2	3
10 Tip the Scales	30 (20+10)	2	4	2.5
11 Angler Artefacts	30 (20+10)	2	4 (time wise)	3
12 Salvage Operation	30 (20+10)	1	3	2
13 Statue Rebuild	30	3	4	2
14 Forum (Might do some only)	35 (5+5+5+5+5+5+5)	10	5	5
15 Site Marking (Won't unless on the way)	30 (10+10+10)	7	5	3

MAX RUN LINK: https://www.youtube.com/watch?v=c62qryVwEQw

Green: Might Do

Red: see again

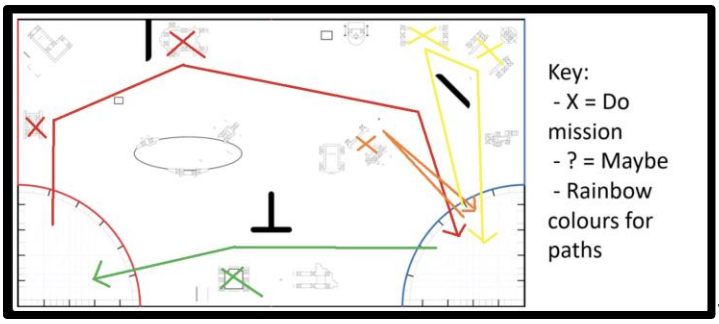
Blue: Will DO

None: won't do

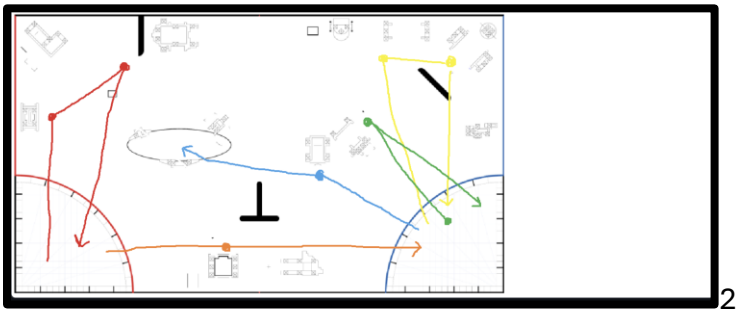
MECHANISMS for the missions we are doing

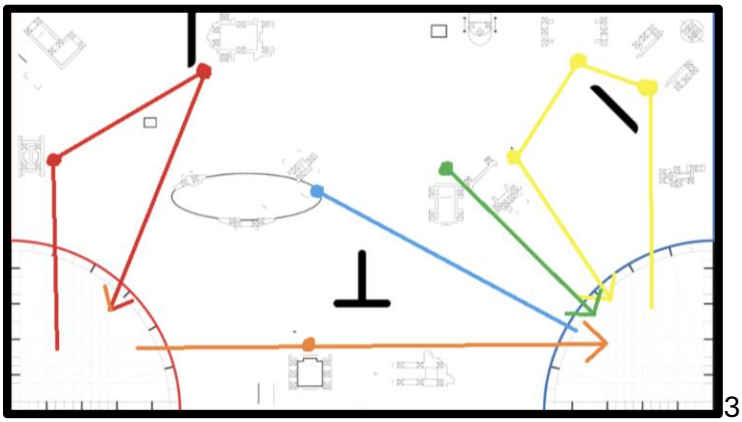
(Nationals)

This was our mission map plan where we have listed all the missions and then as a team made this brainstorm page showing whether we think we will do it (blue), might do it if we have time (green) or will not do because it would take too long and it wouldn't fit our robot. (none)



1st version (regionals)


2nd version (regionals)


3rd version (regionals)

4th version (nationals)

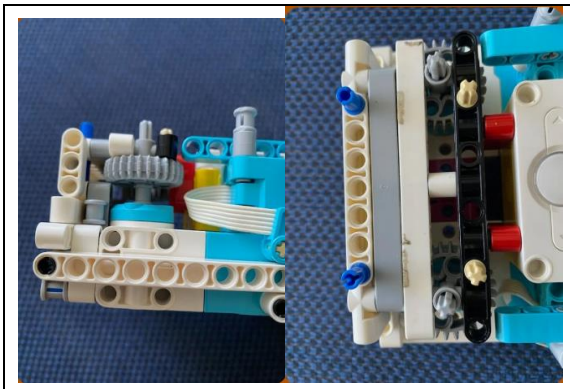
5th version (nationals)

DESIGN

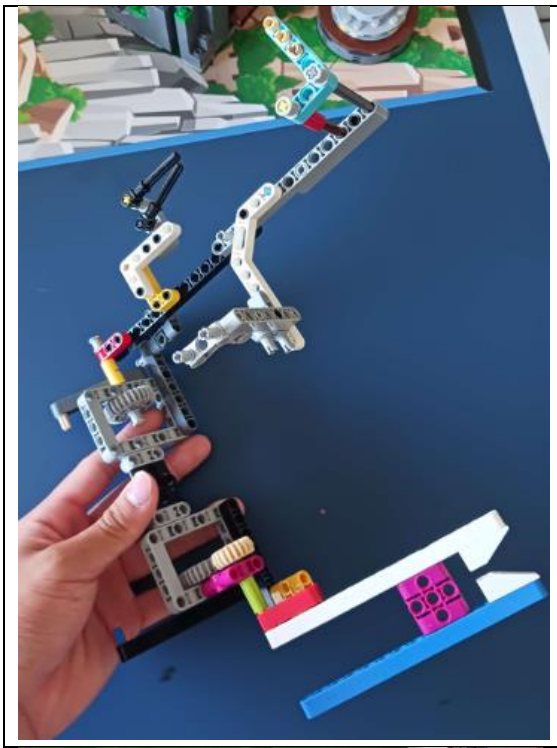
Clear evidence of building and coding skills in all team members:

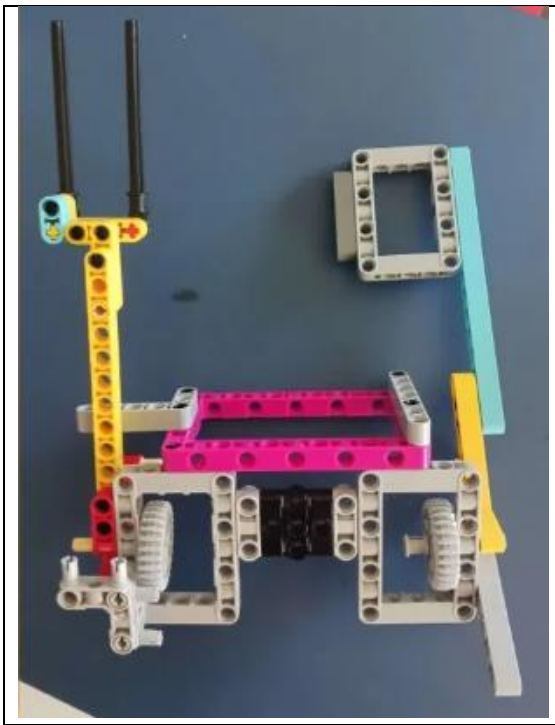
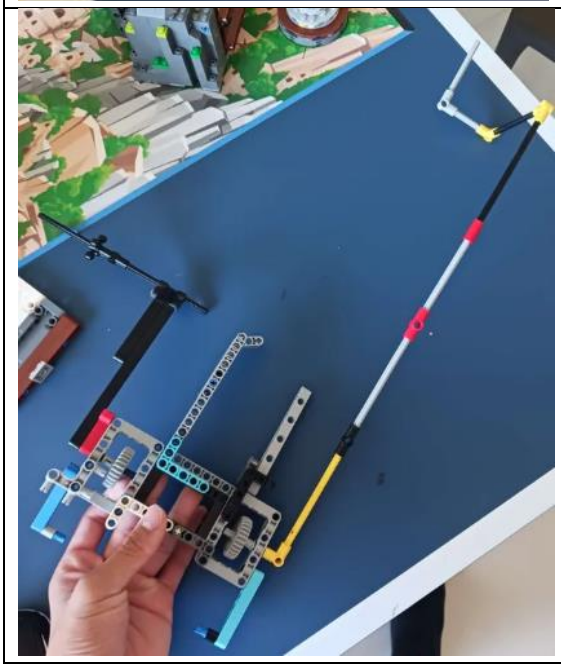
Each person created around one attachment used within the runs, despite not all of them being used, they were considered and tested before deciding to not use it. We all collaborated with each other to construct the attachments so that most of them could both be attached to the robot and be necessary within the run. We also made sure that everyone would get a chance to code the robot showing what they can do. We also used GitHub so that anyone can just clone our code and work on it at home. If they needed help the more experienced coders on our team would help them and teach them.

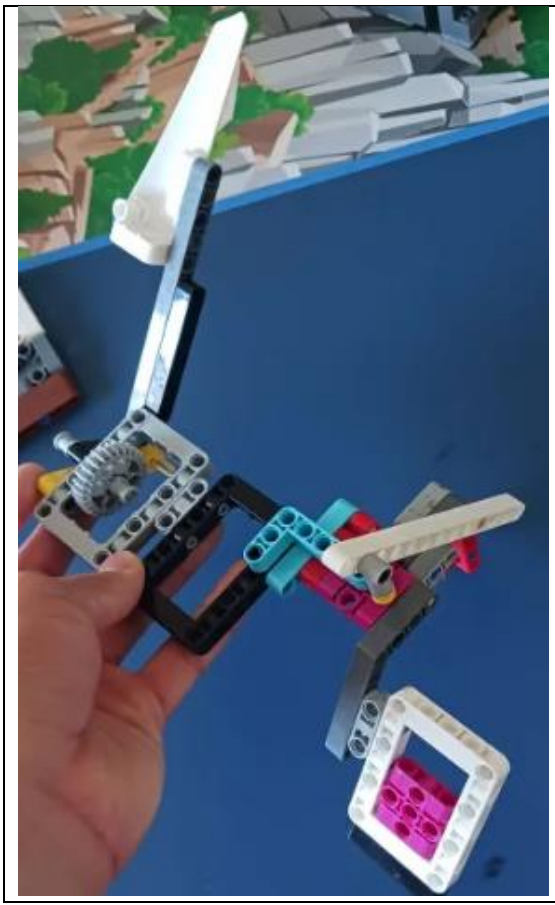
Clear evidence that all team members contributed ideas:



Gearbox
All our attachments are made on separate gearboxes and quickly attached to the motors on the top of the robot. This allows for efficient pit-stops in between runs.

	**Attachment 1** This attachment is used for the mission surface brushing, and to recover the soil. This attachment is based on the engineering principles of reliability and functionality. It is very reliable for doing the surface brushing as it works all the time because of the funnel at the end of the arm and the other arm allows us to do the soil missions at once for functionality.
	**Attachment 2** This attachment is used to complete both components of salvage operation as well as marking the area with a flag. This attachment was made using the engineering principles of problem solving and reliability. We wanted to put a flag into the pirate ship at the same time as we did it and came up with an idea to push the ship and then do the arm and let go of the flag which is also extremely reliable.

	**Attachment 3** This attachment is used to complete both components of the mineshaft explorer and the careful recovery. This attachment is based of the engineering principles of simplicity and efficiency. This mechanism allows us to do both missions at once for maximum efficiency and it is very simple with simple gearboxes to attach it easily onto the robot.
	**Attachment 4** This attachment does a lot of missions including the forge, who lived here, raising the roof of the market and heavy lifting. This uses the engineering principles of efficiency and maintainability. It actually does two of the missions with a passive mechanism for maximum efficiency and it is easy to put back together.



Attachment 5
This attachment does Statue Rebuild combined with another arm to raise the goods and pull out the item. It is designed based off the robot principle of simplicity. It has a simple arm to do the statue rebuild on a simple and standard gearbox as well as 2 passive mechanisms to the right side.

Issues we came across and how we fixed them:

- The centre of gravity of the robot was too high which mean there were inaccuracies in movement as the robot swayed when moving. To fix this we moved heavy motors as well as the hub/brick lower
- Gears were not connected well which meant attachments on gears slipped and didn't turn well. To fix this we added pieces to pin down the gears and attachments to reduce gear slippage.
- Stability wheel caused inaccuracies in movement. This wheel would keep moving even after we have stopped moving so to fix this we :
    1. Came up with 3 possible solutions.
        a. Non-Friction
        b. Caster Ball
        c. Plastic Nub
    2. Tested them (turning and driving).
    3. Determine which was best which ended up being a caster ball.

How we improved from last year:

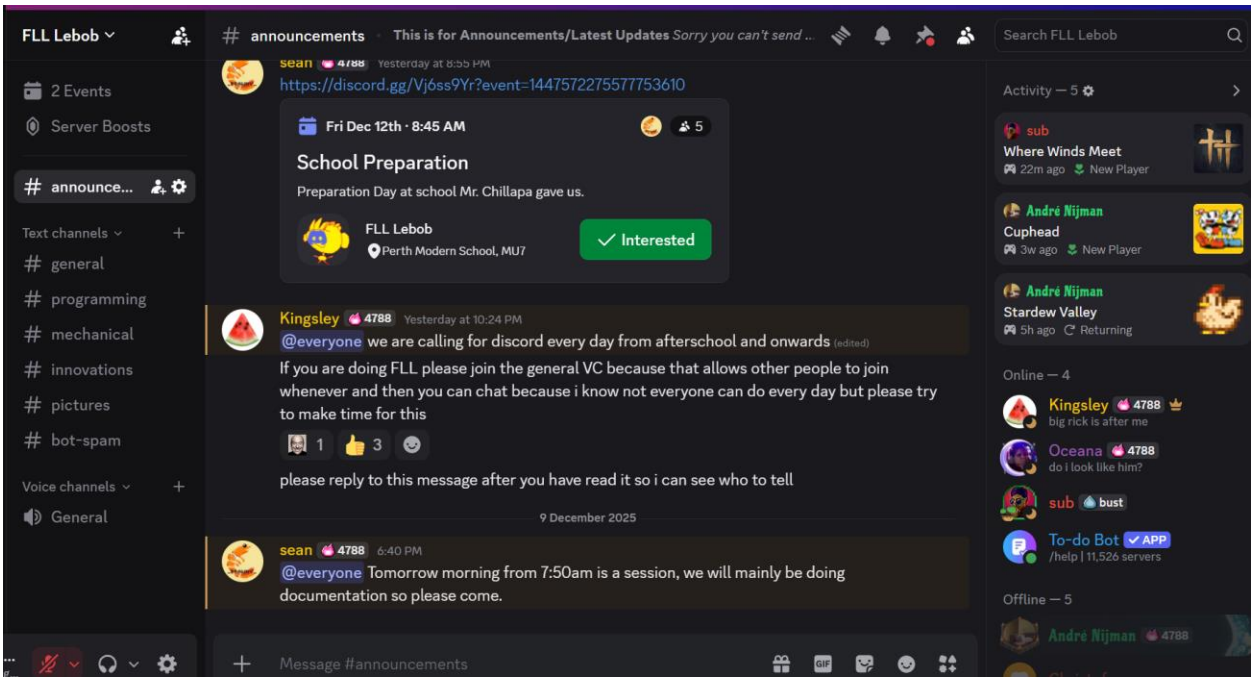
Robot problems - Last year, we didn't do very well at nationals for the robot game because we changed the whole robot and didn't have enough time to code it so this year, we decided to change the path of the robot instead of changing the robot which allowed us to code the game in time.

Logging - Another problem was the fact that we did not have enough proof of progressive testing but now we have a logbook that is in the documentation booklet.

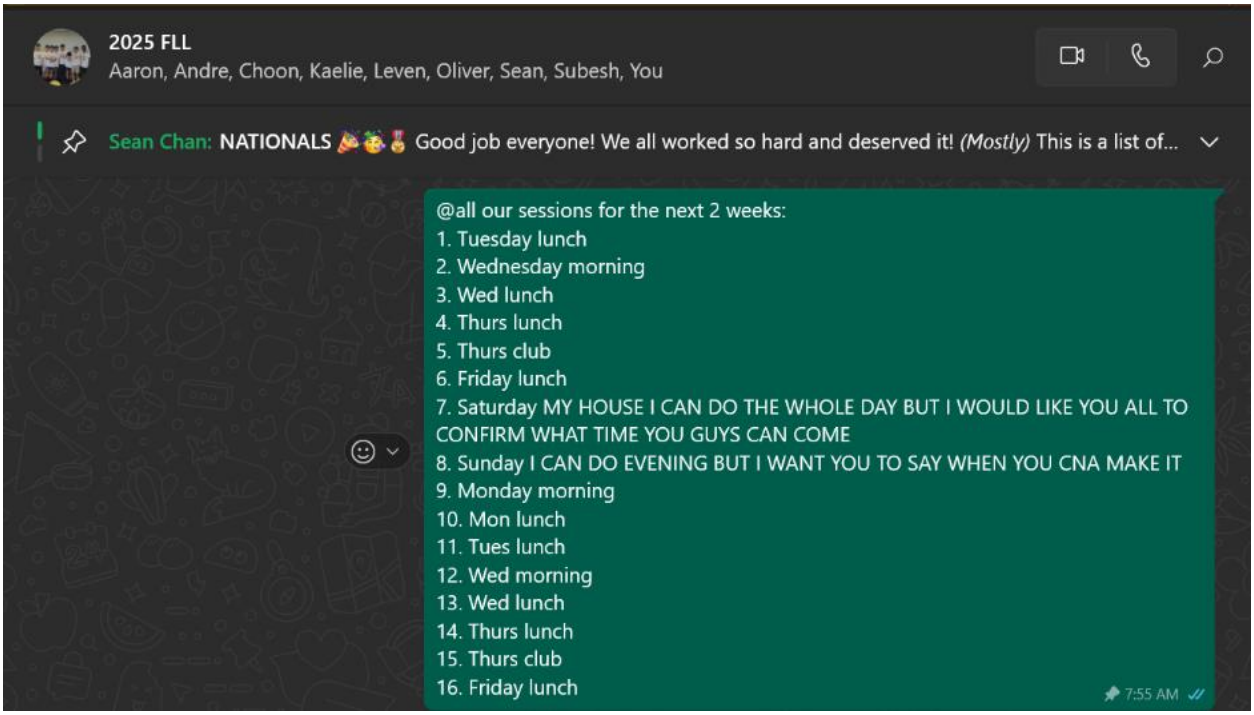
Communication - Last year we found that communicating was hard especially through emails, so this year we decided to have a WhatsApp group chat and a discord group chat where we can talk and send photos, videos and communicate. Our Discord group has an announcement, general, programming, mechanical, innovations, pictures and bot spam group as well as a voice chat for calls.

Discord Channels -

- Announcement – For important announcements for the team to see.
- General – For general chats where we talk about general things like meeting etc.
- Programming – For programming and coding from home because we have theory coding
- Mechanical – For talking about mechanical structure of the robot as well as research for the robot and the chassis
- Innovations – All the innovations electronics, coding, research and 3D printing is talked here by the team
- Pictures – Pictures are sent here for the PowerPoints and documents
- Bot spam – This was created because we have a to-do bot where we have assigned tasks for it and to access it you need a command and it takes up a lot of space in the chat where we can miss important information.
- Voice chat – This is used when we need to call overnight or on the weekends when we can't meet so we can finish PowerPoints and documents



Above is our discord group



This is our WhatsApp group, allows for faster communication, but we use discord as out main form of communication.

CREATE

Clear explanation of innovative attachments and their purpose:

See page 7-12

Clear explanation of innovative code and/or sensor use:

We based our code on programming principles to ensure our code is robust, readable, maintainable and modular.

Using PID: Because we used PyBricks for coding our robot, we can set it so that the drive base uses the built-in gyro in the spike to align our movement for better accuracy.

Resetting the angle of the mechanism motors: We made a function to do this, move until stalled, which moves the motor at a certain speed until it is unable to move physically at all. This lets us align the motor to a certain surface so that we can have better accuracy and so that people don't need to set the mechanism to the right position in the pit-stops.

Automatically Updating Menu using Decorators: We used a decorator to automatically add new functions of runs to our robot menu, allowing us to save time when coding runs.

How our code works:

This system uses Object-Oriented Programming to keep the robot's code organized and efficient. A Robot class manages motors, drive settings, gyrometer, and arm movements, while a PID class improves accuracy with angle wrapping and precise turning calculations. The MissionControl class provides a user interface on the hub, letting users select missions during matches, and it also manages animations, gyro resets, drive profiles, and timing. Missions are easily added with an `@mission()` decorator, and each mission function leverages Robot helpers for reliable driving, turning, and arm control.

pidcontroller.py

src > robot > pidtesting > pidcontroller.py > ...

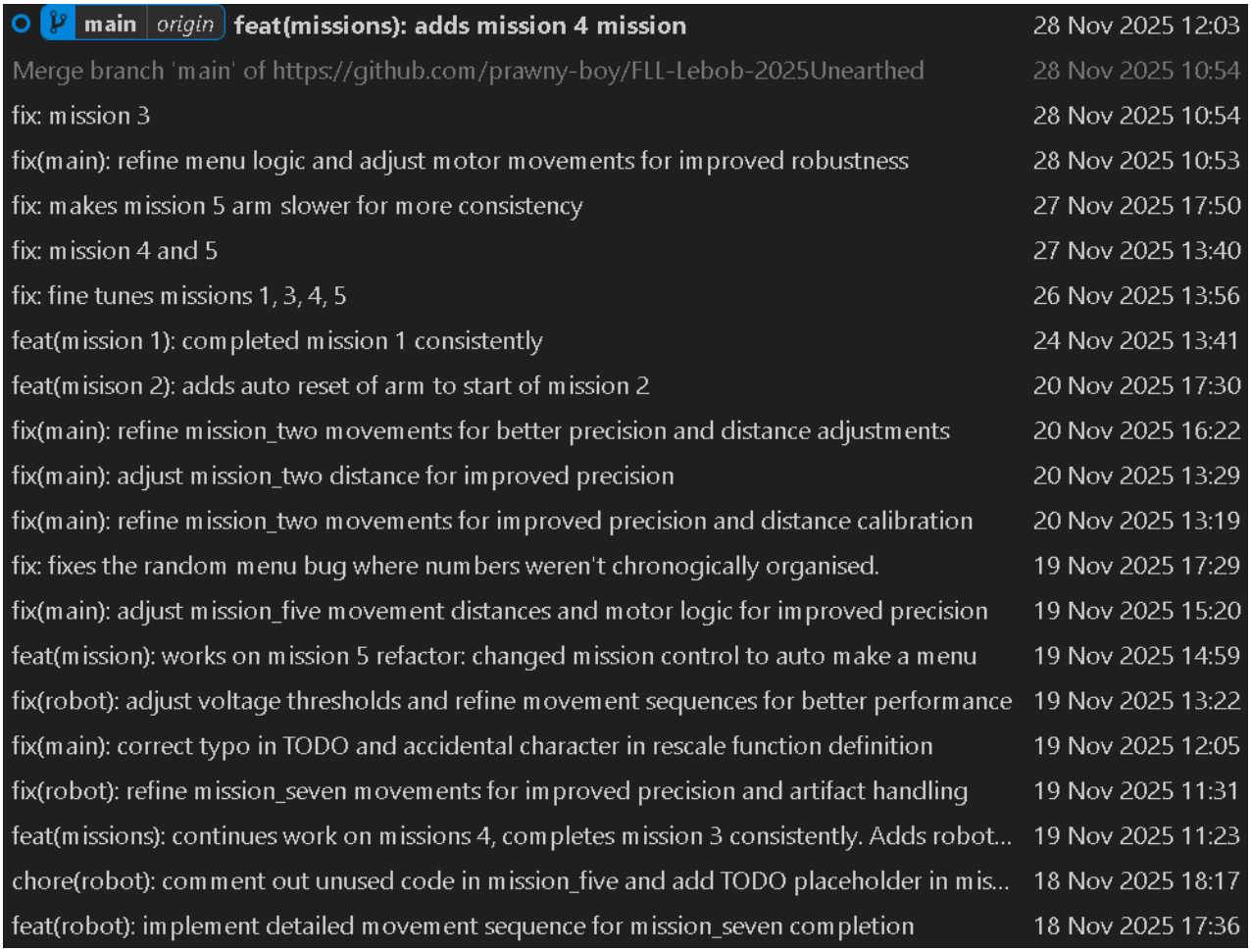
```python
"""Standalone PID utilities for testing motor control strategies."""

from __future__ import annotations

from dataclasses import dataclass
from typing import Tuple


@dataclass
class PIDController:
    kp: float = 1.0
    ki: float = 0.0
    kd: float = 0.0
    output_limits: Tuple[float, float] = (-100.0, 100.0)
    integral_limit: float | None = None

    def __post_init__(self) -> None:
        self.integral = 0.0
        self.previous_error = 0.0

    def reset(self) -> None:
        self.integral = 0.0
        self.previous_error = 0.0

    def _clamp(self, value: float, limits: Tuple[float, float]) -> float:
        low, high = limits
        return max(low, min(high, value))

    def _update(self, error: float, dt: float) -> float:
        if dt <= 0:
            raise ValueError("dt must be positive")
        self.integral += error * dt
        if self.integral_limit is not None:
            self.integral = self._clamp(self.integral, (-self.integral_limit, self.integral_limit))
        derivative = (error - self.previous_error) / dt
        output = self.kp * error + self.ki * self.integral + self.kd * derivative
        self.previous_error = error
        return self._clamp(output, self.output_limits)

    def to_speed(self, target_speed: float, measured_speed: float, dt: float) -> float:
        """Return duty cycle to reach target speed (deg/s)."""
        error = target_speed - measured_speed
        return self._update(error, dt)

    def to_angle(self, target_angle: float, measured_angle: float, dt: float) -> float:
        """Return duty cycle to drive toward desired angle (degrees)."""
        error = target_angle - measured_angle
        return self._update(error, dt)

    def to_angle_with_speed(
        self,
        target_angle: float,
        measured_angle: float,
        measured_speed: float,
        max_speed: float,
        dt: float,
    ) -> float:
        """Nested control: angle error converted to speed target and fed into the speed loop."""
        angle_error = target_angle - measured_angle
        desired_speed = self._clamp(angle_error, (-max_speed, max_speed))
        return self.to_speed(desired_speed, measured_speed, dt)


__all__ = ["PIDController"]
```

This is a picture of our PID code

ITERATE

Clear evidence of repeated testing of their robot and code:

We have our commit changes for our code on GitHub, and it shows how we have repeatedly tested and improved all our code over a long period of time. There should be a timelapse running. These screenshots of all our commits should show this. We have many logs of each run we have done since regionals in our documentation folder as well:



feat(robot): add ability to override and restore drive settings with optional speed parameter	18 Nov 2025 13:47
fix(robot): refine mission_three movements for improved precision and artifact uncovering ...	18 Nov 2025 13:32
fix(robot): adjust mission_three movements for better alignment and artifact handling	17 Nov 2025 13:33
docs: add contributor guide and expand README with setup instructions and project details	16 Nov 2025 23:10
Merge branch 'experimental'	16 Nov 2025 22:54
chore(main): add comment for testing git signing	16 Nov 2025 22:47
chore(ide): add JetBrains IDE project configuration files bro it keeps being annoying and w...	15 Nov 2025 19:57
fix: gitignore bug	14 Nov 2025 17:12
removes pycache in sub directories	14 Nov 2025 17:12
removes pycache files and adds gitignore to ignore them when pushing to repo. added ye...	14 Nov 2025 17:08
Revamp testing utilities	14 Nov 2025 14:21
Refactor robot control and missions	14 Nov 2025 14:06
Add robot control and pathfinding features for FLL Unearthed 2025	14 Nov 2025 21:47
fix(mission): update mission three driving sequence for improved performance	14 Nov 2025 21:30
fix(main): remove confirmation comment for DRIVEBASE_AXLE_TRACK value	14 Nov 2025 21:28
fix(mission): adjust drive distances and motor rotations for mission one and three	14 Nov 2025 21:25
Merge pull request #2 from prawny-boy/small-error-fix	14 Nov 2025 13:07
feat(pathfinding): add pathfinding and visualization modules with PID controller integration	14 Nov 2025 20:59
Merge branch 'main' of https://github.com/prawny-boy/FLL-Lebob-2025Unearthed	14 Nov 2025 12:44
feat(missions): further implements mission one. fix(pid movement): continues work on smar...	14 Nov 2025 12:44
Revise README for Unearthed 2025 season	12 Nov 2025 17:50
Include Apache License 2.0	12 Nov 2025 17:49
feat(missions): implements a rough estimate of missions 4 and 5, untested. Also adds progr...	11 Nov 2025 21:13
feat(smart movement): adds smarter turn in place and drive for distance functions that use ...	9 Nov 2025 10:52
feat(stopwatch): adds a stopwatch for missions using the StopWatch pybricks class that sho...	8 Nov 2025 09:26
feat(use gyro): adds a optional use_gyro parameter to the Robot class that controls if the r...	8 Nov 2025 09:11
feat(mission three): improves mission three so that it can do two missions consistently. Refe...	8 Nov 2025 08:50
fix: fixes turn_in_place function bug	3 Nov 2025 14:51
feat(missions): implements missions 1 and 2, both untested.	16 Oct 2025 17:27
feat(path 3): added robot game strategy version three path.	16 Oct 2025 15:51
refactor: refactors code to prepare for this year's season.	25 Sep 2025 15:51
feat(missions strats files): adds mission robot game strategy files for reference.	21 Aug 2025 16:05
test: continues testing with interactive path planning and path finding.	21 Aug 2025 16:11
test(pid): tests pid implementation for movement of robot.	17 Aug 2025 22:09
feat(setup): Environment Setup and Initial Files with base code initial commit with basic pro...	13 Aug 2025 21:15

Clear evidence of improvements based on testing:

To ensure our runs were consistent, we constantly improved our code and mechanisms after running them repeatedly. Although it was very slow, we ended up with consistent runs which can be run multiple times without failing. The commits above also show the many changes and fixes we made to the code, and in the documentation file there is a log of all the runs that we have done.

COMMUNICATE

Detailed explanation of process and lessons learned:

“It’s about the friends we made along the way” – Kingsley, a member of our team

This was what we learnt in FLL, to achieve a lot, but also know that we can’t make it perfect, unlike those 545 MAX points teams. We just need to do our best.

Working Together: As a team we learnt to work together, even during the hardest times.

Division of Work: We found that having few people on many different things was better than having lots of people on one thing, overcrowding the area.

Gracious Professionalism: We learnt to be GP!

Staying on Task, Not getting distracted, Longer Attention Span: These are some advantages we gained.

Healthy Work Environment: To make everyone happy and work better.

No big changes: Last year, we changed the whole robot before nationals. Didn’t have enough time to code it. This year we changed the path instead, didn’t have to code it that much.

Documentation: Last year, we didn’t have proof of progressive testing. This year we have a logbook and documentation.

Communication: Communication was confusing. We used a WhatsApp group and Discord group chat this year. It was better for talking, sending photos, announcements and organisation.

Organisation: We used trello to organise us and it allowed us to know what to do and work faster than last year.

Team clearly shows pride or enthusiasm for their work:

Our team is extremely proud of our robot in many ways; here are statements from each person in our team.

Sean: I am proud of how good our robot turned out, considering the short amount of time we had to build it. We achieved an amazing score!

Andre: I really appreciated our teams effort this semester, everyone put their best work into this.

Kingsley: I am very proud of making the robot and some of its attachments. Our teammates collaborated and worked very well together as well as listening and interpreting different things.

Chris: I am very happy to work on the robot this year, as I generally only worked on the innovation project in my last team. Having the chance to work on the robot allowed me to expand on my existing knowledge and abilities.

Leven: I am glad that I was able to gain this experience, as it was only my second year of robotics, being able to contribute ideas to all aspects of our project, designing a mechanism, working on documents, and being on the board operating the robot. I think it has been valuable experience for my teamwork and cooperation skills.

Oliver: I think our team collaborated very well, and we all cooperated to complete the mission to the best of our ability.

Aaron: I think that my team helped me understand the code that they wrote, told me what errors to fix, and collaborated well across all categories in general. The group chats made it easy to be organised and to communicate.

Subesh: I believe that our team worked cohesively throughout this semester, despite setbacks with delays in receiving the mission models. I believe we were organised and that everyone did their part in all parts of FLL.