Paly Robotics Team #8 Scouting Documentation

Authors: Robbie Selwyn, Ailyn Tong, Alex Tarng

Table of Contents

I. Introduction and Structure

Motivation and Issues with Previous Season

In past years, scouting was something thrown together right before competitions. Most of the team did not fully appreciate the potential impact that good scouting can have on a team's performance.

We decided to create an organized and effective scouting system after a piece of scouting data played a crucial role last season, allowing us to attend the 2016 FRC World Championships. The new system was one of the largest factors influencing our performance this year: we were consistently a top first pick for alliances and played in the Roebling Semifinals at the Houston World Championships.

Timeline and Resources

The work of developing our scouting system began in August 2016, when we first started discussing the idea of creating a new scouting system. We chose to use the collection-viewer-server model that is detailed in the further sections. Soon after we decided on the structure, we began working on the viewer app, the Python server, and the collection app. We decided to make a version of the collection app for Stronghold, so that we could get an idea of how the app would work. We tried to complete as much game-independent work as possible for the viewer app in the fall, but we still had a considerable amount of work on the app throughout the season, which included the QR code communication, the match screen, and pit scouting section.

Competition Structure

During competition, we divide the scouts into "squadrons" of at least six members, including a designated lead and second-in-command. Squadron leads are responsible for making sure their group completes assigned scouting tasks. Additionally we have a stands captain, who is responsible for coordinating between squadrons, communicating with drive team, and discussing the abilities of teams with other strategy members.

Squadrons split into pairs for pit scouting. At least one member in each pair will have a compatible phone with the viewing app, which is used to collect pit scouting data as well as view all uploaded data and statistics.

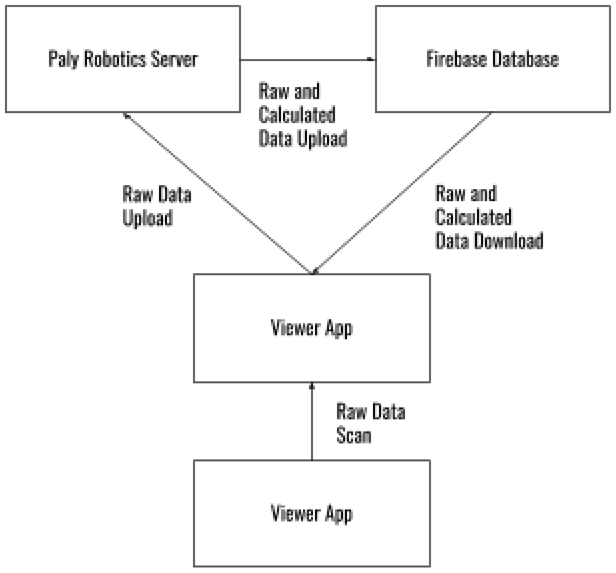
II. Implementation

This year, we implemented our scouting system with three main components. The first device is the collection app. This app runs on iPads, and allows scouts to collect data during matches. After a scouting session, a QR code containing all the raw data for that match is generated, which can then be scanned and uploaded to the server via the viewer app. The second device, the viewer app, is used to view data, conduct pit scouting, and work on match strategy. The third device is the server, which receives data that is scanned by the viewer app. All the data that it receives is uploaded to Firebase, where it can be accessed by the viewer app.

Information about each component is broken down into sections below.

Server

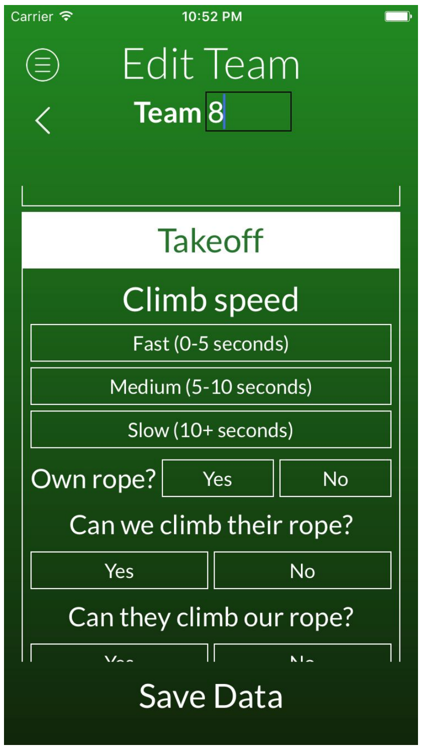
The server for our scouting app was written using Python (2.x), and it uses the **flask** framework to run. The server is broken down into several files, each of which accesses separate data locations. The first external service that is accessed by the server is Firebase, which is where all our data is stored. Once the server receives data uploaded by the app, it inserts the raw data and begins to calculate our different statistics.

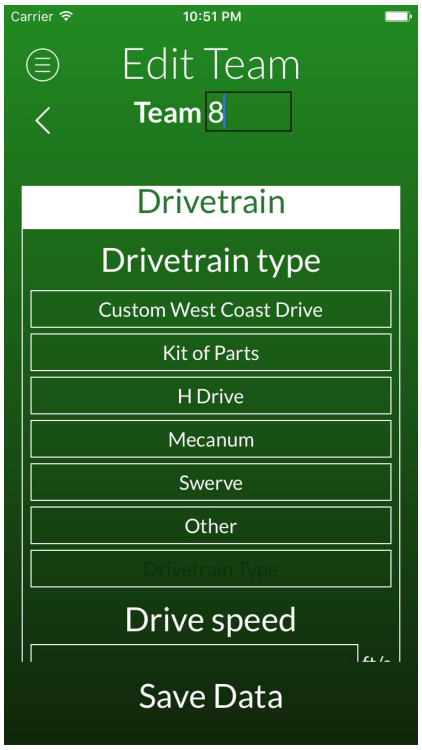


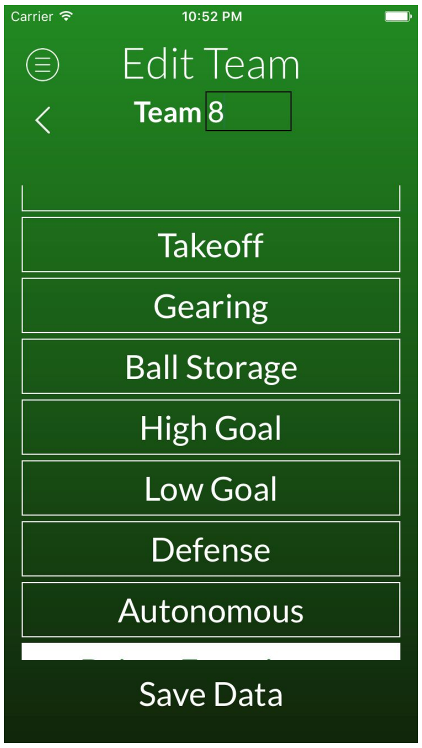
The second service that is accessed by the server is TheBlueAlliance, which we use to obtain the match schedule, current team record, and the number of rotors that were scored in each match (for QA purposes). Throughout competitions, we run several quality assurance checks on all of the data. Using TheBlueAlliance, we cross-checked the scouted number of rotors scored against the total number of gears that were reported by our scouts. This allowed us to re-scout any matches that were incorrectly scouted.

Viewer App

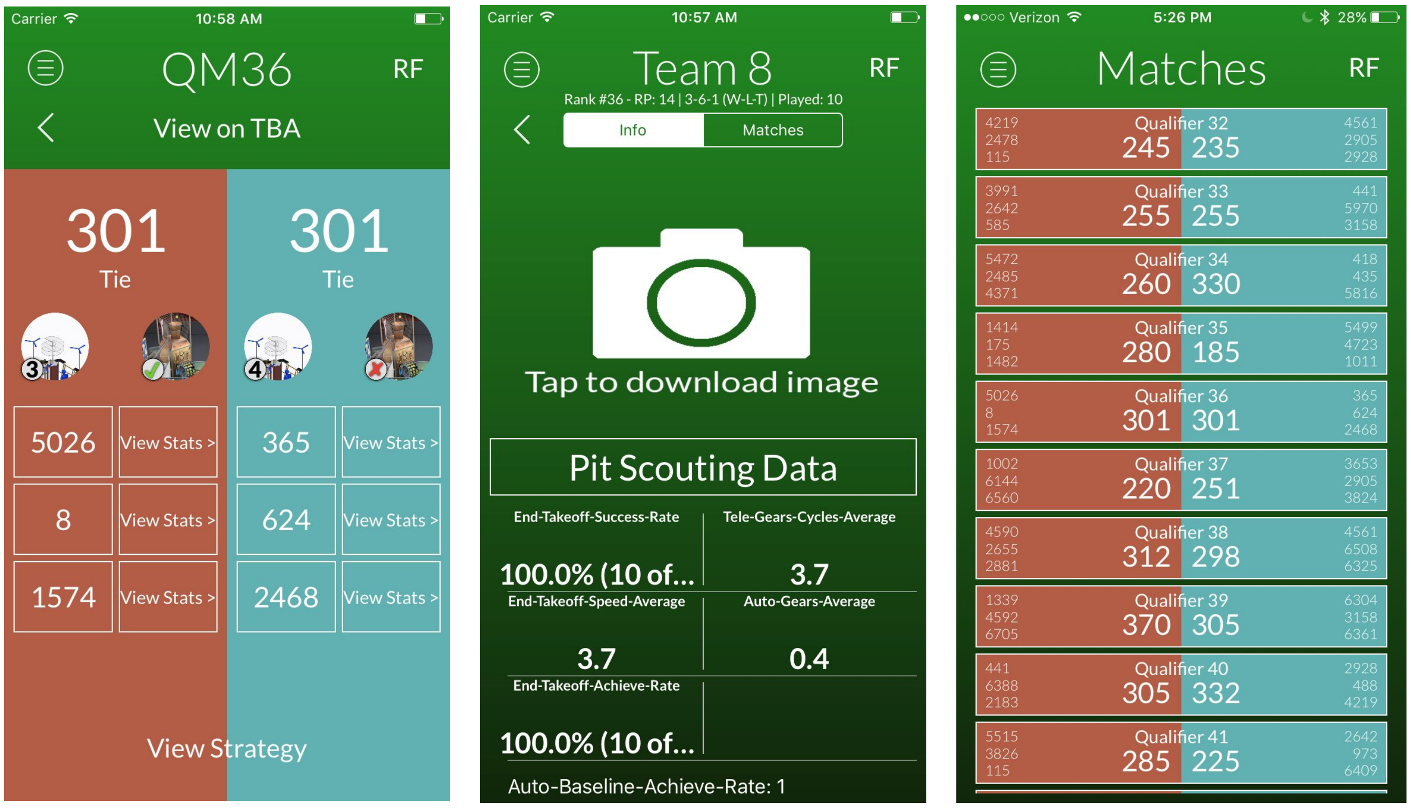
The second component of our scouting system is the viewer app, which was written for use on an iPhone (any size). The viewer app had several main features. First, we used the app to pit scout robots. The app allows scouts to enter data about robots, and save the data on the device for later upload when the scouts finish the rounds of pit scouting.







The second main app feature was the ability to view data. The app can view calculated team data, pit scouting data, data from the individual matches, and an image of the robot (taken by camera and uploaded manually to Dropbox). For pre-match strategy, we created what is called the "pre-match" page, which gives a synopsis on the performance of all teams in a given match. This allows our team to identify trends in statistics like gear counts and climbs, and also to quickly review our scouts' notes on past matches, all on one screen. This screen turned out to be one of the most important aspects of the app, as all of the relevant data from each of a team's past matches was available in one page. In addition, the app allows users to sort by several important statistics, such as gears delivered, climb rate, gears delivered in auto, and climb speed.


Carrier 10:58 AM

QM36 RF

View on TBA

301 Tie | 301 Tie

5026	View Stats >	365	View Stats >
8	View Stats >	624	View Stats >
1574	View Stats >	2468	View Stats >

View Strategy

Carrier 10:57 AM

Team 8 RF

Rank #36 - RP: 14 | 3-6-1 (W-L-T) | Played: 10

Info | Matches

Tap to download image

Pit Scouting Data

End-Takeoff-Success-Rate	Tele-Gears-Cycles-Average
100.0% (10 of...	3.7
End-Takeoff-Speed-Average	Auto-Gears-Average
3.7	0.4
End-Takeoff-Achieve-Rate	
100.0% (10 of...	

Auto-Baseline-Achieve-Rate: 1

●●○○○ Verizon 5:26 PM 28%

Matches RF

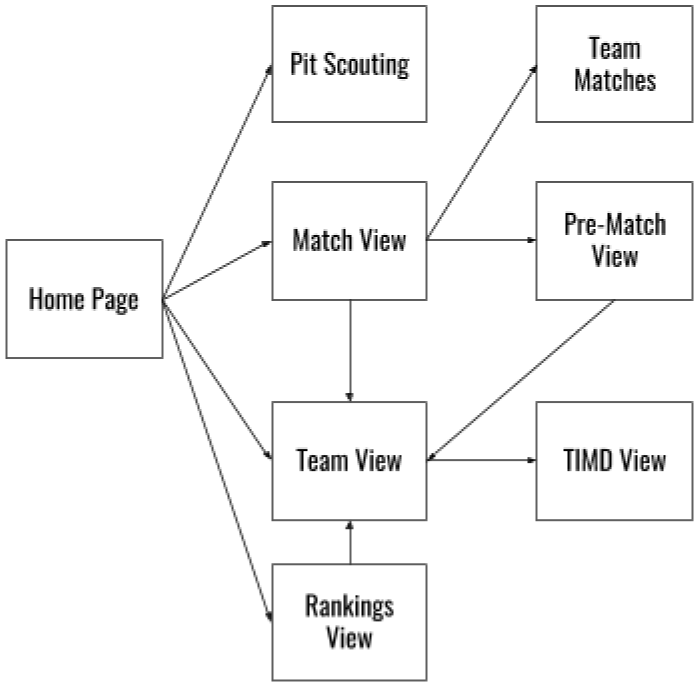
4219 2478 115	Qualifier 32	245	235	4561 2905 2928
3991 2642 585	Qualifier 33	255	255	441 5970 3158
5472 2485 4371	Qualifier 34	260	330	418 435 5816
1414 175 1482	Qualifier 35	280	185	5499 4723 1011
5026 8 1574	Qualifier 36	301	301	365 624 2468
1002 6144 6560	Qualifier 37	220	251	3653 2905 3824
4590 2655 2881	Qualifier 38	312	298	4561 6508 6325
1339 4592 6705	Qualifier 39	370	305	6304 3158 6361
441 6388 2183	Qualifier 40	305	332	2928 488 4219
5515 3826 115	Qualifier 41	285	225	2642 973 6409

(Photos of the Match Review, Team View, and Match List Screen)

Easy accessibility of all data and the ability to navigate the app according to the user's train of thought were major constraints in the design of the viewer app. The user can begin viewing data by entering the Team List screen or Match List screen from the menu, and from there, the user is able to access the data summary for any team or match. The team screen allows the user to view all matches in which the team is scheduled to play, and by tapping the match, the user is brought to the match screen. Vice versa, all teams in a match are displayed on the match screen, and tapping them brings the user to the team's data summary. In addition, by tapping the "View Stats" button on either screen, the user can view the raw data from any team in any match for which scouting data exists. Most importantly, as the user navigates through the screens, each screen is added to a navigation stack, allowing the user to backtrack by tapping the back button.

The viewer app is also able to scan QR codes from the collection app and upload the raw data to the server via flask. A summary of the raw data is displayed on the screen, allowing the user to double check the data before uploading. At the team's first regional, data upload time was about 30 seconds. Through the implementation of multithreading on the server side and a more efficient Firebase upload scheme, the upload time of raw data from the viewer app was drastically improved to less than 2 seconds.

A planned, but unimplemented feature of the viewing app was the ability to create pick lists, local to a user's device. This would allow the user to create multiple lists of teams, with easily accessible summaries of each. For example, the user would be able to expand a brief description of each team on the list, including custom notes from the user. The ability to generate multiple pick lists would allow the user to sort teams based on different criteria, according to whether the teams were to be first picks or second picks, as well as strategic counters to opponent alliance compositions. During alliance selection, the user would be able to enter "Alliance Selection Mode" on the app, in which tapping a team would cross it out, indicating that the team is unable to be selected, allowing the team representative and strategists to easily make decisions about picks.



The viewer app also has the ability to report bugs, although this feature was never used in practice.

Collection App

The last component of our scouting system was our collection app, which was used by our scouts in the stands to scout matches. This app, which ran on team-provided iPads (without internet), allowed the scouts to easily enter all a robot's actions, as well as its attempted actions. The app has a screen for autonomous, teleop, and a screen where the scouts can enter overall information about the robot (driver skill, speed, notes, etc.). The information exchange between the viewer app and the collection app happens through QR codes, which are generated by serializing all the data collected throughout the match. (The following page has images of the collection app.)

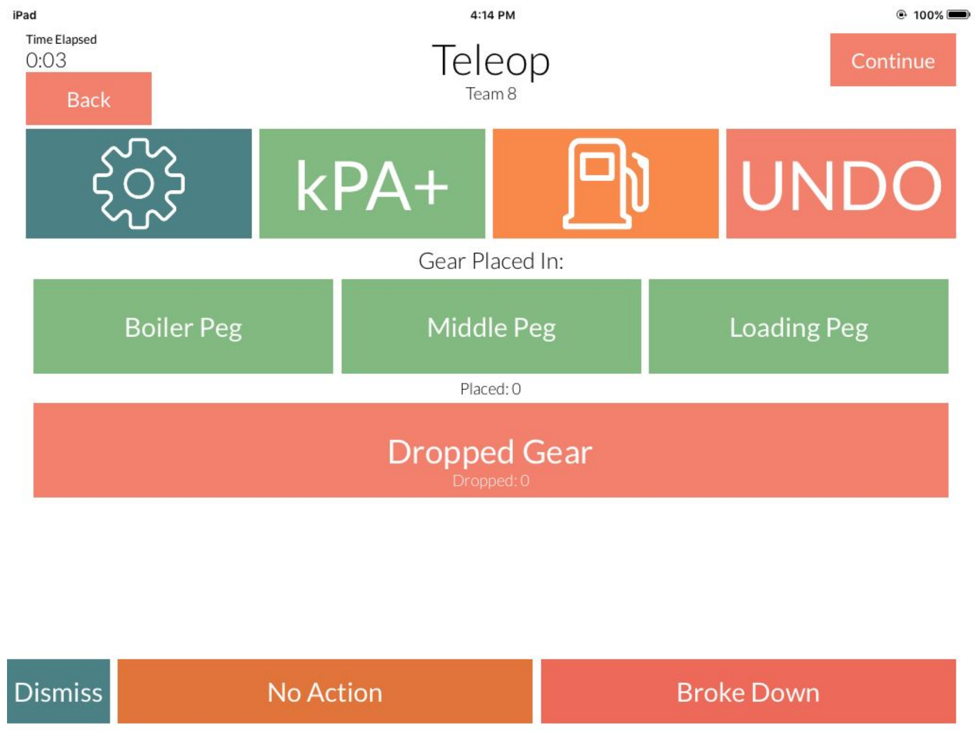
The collection app was optimized for ease of use for scouting in fast-paced matches. The autonomous and teleop screens (which were almost identical) has three main sections, each section accessible by a row of buttons at the top. These three sections were Gearing, Fuel, and Intake. By dividing up the possible robot actions into multiple sections, we were able to make the buttons larger, reducing screen clutter and making it easier to tap the correct button, both of which were essential to minimize the time for which the scout has to look away from the match. Tapping a button also flashes it white, allowing scouts to see which button was pressed in their peripheral vision.

During autonomous and teleop, a timer is visible in the upper left. This timer serves two main functions. First, when the timer reaches 15 seconds elapsed in the autonomous phase, the "Continue" button in the upper right continuously flashes white, alerting the scout that the autonomous phase is supposed to have ended and reminding them to continue to the teleop phase. This solved the issue of scouts forgetting to tap continue after the end of the autonomous phase. In addition, the timestamps of certain actions are recorded and used to generate cycle times. The implementation of cycle times was a trial this year, to test whether collecting such data would be useful. In the calculation of the cycle times, certain edge cases were taken into account in order to minimize error. For example, the time of the first gear or fuel cycle in teleop was not calculated, due to the result of the robot's autonomous having a possible effect on the first cycle's time. In addition, the app attempts to handle joint gear/fuel cycles. However, we found that even taking these precautions, calculated cycle times were still extremely inaccurate, and as a result, cycle time data generated from the app was rarely used in strategy discussions.

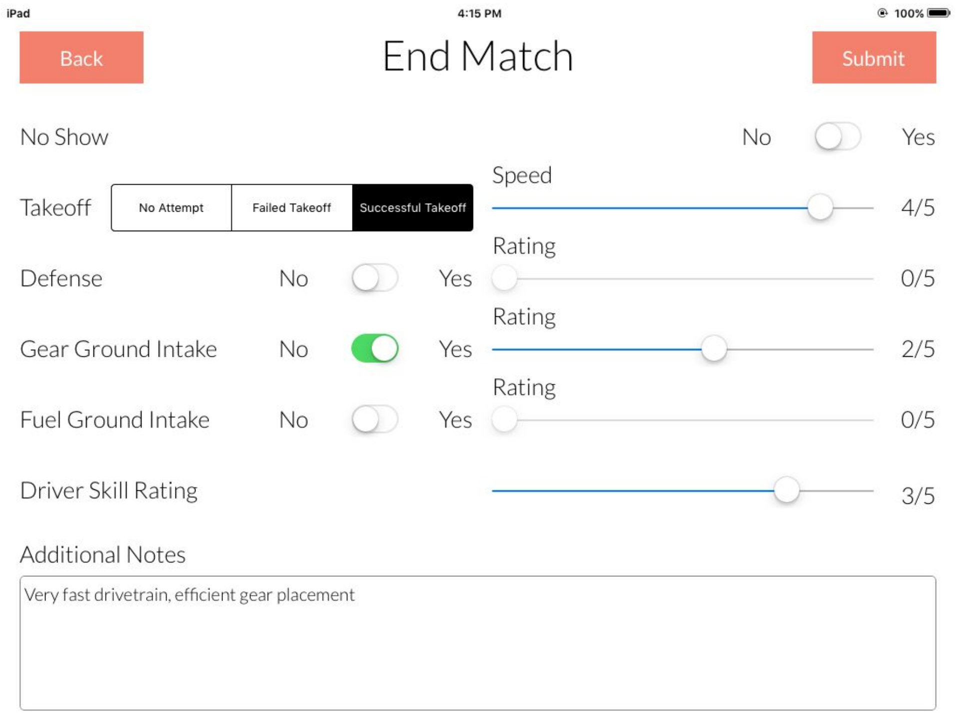
At the end of a match, the scout can input qualitative and subjective quantitative data not otherwise gathered. If a robot has performed a certain function during the match (such as fuel ground intake), the scout is asked to rate the effectiveness of the performed function. In addition, the notes section allows for detailed quantitative observations. The notes section was one of the most useful data points in strategic discussions.

After the scout taps "Finish" after a match, the collection app parses the raw data and puts it into CSV format, from which a QR code is generated and displayed, taking up the majority of the screen to allow for ease of scanning. Since there is an upper limit to the density of a QR code that a phone can scan, this also allows for a larger character limit in the notes section. In addition, the raw data is displayed next to the QR code.

At our two regionals, the scout had to manually input the team and match number in order for the data to be correctly uploaded to the server. However, due to poor visibility of team numbers at the start of a match as well as confusion as to what number the current match was, some data was lost due to mistakes in team or match number. Therefore, for the World Championships, we added the ability for the collection app to fetch a match schedule from The Blue Alliance and automatically assign a team number according to the iPad's device ID. This solved the issue of incorrectly typed numbers, and as a result, there was almost no missing data at CMP.



(Teleop Scouting Screen)



iPad 4:15 PM 100%

Back | End Match | Submit

No Show: No [Yes]

Takeoff: No Attempt / Failed Takeoff / [✓ Successful Takeoff]

Speed: 4/5

Defense: [✓ No] Yes — Rating 0/5

Gear Ground Intake: No [✓ Yes] — Rating 2/5

Fuel Ground Intake: [✓ No] Yes — Rating 0/5

Driver Skill Rating: 3/5

Additional Notes: Very fast drivetrain, efficient gear placement

(Match Review Screen)

III. Data Collected

Throughout each competition, we collected thorough data on a robot's performance in a match, called Team In Match Data (TIMD), and this was used to calculate many overall team statistics. Below, the full data model of what was collected is shown.

Match Data Descriptions: These describe the robot's performance in a single match.

`match/auto-baseline`: Whether or not the robot crossed the baseline in Auto.

`match/auto-fuel-high-cycles`: Number of high-efficiency boiler fuel cycles the robot shot in Auto. A full or near-full robot "hopper" counts as 1 cycle; anything less than ~75% capacity counts as ½ cycle.

`match/auto-fuel-high-positions`: Position from which the robot shot fuel into the high-efficiency boiler (inside/outside key) in Auto.

`match/auto-fuel-intake-hopper`: Whether or not the robot used the hopper in Auto.

`match/auto-fuel-low-cycles`: Number of low-efficiency boiler fuel cycles the robot dumped in Auto. A full or near-full robot "hopper" counts as 1 cycle; anything less than ~75% capacity counts as ½ cycle.

`match/auto-gears`: Number of gears the robot successfully placed onto the airship in Auto.

`match/auto-gears-failed`: Number of attempts the robot failed to place a gear in Auto.

`match/auto-gears-failed-positions`: The position (loading station side, center, boiler side) from which the robot failed to place a gear.

`match/auto-gears-intake-ground`: Whether or not the robot used gear ground intake in Auto.

`match/auto-gears-position`: The position (loading station side, center, boiler side) from which the robot successfully placed a gear.

`match/auto-robot-broke-down`: Whether or not the robot broke down in Auto.

`match/auto-robot-no-action`: Whether or not the robot failed to move in Auto.

`match/end-defense`: Whether or not the robot played any defense during the match.

`match/end-defense-rating`: Human rating of the robot's defensive play from 1 to 5 (-1 if not applicable).

`match/end-driver-rating`: Human rating of the driver's driving ability from 1 to 5.

`match/end-fuel-ground-intake`: Whether or not the robot used fuel ground intake during the match.

`match/end-fuel-ground-intake-rating`: Human rating of the robot's fuel ground intake ability from 1 to 5 (-1 if not applicable).

`match/end-gear-ground-intake`: Whether or not the robot used gear ground intake during the match.

`match/end-gear-ground-intake-rating`: Human rating of the robot's gear ground intake ability from 1 to 5 (-1 if not applicable)

`match/end-no-show`: Whether or not the robot failed to show up to a match.

`match/end-notes`: Open-ended qualitative notes from the scout.

`match/end-takeoff`: Whether the robot 1) did not attempt, 2) failed, 3) succeeded takeoff. Climbing the rope partially and stopping/falling and trying to catch onto the rope both count as failed attempts.

`match/name`: Name of the scout.

`match/tele-fuel-high-cycle`: Number of high-efficiency boiler fuel cycles the robot shot in Teleop. A full or near-full robot “hopper” counts as 1 cycle; anything less than ~75% capacity counts as ½ cycle.

`match/tele-fuel-high-cycle-in-key`: Whether or not the robot shot fuel into the high-efficiency boiler from within the key in Teleop.

`match/tele-fuel-high-cycle-out-of-key`: Whether or not the robot shot fuel into the high-efficiency boiler from outside the key in Teleop.

`match/tele-fuel-high-cycles-time`: Average time (in seconds) the robot takes to complete one high-efficiency fuel cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`match/tele-fuel-intake-hopper`: Whether or not the robot collected fuel from hoppers in Teleop.

`match/tele-fuel-intake-loading-station`: Whether or not the robot collected fuel from the loading station in Teleop.

`match/tele-fuel-low-cycles`: Number of low-efficiency boiler fuel cycles the robot shot in Teleop. A full or near-full robot “hopper” counts as 1 cycle; anything less than ~75% capacity counts as ½ cycle.

`match/tele-fuel-low-cycles-times`: Average time (in seconds) the robot takes to complete one low-efficiency fuel cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`match/tele-gears-cycles`: Number of gear cycles the robot completes in Teleop. A gear cycle consists of driving to the loading station, acquiring a gear, then driving back to the airship and placing the gear on a peg.

`match/tele-gears-cycles-times`: Average time (in seconds) the robot takes to complete one gear cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`match/tele-gears-dropped`: Total number of gears the robot dropped in Teleop.

`match/tele-gears-intake-dropped`: Number of gears the robot attempted and failed to intake from the ground in Teleop.

`match/tele-gears-intake-ground`: Number of gears the robot successfully managed to intake from the ground in Teleop.

`match/tele-gears-intake-loading-station`: Number of gears the robot successfully managed to intake from the loading station in Teleop.

`match/tele-gears-position-boiler`: Number of gears the robot scored on the boiler side peg in Teleop.

`match/tele-gears-position-middle`: Number of gears the robot scored on the center peg in Teleop.

`match/tele-gears-position-loading`: Number of gears the robot scored on the loading station side peg in Teleop.

`match/tele-robot-broke-down`: Whether or not the robot broke down in Teleop.

Team Data Descriptions:

These are calculated based on the results of each robot in several matches.

`team/Auto-Baseline-Achieve-Rate`: # success/matches the robot reached the baseline in Auto.

`team/Auto-Fuel-High-Cycles-Average`: Average number of high-efficiency boiler fuel cycles per match the robot shot in Auto.

`team/Auto-Fuel-High-i-Position-Prob`: Proportion of the time the robot shoots into the high-efficiency boiler from inside the key in Auto.

`team/Auto-Fuel-High-o-Position-Prob`: Proportion of the time the robot shoots into the high-efficiency boiler from outside the key in Auto.

`team/Auto-Fuel-Intake-Hopper-Average`: Proportion of the time the robot intakes fuel from a hopper during Auto.

`team/Auto-Fuel-Low-Cycles-Average`: Average number of low efficiency-boiler fuel cycles per match the robot shot in Auto.

`team/Auto-Gear-Counts`: Lists the number of gears the robot scored in Auto each match.

`team/Auto-Gears-Achieve-Rate`: # success/matches the robot scored at least one gear in Auto.

`team/Auto-Gears-Dropped-Average`: Average number of gears per match the robot dropped in Auto.

`team/Auto-Gears-Intake-Ground-Average`: Average number of times per match the robot used gear ground intake in Auto.

`team/Auto-Gears-b-Position-Prob`: Proportion of times the robot attempts to score a gear on the boiler peg in Auto.

`team/Auto-Gears-b-Success-Rate`: # success/attempts the robot successfully scores a gear on the boiler peg in Auto.

`team/Auto-Gears-l-Position-Prob`: Proportion of times the robot attempts to score a gear on the loading station peg in Auto.

`team/Auto-Gears-l-Success-Rate`: # success/attempts the robot successfully scores a gear on the loading station peg in Auto.

`team/Auto-Gears-m-Position-Prob`: Proportion of times the robot attempts to score a gear on the center peg in Auto.

`team/Auto-Gears-m-Success-Rate`: # success/attempts the robot successfully scores a gear on the center peg in Auto.

`team/Auto-Robot-Broke-Down-Average`: Average proportion of times the robot broke down in Auto.

`team/Auto-Robot-No-Action-Average`: Average proportion of times the robot did no action in Auto.

`team/End-Defense-Average`: Average proportion of times the robot plays any defense in a match.

`team/End-Defense-Rating-Average`: Average scout rating of the robot's defensive ability from 1 to 5. A non-applicable match gets an automatic rating of -1.

`team/End-Driver-Rating-Average`: Average scout rating of the driver's driving ability from 1 to 5.

`team/End-Fuel-Ground-Intake-Average`: Average number of times per match the robot used fuel ground intake.

`team/End-Fuel-Ground-Intake-Rating-Average`: Average scout rating of the robot's fuel ground intake ability from 1 to 5. A non-applicable match gets an automatic rating of -1.

`team/End-Gear-Ground-Intake-Average`: Average number of the times per match the robot used gear ground intake.

`team/End-Gear-Ground-Intake-Rating-Average`: Average scout rating of the robot's gear ground intake ability from 1 to 5. A non-applicable match gets an automatic rating of -1.

`team/End-No-Show-Average`: Average number of times the robot did not show up to a match.

`team/End-Takeoff-Achieve-Rate`: # success/matches the robot completed takeoff.

`team/End-Takeoff-Speed-Average`: Average scout rating of the robot's takeoff speed from 1 to 5. A non-applicable match gets an automatic rating of -1.

`team/End-Takeoff-Success-Rate`: # success/attempts the robot completed takeoff.

`team/Loading-Station-Reliability`: Proportion of times the robot successfully collects a gear from the loading station.

`team/Reliability`: Proportion of times the robot is fully functional in a match

`team/Strategy-Rate-End-Defense`: Proportion of times the robot plays defense.

`team/Strategy-Rate-Tele-Fuel-High-Cycles`: Proportion of times at which the robot completes high-efficiency fuel cycles in Teleop.

`team/Strategy-Rate-Tele-Fuel-Low-Cycles`: Proportion of times at which the robot completes low-efficiency fuel cycles in Teleop.

`team/Strategy-Rate-Tele-Gears-Cycles`: Proportion of times at which the robot completes gear cycles in Teleop.

`team/Tele-Fuel-High-Cycles-Average`: Average number of high-efficiency fuel cycles per match the robot completes in Teleop.

`team/Tele-Fuel-High-Cycles-In-Key-Average`: Average number of high-efficiency fuel cycles per match the robot shoots from inside the key in Teleop.

`team/Tele-Fuel-High-Cycles-Out-Of-Key-Average`: Average number of high-efficiency fuel cycles per match the robot shoots from outside the key in Teleop.

`team/Tele-Fuel-High-Cycles-Times-Average`: Average time (in seconds) the robot takes to complete one high-efficiency fuel cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`team/Tele-Fuel-High-In-Key-Position-Prob`: Proportion of times the robot shoots fuel into the high-efficiency boiler from inside the key in Teleop.

`team/Tele-Fuel-High-Out-Of-Key-Position-Prob`: Proportion of times the robot shoots fuel into the high-efficiency boiler from outside the key in Teleop.

`team/Tele-Fuel-Intake-Loading-Station-Average`: Average number of times per match the robot collects fuel from the loading station in Teleop.

`team/Tele-Fuel-Low-Cycles-Average`: Average number of low-efficiency fuel cycles per match the robot shoots in Teleop.

`team/Tele-Fuel-Low-Cycles-Times-Average`: Average time (in seconds) the robot takes to complete one low-efficiency fuel cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`team/Tele-Gear-Counts`: Lists the number of gears the robot scored in Teleop each match.

`team/Tele-Gears-Boiler-Position-Prob`: Proportion of times the robot places a gear on the boiler peg in Teleop.

`team/Tele-Gears-Cycles-Average`: Average number of gear cycles per match the robot completes in Teleop.

`team/Tele-Gears-Cycles-Times-Average`: Average time (in seconds) the robot takes to complete one gear cycle in Teleop. Cycles begin and end upon scoring/failing to score.

`team/Tele-Gears-Cycles-Upper-Limit`: Maximum number of gears the robot has scored in one match in Teleop.

`team/Tele-Gears-Dropped-Average`: Average number of gears per match the robot drops while attempting to score in Teleop.

`team/Tele-Gears-Intake-Dropped-Average`: Average number of gears per match the robot drops while intaking in Teleop.

`team/Tele-Gears-Intake-Ground-Average`: Average number of times per match the robot uses gear ground intake in Teleop.

`team/Tele-Gears-Intake-Loading-Station-Average`: Average number of times per match the robot uses loading station gear intake in Teleop.

`team/Tele-Gears-Loading-Position-Prob`: Proportion of times the robot places a gear on the loading station peg in Teleop.

`team/Tele-Gears-Middle-Position-Prob`: Proportion of times the robot places a gear on the center peg in Teleop.

`team/Tele-Gears-Position-Boiler-Average`: Average number of gears per match the robot places on the boiler peg in Teleop.

`team/Tele-Gears-Position-Loading-Average`: Average number of gears per match the robot places on the loading station peg in Teleop.

`team/Tele-Gears-Position-Middle-Average`: Average number of gears the robot places on the center peg in Teleop.

`team/Tele-Robot-Broke-Down-Average`: Average number of times the robot breaks down in Teleop.

`team/Tele-Robot-No-Action-Average`: Average number of times the robot does nothing in Teleop.

Sykes Data Descriptions: Statistics taken from FRC 4536 Caleb Sykes' database on Chief Delphi. More information can be found here.

`team/Sykes-4-Rotor-RP-Achieved`: Calculated contribution for getting an extra RP for the fourth rotor.

`team/Sykes-Auto-Fuel-High`: Calculated contribution for the number of kPa a robot shot into the high-efficiency boiler in Auto.

`team/Sykes-Rotor-1-Auto`: Calculated contribution for the first rotor in autonomous.

`team/Sykes-Rotor-3-Engaged`: Calculated contribution for the third rotor during teleop.

`team/Sykes-Rotor-4-Engaged`: Calculated contribution for the fourth rotor during teleop.

`team/Sykes-Teleop-Fuel-High`: Calculated contribution for points scored from high efficiency goals

`team/Sykes-Total-Points`: Calculated contribution for total number of points scored in a match (OPR).

`team/Sykes-Total-Scored-Gears`: Calculated contribution for number of gears scored in a match.

`team/Sykes-Winning-Margin-Elo`: Elo value calculated using historic data on winning margins.

`team/Sykes-kPa-Added`: Calculated contribution of kPa added.

`team/Sykes-kPa-Bonus-Achieved`: Calculated contribution for achieving the kPa bonus.

IV. Creating a Picklist

Common Misconception from This Year: One size fits all.

This year, there was no robot that would be the best pick for every alliance. We found that the best way to handle alliance selection was to create a list of robots by all the top factors, and then assess what alliance we would be against to try and more accurately pick our robot. We created our list by using the app to sort by gear count, and then filtering out for various issues we found. For example, a team would be removed from the list if they had below a certain climb threshold (at CMP, we made sure to only have robots with 8 or more climbs out of 10). At the Silicon Valley Regional, this did not come into play; however, this was absolutely vital to forming our alliance at both the Ventura regional and at the Houston Championships.

Before each alliance selection, we familiarized ourselves with the data on all the captain teams, as well as the data about the teams that would likely be first picked. Although we were not captains, we immediately started thinking about the strategy we would face as soon as we saw the teams we were against. This allowed us to advise our captains on the third robot on our alliance to counter that strategy. The first example of this was at the Ventura Regional (week 3), where we were the first pick of the #3 seed alliance. Especially at the beginning of the season, the “snake” draft favored the 6-8th seeded alliances, so we knew we could be up against a 4 rotor alliance. After the #6 alliance finished their second pick, we knew that the field would not be deep enough to create an alliance that could “out gear” the opposing alliance. Because of this, we picked a team that had a strong drivetrain, knew how to play defense, and could shoot some fuel in auto to break a tie. The second instance of this was at the Houston Championships, where we were the first pick for the 5th seed alliance. We knew at the start that both us and our alliance captain could score a gear in auto, and because we were up against one of the best shooting robots, we knew that we would need a two rotor autonomous to have a chance. Because of this, we picked a team that would be able to help us complete the 2 rotor autonomous and allowed us to make it past quarterfinals.