**FireWall Script**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pranav: We refined our mission strategy 25 times, while increasing wall range, and utilizing Python for async. We focused on missions involving collecting and dropping things, simultaneous missions, and used smaller zones for easy navigation, optimized our path plan greedily to maximize points. Our top score is 515 with 5 runs:

* Run 1: Collect samples and make space simplifying navigation for other runs.
* Run 2: Complete left-side missions
* Run 3: Quickly drop collected samples in the boat and the coral segments.
* Run 4: Travels to the right side, while completing missions along the way.
* Run 5: Complete right-side missions

Advik: Our robot, FireWall, features an innovative front wall that moves in all four directions for easier navigation, during robot movement and asynchronous mission completion. We created simultaneous movement with our wall by connecting the side-to-side motor on the vertical motor. After qualifiers we increased our extended our vertical and horizontal range by 10 studs by adding more teeth to our rack. After qualifiers we increased our range we extended our vertical and horizontal range by 10 studs by adding more teeth to our rack. We did over 20 stability iterations. We used plates to keep a beam connected to our rack in place and then we used two racks instead of one. We used a gyro sensor for accurate navigation. We decided not to use color sensor due to varying lighting conditions across competitions and there are fewer lines for accuracy. We learnt that having multiple connection points and dimensions of connection was vital for both robot and attachments stability.

Mohith: By utilizing our wall’s variety of motion, we innovated our attachments to create attachments that were modular, compact and easy to use. We innovated our attachments to be easily used by the wall and rely on our wall’s motion.

Divith: We split up into groups and made sure every two hours we would switch, making sure that everyone could code and innovate on our attachments. Saanvi and I coded run 5 and we have a dropper mechanical alignment, and reused attachments to do 3 missions

Saanvi: Advik and I coded run 1 and we built the multi purposed krill collector and mast attachment which we made way smaller to ease navigation, multiple iterations to collect seabed sample

Mohith: Pranav and I coded run 2 and we are doing 3 missions simultaneously and did multiple iterations to adjust the mission offsets and diver lifter and dropper.

Pranav: Advik and I coded run 4 and we built shark passive dropper attachment

Advik: I helped everyone with overall debugging

Saanvi: Even though people worked on different runs we helped each other code, debug, build attachments, and deal with any problems that occurred.

Pranav: For our design process, at the start of the season we had a 2-hour brainstorming discussion on whether we should have used the wall or not, and in the end we decided to build a prototype of the wall and compare with our last year robot. We tested it for accuracy, ease of changing attachments, and navigation before deciding to keep it.

Saanvi: Our Testing process had two methods to ensure reliability:

* 10 out of 10 Method: Ran the code 10 times to identify consistently failing missions, helping us pinpoint issues in the code or attachments.
* Table Method: Checked if all missions were working properly based on our attachments and code.

For our debugging process, we created a debugging cheat sheet that our team members could use whenever we needed help with debugging.

Divith: For code, we created a common code library FlamingAPI . We created multiple functions for gyro turning, proportional gyro straight, and Moving wall.

* Last year we used Python 2 and Gyro sensor for proportional gyro straight. We used a timer-based solution and our runs were not so reliable as timer was not granular to milliseconds. So, this year we started with block coding. But later we discovered that Python 3 had many fixes for better timer and asynchronous functions.
* For Proportional Gyro Straight we created many versions
  + We created a timer based and compared with relative motor position based and saw that was more accurate for runs.
  + We then tried steering and compared it with tank turn and learnt that tank turn was more reliable.
* The attachment weight and wall positioning affected the gyro proportional error, so we had to tune that for every run.

Saanvi:

* We also learnt that we had to tune the delay before asking yaw from gyro sensor otherwise it would get stuck and start returning –180 instead of correct angles.
* We also debugged and learnt that gyro sensor returned incorrect values for first 300 milliseconds randomly, so we ignored these values by adding a delta from our current angle.
* For gyro turning, we innovated to create a new Decel-Gyro Pro function that allows more accurate and faster gyro turning based on the delta between target and current yaw. We discovered that this was more reliable than last year gyro calibration method.
* We also learnt and used asynchronous programming from Python to make our wall movement the same time as the robot to make runs more efficient.

Mohith: We used some really helpful building and coding resources to help support our mission strategy. We took inspiration from Brickwise and Brickering for their moving wall robot examples, and YouTube videos for inspiration for attachments. Prime lessons for Python3, First tech forum for gyro problems, and Spike prime API were our top 3 coding resources. We have dozens of citations for coding and building resources we have listed here.