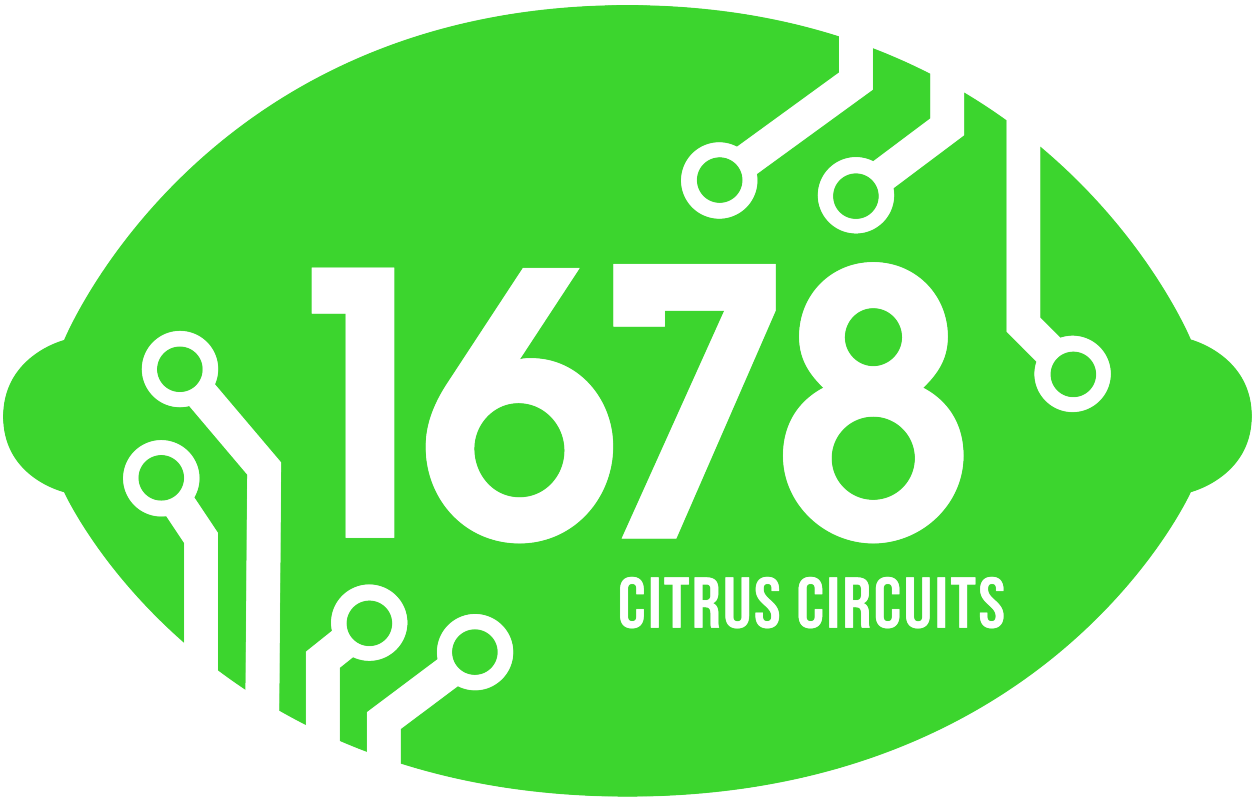

2019 Citrus Circuits
Electronic Scouting System

1678 Software Scouting Team

Ben Bobell, Carl Csaposs, Carter Luck, David Carlip, Ellie Kim, Emily Lin, Ethan Talbert, Helena Young, Jake Roggenbuck, Jude Manansala, Justin Yeung, Kate Unger, Ludi Wang, McKenna Lincoln, Nathan Solomon, Ray Fabionar, Teo Honda-Scully

November 1, 2019

Contents

Chapter 1

Introduction

The Scouting Whitepaper is a technical document that describes FRC Team 1678: Citrus Circuits' electronic scouting system. To scout, we use electronic devices to collect, process, and visualize data. These devices are operated by trained team members, whom we call scouts, to record statistics about robots at the competitions we attend.

The purpose of the Scouting Whitepaper is to publicly provide documentation on the development of our scouting system, which we invite other FIRST Robotics Competition (FRC) teams to replicate.

Scouting aids us in achieving our goal of winning an FRC competition by providing insights about every team at the competition. This allows us to formulate match strategies and create a picklist that best suits our strategic needs. Even if a team is not the alliance captain, having a picklist is useful for choosing a second pick robot.

Citrus Circuits scouts electronically for more accurate and efficient data collection. Electronic scouting also allows for more complex calculations, enabling a wider range of data visualization. Most importantly, it provides real time communication between collection, processing, and visualization.

To fully understand our system, it is important to know the 2019 FRC game, Destination: Deep Space, as our scouting system adapts to the current year's game. Information about the 2019 game can be found here: `https://www.firstinspires.org/resource-library/frc/archived-game-documentation`

Chapter 2

Technical Details

2.1 Overview

Our scouting system is comprised of three components: collection, processing, and visualization. There are three primary mobile applications that are responsible for data collection: the Scout, Super Scout, and Pit Scout apps. They collect objective and subjective data for matches, alliances, and individual robots. The collected data is transferred to our database, which is separated into two classes of data: raw, non-calculated data and calculated data. The server uses the raw data to compute the calculated data. Both raw and calculated data are visualized on Android and iOS mobile applications, as well as on the Picklist Spreadsheet. The Picklist Spreadsheet is used for the creation of our picklist, and the mobile applications are used to develop match strategies and modify our picklist. All components of our scouting system work together to provide us with valuable information to help us form effective match strategies and to aid in our picklist creation.

2.2 Platforms and Languages

Platform	**Language**	**Use**
–	Python	Server - runs calculations
Android	Java	Scout, Super Scout, Android Viewer app
iOS	Swift	iOS Viewer app
Ionic	Javascript, HTML	Pit Scout web app
Flutter	Dart	QR Scan app (data transfer for Android and iOS)
Google Sheets	Google App Script	Picklist Spreadsheet
Firebase	–	Cloud storage database - stores raw and calculated data
SmugMug	–	Cloud photo database - stores pictures of robots

2.3 Data Transfer

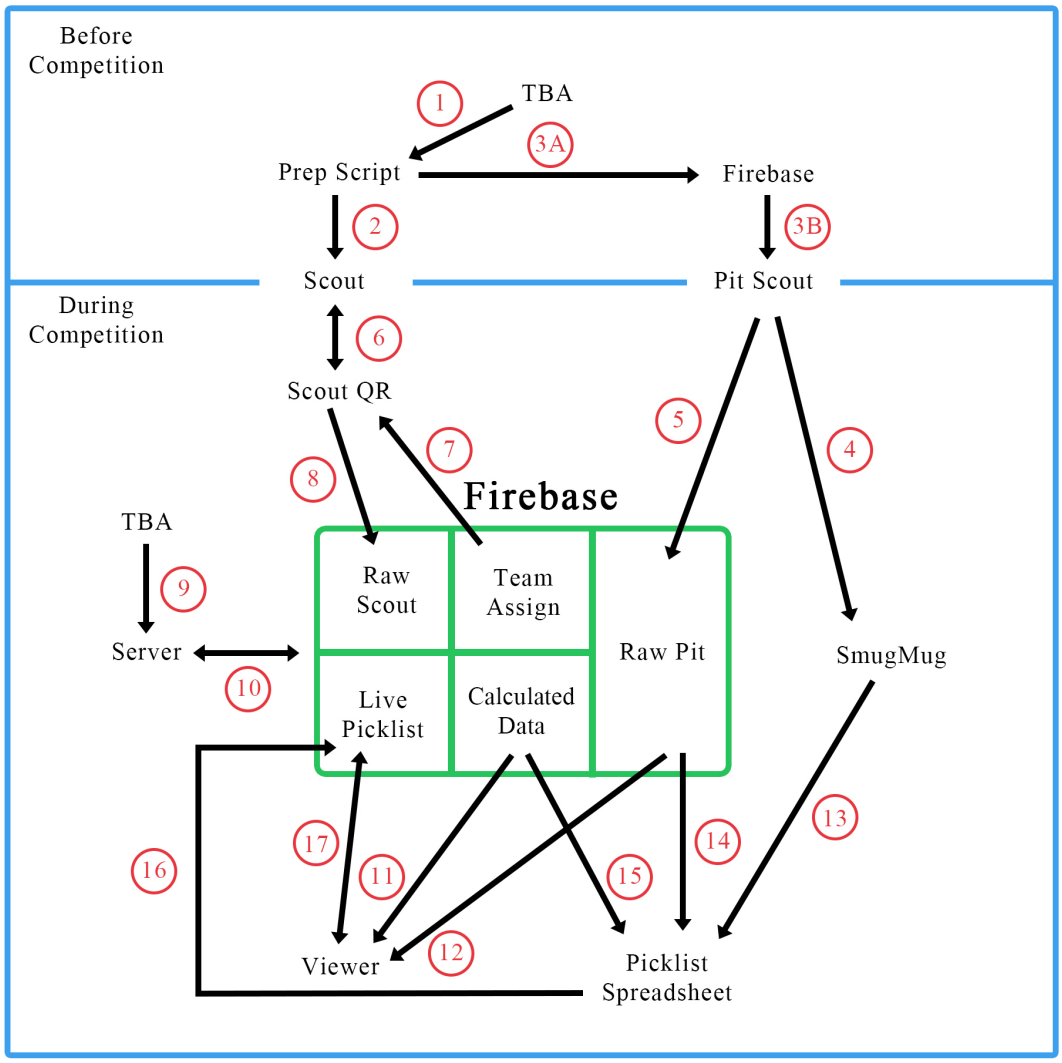

Figure 2.1: Diagram of data transfer in our scouting system

Before Competition:

1. The Server retrieves the match schedule and list of teams (team name and team number) from The Blue Alliance Read APIv3.
2. The Server sends the match schedule to the Scout app using Android Debug Bridge over USB. The Scout app uses the match schedule for team assignment.
3. The Server sends the team information to the Pit Scout app via Firebase.

During Competition:

4. The pit scout captures photos and uploads them to SmugMug, an online database for photos.
5. The Pit Scout app sends collected data to Firebase.
6. To communicate with Firebase, the Scout app leverages the data connection of a scout's personal phone with the Scout QR app.

(a) Scout QR app:
The Scout QR app leverages the data connection on a scout's personal phone to transfer data between the Server (via Firebase) and Scout app.

Figure 2.2: QR Screen

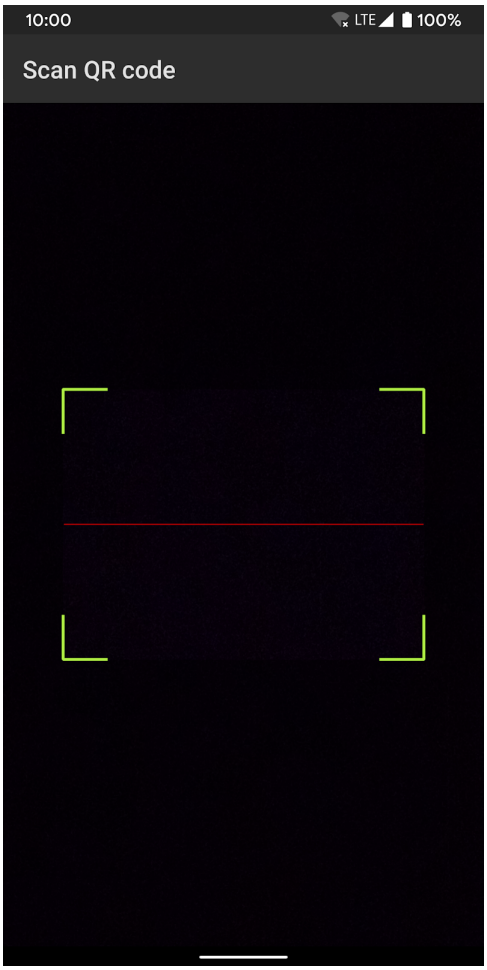

Figure 2.3: Scan Screen

Figure 2.2 displays a QR code used to transfer team assignment data from the Server to the Scout app (see #7). Pressing the Scan button (camera icon) in the lower right corner opens the screen in Figure 2.3, which scans QR codes generated by the Scout app and sends them to the Server (see #8).

(b) Compression:
A QR code can store a maximum of 1273 binary characters at 30% error correction. (source: `https://www.qrcode.com/en/about/version.html`) To maximize error correction level and minimize QR code size, the Scout app compresses JSON data for a match before displaying the data in a QR code. During compression, whitespace and unnecessary separators are removed, and known strings (e.g. data field names) are replaced with a single character.

7. Via the Scout QR app, the Scout app receives team assignment data from Firebase.
8. Via the Scout QR app, the Scout app sends the data for a match to Firebase.
9. The Server retrieves and caches team seeding and match results from The Blue Alliance Read APIv3.
10. The Server retrieves and caches data from the "Raw Scout" section of Firebase and sends data to the "Team Assignment" and "Calculated Data" sections of Firebase.
11. The Viewer app retrieves Match data, Team data, Team in Match data, and seeding (predicted and actual) from Firebase.

12. The Viewer app retrieves pit data from Firebase.
13. The Picklist Spreadsheet retrieves photos from SmugMug.
14. The Picklist Spreadsheet retrieves pit data from Firebase.
15. The Picklist Spreadsheet retrieves Team data and Team in Match data from Firebase.
    (a) Before picklist creation, data from the competition is exported into two separate CSV files. One CSV file has each team with its respective team data (ex: averages, pick abilities). The other CSV file has each team in each match and their respective TIMD (Team In Match Data). These CSV files are imported into the Picklist Spreadsheet.
16. After the creation of our initial picklist, the Picklist Spreadsheet sends the picklist to Firebase, to be edited in the Viewer during the second day of qualification matches.
17. During the second day of qualification matches, the Viewer app sends and receives updates to the “Live picklist” section on Firebase.

2.4 Collection

2.4.1 Scout

The Scout is an objective match data collection app used by scouts. It uses a map of the field for data entry, and QR codes for simple and reliable data transfer.

The Scout app is comprised of four primary screens: the Match Data Screen, the Pregame Screen, the Map Screen, and the Data Check Screen.

2.4.1.1 Match Data Screen

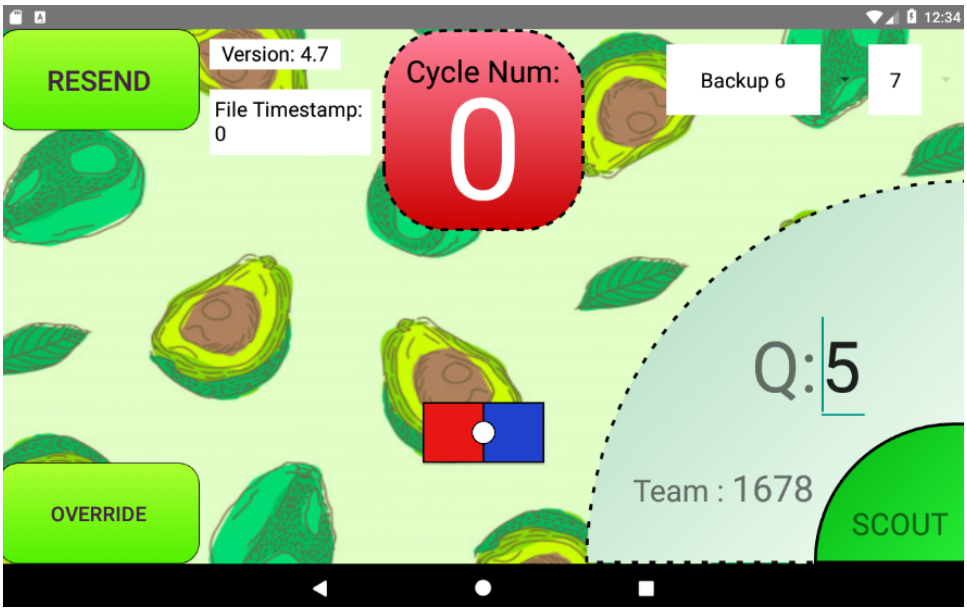

Figure 2.4: Match Data Screen

- Scout identification (each active tablet has a unique ID number between 1 and 18)
- Scout name (the inputted name of the scout)
- Qualification match number (which can be manually overridden by the scout, updating the robot assignment)
- Assigned robot’s team number

- Assigned robot's alliance color

It also allows the user to:

- Set the field orientation from their perspective to be used on the Map Screen
- Resend collected data from previous matches
- Select their robot assignment method out of three options: QR, File, and Override

Our primary assignment system is the QR system. This allows us to rotate our scouts based on their data accuracy, so that more accurate scouts are paired with less accurate scouts. If the QR system does not function properly, we use the File system as a backup, and if that also fails, we use Override as a final resort.

- QR:
  - Allows the user to scan the QR displayed on our mobile QR Scan app
  - Varies scout distribution based on each scout's scout precision rank (SPR)
    - Distributing scouts based on SPR allows for an even distribution of scouts with high and low accuracies assigned to each robot.
- File:
  - Used when QR scanning does not function properly
  - Distributes scouts based on their scout ID using a predefined algorithm
  - Shuffles the assignments every match
    - This prevents the same scouts from always being assigned to the same robot, which allows for more accurate SPRs.
- Override:
  - Used when QR and File do not work
  - Allows scouts to manually input the robot team number and alliance color

2.4.1.2 Pregame Screen

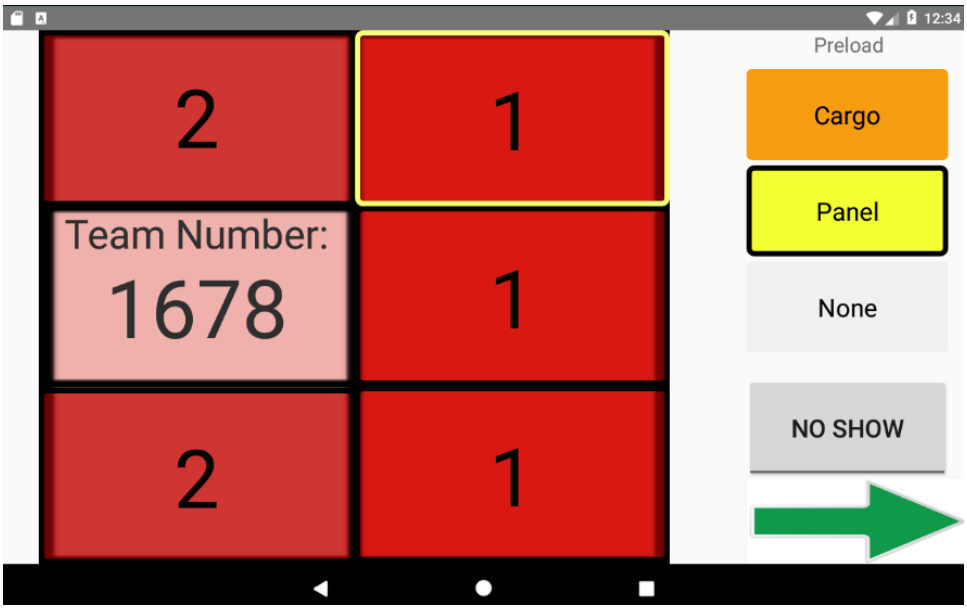

Figure 2.5: Pregame Screen

After scouts ensure the Match Data Screen is displaying the proper match and user data, they are brought to the Pregame Screen, which must be filled out before the match begins. On the Pregame Screen, users input:

- The robot's starting position on the visual resembling the Hab platform (oriented based on previously set field orientation and robot's alliance color)
- Preloaded game element in the robot (cargo or hatch panel)
  - This input determines the map variation that will first be displayed in the following Map Screen (intake mode, placement mode cargo, or placement mode hatch panel).
- Whether or not their robot showed up to the match

2.4.1.3 Map Screen

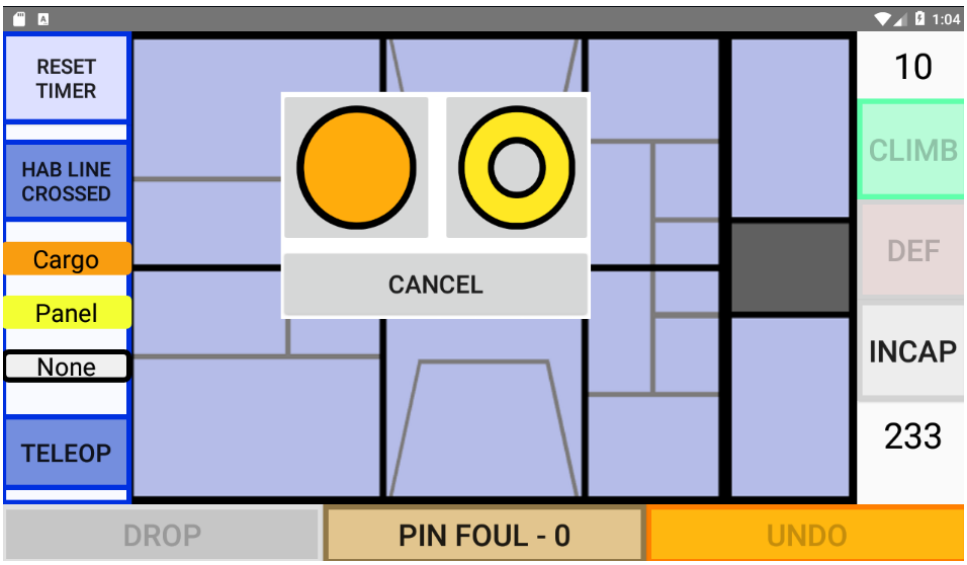

Figure 2.6: Map Screen (sandstorm bar on left and intake mode with game element popup)

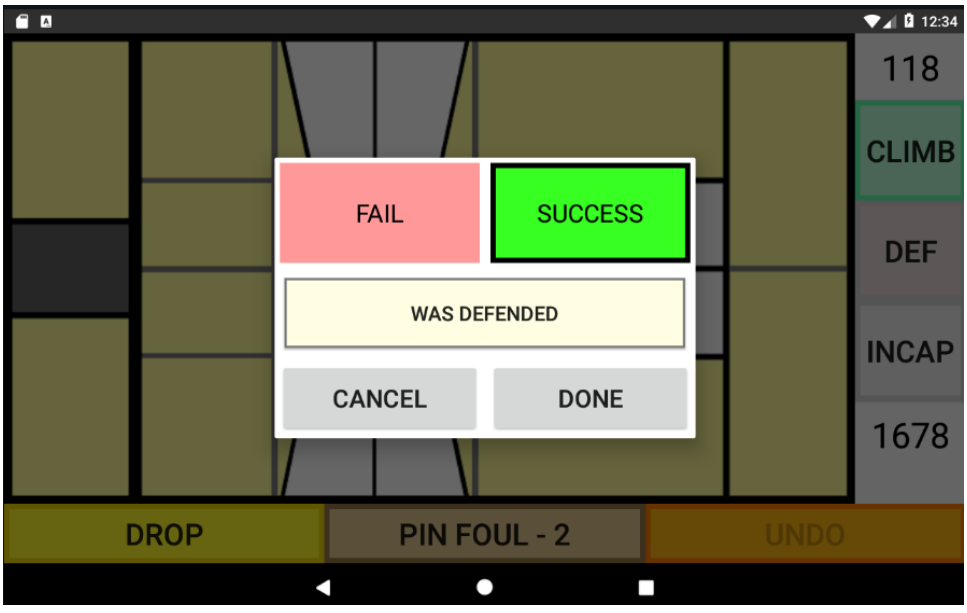

Figure 2.7: Map Screen (cargo ship hatch panel placement dialog)

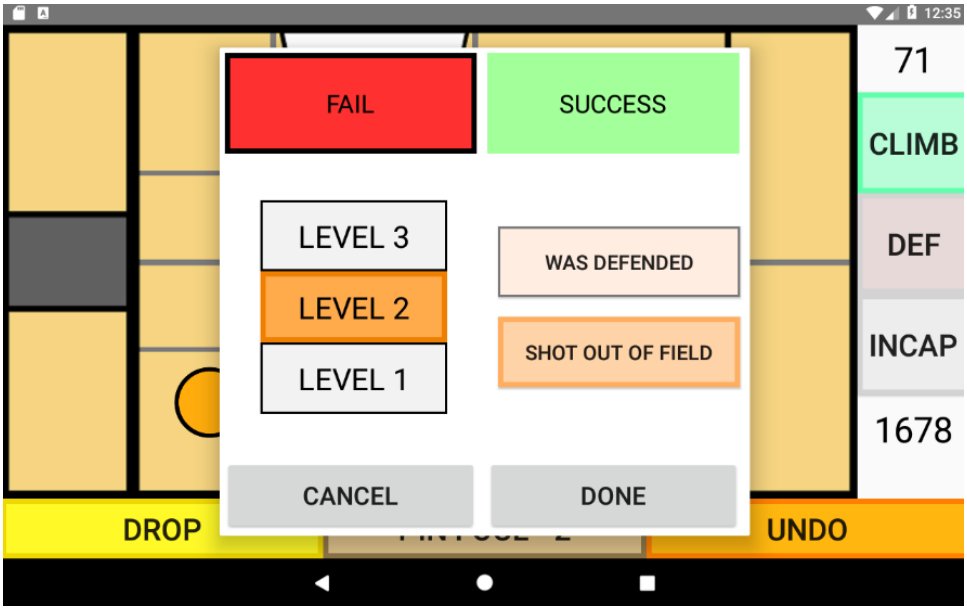

Figure 2.8: Map Screen (rocket cargo placement dialog)

Scouts are then brought to the Map Screen with the Sandstorm section visible. Users whose robot did not show up to the match are taken directly to the Data Check Screen (Section 2.4.1.4). The Map Screen is comprised of:

- A map that primarily displays the assigned alliance's side of the field, with only a small section delegated to the opponent's side of the field for recording defense. The map is split into eight zones, in which robots can intake game objects, and sections for the rockets and cargo ship, where game objects can be placed.
  - Intake popups and placement dialogs appear when the intake/placement zones on the map are pressed. The popups and dialogs collect further information on the respective intake/placement.
- A Sandstorm section that hides the opponent's side of the field and allows the user to collect Sandstorm-specific data
- Various buttons surrounding the map
- A background timer recording every action with a timestamp
  - The timer is manually activated by the user at the start of the match.
  - This information is used during Consolidation (Section 2.5.1) and in calculations such as intake-placement cycle times and when the robot went to climb.

2.4.1.4 Data Check Screen

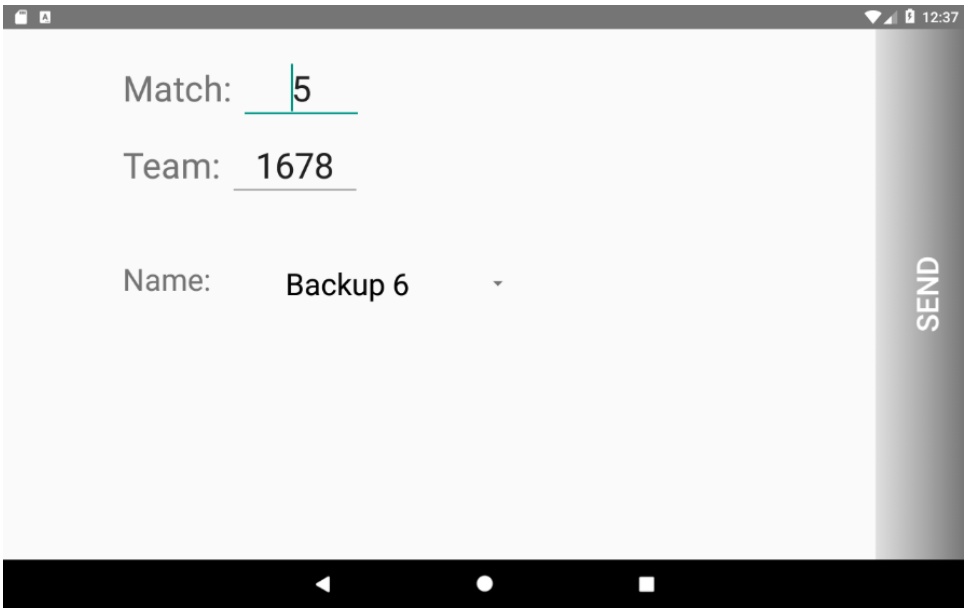

Figure 2.9: Data Check Screen

Once the time reaches zero out and the match is over, an arrow appears allowing the user to continue to the next screen, the Data Check Screen, in which they can:

- Edit their scout name
- Edit the match number
- Edit the scouted robot's team number

After ensuring all the information displayed on the Data Check Screen is correct, the user is brought to a QR code containing all the compressed, collected data from that match. The scout scans the QR code to send the collected data to the database, and returns to the Match Data Screen (Section 2.4.1.1).

2.4.2 Super Scout

The Super Scout app collects subjective match data and is used by super scouts. It uses a ordinal driving ranking system, a cardinal defensive ranking system, and a QR system for simple and reliable data transfer. The purpose of the Super Scout app is to collect subjective data to measure defensive, counter-defensive, and driving abilities of each robot at a competition. What differentiates the Super Scout app and the Scout app is the type of data collected. The Scout app collects objective data while the Super Scout app collects subjective data—data that requires human input and intuition to record.

There are two super scouts assigned to each alliance. One super scout documents every defensive and counter-defensive action between robots throughout a match, and the other inputs subjective data about driver ability at the end of each match.

The app has three main screens: the Pre-Match Screen, the Teleoperated-Defensive Ranking Screen, and the Driver Ability Screen.

2.4.2.1 Pre-Match Screen

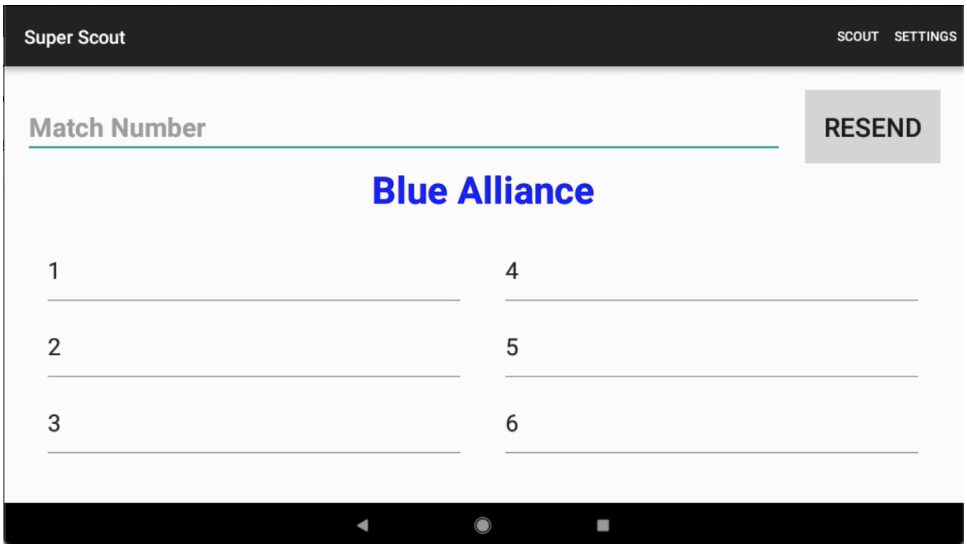

Figure 2.10: Pre-Match Screen

- Displays which alliance the super scout will be scouting and each robot on the field.
- Has a manual override feature in case the automatic file assignment system fails.

2.4.2.2 Teleoperated-Defensive Ranking Screen

Figure 2.11: Teleoperated-Defensive Ranking Screen

Figure 2.12: Pushing battle dialog menu

- Used by the super scouts to rank the defensive and counter-defensive abilities of each robot on the field (defensive for their assigned alliance and counter-defensive for the opposing alliance).
- The super scout also records any pushing interactions between robots on their alliance - whether or not they were able to out-perform an opposing robot in a pushing battle.

2.4.2.3 Driver Ability Screen

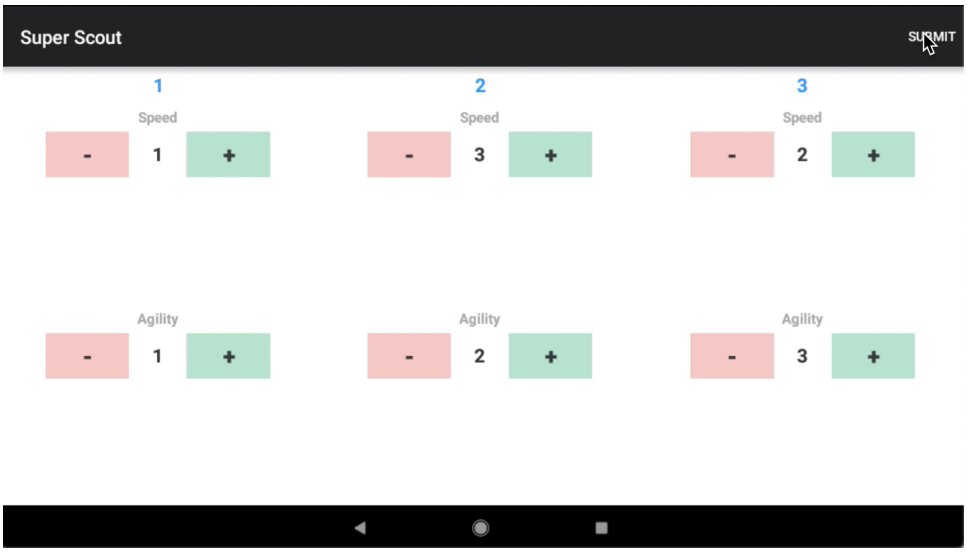

Figure 2.13: Driver Ability Screen

- At the end of the match, super scouts input the speed and agility of each team on their alliance for use in calculating each robot's driver ability.

2.4.3 Pit Scout

The Pit Scout app is a objective and subjective data collection app used to collect physical properties of robots in the pit area of an FRC competition.

During the practice and qualification matches, we record photos and objective data by visiting each team in the pits and asking them questions about their robot. In the 2019 FRC game, we collected subjective data by testing how well teams could drive onto our triple-climb ramps, and ranking them on a cardinal scale.

The Pit Scout app has two main screens: the Teams List Screen and the Team Info Screen.

2.4.3.1 Teams List Screen

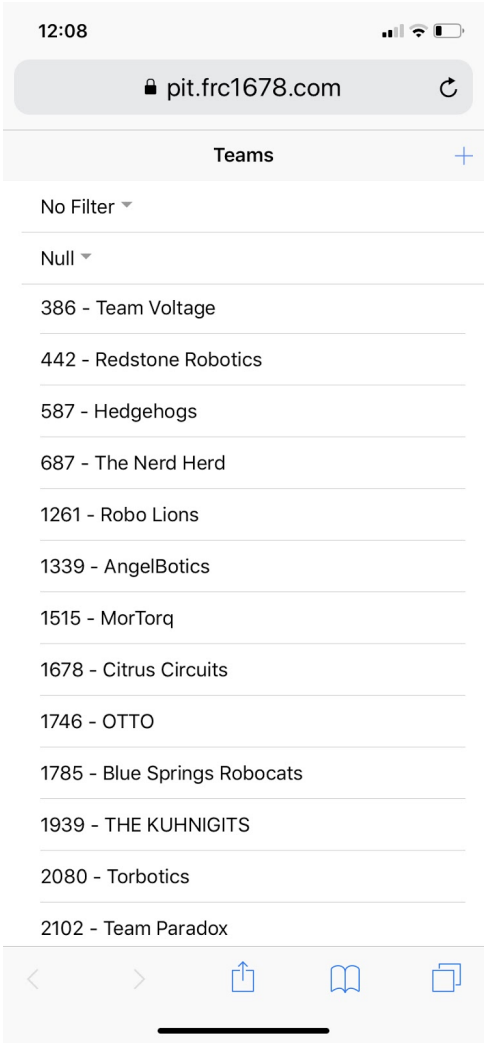

Figure 2.14: Teams List Screen

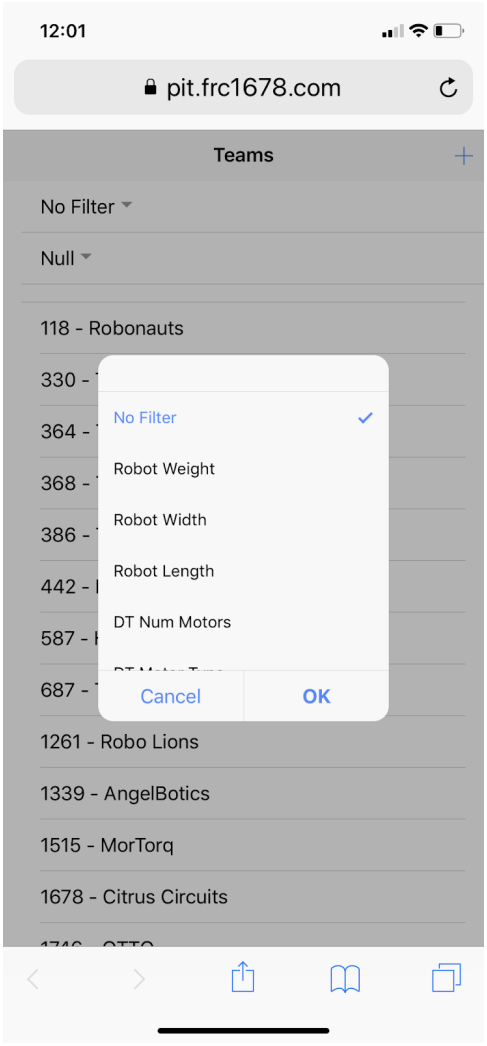

Figure 2.15: Filter dialog menu

The Teams List Screen consists of a list of every team at the competition, as well as a filter and an “add team” feature.

- Team List: List of teams at the competition, each linked to their respective Team Info Screen
- Filter: Used to filter teams based on a value for a specific data field
- Add Team: Used if a team is missing from The Blue Alliance

2.4.3.2 Team Info Screen

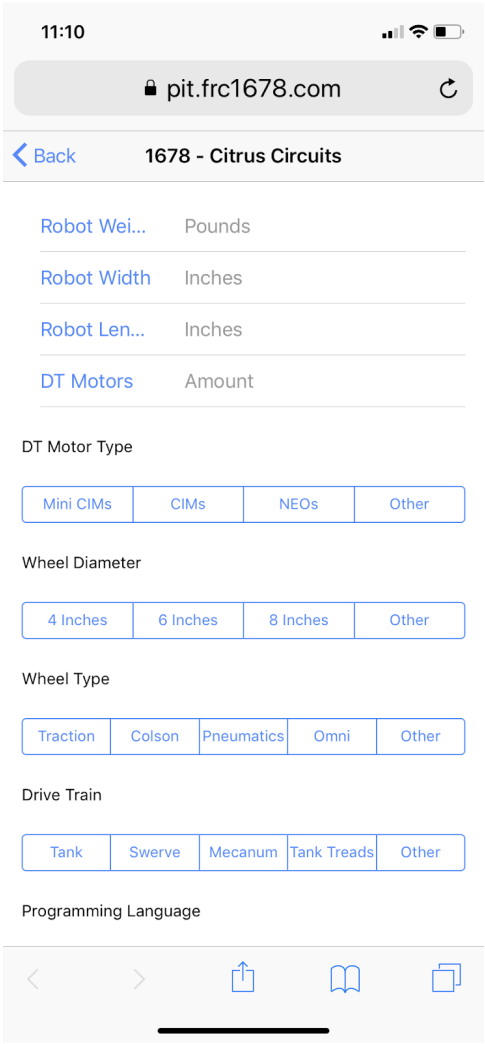

Figure 2.16: Team Info Screen

The Team Info Screen uses text fields, segmented controllers, and switches to record data.

Unlike the Scout and Super Scout apps, the pit data collected from each team does not change every match. The physical properties of a robot, such as the dimensions and weight, are constant throughout a competition, and can be measured before qualification matches begin.

2.5 Processing

2.5.1 Computations

2.5.1.1 Pick Ability

Strategists begin developing our picklists by using computer-generated lists of teams, and editing them after the first day of qualification matches. For each robot, the server calculates two values to summarize how much they would contribute to our elimination-round alliance (not any alliance): first and second pick ability (which are used to create first drafts of our first pick and second picklists, respectively). Each ability is a weighted average of a team's offensive and defensive capabilities. The first and second pick calculations use different weights since we look for different traits in a first and second pick. Pick ability roughly represents how much of a point swing a team will contribute to our alliance—the more, the better.

Throughout the season (even during competitions), we adjust the weights used to calculate pick ability.

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_abilities.py`

2.5.1.2 Points Prevented

Points prevented is a computation attempting to objectively quantify a robot's defensive ability in a single match. Instead of judging defensive robots based on subjective observations, points prevented is calculated based on objective metrics. These metrics are the amount of the opponent's cycles stopped by the defending team and the amount that the opponent team's average cycle time is slowed down. Every scoring cycle is timestamped and marked as defended or undefended. Additionally, the time is recorded whenever a robot crosses the midfield line. Since the 2019 FRC game only allows one defender on the other side of the field at a time, we can determine the cycles which a robot defended in a match. The calculation for points prevented is as follows:

$$pointsPrevented = cargoCyclesPrevented \times 3 + panelCyclesPrevented \times 2$$

$$cyclesPrevented = defenseDrops + defenseCyclesLost$$

$$defenseDrops = (defenseDropRate_{match} - freeDropRate_{overall}) * defenseCyclesDropped_{match}$$

$$defenseCyclesLost = (defenceCycleTime_{match} - freeCycleTime_{overall}) \times (defenseCycles / freeCycleTime_{overall})$$

- $pointsPrevented$ is the number of points a robot doesn't score due to defense in a match
- $cargoCyclesPrevented$ is the number of cargo cycles a robot doesn't complete due to defense in a match
- $panelCyclesPrevented$ is the number of panel cycles a robot doesn't complete due to defense in a match
- $cyclesPrevented$ is a generic term for either cargo or panel cycles prevented
- $defenseDrops$ is the number of dropped pieces caused by defense in a match
- $defenseCyclesLost$ is the number of cycles a robot doesn't complete due to defense based on time in a match
- $defenseDropRate_{match}$is the percentage of cycles a robot drops under defense in a match
- $freeDropRate_{overall}$is the percentage of cycles a robot drops without defense throughout a competition
- $defenseCyclesDropped_{match}$ is the number of cycles a robot doesn't complete due to defense in a match
- $defenseCycleTime_{match}$is the time it takes a robot to complete a cycle under defense, on average, throughout a match
- $freeCycleTime_{overall}$ is the time it takes a robot to complete a cycle, on average, throughout a competition
- $defenseCycles$ is the number of cycles a robot completes under defense in a match

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_defense.py`

2.5.1.3 Scout Precision Ranking (SPR)

The server measures each scout's accuracy by comparing that scout to the other scouts that collect the same data. (Since there are 18 scouts per match, we have 3 sets of data for the same robot in a match.) This accuracy is represented in a single number called the Scout Precision Ranking (SPR). SPR serves two purposes: breaking ties in the consolidation process (Section 2.5.1.5) and assigning scouts to robots.

For each robot in each match, a scout is evaluated to agree or disagree with the majority of the three scouts for each of the following categories: number of game piece intakes, number of game piece placements, number of game piece drops, number of pinning fouls caused, climb, robot incapacitation, and number of center line crosses. A scout's SPR is the percentage of their category evaluations that agree with the majority.

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_sprs.py`

2.5.1.4 Normalized Driver Ability

Normalized driver ability is a calculation used for determining how well a team can drive, which helps determine how well they can play defense. For each match, super scouts ordinally rank each alliance member's speed and agility from 1 (worst) to 3 (best).

$$driverAbility = agilityZScore * agilityWeight + speedZScore * speedWeight$$

Where "Z-score" is the number of standard deviations a data field is above the mean.

Then, each team's driver ability is scaled based on the driver ability of all of the other teams who have been on their alliance. To do this, the driver ability of each other team that the team has played with is scaled between 0 and 1 based on the highest and lowest driver abilities in the competition.

$$scaledDriverAbility_{team} = (driverAbility_{team} - min(DA))/(max(DA) - min(DA))$$

Where DA is the set of driver abilities for each team in the competition.

Then, each team's driver ability is normalized by multiplying the team's driverAbility by the average scaledDriverAbility for every team they have been on an alliance with at the competition:

$$normalizedDriverAbility_{team} = avg(scaledDriverAbility) * driverAbility_{team}$$

This prevents teams from being ranked lower or higher because of stronger or weaker than average alliance partners.

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_abilities.py`

2.5.1.5 Consolidation

In order to have the most accurate data possible, our scouting system has three people scout every robot in each match. This redundancy ensures accurate data even if one scout makes a mistake. The raw data from a single scout is called a temporary Team in Match Data (tempTIMD). 3 tempTIMDs are consolidated into one Team in Match Data (TIMD), which data we use to represent what a robot did in a match. In order to consolidate the three tempTIMDs into one TIMD, the tempTIMDs are first separated into their two component parts: timed data and non-timed data.

Consolidating non-timed data is fairly straightforward. When we have three scouts, we simply go with majority rule. If all three scouts disagree, we go with whichever scout has the highest SPR. In cases where only two tempTIMDs are provided, we go with whichever scout has the highest SPR. If only one tempTIMD is provided, there is no need to consolidate data at all.

Consolidating timed actions is more difficult. First, we determine the correct number count of each type of action (e.g. intakes, placements, drops, etc.) by majority rule (using SPR as a tiebreaker). If the tempTIMDs don't agree on the item count, we line up actions from each scout's timeline (list of timestamped actions, in sequential order) to minimize time differences – we do this by lining up the closest actions (by timestamp), then the next closest, and so on until we have the desired number of actions. This process requires removing extraneous actions in a tempTIMD after lining up the actions, and requires handling for missing actions in a tempTIMD.

All data for one action can be consolidated by majority rule, with SPR as a tiebreaker. To consolidate the timestamp of a timeline action, server calculates a weighted average of the times from each tempTIMD, where the weight is the reciprocal square Z-score for each time:

$$actualTime = \Sigma_{t \in T} t * (t - \bar{T}) - 2)/(\Sigma_{t \in T}(t - avg(T))^{-2}$$

where T is the set of times given for each tempTIMD

Code: `https://github.com/frc1678/server-2019/blob/master/consolidation.py`

2.5.1.6 Elo Pushing Ability

Elo pushing ability is a measure of how good a robot is at pushing other robots. We use the Elo rating system because our only measure of a robot's pushing ability comes from who wins and loses in pushing battles (teams are only compared against other teams). The Elo rating system is designed to measure players' abilities in zero-sum games (such as one-on-one competitions).

Robots start with 500 Elo. Next, the server goes through all the pushing battles in order, and updates robots' Elo ratings after each pushing battle. The amount a robot's Elo changes is given by the following equation:

$$\Delta E = K * winValue * [1/(1 + 10^{-(E_w - E_l)/C})]$$

- ΔE is how much Elo the winner gains, and how much the loser loses
- E_l is how much Elo the loser of the pushing battle had before losing, and E_w is how much the winner had before the pushing battle
- K is a constant equal to 100, which represents the maximum amount of Elo a robot can lose or gain from a single pushing battle
- C is a constant equal to 160. If robot A has 160 more Elo than robot B, robot A is estimated to win 10 out of 11 matches against robot B
- $winValue$ is 1 if super scouts report a "big win", and 0.6 if they report a "small win".

The part within [square brackets] is the winner's predicted win chance. Thus, a team's win chance is a logistic function of how much more Elo they have than their opponent.

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_pushing_ability.py`

2.5.1.7 Predictions

As we collect data about teams in competition, we are able to predict the scores of matches fairly accurately. These predicted scores are used to determine which matches will be more difficult for us, and, therefore, the matches we should invest more time into strategizing for. Predicted scores are also used when calculating the predicted rankings of teams (see below). The predicted score of an alliance is calculated using the following expression:

$$predictedSoloPoints_{team1} + predictedSoloPoints_{team2} + predictedSoloPoints_{team3} + predictedAllianceClimbPoints$$

Predicted scores are also used to calculate predicted Ranking Points (RPs). Predicted RPs are calculated for each alliance in each match. The predicted RPs for a team is the sum of the predicted RPs for each of their alliances in their matches. Predicted RPs for teams are ranked to create predicted seeds. The calculation for predicted RPs considers predicted scores (a win is worth 2 RPs), and bonus RPs. The predicted 15-point climb bonus RP is calculated based on each of the teams' previous climbing performance. The predicted complete rocket bonus RP is calculated based on the scoring abilities of the best two teams on an alliance.

Given the standard deviation of a data field, we can use a cumulative distribution function[1] to find the probability that a value (x) in that data field exceeds another value (μ). For example, if x is the average number of hatch panels a team can score in a match and μ is 6, we can find the probability (P) that a team will score enough hatch panels to get a rocket RP. We do the same for number of cargos scored per match, so then we get

$$P(rocketRP) = P(enoughCargo) * P(enoughPanels)$$

To calculate an alliance's chance of getting the climb RP, we use each team's climb success rates (calculated separately for each level) to find the chance that one robot will climb to level 3 and another will climb to level 1, or that two robots will climb to level 2 and one will go to level 1 (these are the 2 possible combinations for an alliance to score at least 15 points in climbs).

Using predicted score, climb RP chance, and rocket RP chance, we can predict each team's seeding at the end of competition. Predicted seeding allows us to predict the top-seeded alliances in the alliance selection process, and tailor our picklist around those alliances. Additionally, predicted seeding helps us with our later qualification match strategy to determine how many RPs we need to seed in the highest position possible (to give our team the best

[1] `https://en.wikipedia.org/wiki/Cumulative_distribution_function`

picking position).

Code: `https://github.com/frc1678/server-2019/blob/master/calculate_predictions.py`

2.5.2 Monitoring

During competitions, the server runs on a computer's terminal, which facilitates the monitoring of the program. Running the server on terminal means that developers can see whenever an error occurs in the program, or if calculations aren't working properly.

Along with looking at the terminal to check if there are any errors in the server, developers periodically check the database to ensure all server calculations are accurate and do not need to be fixed.

After making sure that all the data is accurate, the outputs of the weighted calculations such as first and second pick abilities are examined. If needed, the weights for these calculations are adjusted at the strategists' discretion.

2.6 Visualization

2.6.1 Viewer

The Viewer app (written separately for Android and iOS) allows strategists to access scouting data on a smartphone. The Viewer app is used to create our team's qualification match strategy and to modify our picklist.

2.6.1.1 Match Schedule Screen

Figure 2.17: Match Schedule displaying match score and ranking points (iOS)

The match schedule page displays the competition's qualification match schedule, the teams in each match by alliance color, and the score and ranking points each alliance earned in a match, or is predicted to earn in a match.

2.6.1.2 Seeding Screen

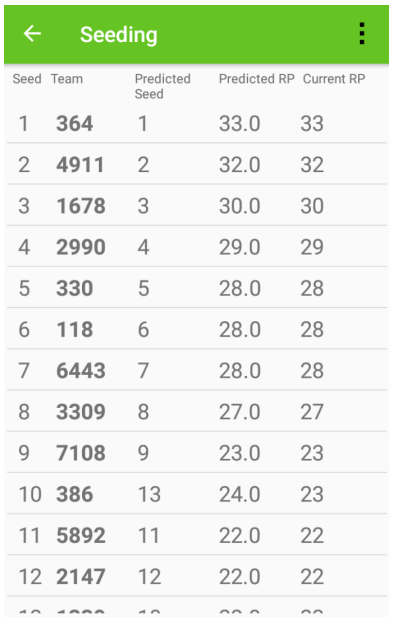

Figure 2.18: Seeding page (Android)

The seeding page shows the current ranking of teams by ranking points.

2.6.1.3 Team Details Screen

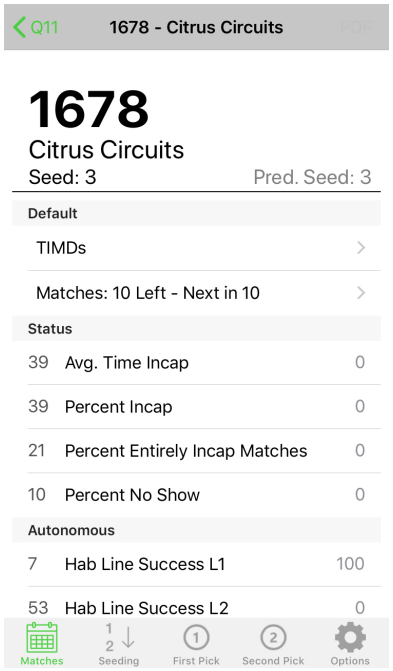

Figure 2.19: Team Details for 1678 (iOS)

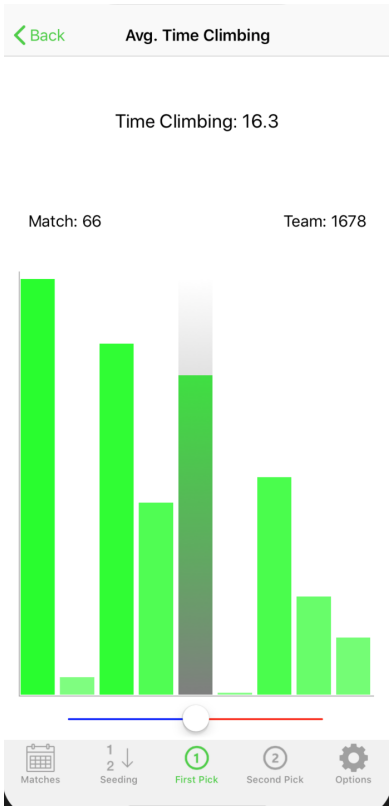

Figure 2.20: Graph of Average Time Climbing for team 1678 showing the values from each match. When a bar is tapped, the match number and value are displayed (iOS)

Each team details page contains overall information about the team. Data is separated into categories (such as Sandstorm or Teleoperated). Most calculated data (including averages, maxes, and percentiles) will, if pressed, take the user to a graph of that data field across a team's matches. For example, the calculated datapoint avgCargoScored would show a graph with the number of cargoScored for each of the team's matches.

2.6.1.4 Match Details Screen

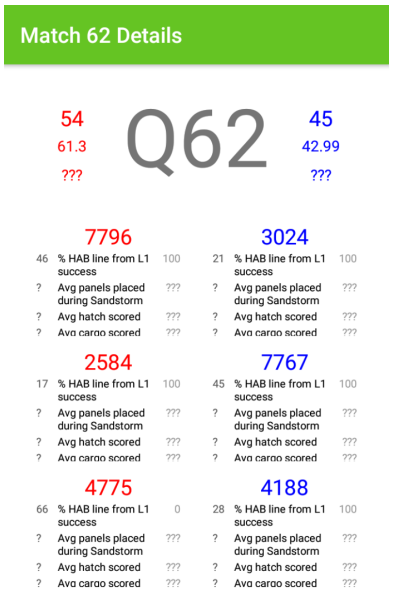

Figure 2.21: Match Details (Android)

Users open the match details page by tapping on a match from the match schedule page. The page displays the score of the match, the predicted score, the chance of each alliance winning, as well as any other user-selected data fields. The Match Details page is primarily used to create the strategy for a qualification match.

2.6.1.5 Live Picklist Screen

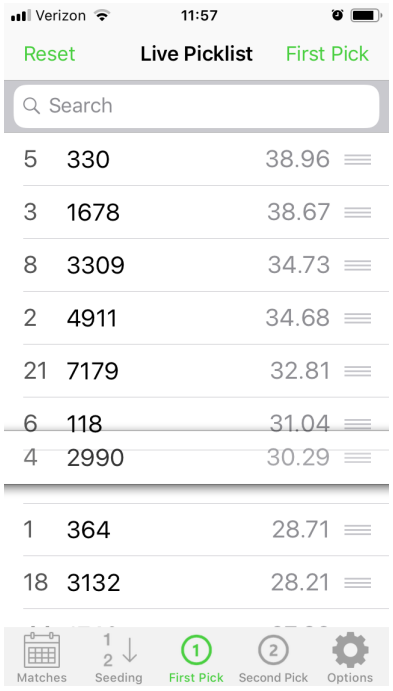

Figure 2.22: A team being moved in the Live Picklist (iOS)

The live picklist is used by strategists to update the picklist on the second day of qualification matches in real time.

2.6.1.6 Options Screen

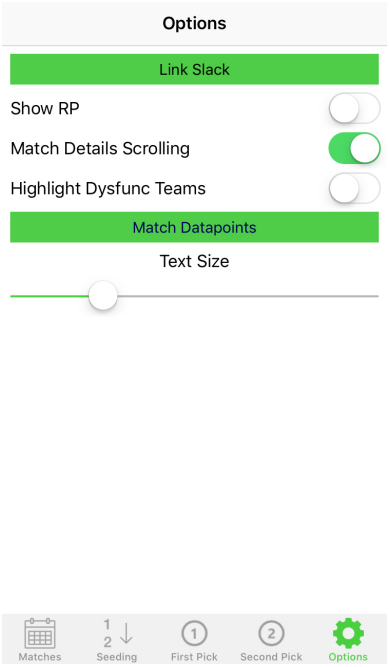

Figure 2.23: Options page (iOS)

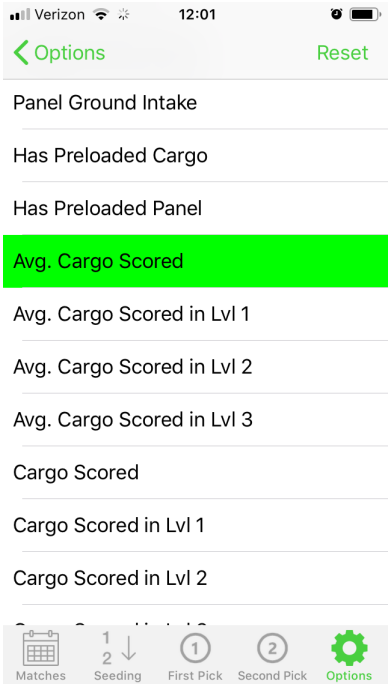

Figure 2.24: Selecting match data points (iOS)

The Options page allows users to customize the app's UI (user interface) and functionality. These options includes the ability to toggle ranking point display on the match schedule and highlighting teams with a high rate of not moving during matches. The user can also select the data fields displayed in the Match Details page.

2.6.1.7 Instabug

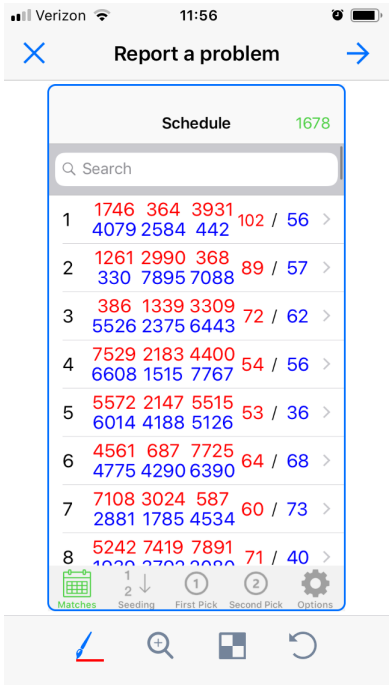

Figure 2.25: Reporting a bug with Instabug

Instabug is a third party platform used in the Viewer app to allow users to report bugs and crashes. It sends bug reports to an online account for developers to review.

2.6.2 Picklist Spreadsheet

Citrus Circuits creates its picklist in the evening after the first day of qualification matches, where strategists use data to rank the teams in order of ability to contribute on our alliance. During the alliance selection process, we pick the highest available team on our picklist. The Picklist Spreadsheet displays scoring ability, defensive ability, images, match videos, and graphs for each team.

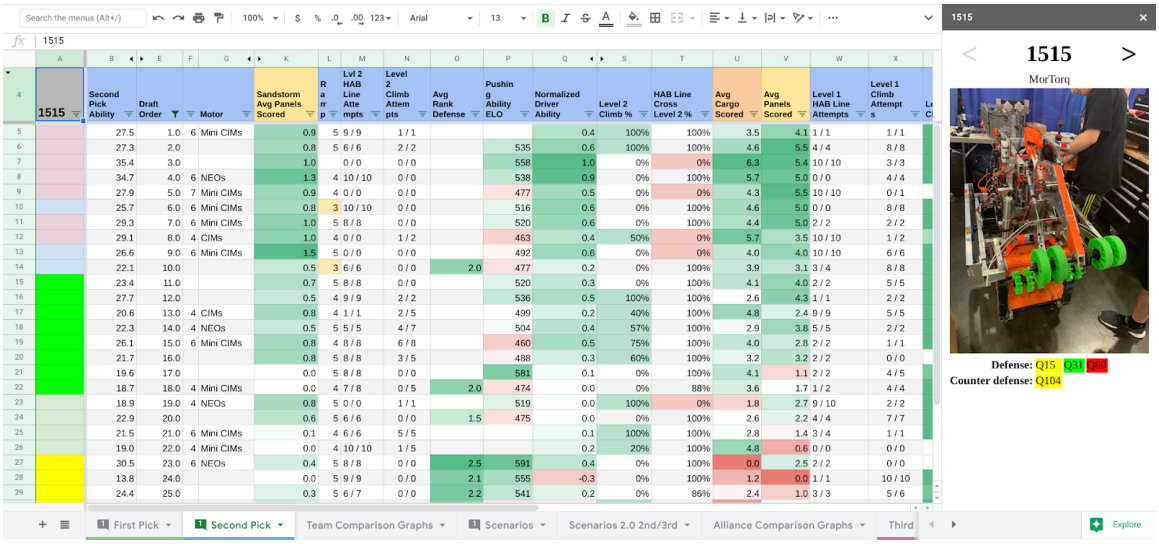

Figure 2.26: Screenshot of Picklist Spreadsheet (team numbers have been replaced with blank cells in the left-most column)

2.7 Hardware

2.7.1 Tablets

Citrus Circuits used four types of tablets during our 2019 Season: the Fire 7 (5th generation), Nexus 7 (2012), Nexus 7 (2013), and Lenovo Tab E7. The Fire 7 and Nexus 7 tablets were used to run the Scout app and the Lenovo Tab E7 was used to run the Super Scout app.

2.7.2 Mobile Phones

Software Scouting utilizes the Android and iOS mobile phones of Citrus Circuits team members for the Pit Scout, Viewer, and QR Scan apps.

2.7.3 Video System

Citrus Circuits' video system uses a Canon Vixia HF100 and 4 TB hard drive setup to record match footage. The videos are used by the Citrus Circuits drive team and strategists, making it easier to identify and watch robots first-hand. They are also used to review our own matches, and to coach other teams on defense. In the 2019 FRC game, match video was used in the creation of our picklist to identify top-performing defensive teams. The video system is ideally powered by the main power supply, but if they are too far apart, the system can rely on its own power supply.

2.7.4 Hotspot

Citrus Circuits uses a Netgear LB2120 modem to connect to a cellular network via Ethernet. This is a wired hotspot that does not use WiFi (which is prohibited at FRC events). The modem is connected to a network switch, which is connected to the main server computer and a backup computer. We have two SIM cards, one from T-Mobile and one from AT&T. If we are unable to connect with one carrier, we swap out the SIM cards.

2.7.5 Competition Power Supply

We use old robot batteries to power the wired hotspot, video system, and tablet charger system. To do this, we use a Potek P1500 500W power inverter.

Chapter 3

Logistics and Process

3.1 General

3.1.1 Team Culture About Scouting

Citrus Circuits fosters a positive culture around scouting, emphasizing its importance to our team's competitive success. We dedicate many hours during competition season to train scouts/super scouts and dedicate over half of our 44-person travel team to scouting during competition. Additionally, the entire team, including mentors, emphasizes the importance of scouting on our team, and how it contributes to our competitive success.

3.1.2 Idea Sharing Culture

This year's Software Scouting subteam strived to create a very welcoming culture in which developers feel very comfortable sharing their ideas, asking questions, admitting when they don't know something, and contributing to discussions. One procedure that came as a result of this philosophy and our idea sharing culture is the Crazy and Random Idea Box (CRIB). The CRIB is a short Google Form that records a developer's name, with a single sentence description of their idea, and an optional space to elaborate. Any idea can be submitted to the CRIB, and crazy and random ideas are especially encouraged (since some of Software Scouting's greatest improvements originally started as seemingly-crazy ideas). Occasionally, the contents of the CRIB are reviewed, and ideas are considered and often implemented.

3.1.3 App Groups

The Software Scouting subteam is made up of several app groups: the Scout, Super Scout, Pit Scout, QR Scan, iOS Viewer, and Android Viewer apps, as well as Server. Each member of Software Scouting belongs to one or two of these groups and works with the other developers in that group to develop their assigned sections of the system.

3.1.4 Project Managers

Each app group has a project manager responsible for working with the group to delegate tasks. Project managers and the subteam lead meet weekly during build season and bi-weekly during competition season to discuss progress, training, and future goals.

3.2 Offseason

3.2.1 Training

Citrus Circuits uses a peer-to-peer teaching system. Veteran members are responsible for the training of new members, from creating curriculum to guiding them through it. New members will need to work together and help each other to complete their training.

3.2.2 Continued Development

While veteran members help train new members during the offseason, they also continue development, which is then tested during offseason competitions. This is to test out possible changes to be implemented in the upcoming season, and to improve the software development skills of veteran members.

3.3 Development

3.3.1 Scout User Interface Prototypes

After the 2019 FRC game was released and we had decided which data fields to collect, our scouting developers discussed possible user interfaces (UI) to use on the Scout app. After completing two prototypes, several users evaluated how intuitive and effective each prototype was, allowing us to determine which UI we would use for the 2019 Scout app.

3.3.2 Field Tests

As part of the testing process, we schedule regular tests of the entire scouting system (called field tests) before each competition. These tests simulate the competition environment as realistically as possible. This allows us to understand how our system might break during competition and how to prevent it.

3.3.3 Review and Testing Process

Before any code is merged into the main codebase, it must pass a series of reviews and tests:

3.3.3.1 Reviews

- Buddy Review: A line-by-line code review process done with someone who practices the same programming language, preferably with a member of the same app group. This is often helpful with catching syntax errors or other common problems.
- Peer Review: A summary of what the code does and the logic behind each action completed with someone of a different app group. This helps identify logic issues or confusing code.

3.3.3.2 Tests

- Edge Test: Completed by someone in the same app group as the code developer, and is specifically used for testing edge cases. The tester in the Edge Test is intentionally trying to crash the app to discover uncommon bugs. This is critical for catching crashes before they merged into the main codebase.
- User Test: Completed by a user of the app to see if the general functions and specific features of the app work. This test helps with user communication and catching problems or crashes in an app. If testing a collection app while match video is available, the tester should watch a match and collect that match's data to simulate a competition environment.

In addition to this review system, only project managers can merge code into the main codebase to ensure all reviews and tested are completed before code is merged.

3.4 Competition Season

3.4.1 Before Competition

3.4.1.1 Scout Training

Before every competition, we test our software and train our scouts. Our lead scout (responsible for the organization of scouts in the stands at competition) and assistant lead scout run scout training. The purpose is to familiarize scouts with the Scout app, so scouts are able to input data without locating where to input it on the screen.

Each session of scout training lasts for 2.5 hours. We have multiple sessions of scout training before each competition. If it is a scout's first competition, they must attend two sessions. If not, they are required to attend one.

1. Prior to scout training, the scouts are required to read the Scout app user documentation.
2. Scout training begins with a brief description about the importance of scouting.
   (a) Understanding the value of scouting is critical to keeping scouts motivated and up to standards (for behavior and scouting ability) and on task (at training and competition).

3. We project the Scout app from a tablet, demonstrating the Scout app and its different features.
4. Scouts take a quiz on the Scout app user documentation.
5. After everyone has finished the quiz, the lead scout goes over the answers.
6. Scouts spend the remainder of the time scouting old match videos. All scouts scout the same robot.
    (a) Sometimes we play match videos at 125% to 200% speed, so that scouting at normal speed will seem easier afterwards. Software Scouting developers attend to discern and discuss any discrepancies (especially missed cycles or obvious mistakes). Every discrepancy is identified and reviewed in the match video to help scouts avoid making common errors in data entry during competition.

Scout training also serves as a systems test, where scouts can report bugs. Additionally, training allows the lead scout to ensure that the QR Scan app is successfully downloaded and working on all scouts' phones.

Super scout training consists of watching match videos, then discussing how to record subjective metrics such as speed, agility, and defense alongside a strategy mentor.

3.4.2 Competition

3.4.2.1 Competition Roles

At competition, we have the following people operating our scouting system:

- 1 lead scout
- 1 assistant lead scout
- 18+ scouts
- 5 super scouts
- 1 pit scout
- 2 video system operators
- 4 Software Scouting developers

The lead scout is a student responsible for managing all the scouts and super scouts in the stands at competition. They keep track of which scouts are on break, manage the behavior of scouts, make announcements to scouts, take roll, etc. The lead scout must be very responsible and respected by other students. The assistant lead scout supports the lead scout, and leads the scouts at training or competition when the lead scout is unable to do so. Of the four designated Software Scouting developers at competition, two run the server, one is available to fix the scout app, and another is available to fix other apps. Software Scouting developers are scouts when they are not needed to fix bugs.

3.4.2.2 Scout Rotation

During competition, 18 scouts and 4 super scouts scout at a time. The lead scout determines a rotation schedule ahead of time to ensure that all scouts get an equal number of breaks. Scout shifts rotate every 30 to 60 minutes.

3.4.2.3 Updates During Competition

Updates made to apps are applied between matches by swapping active tablets out for extra tablets, to avoid the loss of collected data during app updates. The lead scout and Software Scouting developers work together to make this process go smoothly.

3.4.3 After Competition

3.4.3.1 Debriefs

After each competition, Software Scouting debriefs for 2-3 hours, ensuring that everyone has a voice and offers their unique perspectives. We explore what went well and what could have gone better for the scouting system before and during the competition. We also use the time to re-evaluate what data we need to collect, calculate, and visualize, so that we can improve our system for the next competition. Debriefs allow us to adapt our scouting system to new information we receive and observe after each competition, ensuring that our system continues to grow with each competition.

Chapter 4

Lessons Learned and System Improvements

4.1 2018 Offseason Improvements

Citrus Circuits' scouting system is continuously improving. The following section includes improvements made to our scouting system for the 2019 FRC season based on the lessons we learned from the 2018 season:

4.1.1 Map-Based Scouting System

- Our previous user interface (UI) layout of buttons and counters was unorganized, causing scouts to spend too much time looking at the screen. As a result, scouts would miss robot actions on the field.
- FRC teams 3005 and 1619 inspired us to redevelop the Scout app with a map-based UI. Scouts found it more intuitive and were able to input data quicker without spending as much time looking at the tablet screen. We were also able to collect much more data with a single tap using the map-based Scout app (in one tap, we could record when and where a game element was intaken).

4.1.2 QR Data Transfer and Robot Assignment

- At the beginning of our 2018 season, data transfer between the online database and scouting tablets required using a Bluetooth connection. This data transfer was not reliable, and the tablets often had to be plugged into a computer or connected to WiFi outside the venue for manual data transfer.
- During the 2018 season, we developed and tested the following data transfer systems: Bluetooth Internet Sharing, Bluetooth OBEX File Transfer, and QR codes. We settled on QR code data transfer because it was significantly more reliable than Bluetooth data transfer, and more convenient than USB data transfer.

4.1.3 Redevelopment of the Server

- In the 2018 season, our Server had code that was difficult to read and sustain, and contained a number of bugs. The messy, uncommented code made it difficult to understand, sustain, and fix existing code, and add additional code. In the 2018 offseason, Software Scouting members rewrote the server code using better style guidelines, commenting procedures, and more robust logic, making it much easier to understand and to hotfix at competitions.
- The 2019 Server uses the PEP8 style guidelines for Python, enforced with a linter (Pylint). Additionally, the commenting procedures are much stricter, requiring docstrings for every file and function, and comments for code that is not self-explanatory.

The 2019 Server code can be found at `https://github.com/frc1678/server-2019`

4.1.4 Pre-Scouting and Practice Match Scouting

- At the beginning of each competition, there was previously very little data on each team for our strategists to plan matches with. In the past, we've informally collected data during practice matches and by watching match videos from previous competitions. We realized that we could develop software for this data collection, and incorporate this data into our scouting system for easier and more useful collection, processing, and visualization.
- We developed a web form using Google App Script that outputs to a Google Spreadsheet to scout practice matches and online videos of previous competitions to give us the basic data needed to strategize for our early qualification matches.

4.1.5 Debriefs

- In past years, we did not spend enough time debriefing competitions. Additionally, many debriefs only included a handful of voices and opinions.
- Beginning in the 2018 offseason, we began doing more comprehensive debriefs, and strived to increase participation. Having detailed debriefs allows us to utilize new information to enable our scouting system reach its full potential. The majority of improvements made to our scouting system come as a result of our debrief process, which allows us to highlight and document the most pressing issues with our system.

4.1.6 Feature Cutoff

- Last year, we continued to develop and program additional features well into competition season, sacrificing the reliability of our previously programmed features by using our time to rush the implementation of features (causing them to be unreliable), rather than to test our system.
- This was partially remedied by our feature cutoff, which was a set date in which we transitioned from adding features to testing our system, resulting in a more thoroughly tested and reliable system. While this improved the reliability of our system, we are looking to change our development procedures to focus on reliability first.

4.1.7 Field Tests

- In the 2019 season, we ran full-system field tests to simulate the competition environment. The 2019 field tests involved all scouting developers, as well as other members of Citrus Circuits that were not in the Software Scouting subteam. We tested every component of our system by collecting data from match video, processing data, and displaying data. This allowed us to identify bugs in our system, specifically during data transfer, which is imperative to the components of our system working together.

4.1.8 Better Git and GitHub Practice

- Scouting developers have struggled with their Git and GitHub practice due to a limited understanding of Git and GitHub and experience with a collaborative Git and GitHub workflow.
- To address this, we created specific training (including practice assignments) for Git and GitHub for new and veteran developers.

4.1.9 Code Review and Testing

- In past years, we have had no code review or testing procedures, allowing buggy and inefficient code to be merged into our main codebase. This resulted in many issues with the system, especially during competitions, and meant that only one developer understood the code they had written.
- We implemented a thorough code review and testing procedure (Section 3.3.3) before code could be merged into the master branch. This allowed us to avoid logic errors, common bugs, and edge case bugs, as well us give multiple developers an understanding of the code, which allows them to better debug code, modify code, and write new code.

4.2 2019 Mid-Season Improvements

Citrus Circuits develops our scouting system during build season in preparation for our first competition, and continues to update and refine our apps throughout the competition season to provide the most useful and accurate data for picklist creation and match strategy. As our strategic understanding of the game improved during the season, so did our understanding of what data was important. During the 2019 season, the following major mid-season redevelopments were implemented in the Scout app, Super Scout app, Server, and Picklist Spreadsheet.

4.2.1 Scout

- Simplify and remove unnecessary or complicated features: simplify incapacitation, remove alliance station and end position inputs.
  - After observing what collected data was necessary in calculations and useful to display, we realized that we were collecting extraneous data: types of incapacitation (broken mechanism, tipped over, etc.), alliance station position, and end position. This resulted in excess work for the scouts. By removing extraneous data, scouts were able to focus on collecting the important, relevant data more accurately.
- Emphasize defensive capabilities data collection: add a counter for the number of cycles defended, record whether a robot was defended when it placed a game element.
  - Our elimination match strategy for the 2019 FRC game requires our alliance to have a strong defensive robot, and therefore, we needed to collect, calculate, and visualize more information relevant to the robots' defensive capabilities. This data was primarily collected in the Super Scout app, but the Scout app also underwent similar changes, collecting data to better objectify a robot's defensive capabilities.

4.2.2 Super Scout

- Re-develop to focus on defensive capabilities data collection: add cardinal ranking system for defense and counter-defense, add tracking of pushing battles.

4.2.3 Server

- Add calculations: Match predictions (score, RP)
  - During the 2018 season, we had both team and match predictions, but due to the redevelopment of the Server, we were unable to implement these calculations until mid-competition-season. We decided to implement match predictions since they were very helpful in determining which matches we should focus our strategic efforts on.
- Better measurement of defensive capabilities: pointsPrevented (Section 2.5.1.2), pushing battle Elo (Section 2.5.1.6)
- More efficient querying of APIs: Improved caching system for data from The Blue Alliance.
  - During the 2019 competition season, we encountered issues with the efficiency of the server. Due to inefficient queries on The Blue Alliance's APIv3, the server ran slowly, causing delays in data processing (and therefore, visualization). To address this, we added a local caching system for all requests made from our server to The Blue Alliance.

4.2.4 Picklist Spreadsheet UI Improvements

- The Picklist Spreadsheet is designed to make our picklist creation meeting as efficient as possible. To accomplish this, the UI (user interface) of the spreadsheet must provide quick access to relevant data, or data won't be utilized in the picklist creation process.
- In the 2018 offseason, we added a sidebar (written in Google App Script) to display robot photos and link directly to match video on The Blue Alliance. Additionally, we overhauled the graphing system to be more reliable and visually readable, added a settings page for easier configuration, and simplified the spreadsheet formulas.

4.3 Future Improvements

While the 2019 scouting system was our most effective and reliable scouting system, it did have a number of flaws. The following is a list of weaknesses present in our 2019 scouting system and how we plan to improve those areas in the future:

4.3.1 Data Fields Collected

- Our 2019 system collected excess data that went unused. This issue came from the way we selected data fields to collect at the beginning of the season: we determined which data fields we wanted to collect before selecting what data we need to display, resulting in a surplus amount of collected data. We then designed our data collection apps to collect these extraneous data fields, making data entry difficult, thus decreasing the accuracy of the collected data.
- For the upcoming season, we plan to first evaluate what we want our output to be (what we want to visualize) and work backwards to determine the input (what we need to collect). Also, we plan to factor in the accuracy cost of adding a collected data field to avoid compromising our data accuracy. This will result in a system where a majority of the collected data is used.

4.3.2 Programming Languages and Platforms

- We used numerous programming languages and platforms (Section 2.2), resulting in less collaboration and smaller developer teams. Additionally, it was difficult to find documentation and create training for the several languages and platforms we used.
- To prepare for the 2020 season, we have discontinued support for iOS, and plan to use 2 programming languages: Kotlin for collection and visualization, and Python for processing.

4.3.3 User-Developer Communication

- Due to a lack of consistent communication and system releases, we were unable to properly gauge what our users needed. This caused us to collect, calculate, and display unnecessary data fields while missing important ones.
- This year, we hope to reach out and communicate with our users more to better understand their needs, which we will then incorporate into our processes. Additionally, we plan to get frequent user feedback with frequent system releases to recognize whether we are going in the right direction with each release.

4.3.4 Git and GitHub Practice

- Git and GitHub practice has improved compared to previous seasons, but has room for improvement.
- We plan to thoroughly train all scouting developers to improve our understanding of how Git and GitHub works and reinforce good Git and GitHub practices.

Chapter 5

System Analysis

5.1 Accuracy of Data

5.1.1 Predicted Ranking Points

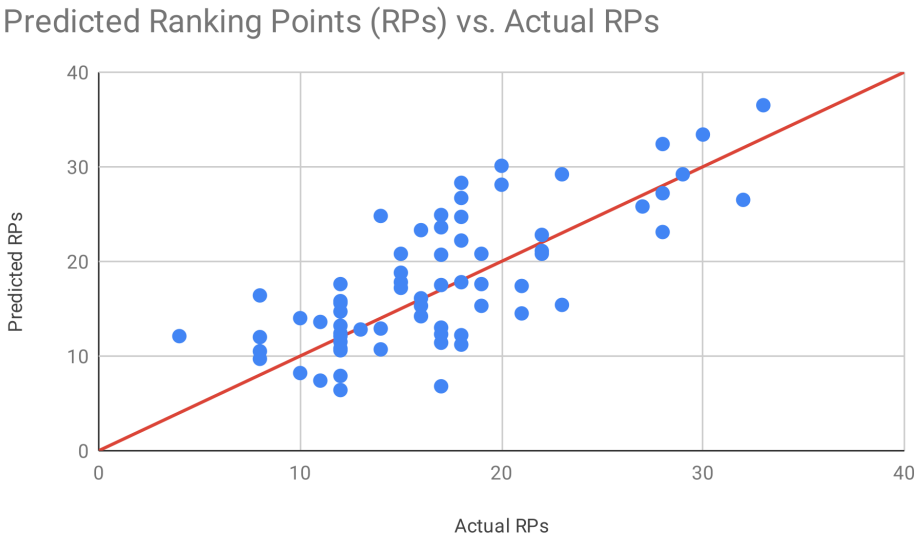

Figure 5.1: Predicted Ranking Points (RPs) vs. actual RPs

Figure 5.1 shows a comparison between the predicted Ranking Points (RPs) and the actual RPs of each team in the competition based on prescouting data. Each point is a team, the x-axis represents the team's actual number of RPs, and the y-axis represents the team's predicted number of RPs.

5.1.2 Predicted Score

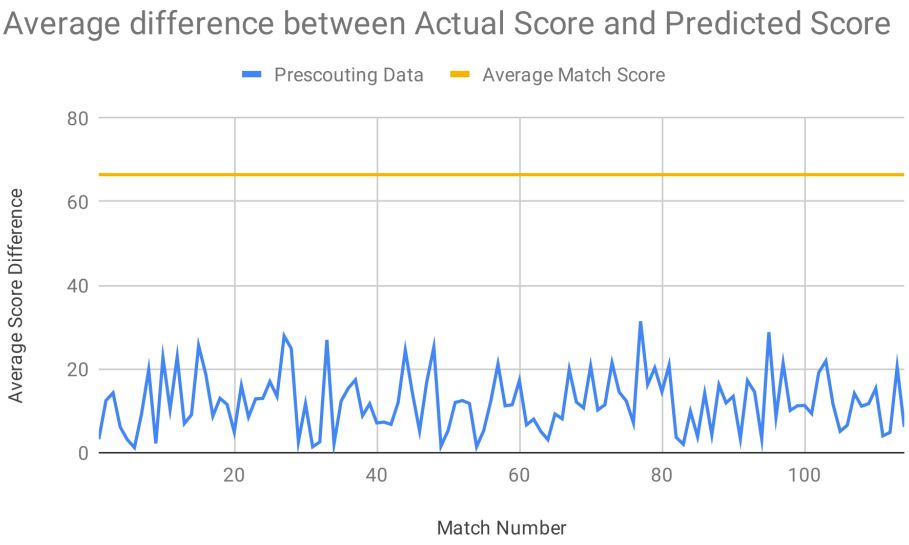

Figure 5.2: Average difference between predicted score and actual score for each alliance by match, based on prescouting data

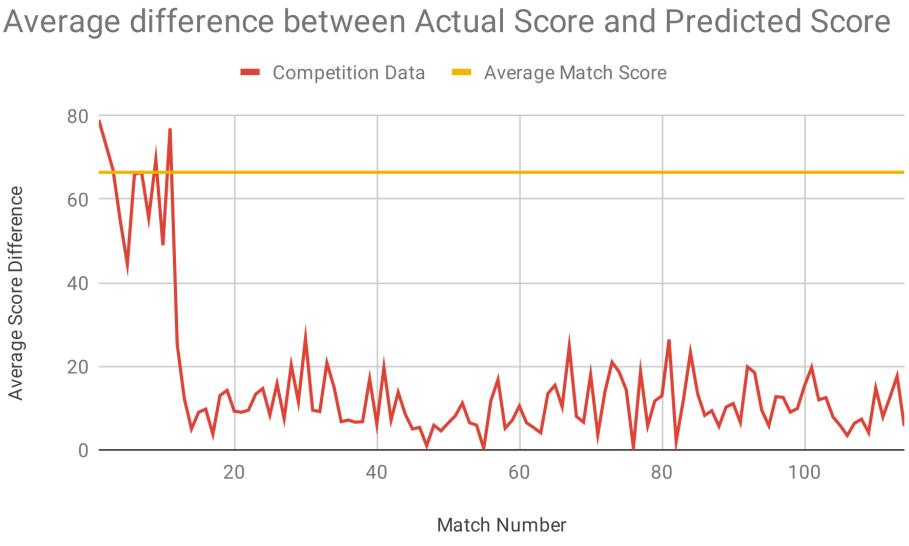

Figure 5.3: Average difference between predicted score and actual score for each alliance by match, based on competition scouting data

Over the course of the competition, predicted score became more accurate as more data was collected for each team. During the second day the Carver Division, the Server correctly predicted the winning alliance in 75% of matches.

5.1.3 Scored Game Piece Counts

We tested the accuracy of our objective match data against data from The Blue Alliance (TBA). For each match, we added together the average amount of cargo and panels scored by each team in an alliance and compared the number of game pieces scored by each alliance to data we retrieved from TBA. We found that our scouts were off by approximately 0.5 panels and 0.5 cargo for each alliance in a match.

5.2 Analysis of Consolidation

In order to determine whether our 18-scout collection system (for objective match data) is more useful than a similar 6-scout system, we analyzed how many times two scouts agreed on the number of game pieces scored and one scout disagreed, to show how many times having 18 scouts improved the accuracy of our data. We found that our accuracy for the number of game pieces scored by a team in a match improved by about 20%, meaning that consolidation was vital to the data accuracy of our scouting system.

Chapter 6

Conclusion

6.1 Overall

The Citrus Circuits 2019 Scouting System was the most effective and reliable ever, thanks to our pre-season and mid-season redevelopments, as well as the new processes, procedures, and debriefs implemented in the Software Scouting subteam. Reliable, efficient, and easy to use, the 2019 Scouting System allowed for flexible and powerful strategic analysis that was crucial in helping us reach the World Championship Einstein field for the seventh year in a row.

6.2 Human Resources Utilized

To create our electronic scouting system, the Software Scouting subteam requires many resources, primarily the time and commitment of many dedicated software developers. During the 2019 season, the Software Scouting subteam was comprised of 17 members. Our season is split into two sections: the offseason and the build/competition season.

- Offseason (June–December):
  - Mandatory meetings two nights a week (5 hours total)
  - Used for training new and veteran members, as well as discovering new ways to improve our system for the upcoming season
- Build and competition season (January–April):
  - Four mandatory meetings a week (21 hours total)
  - Used for developing, programming, and debugging the scouting system

In order to complete our scouting system in time, we often end up working during extra team meetings and outside of meetings to meet deadlines and complete all required work. Scouts, super scouts, and strategists also spend several hours before each competition training on their respective app (Section 3.4.1.1), allowing them to best utilize the app and collect accurate data during competition.

6.3 Resources

6.3.1 GitHub

All of the code from the 2015–2019 scouting systems can be found on GitHub at: `https://github.com/frc1678`. It is open-source and available for use under the MIT License.

6.3.2 Past Whitepapers

Citrus Circuits first used a custom-built electronic scouting system in 2013, and has used such a system every year since. We release an annual whitepaper with the details of that year's scouting system. For past years', please reference our website: `https://www.citruscircuits.org/scouting.html`

6.3.3 User Documentation

The following documentation was authored by members of the Software Scouting subteam to be read by users of the apps (scouts, super scouts, strategists) prior to Scout Training. The documents are currently out of date and are meant to provide an understanding of the layout we use to create our user documentation, rather than a guide on using our collection applications.

6.3.3.1 Scout

`https://1678scouting.com/2019whitepaper/scout_user_documentation`

6.3.3.2 Super Scout

`https://1678scouting.com/2019whitepaper/super_scout_user_documentation`

6.3.3.3 Pit Scout

`https://1678scouting.com/2019whitepaper/pit_scout_user_documentation`

6.3.4 Additional Resources

6.3.4.1 Citrus Circuits Fall Workshops: Scouting System Development

Citrus Circuits students and mentors put on several workshops/seminars focused on different aspects of our team. The Scouting System Development workshop covers the various aspects of our scouting system, and how the lessons we've learned can be applied to other FRC teams.

Citrus Circuits Fall Workshops: `https://www.citruscircuits.org/fall-workshops.html`

6.3.4.2 Simbots Seminar Series: Scouting and Match Strategy

Seminar by the renowned mentor of Team 1114 Karthik Kanagasabapathy. Covers various methods of scouting, development of match strategy, and the foundations of mathematical analysis techniques in FRC.

`https://www.youtube.com/watch?v=l8syuYnXfJg`

6.3.4.3 Overview and Analysis of FIRST Stats

Comprehensive overview of mathematical analysis of FRC and FTC teams. Covers concepts ranging from simple analyses such as OPR, DPR, and CCWM to more complex methods such as WMPR, EPR, and MMSE techniques. Some knowledge of linear algebra required.

`https://www.chiefdelphi.com/t/overview-and-analysis-of-first-stats/144569`

6.3.4.4 The Blue Alliance

Excellent website for viewing data on FRC. Provides an API for easily pulling data from the site for use in custom scouting system software.

`https://www.thebluealliance.com`

6.4 Contact Us

We are interested in helping to develop the FRC scouting community and opening the power of an electronic scouting system up to other teams. If you have any questions, please contact us at `softwarescouting@citruscircuits.org`.