FRC Control System Usability Analysis

An Interactive Qualifying Project
submitted to the faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Bachelor of Science Degree
On December 19, 2013

By:
Alexander Henning
Brendan McLeod
Fredric Silberberg

With WPI Advising from:
Brad Miller & Ken Stafford

EXECUTIVE SUMMARY

FIRST (For Inspiration and Recognition of Science and Technology) is a nonprofit organization founded to help get students interested in STEM (Science, Technology, Engineering and Mathematics) education. FIRST runs a robotics competition called FRC (FIRST Robotics Competition) which is for students in grades 9-12. The students build robots in a yearly competition, which takes place over the first few months of every calendar year. The control system consists of the electronic components and the software tools that teams use to control their robots. In 2015, FRC will be moving to a new control system, manufactured by National Instruments. Due to this transition, it is a good time to evaluate the current state of the control system, looking at what enables team success and what can be improved.

In this IQP (Interactive Qualify Project), we conducted a survey to see which software and documentation features of the current control system are helping teams succeed, what features could be improved, and what features could be added. To get a feel for the issues and success of the current control system, we started by conducting interviews with teams at a post-season event, Mainely Spirit. Next, we analyzed the results of our interviews and created a comprehensive survey on these issues. We tested this survey by distributing it to a FIRST-interest email alias at WPI. We used this feedback to modify the survey, ensuring that we were asking the correct questions and that the display logic worked. We then distributed the final survey to teams through social media, including ChiefDelphi (a popular FIRST-related forum), several FIRST-related Facebook groups, and Twitter. We also had several mentors on other teams spread the word about the survey.

Our survey had 176 respondents including both mentors and students. These split relatively evenly between the two groups. We analyzed the demographics of our response population and compared them to the demographic information we have from FIRST to help determine what segment of the population responded to our survey. We also compared the language use of our respondents to the real-world language data, provided to us by National Instruments. Finally, we looked at how involved our respondents were with the control system to see if there were any more correlations.

With these results, we were able to generate a list of seven different recommendations for the control system team. Our first recommendation centers around the official Technical Resources page on the FRC website, which is the official documentation source for teams. We determined that most people expect to find information on this page, and that it is currently very hard to find information on the page. We also recommended that the current short, memorable URL be updated, to allow teams to have easy access to the page.

Our next recommendations focused on the ScreenStepsLive documentation source. Out of all the documentation sources, ScreenStepsLive was the least known, however most teams who knew about it used it and found it to be among the most useful documentation sources. It is also missing some key areas of documentation and is programming-language biased at times. We recommended increased promotion of ScreenStepsLive, and updating it with more task-based examples for multiple programming languages.

We also looked at the training videos that have been produced by various members of the control-system development team. Teams felt that some videos were useful, but many were out of date, and that the videos were often missing topics they were looking for. We recommended that the control system developers update the videos to be relevant to the current elements of the control system, and to produce more videos for other tasks that teams need to accomplish.

One of the tools we looked at as part of this IQP is the FRC Simulator. This simulator allowed teams to test code on a virtual robot. Teams who used the simulator felt that it was useful, but most teams were unable to use the simulator due to lack of compatibility with custom robots or certain programming languages. We recommended that the simulator continue to be supported, and that support for custom robots and all of the official FRC languages be added.

Another tool we looked at is RobotBuilder. RobotBuilder is a tool for the Java and C++ programming languages that allows teams to visually design their robot code framework and then export that framework to real code. Teams who used RobotBuilder were very satisfied with the experience. Therefore, we recommended that the control system team should continue to support the RobotBuilder, and to update it as necessary to support the 2015 control system.

Finally, we looked at the current feature-parity between all of the different languages. We feel that the current parity is excellent, since teams are choosing to use languages based on their knowledge of the language itself, and not based on whether the Control System components are better supported in a specific language. This is good for teams, since it means that they do not feel forced into choosing a particular language because other languages do not have the features they need. We recommend keeping that this level of parity into the future.

The general view of respondents to our survey is that the current elements of the control system are high quality and do not hinder team success. However, there needs to be more documentation and it needs to be better organized. The tools need to support more scenarios, such as the simulator supporting custom robots. The quality of the currently produced documentation is high, but sections are often missing or very hard to find. Our general recommendation to the control-system development team is to keep the level of quality that they are currently producing, but to consolidate and complete the documentation, and make the available tools more flexible.

TABLE OF AUTHORSHIP

Editing	All
Executive Summary	Silberberg
Chapter 1: Introduction	Henning/McLeod
Section 1.1: Structure of this document	Henning
Section 1.2: The Goal	Henning
Section 1.3: Motivation	Henning
Section 1.4: Success in the FRC Control System	McLeod
Section 1.5: Objectives	All
Section 1.6: Conclusion	McLeod
Chapter 2: Background	All
Section 2.1: Overview	All
Section 2.2: FIRST Robotics Competition at a Glance	Henning
Section 2.3: Programming for Students	Henning
2.3.1: Integrated Development Environments	Henning
2.3.2: Graphical and Text-Based Programming	Henning
2.3.3: Code Scaffolding	Henning
2.3.4: Simulation	Henning
2.3.5: Documentation	Henning
Section 2.4: Control System Usability	McLeod
2.4.1: FRC Driver Station	McLeod
2.4.2: FRC Dashboard	McLeod
2.4.3: FRC Radio Configuration	McLeod
2.4.4: Conclusion	Silberberg
Section 2.5: Interview Techniques	Henning
2.5.1: Goals and Interviews	Henning
2.5.2: Structured vs. Unstructured Interviews	Henning
2.5.3: Interview Method	Henning
2.5.4: Number of Questions and Interview Duration	Henning
Section 2.6: Survey Techniques	Silberberg
2.6.1: Blocking Factors	Silberberg
2.6.2: Censuses	Silberberg
2.6.3: Random Sample	Silberberg
Section 2.7: Online Survey Tools	McLeod
2.7.1: Available Tools	McLeod
2.7.2: Paragraph and Single Word Answers	McLeod
2.7.3: Multiple-Choice Question	McLeod
2.7.4: Display Logic	McLeod
2.7.5: Exportable Data	McLeod
2.7.6: Randomizable Questions	McLeod
Section 2.8: Conclusion	McLeod

Chapter 3: Methodology	Henning
Section 3.1: Interviews	Henning
3.1.1: Interview Details	Henning
3.1.2: Method	Henning
Section 3.2: Pre-Survey	Henning
3.2.1: Pre-Survey Details	Henning/Silberberg
3.2.2: Method	Henning
Section 3.3: Online Survey	McLeod
3.3.1: Survey Size	Silberberg
3.3.2: Survey Questions	McLeod
0 Distribution Method	Silberberg/McLeod
Chapter 4: Results	McLeod
Section 4.1: Interview Results	McLeod
4.1.1: Knowledge of Tools	McLeod
4.1.2: Looking for Help	McLeod
4.1.3: Libraries and Objects	McLeod
4.1.4: Different Help Formats	McLeod
4.1.5: Summary	McLeod
Section 4.2: Pre-Survey Results	Silberberg
4.2.1: Conflating Answers	Silberberg
4.2.2: Respondent Involvement	Silberberg
4.2.3: Other Fixes	Silberberg
4.2.4: Summary	Silberberg
Section 4.3: Survey Results	Henning
4.3.1: Demographics	Henning
4.3.2: Programming Language Choice	Henning
4.3.3: Team Success	Henning
4.3.4: Knowledge of Tools and Documentation	Henning
4.3.5: Summary	Henning
Section 4.4: Conclusion	Henning
Chapter 5: Recommendations	McLeod
Section 5.1: Re-Structuring of FRC Technical Resources Page	McLeod/Silberberg
5.1.1: Making Resources more Accessible	McLeod
5.1.2: Better Organization of Technical Resources	McLeod/Silberberg
Section 5.2: Better Promotion of ScreenStepsLive	McLeod
5.2.1: Increase References to ScreenStepsLive	McLeod
5.2.2: ScreenStepsLive Popularity	McLeod
5.2.3: Renaming ScreenStepsLive	McLeod
Section 5.3: Increase the Task-Based Documentation on ScreenStepsLive	McLeod
Section 5.4: FRC Simulator	McLeod
5.4.1: Continue Providing the FRC Simulator	McLeod

5.4.2: Expand to other Languages and Support other Robots	McLeod
Section 5.5: Maintain Current Status of RobotBuilder	McLeod
Section 5.6: Continue Providing Dashboards	McLeod
Section 5.7: Language Feature Parity	McLeod
Chapter 6: Conclusion	Silberberg
References	Silberberg
Appendix A: Script for Interviews	Henning
Appendix B: Pre-Survey Materials	Silberberg
Message to WPI Students	All
Pre-Survey Questions	All
Appendix C: Final Survey Materials	All
Survey Flowchart	McLeod/Silberberg
Final Survey	All
Appendix D: Final Survey Results	Silberberg/Respondents

Table of Contents

TABLE OF FIGURES

CHAPTER 1: INTRODUCTION

Section 1.1: Structure of this document

This paper discusses the goals, background, methodology, results, and analysis for the “Improving FRC Team Success” IQP. Discussed in this chapter are the goals that guided this project. Chapter two discusses the relevant background research in understanding and accomplishing the project’s goals. Next, chapter three discusses the methodology of this investigation. Then, chapter four analyzes the data we gathered and provides our suggestions for improving team success with the FRC Control System. Chapter 5 presents a list of recommendations and their supporting evidence. Finally, chapter six summarizes the discussed material and concludes this report.

Section 1.2: The Goal

Our goal was to generate a list of action items that will allow the new control system designers to help teams be as successful as possible. This project analyzed the parts of the control system that encompass the software and tools provided to teams for interfacing with common FRC hardware. These improvements fall into three categories: current well-done features that which need to be kept, features that are done poorly which should be removed or improved, and new features that can be added.

Section 1.3: Motivation

The FIRST (For Inspiration and Recognition of Science and Technology) Robotics Competition (FRC) has had the same control system for the past five years. In 2015, FIRST, in partnership with National Instruments, is releasing a new control system called the RoboRIO. With this major change in the control system, now is an ideal time to evaluate the current control system with an eye for improvements.

Software for the 2015 control system is still under development, making now an opportune time for this study. The primary push to get the software ready for release will occur in 2014, which leaves time to incorporate the results of this study into the software and documentation of the new control system. Since much is already changing, it is easier to make large changes that may be backwards incompatible without confusing existing teams since there is an expectation of change.

Section 1.4: Success in the FRC Control System

In order to evaluate how to improve teams’ success, it is first necessary to define success in the context of the FRC Control System. It can be difficult to create a definition for ‘success’ with the control system for a robot programming competition. Although many may define success in a competition as winning, success with respect to the control system is more complicated. There are a fixed number of winners, regardless of whether or not the control system enables teams' success. Instead, this study uses another measure.

One way to find a measurement is to look at the 2013 FRC Game "Ultimate Ascent". In this game, students built robots that could score in two challenges: climbing a pyramid, and scoring Frisbees in goals. Students needed to overcome several control issues to solve these challenges. For example, teams that created a robot to climb the pyramid may encounter control issues such as closed and open loop position control, signal timing, state machine design, and sensor interfaces. Focusing on shooting systems, teams may see challenges such as velocity control loops, vision tracking via targets, precision drive stability, sensor feedback, and system integration.

It is not possible to settle on a list of robot tasks for teams to accomplish as the measure, since teams may have different goals. Instead, a measure is "How well did a team accomplish their respective goals?" For example, if a team only had a robot that climbed the pyramid, they have not failed if they cannot vision track, since they did not design their robot to perform that task. Therefore, our starting definition for success must include how well teams were able to accomplish their goals.

Unfortunately, even this is not a sufficient definition of success with respect to the control system. We also considered how fully teams were able to accomplish their goals. If they were not able to accomplish these goals, we also evaluated whether the inhibiting factor was controls-based. For example, if a team fails at their goal to climb because the arms snapped trying to climb, the control system is not the factor preventing the team from achieving their goal.

A final factor we considered is that teams select goals that they feel they can accomplish. To that end, teams may avoid goals that they feel are beyond them. This can include tasks that involve vision-tracking, autonomous movement or complex goals. Therefore, when determining whether the control system enables teams' success, we evaluated whether it caused them to avoid goals they would have chosen.

Since we are measuring teams' successes relative to their goals, it means that success changes from team to team and we considered this throughout this study. These complications make measuring success less straightforward. However, understanding success in the context of the FRC Control System is necessary to improve it.

Section 1.5: Objectives

Ideally, our deliverables would include a list all control system features that enable every team to succeed. Given our time and information constraints, we instead recommended as many possible improvements as we could with as high statistical confidence as possible. Our minimum target was to determine where the control system currently succeeds and where the control system developers can make improvements. Our recommendations are backed with a 90% confidence level.

Section 1.6: Conclusion

Development of the software tools and documentation for the FRC Control System is critical to the competition. Without these, students may struggle at programming robots, making them unable to compete. The FIRST control system team aims to provide everything necessary to make it easy for students to learn to program robots successfully. The creation and development of tools and documentation relies on knowledge of what teams need and where they struggle. This IQP aims to provide the information necessary to make improvements.

CHAPTER 2: BACKGROUND

Section 2.1: Overview

Robots and robotics competitions offer many opportunities for teaching students important topics and skills including STEM (Science, Technology, Engineering, and Math), teamwork, and organizational skills. To take advantage of robotics and use it as an educational experience, it is important to make sure that the tools enable success.

This section discusses the following:

- FIRST Robotics Competition at a Glance
- Programming for Students
- Control System Usability
- Interview Techniques
- Survey Techniques
- Online Survey Tools

Section 2.2: FIRST Robotics Competition at a Glance

FIRST is a nonprofit founded by Dean Kamen, an inventor and entrepreneur. It aims to inspire kids to go into STEM fields. According to Kamen, the program’s vision is "To transform our culture by creating a world where science and technology are celebrated and where young people dream of becoming science and technology leaders."[1]

FRC is a competition held by FIRST and aimed at teams of high school students. Every year, FIRST announces a new challenge to approximately 2,500 teams. Teams then have six and a half weeks to design, build, and program a robot. The typical robot is just under 140lb and around 5 feet tall.

While the exact skills developed depend on the resources the team has, working on a FRC team exposes students to many STEM and non-STEM skills. STEM areas include the mechanical engineering skills necessary to build the robot, the electrical engineering skills necessary to wire it and add sensors for effective control, and the computer science skills necessary to program it to compete. Non-STEM skills include teamwork, fundraising, planning, communication, and organization.

There have been several studies comparing the outcome of students who participated on FIRST teams to students who did not. One study assessed the impact of FRC on students’ attitudes towards science with the TORSA (Test of Science Related Attitudes). It found that students with FRC experience

[1] (USFIRST n.d.)

view science more positively. This is important because "Positive views toward science are often viewed as an important correlate to achievement in science."[2]

Benefits of participating in FIRST go beyond a more positive view of science. Another survey found that FIRST alumni on average performed better in important areas. The following are true of "average" FIRST alumni participating in the survey:

- Students have higher high school grades,
- Students have higher science SAT and ACT scores than the national average,
- Students are more likely to graduate high school and go to college,
- Students are more likely to be employed,
- Students are more likely to major in engineering.

These are very promising results. Unfortunately, due to the nature of the survey, it cannot distinguish "whether this strong performance in high school was the result of involvement in FRC, or whether FRC attracted strong students, or both." Regardless of which condition is true, 94% of study participants felt that they learned new skills and that 89% felt FIRST increased their self-confidence. FIRST definitely seems to have made a difference to its participants.[3]

Section 2.3: Programming for Students

FIRST provides teams with a number of real world programming tools designed to make it easy for students to get started. It is important to understand whether each tool contributes to enabling success. Some of these are listed below:

- NetBeans IDE (For programming in Java)
- Eclipse IDE (Wind River for programming in C++ and standard Eclipse for programming in Java)
- LabVIEW Development Environment (For programming in the LabVIEW dataflow language)
- RobotBuilder (Graphical tool to help generate the scaffolding for Java and C++ programs)
- LabVIEW Simulator (Currently supports three robots and only works with LabVIEW)
- Documentation (Online, JavaDoc, Doxygen, Videos, etc.)

2.3.1: Integrated Development Environments

FIRST provides an IDE (Integrated Development Environment) for each of the three officially supported languages. They are the NetBeans IDE for Java, Wind River IDE for C++, and the LabVIEW development environment for LabVIEW. While the IDEs are all different, they provide some common

[2] (Welch 2010)
[3] (Melchior 2005)

behavior. One useful feature is integration with WPILib (the FRC-provided libraries for teams developing in all supported languages) to pop-up built-in help for classes, methods, VIs (virtual instruments, used in LabVIEW), and other WPILib features. For example, if a student is wondering what the `RobotDrive` class does, they can just type it, and a short paragraph describing the class and its basic usage pops up.

The IDEs also have other useful features for students. One such feature is that they all come with template programs and example programs that students can use as a starting point when programming their robot. All environments support running code and debugging code with an integrated debugger. In addition to the added features for FIRST, these IDEs have a large number of common features for developing and structuring code. These features are available to students and can be very helpful, but whether students use them depends on the team and the amount of code they develop.

2.3.2: Graphical and Text-Based Programming

FIRST distributes development environments for the three officially supported languages. These are programming languages used in industry, so students can carry these experiences to future jobs and college. However, since the focus is on helping high school students learn about the general concepts of programming in a six and a half week period, there are special considerations. One area to look at for comparison and ideas is programming languages targeted at children. This section focuses on discussing previous research and its applicability in the FRC context.

One study discusses problems with early programming languages used to teach children. The problems they found are worth considering when trying designing an environment to help teach students to program. Three main problems commonly found are as follows:

- "Early programming languages were too difficult to use, and many children simply couldn't master the syntax of programming;
- "Programming was often introduced with activities (such as generating lists of prime numbers and making simple line drawings) that were not connected to young people's interests or experiences; and
- "Programming was often introduced in contexts where no one could provide guidance when things went wrong—or encourage deeper explorations when things went right."[4]

This is a good set of problems to avoid. For any FRC programming language, the activity is making robots perform tasks, which is a very engaging subject with children.[5] Most FRC teams have mentors to

[4] (Resnick 2009)
[5] (Johnson 2003)

help students, so students should be able to get help when things go wrong and get guidance to go deeper and add extra functionality to the robot to make it more competitive.

Java and C++ are traditional text-based languages, where programs consist of human-readable text that is compiled (a process in which the program is translated to a form that a computer can understand) for the robot controller. These languages are both C (an earlier programming language) derivatives, and therefore have very similar syntax. LabVIEW, however, is a graphical dataflow language, and not a text-based language. Users model the program as a flowchart, and then the LabVIEW compiler compiles the flowchart. Engineers and scientists consider this intuitive, as they are already use flow charts for other applications.[6]

It is impossible to improve the syntax of these languages without defining new languages, which creates compatibility issues. However, it is possible to generate some of the syntax for the programmer in certain contexts. Generating code can help not only with the syntax that is confusing to beginners, but also with the structure and organization of the program. Finding the correct program organization patterns can be very confusing when programming a robot or even just learning to program for the first time. This is why most learning environments start with a template of a good structure for the user to expand on, as described in multiple studies.[7,8] All of the FRC programming environments provide starting templates in addition to code for example robots that accomplish tasks.

2.3.3: Code Scaffolding

One technique for helping students learn to program is to provide code scaffolds. "Scaffolds are tools, strategies, and guides that can support students in regulating their understand [sic] of complex topics".[9] RobotBuilder tries to go beyond providing a generic template and helps generate some code based on user input. This generated code acts as a starting point for the user and is known as 'scaffolding'. All of the three IDEs for the three programming languages also provide example projects that can act as scaffolds, though not as customized to the team's robot. In code scaffolding, the environment or lesson provides boilerplate code and difficult code for the user. The programming environments usually provide boilerplate code as templates in files and typically hide the difficult code in some form of library or framework.

[6] (Gilder 1992)
[7] (M. Kölling 2010)
[8] (M. Q. Kölling 2003)
[9] (Azevedo 2004)

One example of Java code scaffolding for children is the Greenfoot integrated development environment.[10] It provides graphical representations of the class structure and ways of creating and instantiating classes. When the user creates a new class, the Greenfoot environment creates it using a template, so that the user does not face a blank screen. Greenfoot does this to follow the principle that programming environments should "Never start with a blank screen. Starting from a blank screen requires design, and is an advanced exercise. It is something students encounter later, but not as the first contact."[11] This sentiment currently carries through the FRC programming environment in the form of both templates and example robot programs that users can start with and adapt to their needs.

The scaffolding should provide a useful template to help them learn and provide some examples of where to go. Greenfoot does this by generating a Java template for the object type created. Other languages, like Alice, provide a graphical interface for programming.[12] These generally use many graphical cues to help scaffold.

Using scaffolding to generate boilerplate using a knowledge of the desired outcome is not limited to education. It is also common to use a graphical interface that generates a large portion of boilerplate code and some useful code when using many GUI (Graphical User Interface) Builders. The user models the GUI with a graphical interface, using drag and drop controls to generate the code. Usually the generated code has hooks so that the user can extend the resulting code to have the additional functionality, such as the ability to react to button presses and text selections. Some GUI builders, such as the GUI builder for Petri-Net, try to generate all the code from the desired behavior specified in the GUI.[13]

2.3.4: Simulation

In addition to the difficulties surrounding writing code for a robot, programmers on FRC teams often do not have access to the robot until near Stop Build Day, the end of the build season. This is because other team members are usually still working on the robot until that point. After Stop Build Day, the robot must be bagged or shipped and cannot be worked on. In some cases, the students cannot test their code until the day of the competition. One approach to alleviate this problem is through simulation. Currently, NI provides a simulator that only works with LabVIEW and allows students to write programs and test code without access to a real robot. In its current state, it is not possible to import a team's robot; however, it provides three robots that students can program for to facilitate learning. They can also test

[10] (M. Kölling 2010)
[11] (M. Q. Kölling 2003)
[12] (Kelleher 2005)
[13] (Gomes 2010)

the logic of their code if their robot is similar enough to an existing simulated robot. Improvements for simulation of FRC robots that teams can import are currently under investigation.

2.3.5: Documentation

FIRST provides documentation to teams in a variety of formats. One form is the aforementioned integrated help. As software developers with an understanding of the libraries, we often find it among one of the most helpful forms of documentation when programming. When volunteering at events, we have found that other developers feel similarly. Other common forms of developer documentation of the library include JavaDoc (for Java) and Doxygen (for C++). These provide easy access to all of the objects and methods in the library.

Students on FRC teams are not always developers, so the integrated documentation is not enough. For more high-level documentation, they have many options to get a basic understanding of the libraries. ScreenStepsLive has task-oriented documentation for how to use WPILib to accomplish tasks such as moving a motor, driving a robot, tuning a PID controller, dynamically choosing an autonomous mode from a dashboard and many other common tasks. For the more visually oriented teams there are varieties of videos on FRC Mastery and from Brad Miller (hosted on YouTube).

There are also some in-person options for teams to get help. On kickoff weekend, there are a number of Quick Build Day events. At these events, experienced mentors teach whoever is present how to build the KitBot (the basic, drivable robot that is included in all kit-of-parts) and write drive code. Experienced event organizers and teams often host a number of training talks and seminars about how to use WPILib effectively.

Section 2.4: Control System Usability

While using the FRC Control System, there are several non-programming tools to assist students. These include the FRC Driver Station, FRC Dashboard, and FRC Radio Configuration Wizard. Although not directly related to the programming of the robots, these utilities aid in robot operation and help to improve the team experience.

2.4.1: FRC Driver Station

This LabVIEW program is required to operate an FRC robot. It communicates with the robot over UDP (User Datagram Protocol) sending control data such as joystick inputs, and allows the FMS (Field Management System, used to control the competition field) to control the robot's state. It is the only program allowed to communicate with the FMS during any competition. NI has worked to improve the features and usability of the official program; however, some students still feel that different features and tools are lacking.

2.4.2: FRC Dashboard

There are several versions of this tool to allow the teams to send useful data from the robot to the Operator-Interface (OI). Most commonly used (and the only officially supported) are the NI's LabVIEW Dashboard and WPI's SmartDashboard. Both programs have the same goal of displaying data to the user. The FRC netbooks come with the LabVIEW Dashboard. There are several unofficial versions of the Dashboard, such as ZomB Dashboard. These third-party versions are legal to use in regular competition.

2.4.3: FRC Radio Configuration

Provided to teams in the 2013 season (rather than at events only, as in previous seasons), this tool is designed to help configure the Wi-Fi radios used on and off the field. In previous years, teams were required to log into their radios and configure them manually, often causing problems and failures. Teams are now able to setup their radio in less time. Additionally, this permits teams to implement the same data throttling used by the FMS.

2.4.4: Conclusion

FIRST provides teams with multiple tools to help improve their experience, and to make it possible for competitions to run smoothly. They have also demonstrated dedication to improving the experience year-over-year for teams, such as releasing the Radio Configuration tool during build season so teams could easily set up their radios and replicate field conditions. These utilities help teams to easily set up and communicate with their robots, and help FIRST to provide fun and safe competitions.

Section 2.5: Interview Techniques

Multiple different interview techniques are usable for interviewing FRC teams. These techniques vary in the goal, structure, number and grouping of interviewees, selection bias, length and other factors. It is critical to balance these different factors to produce the best results. Compared to surveys, interviews tend to be more time-consuming, have fewer participants and are generally harder to use to produce statistically valid results. However, they excel at offering insight into qualitative questions and provide the opportunity for the interviewer to investigate further into less detailed answers by asking probing questions.

2.5.1: Goals and Interviews

For an interview to be successful, it must have well-defined goals. The goals should generally focus on qualitative data over quantitative data. One study says "The research objectives center on understanding experiences, opinions, attitudes, values, and processes."[14] This is another way of saying that interviews are best for getting detailed answers to questions and gathering a broad array of information. Interviews can be good for getting interviewees' perspectives and concerns, which the interviewer can then incorporate into a survey that is easier to conduct over a statistically valid sample.

2.5.2: Structured vs. Unstructured Interviews

The amount of structure that is included in an interview can vary widely; one extreme is having very structured interviews. An over-structured interview is essentially just a survey administered by a person instead of a computer; this lacks both opportunity for follow-up and the ability to probe for deeper answers. The other extreme is unstructured interviews; these are very conversational. The main differences from a typical conversation is that the interviewee speaks much more than the interviewer does, and that the interviewer sets the general subject and area of inquiry. Each of these two extremes has differences in future analysis and comparison of collected data.

Comparing structured interviews is more straightforward than unstructured interview results. You can typically apply any analysis and comparison method that you would use for surveys, with the one caveat being that your sample size is likely to be too small to generalize to the general population. On the other hand, unstructured interviews are very difficult to analyze and compare. You have to read the interview transcripts and find threads of conversation on similar topics to compare whether interviewees agreed or disagreed about the topic. This type of analysis can be much harder and more time-consuming to implement when compared to structured interviews.

Fortunately, structured versus unstructured is not a dichotomy. It is possible to have various levels of structuring in an interview. Experts call these interviews semi-structured interviews.[15,16] In a semi-structured interview, the interviewer typically has a small number of high-level questions to the interviewee. The interviewer then has the freedom to probe to get additional information from the interviewee. The interviewer can draw possible probing questions from a precompiled list, or from the context of the conversation. High-level questions categorize the results across all interviewees, making comparing and analyzing the results easier.

[14] (Rowley 2012)
[15] (Bolderston 2012)
[16] (Rowley 2012)

Semi-structured interviews rely on the interviewer asking good probing questions. To do this, the interviewer must be attentive and able to ask relevant questions. The following is a list of generic probing questions that an interviewer can quickly draw on[17]:

- "Please say more about that.
- "Can you give me more details?
- "What is it about ...?
- "Tell me more about …
- "What is your experience with ...?
- "Describe …
- "Imagine …
- "What caused you to ...?
- "What features of (the topic) do you particularly like/dislike ...?
- "How... When... Where... How often do you...?"

The amount of structure chosen depends on the goals of the research.[18,19] For highly exploratory research, it is best to have very open-ended questions. On the other hand, surveys should replace very structured interviews unless having a person administer it is necessary to get sufficient response rate.

2.5.3: Interview Method

There are several methods for conducting interviews. They typically depend on which medium the interviewer chooses to conduct the interview. Conducting face-to-face interviews is different from phone, IM (Instant Messaging), or email interviews. Reasons for choosing different mediums vary widely:

- Geographic nearness of interviewers to interviewees makes in person more feasible.
- Scheduling difficulty makes phone and internet-based methods more convenient. Email can be especially convenient due to its asynchronous nature.
- Removing biases can lead to mediums where interviewers and interviewees cannot see each other.
- When body language and other non-verbal and non-written clues are important, face to face are more important.
- If anonymity is preferred or the topic is embarrassing, it is often better to use internet-based communication.

[17] (Bolderston 2012)
[18] (Bolderston 2012)
[19] (Rowley 2012)

Another major factor is whether to interview a single person or a group. Groups can often provide better results in areas where discussion between members can lead to better ideas and new perspectives. However, groups can be harder to control. As a result, the interviewer can run into issues with groupthink, and shyer members tend not to speak up. It is also harder to schedule groups of people. Given this, it may be better to interview people individually.

2.5.4: Number of Questions and Interview Duration

Two key factors in administering an interview are the number of questions to ask and the duration. The medium and the number of interviewees influence both of these factors. Group interviews typically take longer than individual interviews as there are more people's views to cover. It is important that phone interviews should not take longer than a half hour since the interviewee gets distracted and fatigued leading to lower quality data.[20] Even in person, it is uncommon to have interviews last longer than an hour. It is necessary to take into account these factors when deciding how many questions to ask and how long to schedule. The number of suggested questions varies from four to twelve[21, 22] These numbers depend on how structured the interview is and to what extent the interviewer has the freedom to ask follow-up questions.

Section 2.6: Survey Techniques

A survey of the FRC teams can use multiple techniques. These techniques include survey sizes, sampling proportions, blocking factors, and ensuring response rate. Balancing these factors is important to avoid survey fatigue, ensure a good response rate, and obtain statistically valid information.

2.6.1: Blocking Factors

When conducting surveys, it is important to identify sources of possible bias. One way to address this issue is to split the population into groups by important characteristics. These groups are called blocking groups. If they are not considered, it is possible to miss important variations in data specific to each group. An excellent example would be researching information on how people feel about political parties in America without blocking by party. The results would be uninformative, since the results would aggregated as a whole and no analysis can be done to determine if a party favors certain policies. However, if the data separates respondents by political party, then researchers can draw conclusions about each party, including the types of policies they favor and where they prefer to spend money. For our survey, we can block by several factors. An obvious example is years of participation in the FRC

[20] (Bolderston 2012)
[21] (Rowley 2012)
[22] (Bolderston 2012)

program, which affects the level of experience team members have. Another factor is location, as some locations may have access to more help. Teams near the NI headquarters in Texas have better access to help with LabVIEW, while teams in the New England area have access to the WPI team that develops the C++ and Java libraries.

2.6.2: Censuses

One of the most important factors to consider about a survey is the sample size. Sample size affects the survey in many ways. A census is one option that surveys every member of the population. An important advantage of a census is that all teams are able to give feedback. This helps all teams feel that their feedback is being heard, which can increase satisfaction. It also ensures maximum exposure to feedback, allowing all teams can report problems. Finally, surveying the entire population allows a census provides a very high confidence level.[23]

Censuses also have disadvantages, which are very important to this survey. First, a census results in lots of feedback. Much of this data can be easy to analyze, as most of the answers should be categorical, and can automatically sorted by the survey software. However, for open response questions, it is very hard to go through all of the responses. Since there where 2,508 active teams during the 2013 season, even one open response question has 2,508 potential responses. While there will not be a perfect response rate, and not every open response question applies to all respondents, there is still a large amount of data to process. Additionally, a census is much harder to follow up on. Even if we had a 90% response ratio, that leaves 251 teams to follow up with and attempt to determine why they did not respond to the survey. Finally, censuses can limit the response rate of future surveys, since giving too many surveys leads to survey fatigue.[24] This is a very real concern for FIRST, as they conduct several censuses every year. These surveys include the Kit of Parts satisfaction survey and volunteer surveys, both of which have already gone out to teams.

Determining the sample size necessary for statistical validity in a census is relatively simple. To do this, one must first choose a desired precision level for the survey, designated as L, where L is the level of confidence in the results of the survey. Most censuses target an L value of .95, which is 95% confidence. However, it is acceptable for some surveys to use an L value of .90, which is 90% confidence. For our survey, we should target an L value of .95. Since we are not trying to measure very specific phenomenon and only trying to get an idea of the improvements necessary for the control system, we can still be accurate with an L of .90. The variable in the equation is the population proportion p. p is the proportion

[23] (Henning 2009, 8, 9)
[24] (Henning 2009, 9)

of the target population with a desired characteristic. If p is unknown, you can use 0.25 to get an over-estimate of the necessary sample size. After p is determined, d (the margin of error) is found. This d value is the confidence that the researcher has that the data is within d units of L. Most studies have a d of .03 or .05, but some can range up as far as .1.[25] The final sample size of the survey is n, and z is the t-value (a statistical value for a normal distribution response set based on the confidence level of the survey. The formula refers to this as z_1) of our survey, which is obtained by looking at a distribution chart for the student's t-test distribution. All of these come together in the following formula to give the final sample size for a census: $n = \left[p(1-p) * z_{\frac{1+L}{2}}\right] / d^2$.[26]

2.6.3: Random Sample

Another survey size technique is random sampling. A random portion of the population is selected and surveyed, and then their responses are representative of the rest of the population. For the sample to be statistically valid, there are several considerations. The first is ensuring that random sample is representative of the population, by looking for factors that could affect the outcome of the survey. We can then account for these factors, ensuring that our random sample contains the same proportion of teams with these factors that the full population would contain. These factors are the blocking factors. To ensure that the survey is statistically valid, we attempt to determine the proportions of these populations in the group, and then ensure that the sample contains the same percentage of each group.

Determining sample size for a random sample is dependent on the calculations. There are multiple factors in determining this calculation. The first is the type of data being surveyed, continuous or categorical. The primary blocking factor determines this distinction. Continuous data is a variable that can be any value within a given range. For example, if blocking on the age of each team would be continuous data. The other type of data is categorical. Data is categorical if there are multiple discrete sections, and a team must fall within these sections. For example, we can split the teams into distinct groups by team location.

The data that we are surveying determines the acceptable margin of error. This is a measure of how accurate the results of the study are. The typical margin of error in a survey with categorical data is 5%.[27] Another important factor is the alpha level. This is the risk that the actual margin of error is higher than acceptable. The alpha level determines the t-value used in analysis of result. The typical survey has

[25] (Petruccelli, Chen and Nandram 1999, 256-257, 905)
[26] (Petruccelli, Chen and Nandram 1999, 256-257, 905)
[27] (Kotrlik and Higgins 2001, 45)

an alpha level of .05, which results in a t-value of approximately 1.96 in survey sizes over 120.[28] The final important piece of data for determining survey size for categorical data is the population variance. Variance has a simple definition: the variance is the average of the squared differences from the mean. With categorical data, we cannot obtain the exact variance, but we must estimate it by following this formula: $p/(1-p)$, where p is the proportion of the largest category. All of these factors can then be plugged into Cochran's sample size formula, which is $(t^2 * \sigma)/d$, where t is the t-value calculated earlier, sigma is the variance, and d is margin of error.[29] Depending on the resulting population size, it may also be necessary to apply Cochran's correction formula. This is necessary when the resulting sample size is greater than 5% of the population. This formula is $n_0/(1+\frac{n_0}{Population})$, where n_0 is the original calculated sample size and Population is the total size of the real population.[30]

The final factor is to determine necessary oversampling that will result in the actual response rate being high enough to be statistically significant. While there are multiple ways of getting a good guess via formulas, the best method is to look at past surveys and see what their response rates have been and then to simply use that rate. It is the most accurate method, if the population that the two surveys are targeting has not changed significantly. In the FIRST community, there is a Kit of Parts survey every year. This is a census style survey to all active teams about their satisfaction level with that year's Kit of Parts options. Since the completion of the 2013 Kit of Parts survey, we can use the response rate from that to calculate the necessary oversampling since both surveys target teams that were active during the 2013 season. For the 2013 Kit of Parts survey, the surveyed population was all 2508 teams active during the 2013 season. FIRST requested that every team have a mentor and a student fill out the survey. Of this sample size, 859 unique team responded, and 293 of those 859 teams had multiple members take the survey. This sample included 302 respondents chose not to disclose their team number. If we go with the number of fully complete surveys, 293, that is a completion average of 12%. However, we can go with the next highest number, 859 respondents, since the most important number is the amount of unique team perspectives. This leaves us with a completion rate of 34%.[31]

Section 2.7: Online Survey Tools

Several tools and websites are popular for online surveying. These include Qualtrics, SurveyMonkey, Google Forms, and Constant Contact. They offer many features, both free and

[28] (Kotrlik and Higgins 2001, 46)
[29] (Kotrlik and Higgins 2001, 47)
[30] (Kotrlik and Higgins 2001, 47)
[31] (FIRST 2013)

subscription-based. It is important to choose the most useful one to get the best results. Before we designed the final survey, we gathered some basic requirements on what the software needs to be able to do. A list of our requirements follows: paragraph and single word questions, multi/single checkbox question, skip logic, exportable data, and optionally randomized questioning.

2.7.1: Available Tools

We considered several commonly used online survey tools for this project. Six common survey solutions used by students and professionals are listed below, along with some of the features of each:

Tool	*Cost*	*Skip Logic*	*Exportable Data*	*Randomize Questions*	*Customizable*	*Other Limits*
SurveyMonkey Free	$0.00	No	No	No	No	Ten question limit, 100 responses limit
SurveyMonkey Gold	$300.00	Yes	Yes	Yes	No	SurveyMonkey branded
SurveyGizmo Free	$0.00	No	Yes	No	No	8 Question types, 300 responses max
Google Forms	$0.00	Partial	Yes	No	No	Non-unique responses, maximum responses per survey length
ConstantContact	$15.00	Yes	Yes	No	Yes	N/A
Qualtrics	$0.00*	Yes	Yes	Yes	Yes	N/A

*Table 1: Survey Tool Options. *only with campus license*

2.7.2: Paragraph and Single Word Answers

Often it is not possible to represent a respondent's feelings through a close-ended question. The use of open-ended questions allows the surveyor to get exactly what the respondent is thinking in full details. This type of question has the potential to yield the most information, but due to the difficulty in answering and analyzing, it can be difficult to work with. Additionally, the answers may generate irrelevant data if the respondent does not answer the question properly.

2.7.3: Multiple-Choice Question

In scenarios where there are discrete responses, it is useful to limit the user to selecting one or more choices from a list of options. This allows the respondent to answer the question that best fits how they would respond, as well as making data analysis easier. Due to the closed-ended questions only having a few possible answers, it is faster to complete and allows the survey to ask more questions than a short answer.

2.7.4: Display Logic

When designing a survey for multiple topics, it is desirable to target certain questions to specific people based on their responses. For example, display logic would allow respondents who have used RobotBuilder to answer questions related to the use of it. Otherwise, a respondent may become flustered quickly if he/she is answering questions about a program or tool they have not used.

2.7.5: Exportable Data

Viewing results on a webpage may be useful for some applications, but other applications need to be able to access the data to provide features such as ranking, grouping, or dividing samples for further analysis. In order for that, the software must allow the user to export the results into a format that the team can use to analyze the data.

2.7.6: Randomizable Questions

In a survey with multiple studied programs, it is possible that respondents will have used all of the programs. If this were to occur however, it would result in a large time commitment to answer all of the questions. In a case like this, it may be useful to randomize the topics that the survey asks about, and limit the length of the survey. It would be a helpful feature to prevent lengthy surveys.

Section 2.8: Conclusion

The existing body of knowledge surrounding the FRC Control System and techniques for teaching programming to students is extensive. Techniques and tools used for interviews and surveys are almost as important as the questions themselves when investigating the how the software works. There are many options for interview and survey techniques. These include structured or unstructured interviews; census or random sample surveys; and how to target populations to ensure valid responses. There are also tools available for administering and monitoring surveys, with a plethora of options for survey creation and data analysis. These are among some of the decisions that we had to make when designing the methodology of this study.

CHAPTER 3: METHODOLOGY

Section 3.1: Interviews

We conducted a set of interviews with students and mentors before creating the survey, in order to help ensure that we were asking the right questions on the survey. This meant that we needed to conduct them before the survey was finalized to ensure its validity. This section describes the method used for these interviews.

3.1.1: Interview Details

This section discusses a number of our decisions on how we conducted the interviews. Our research and understanding of FIRST and members of the FIRST community forms the basis of our rationale for these decisions. The following list highlights the main details:

- Location: Post Season Event (Mainely Spirit)
- Structure: Semi-structured
- Duration: Up to half an hour
- Length: Six open ended questions
- Subjects: About six pairs of a mentor and student from an FRC team

We chose Mainely Spirit as the location for two main reasons. First, it has a number of individuals who would be interested in participating in one place. This makes recruiting and coordinating the interviews with participants easier. This was important, since we chose in-person interviews. Second, conducting them at this event fits this project's timeline; it was soon enough that we could integrate the results into the survey, but far enough into the project that the interview could be designed and approved by the IRB in time.

Another early decision was the structure of the interview. We chose in-person interviews to allow smoother communication, better rapport building and less technical issues such as phone failure. Since the interviewer's purpose is to ensure we were asking the right survey questions, we should not conduct an overly structured interview. However, there are a few major points that the interviewers want to address in all of the interviews. This implies that semi-structured interviews was the ideal format. This provides enough flexibility to get feedback from interviewees based on what the participant's interests.

The next decision we made was duration. Generally, a longer interview allows participants to respond to more questions and the interviewer to attain a deeper understanding of the participants' experience with the control system. However, we must balance this with not scaring away potential participants with a long time commitment and not fatiguing them. It is possible that fatigue can result in declining quality of responses as the interview goes on. A half-hour interview allowed us to ask six open-ended questions targeting 5 minutes per question for answers and follow-up questions. According to our research in Section 2.5:, this length of time allowed us to ask all of the open-ended questions we wanted to ask.

Along with duration, another decision was the number of questions to ask. Too many questions leads to not enough time to get into depth on the questions, but too few questions leads to either not as much information or much more unstructured feedback. Based on our half hour duration and previous research, six questions seemed like a good balance that fit our needs and produce reasonable results.

3.1.2: Method

The method used for conducting the interviews was straightforward. The event organizer helped recruit participants using a combination of emails and announcements. Interested pairs of mentors and students signed up for different interview time slots on the day of the event. In some instances, only one team member signed up if the team did not have the relevant student or mentor attending the event.

We interviewed participants at scheduled times. Each interview followed the script in Appendix A. During the interview, there were three researchers present in room. One researcher set up the microphone and was responsible for taking notes of the interview for later reference. Another researcher followed the script, kept the interview focused and occasionally asked probing questions. The final researcher listened intently to what the participants said and focused on trying to understand what the participants were trying to say and ask the right probing questions.

Breaking the roles up this way allowed the interviews to proceed smoothly and ensure that we gathered and recorded the proper information from teams. Each interview resulted in a recording and detailed notes that could be examined. Following the interviews, we analyzed the results to find what teams thought about the control system and to make sure we were asking the right questions in the survey. The following chapter discusses the process of how the actual results influenced the survey.

Section 3.2: Pre-Survey

After we incorporated the results from the interview into the survey, we decided to run a small pre-survey. Goals of the pre-survey included the following:

- Ensure that participants could fully express how they felt
- Ensure that the survey was presented properly
- Verify that the tools worked

Just like the interview, we incorporated the results of the pre-survey into the final survey. This helped ensure that the survey would allow us to answer the questions we wanted to with respect to teams' ability to succeed.

3.2.1: Pre-Survey Details

The pre-survey is very similar to the final survey, while one major difference is the target audience. To accomplish the goals, it is only necessary to test with a small sample of people who were involved with an FRC team for the 2013 game. We wanted to target a small enough population that was unlikely to take the final survey and were easy to contact. For this purpose, we chose to email to a list of WPI community members who indicated interest in FIRST. Since WPI has a large number of FIRST Alumni, this email list seemed likely to have enough relevant participants to get the information we wanted out of the pre-survey.

3.2.2: Method

After incorporating the results of the interviews, we sent the survey was to the email list mentioned above. Appendix B contains the questions asked and the email notification. We waited three days for responses and then analyzed and incorporated the results into the final survey.

Section 3.3: Online Survey

The final survey incorporated feedback from the pre-survey and the interviews. We distributed the survey through multiple social media sources, such as Facebook and online forums. We also determined approximate necessary sample sizes for statistically accurate data, assuming a random sample. However, due to the distribution method, we are expecting to see some amount of bias.

3.3.1: Survey Size

Since we are doing a census, we used the formula specified in Section 2.6.2, which is $n = \left[p(1-p) * z_{\frac{1+L}{2}}\right]/d^2$. For definitions of all these terms, see section 2.6.2:. We do not know p in this survey, so we must use 0.25 as our $p(1-p)$ value. We have multiple possible z values for our survey, one for an L value of $.90$ and one for an L value of $.95$. For $L = .90$, the z value is 1.282, and for $L = .95$, the z value is 1.645. Finally, there are 2 values for d, which are $d = .10$ and $d = .05$. Plugging in all of these values results in the following survey sample sizes:

	$\boldsymbol{d = .05}$	$\boldsymbol{d = .10}$
$\boldsymbol{z = 1.645}$	271	68
$\boldsymbol{z = 1.282}$	165	42

Table 2: Response Size Probabilities

3.3.2: Survey Questions

Our final survey questions fall in seven major categories as shown below. The actual questions are contained in Appendix C. Additionally, Appendix C contains a flowchart depicting the general question flow of the survey.

Category	Feedback
Demographic	• Basic questions about themselves • Team number • Mentor/Student • Team involvement
Goals Selection	• Goals selected • Success with goals • Goals avoided
Failure Feedback	• Issues teams have with goals • Focused on higher level problems
Tool Usage	• Tools used • Documentation used • Tools known • Documentation known
Tool Feedback	• Whether respondents would use unknown tools • Whether they would use tools again • If they have issues with the tool they used
Documentation and Help	• How they get information • Which sources are most useful • How they learn about the control system
Final Thoughts	• Anything else respondents want to let us know

3.3.3: Distribution Method

We distributed the survey through multiple online sources, and used Qualtrics' source and referrer tracking to make sure that we could determine where our survey population came from. The source tracker allowed us to put a parameter in the link we distributed to teams, which Qualtrics recorded with the actual survey data. We were then able to associate answers with sources, and it allowed us to make correlations between where respondents found the survey and their responses on the survey. We used a set of randomly generated numbers for our sources, shown in the table below.

Source
ChiefDelphi Programming Forum Post
FIRST Alumni Facebook Feed
Brendan's Facebook Feed
Fredric's Facebook Feed

Alex’s Facebook Feed
FRC Team 1058 Facebook Feed
Official FIRST Facebook Feed
National Instruments Twitter's Feed (@nifirstrobotics)
Brendan's Twitter Feed (@brendankm7816)
Paul (Mentor on FRC Team 254)
Jamee (Mentor on FRC Team 2648)
Karthik (Mentor on FRC Team 1114)

Table 3: Source Reference Numbers

The referrer tracking in Qualtrics allowed us to see what URL respondents were on before they opened our survey. This allowed our later analysis to see if our links where shared on other websites. For example, if our ChiefDelphi link is shared to another online site, such as reddit.com, we can see that referrer URL was from reddit.com, not chiefdelphi.com.

We took these steps for two reasons. The first is that source and referrer tracking allows helps us to determine where we see bias in our final sample. We can filter out respondents by site and see if our data changes, which is useful for making correlations. The combination of source and referrer tracking also allows us to filter responses. If our survey was posted to a spam board, we would be able to filter out invalid responses by removing all responses from a source with invalid referrer URLs.

3.3.3.1: Primary Distribution Channels

Our primary distribution method for this survey was a combination of social media channels. One of the main channels was a post in the programming sub-forum on ChiefDelphi.com. Brad Miller, the lead developer for Java and C++ versions of WPILib, made this post. We also made multiple posts to Facebook. Each of us made posts on our own walls, and asked friends to repost these. In addition, we made posts to the FIRST Alumni group and to other team’s pages. NI tweeted our survey from their NIFirstRobotics account. Finally, we asked mentors from FRC teams 254, 1114, and 2648 to spread the word about our survey as well.

CHAPTER 4: RESULTS

Section 4.1: Interview Results

We interviewed eight teams, including 12 subjects consisting of eight students and four adult mentors. The results of these interviews were incredibly helpful for ensuring that the close-ended survey questions had choices reflective of actual teams and that we asked the appropriate questions. Using these results, we were able to discover several possible issues that we needed to address in the final survey. We discussed in detail the problems teams had ranging from documentation to personal errors using the FRC Control System.

Several of the teams interviewed discussed similar problems that all had related underlying causes. The interviews showed that one of these underlying causes is the inability of teams to locate tools, documentation, and help. It was clear to us that we would need to ask the FRC population about how they learned what they know about the control system, and where they go when they want to learn more.

4.1.1: Knowledge of Tools

We asked interviews subjects if there were any tools that they feel may have been helpful to the development of their robot's control system. One team was looking for a way to "Allow values to be printed and plotted on a computer to see the output". They were surprised to learn that a tool exists, and does exactly what they were looking for. Another team wanted to simulate their code and have it run on their computer; they were shocked to learn that the tool exists in LabVIEW and FIRST distributes it to all teams.

After several interviews that revealed the same problem, it became clear that we should add questions to the survey regarding how teams get help, and whether or not they know what tools are available to them. If a team does not use a tool, it may be because they do not know it exists, rather than thinking it is too complicated.

4.1.2: Looking for Help

We found that teams struggled with where to find help if they encountered a problem. Often, teams expressed that they loved the integrated help files, but felt that they were often lacking information about how to use the functions and tools. This knowledge prompted us to incorporate questions regarding where teams went when they needed to find information and what types of information they found useful into our survey. It also became apparent that teams needed to look at multiple sources of documentation to solve their problems, and that no one source contained all that they needed. Using that, we added questions regarding what type of information is missing from each source of help.

4.1.3: Libraries and Objects

Although a team may be knowledgeable in the use of NetBeans, Wind River, or LabVIEW, several teams noted that they were unaware of the extent of the libraries available to them. For example, although a team is using Java as their programming language, they may not use the `RobotDrive` class to drive their robot because they do not know it exists. When exposed to the object, they may feel that it is very simple and helpful, but the lack of knowledge of its existence may be what causes it to go unused. This makes us curious as to how we can properly relay vital information like this to teams. Due to this, we have included questions regarding where teams expect to learn about the tools and libraries available to them.

4.1.4: Different Help Formats

After interviewing a few teams, there was a common trend of teams using different types of help depending on what they were trying to accomplish. One team noted that they found videos that showed not only the code, but also what the robot does with that code very helpful for controlling robots. They also noted that integrated help was more useful for solving syntax issues, and language specific problems. Based on that, we asked questions relating the type of problem that a team has, and the type of help that they use.

4.1.5: Summary

Overall, the quality of the feedback received was very good. Most teams felt that the amount of work that was required to program their robot was just right, and they learned a lot from the process. Most teams felt that their goals were successfully achieved, but not without problems along the way. We learned that it is not only important to gather information from the teams that were not successful, but also to learn from teams that encountered issues along the way.

We were able to modify our survey to reflect the information that we feel we need to gather from subjects. Additionally, we changed several opened-ended questions to close-ended questions by adding possible expected responses as multi-checkbox questions. Adding these types of questions not only reduced the time to complete the survey, but also greatly simplified the larger data analysis procedure.

Section 4.2: Pre-Survey Results

We distributed the pre-survey through the FIRST-interest email alias at WPI. There were 33 responses to the survey, and 11 of the respondents fully completed it. Twelve respondents left after the first question (which asked the respondent whether they are a WPI student). Another five respondents ended the survey after question four, which asked respondents how many years they have participated in FIRST. The final five incomplete respondents dropped out somewhere during the initial section of the survey, where respondents are asked about their team’s goals during the 2013 season. All of the respondents who made it past the goals section fully completed the survey.

These responses helped show us several technical issues we missed. They also helped us to ensure we were giving respondents the answers they were looking for, instead of having to write in a response in the other box. Preventing respondents from writing in their own responses helps in the analysis. We also found several questions that were confusing to respondents, as well as other wording that was improved.

4.2.1: Conflating Answers

When we did some practice data analysis on our responses to the pre-survey, we noticed that we had worded one of the questions in a way that conflated two answers. In question 58, we asked respondents about their opinions on all documentation sources they said that they used. Our original question wording was “Overall how useful did your team find the content on the documentation tools?” We then had a four-point scale, with every column having a different set of answers. These ranged from “Difficult to understand; useless” to “Easy to understand; useful”. In our data analysis, we found that we could not tell the difference between whether teams rated a documentation source low for the usefulness of the content, or the ease of use of the source. We changed this in the final survey to be two questions, one asking about the quality of the content in the documentation, and one asking about the ease of use of the source. We also added in another point to the scale, making a five-point scale question. Finally, we took the headers off most of the columns, just putting them on the first and last column. This made the question require less reading on the part of the respondent, and we were able to tell what respondents thought was good or bad about sources in our analysis.

4.2.2: Respondent Involvement

One of the things that we noticed while doing practice analysis on our data from the pre-survey was a large number of write-in answers that dealt with mechanical issues, which were outside the scope of this project. For example, Question 14 asked respondents “Did you have any controls problems shooting Frisbees in teleop? (Select all that apply)”. We got a write-in response saying that the respondent’s team had issues with motors stalling. We believe that we got responses from team members who were not heavily involved in the controls side of the robot, and were instead part of the mechanical design teams. This is fine for the pre-survey, since we were only trying to give the survey a dry run to be sure that it works. For the final survey, however, we wanted a way to filter by involvement in the control system. We added in a new question, “How involved were you with the control system during the 2013 season? (Controls encompasses software and tools for interacting with the CRIO, speed controllers, sensors, etc.)" to help us distinguish between the respondents.

4.2.3: Other Fixes

There were several other corrections made for our final survey. When asked why they chose a particular programming language, multiple respondents used a write-in answer to say that it was due to availability of local help. We integrated this into the final survey as an option under all of the questions regarding language choice. Another minor, but important change was in the wording of one of the pre-worded goals in question 6. Our original wording was "Climb the Pyramid", but we got a few write-in responses where the respondents said that their team wanted to climb the first level of the pyramid. We intended that question to encompass that answer as well, so we changed the wording in the final survey to be "Climb to any level of the Pyramid". We also corrected a couple of display logic errors on questions that were never shown to respondents, but should have been. We also found a few grammatical and spelling errors when doing our practice data analysis and corrected them.

4.2.4: Summary

The pre-survey proved to be a very prudent step in the development of our final survey. We caught several errors that would have made data analysis more difficult, especially with conflating ease of use versus usefulness of content. We were also able to modify our survey to ensure that all respondents could express their opinions on the control system without issues. We were also able to ensure that we could filter respondents by their self-proclaimed involvement with the control system in 2013, and we were able to polish the wording of several questions to ensure that respondents understood the question we were asking.

Section 4.3: Survey Results

This section analyzes the 176 completed responses to the survey and highlights how the response demographics compare to the more general demographics of the FRC population. This influences how strongly the resulting analysis generalizes to the entire population. Since we distributed the survey over a variety of social networks, it is especially important to determine how close the sample was to a true random sample. This section first discusses the demographics of the responses, then looks at how language usage compares to the official numbers gathered by FRC usage reporting software.

4.3.1: Demographics

This subsection discusses the demographics and other metadata of the responses. We will discuss how the respondents found out about the survey, where their teams were located, and their general experience with FRC.

4.3.1.1: Responses by source

As seen in Figure 1, the biggest source of responses was ChiefDelphi, accounting for 64% of our completed responses. The figure shows that the FIRST alumni group on Facebook was our next biggest source of responses. The third biggest contributor were several other Facebook posts on the walls of the researchers and other FRC-related groups. The fourth biggest was NI's (National Instruments) tweet about the survey. Finally, a few mailing groups brought in additional responses.

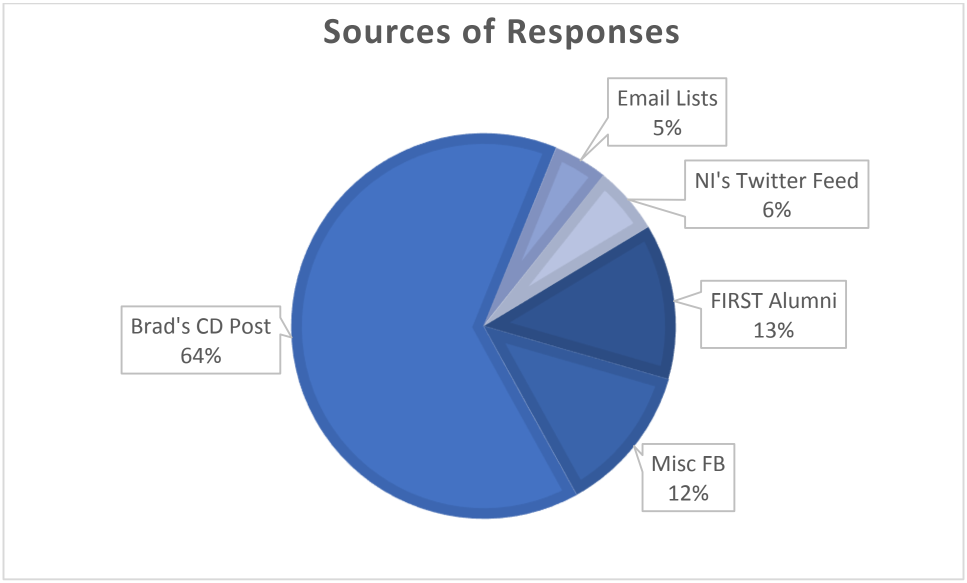


Figure 1 Sources of responses (Question 67).

We tracked responses by inserting a unique ID in the URL of each link. The IDs are related to the social media post or distribution method of the link. While we had more than a dozen unique ID's, two accounted for more than 75% of all the responses we obtained. In Figure 1, we grouped responses to various Facebook posts (excluding the FIRST alumni Facebook group) together since none of them had a significant number of responses. The email lists all include multiple sources, since we had multiple emails sent out.

Based on the sources responses came from, it is easy to see that the types of teams that are active on ChiefDelphi in the fall will be overrepresented compared to teams who are less active on ChiefDelphi during this time period. Another 31% of responses came through teams that responded after getting the message through social media. This means that teams that are not actively checking online are less represented in our data.

4.3.1.2: Map of responses vs. map of teams

The geographic distribution of survey respondents spread across the three countries with the most FRC teams. Seen in Table 4, most responses were from the US, which is home to the majority of FRC teams. Canada has the next-largest number of existing teams. The number of Canadian responses is what one would expect given our sample size and their percentage of the FRC population. Israel is overrepresented by a factor of three, but this is likely due to combination of small sample size and the network effect that occurs when using social media to distribute a survey. Other countries represent a small percentage of the FRC population. No one from these countries completed the survey. Due to this, there may be certain perspectives from teams that do not use English as a native language that this survey does not capture.

Country	Teams (Based off NI data)	Number of Teams
USA	2244	155
Canada	119	10
Israel	47	11
Mexico	37	0
Other	24	0

Table 4: Most common countries and the number of responses from each country (Extracted from team number, Question 2)

Within the responses from the US, the distribution of teams state by state is representative of US teams. States with many teams, such as Texas and California, have a higher percentage of the population in both on the left and right of Figure 2. Some states with less teams did not have any respondents. This is expected with our sample size.

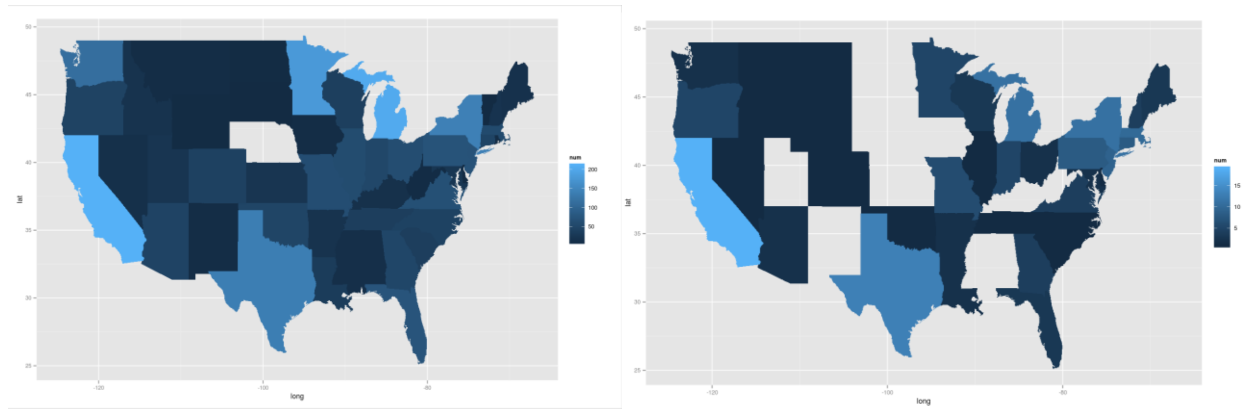


Figure 2: Map of number of teams with the lightest states having the most teams. Left shows the number of teams in each state base off of data from National Instruments and the right shows the number of responses from each state for the survey. (Extracted from team number, Question 2)

4.3.1.3: Mentors vs. Students

Figure 3 shows that the number of responses acquired are nearly evenly divided amongst mentors and students. Due to this, our data is balanced between the opinions of those who were learning to use the control system and those who were teaching how to use it. While we had asked for a response from a mentor and a student on each team, we only had four pairs of student and mentor responses from the same team. This prevents us from comparing what mentors and students in the same situation felt with any statistical significance.

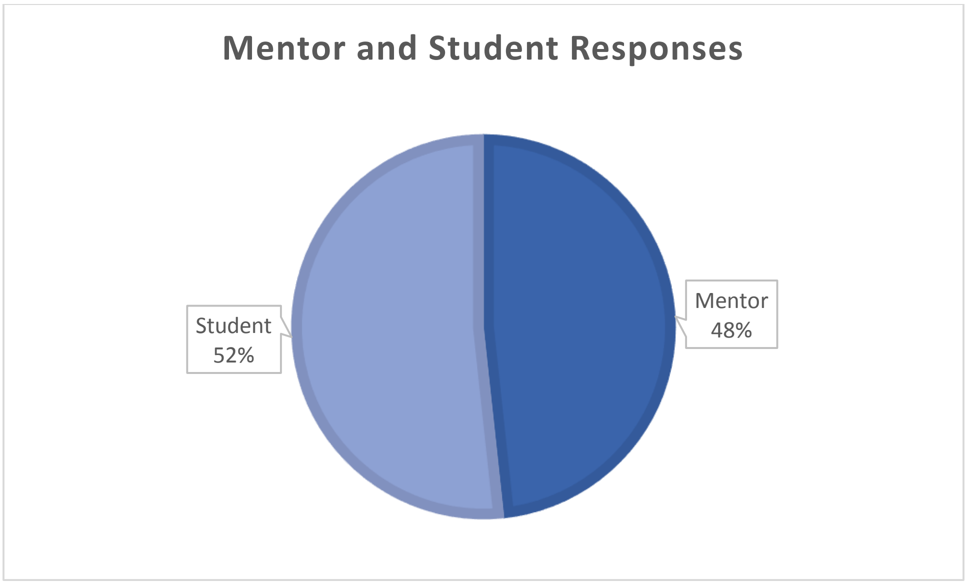
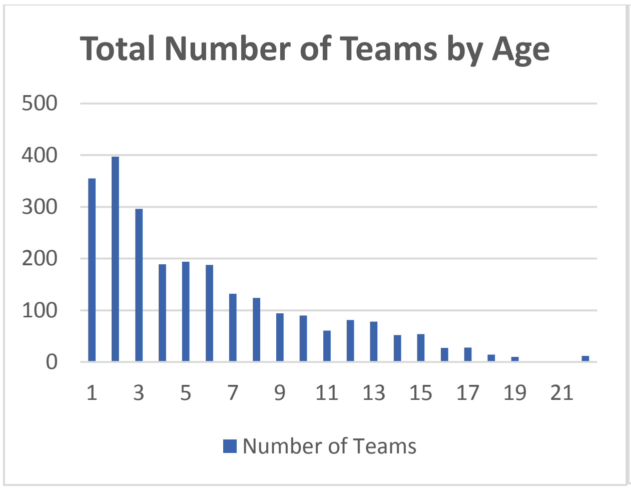


Figure 3: Mentor and Student Responses (Question 1)

4.3.1.4: Team Age

Looking at Figure 4, it becomes apparent that the responses to this survey are disproportionally from older teams. Since our teams are older than a random sample would be, it is important to consider that the teams have had more time to be acquainted with how FIRST works, where to find information and how to organize themselves. This means that there may be problems that affect rookies that this survey does not reveal. A future survey targeting first through third year teams may be useful for identifying problems that younger teams experience and older teams have acclimated to or deal with through organizational knowledge.



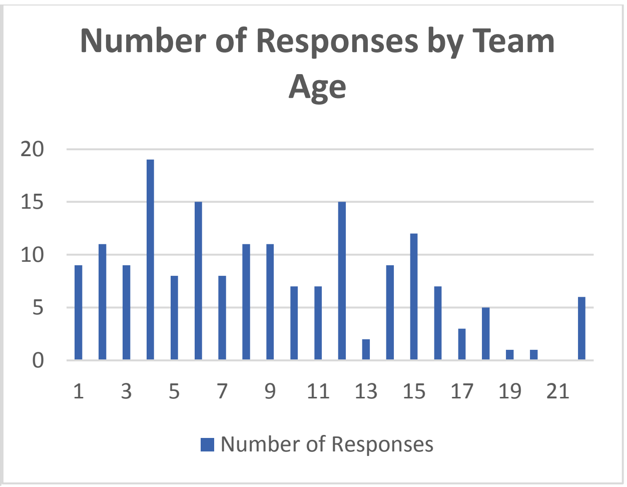


Figure 4: Left shows the number of teams by age; right shows the age of the team respondents are on (Extracted from team number, Question 2).

4.3.1.5: Respondent Experience

As shown in Figure 5, most respondents have four or less years of experience, which is expected given that over half of the responses were students and high school students typically spend four years in school. Some respondents have been participating FRC for many years. The oldest response the survey got was a mentor who has been participating in FIRST for 19 years. Many mentors have over 5 years of experience, so they have experience not only with the current control system, but also with the previous control system. This provides a broader perspective of the control system than only having seen one iteration of the control system.

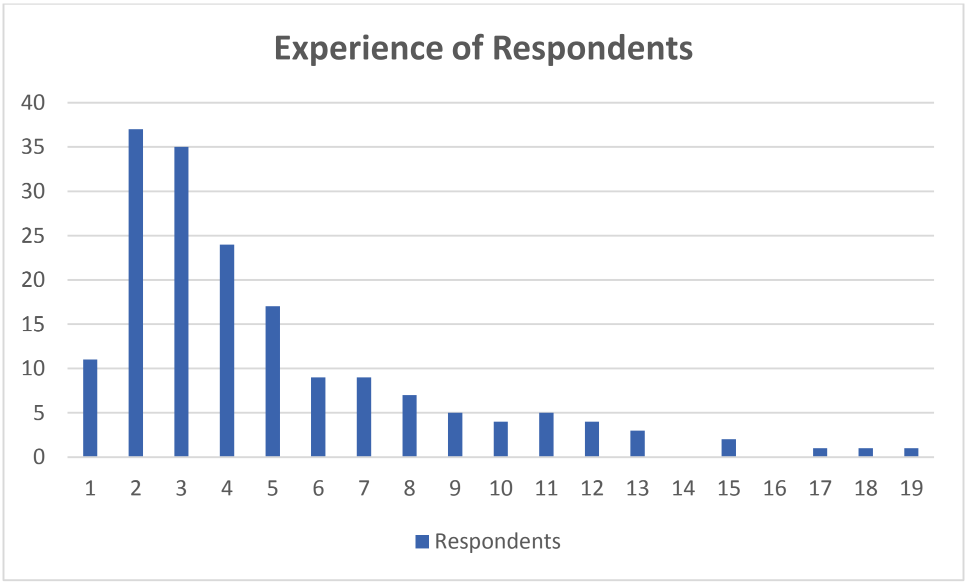


Figure 5: Experience of respondents to the survey (Question 3).

4.3.1.6: Respondent Involvement

Almost 75% of respondents to the survey felt that they were highly involved with the control system on their team. This is shown in Figure 6, which also shows that the most of respondents felt they were at least somewhat involved. Only one respondent did not feel that they were involved at all. This shows that the messages we used on social networks were targeted clearly enough that for the most part, only people involved with the control system responded. The data will therefore reflect the opinions of people who have used at least part of the control system.

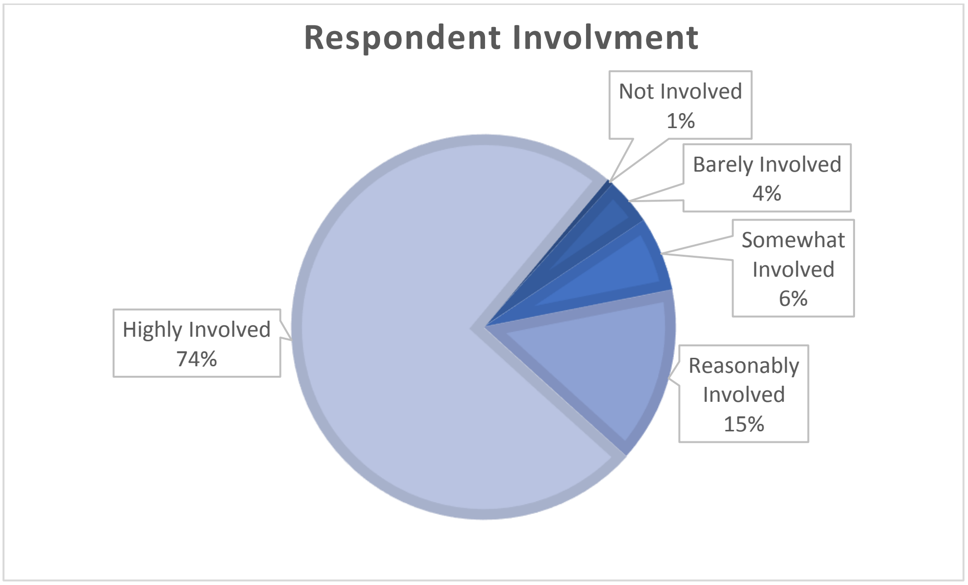


Figure 6: Involvement of respondents with the control system during the 2013 FRC season (Question 4).

4.3.2: Programming Language Choice

When teams connect to the field, the robot reports what programming language the team used along with other usage data. NI has access to this information associated with team number. This allows us to compare the true distribution of team language usage to that of our sample. The NI data includes a small number of teams that are reported as either Python or Unknown. Together, those account for less than one percent of responses. This is too small to determine anything meaningful within our 10% margin of error, so this report ignores them.

Figure 7 shows the true distribution compared to the results of our survey. Looking at this, one can see that while both Java and C++ are overrepresented, it is within our 10% margin of error. Our responses underrepresent LabVIEW by more than can be explained by the margin of error alone. This subsection addresses two possible reasons for the bias: team age and mentor bias.

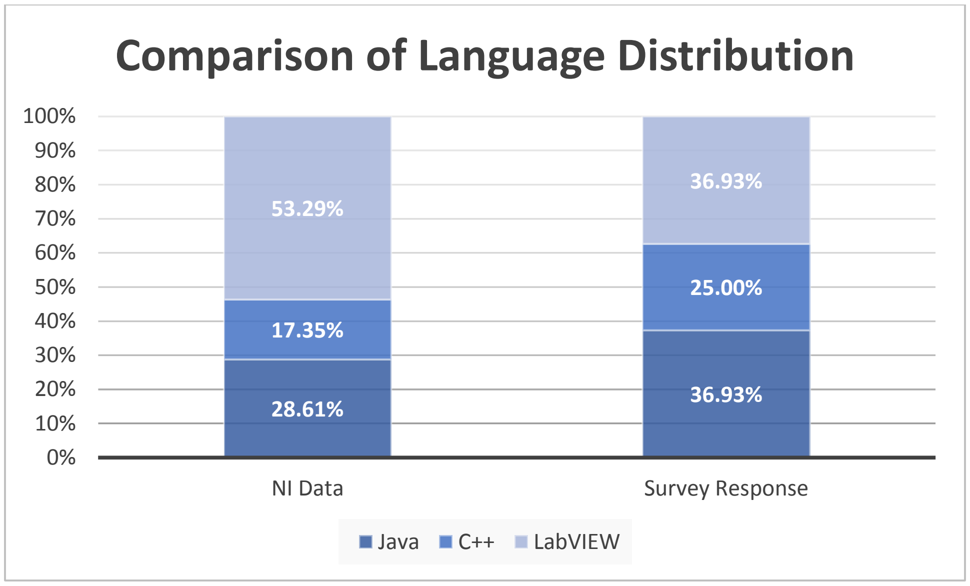


Figure 7: The left hand side shows the NI Data that represents the true distribution, while the right hand side shows the distribution based on the survey data (Question 30).

4.3.2.1: Team Age

Our research shows that team language choice correlates with team age, shown in Figure 8 below. Older teams are typically less likely to use LabVIEW and more likely to use C++ than younger teams. It should be noted that for age categories older than 8 years, there were less than 100 teams competing in the 2013 FRC season, so the variability from year to year increases with older teams. Based on the data from section 4.3.1.4:, it is clear that our data is biased towards older teams that are somewhat less likely to use LabVIEW. This is a large factor in explaining the bias, but alone it is not enough.

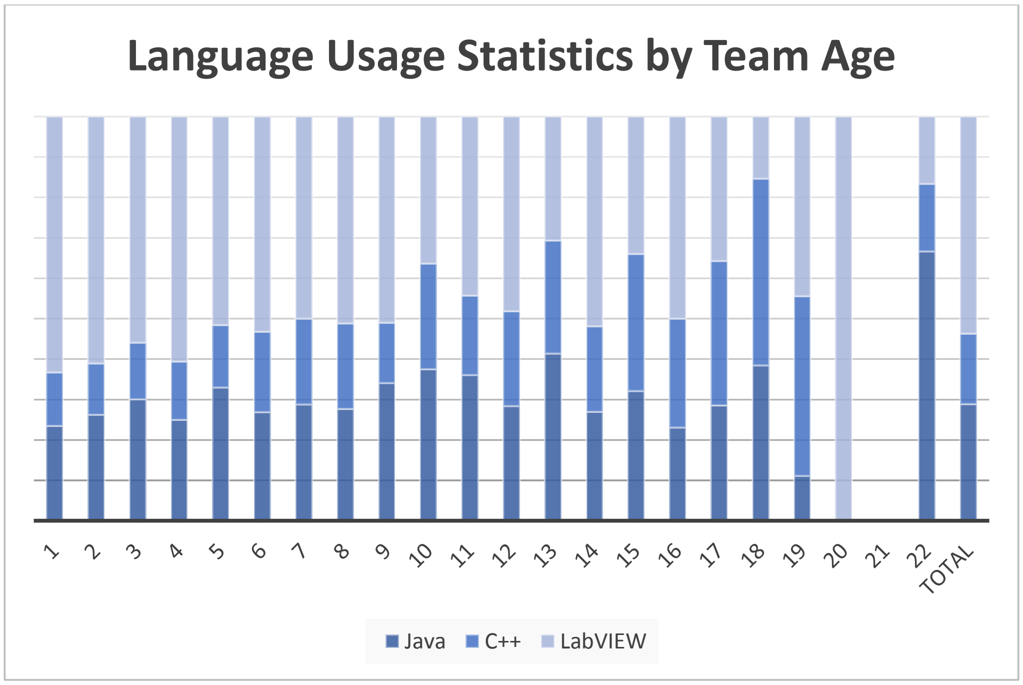


Figure 8: Language usage statistics by team age (Extracted from team number, Question 2 and 30).

4.3.2.2: Mentor Bias

Another possible problem with the language data is that mentors who responded are more likely to use C++ and Java. These mentors are also more involved with the control system on the team. As shown in Figure 9, C++ and Java mentors were significantly more involved than LabVIEW mentors were. One possible reason is that some teams get their programming mentors through AP CS teachers at schools. The AP CS curriculum uses the Java programming language. Another possible contributing reason could be that more industries use C++ and Java, so teams have an easier time finding mentors. While some data hints at reasons that could be contributing issues to the bias, due to the large margin of error, we cannot conclude whether these are actually the source of bias.

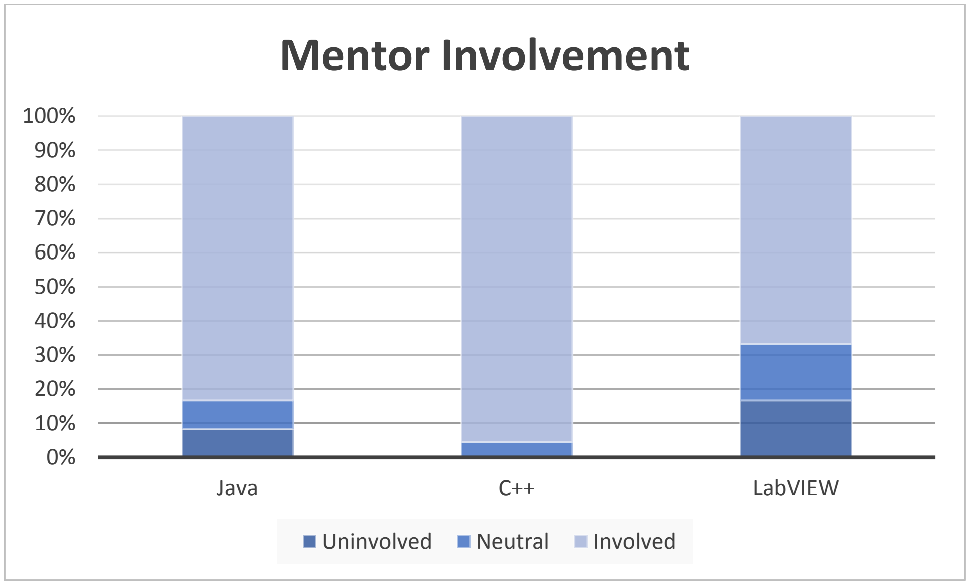


Figure 9: Mentor involvement with the control system. (Question 4)

4.3.2.3: Conclusions

Figure 10 shows that ignoring mentor responses brings us within 10% of the value we would expect to find for all three languages. This indicates that some bias was introduced into our results and as such, we must acknowledge that our findings do not necessarily generalize to the whole FRC population. It also hints that the bias may be stronger in mentors, so it is important to validate that mentors and students have similar responses or further investigate why they respond differently to certain questions throughout the analysis of the data.

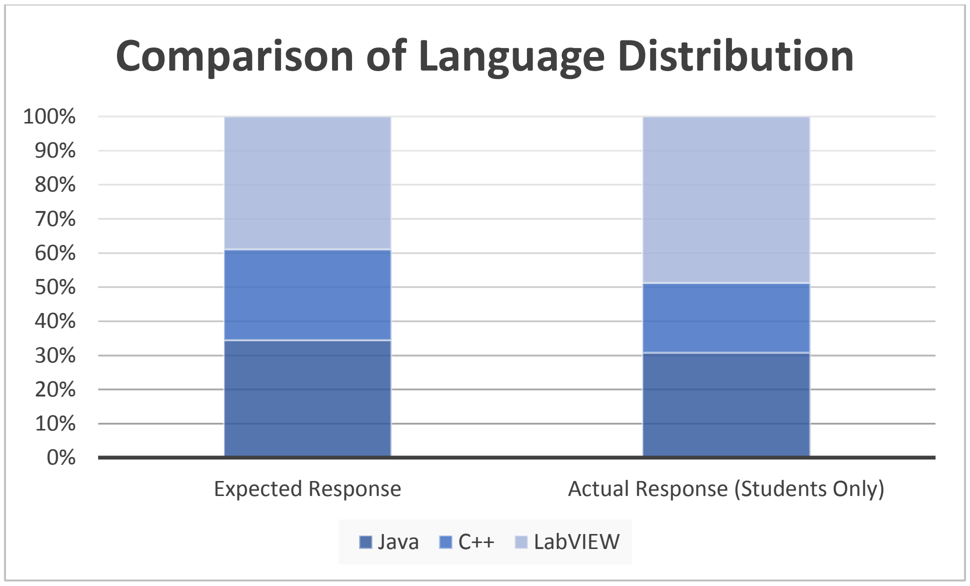


Figure 10: Left shows the expected results adjusted for the age of the actual responses; right shows actual responses of only the students (Question 30).

4.3.3: Team Success

This subsection discusses team success based off of the responses to the survey. It first looks at how sucessful teams felt they were with the control system. It then discusses how successful teams were with various goals that they chose. Finally, we look at how many teams avoided the goals due to controls.

4.3.3.1: Self-reported Control System Success

Overall, teams felt that they were successful with using the control system. Figure 11 illustrates that only 8% of respondents felt that their teams were unsuccessful in at using the control system during the 2013 season. This contrasts with 20% who felt that they were highly successful and another 54% that felt they were successful. The exact meaning of this self-reported success is hard to decipher due to the varied definitions of success that people have. Regardless, the fact that most people felt that they were able to be at least somewhat successful with the control system is a good sign for the current control systems ability to enable success.

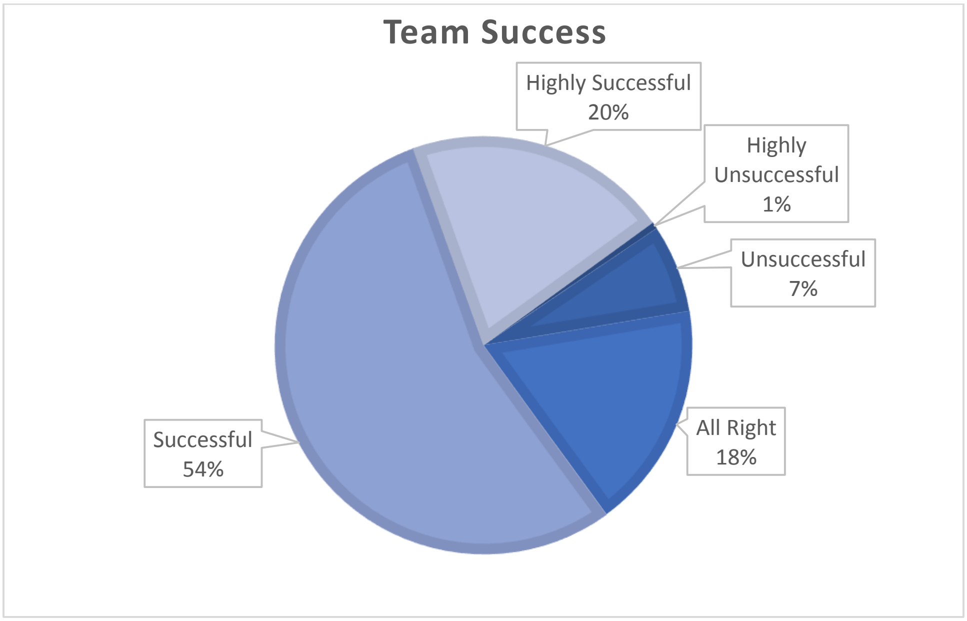


Figure 11: Teams reported success. (Question 5).

4.3.3.2: Success with Goals

In Section 1.4:, we discussed a definition of success in terms of the goals a team chose and their success with those goals. Figure 12 shows teams' self-reported success at goals they chose. Very few teams felt they were unsuccessful due to controls. In fact, almost all of the teams felt that they were successful at shooting Frisbees during both teleop and autonomous. Another interesting piece of data is that a significant number of teams that tried to climb the pyramid had problems due to factors other than controls. Overall, it looks like most teams were either successful with goals they tried or they ran into non-control issues.

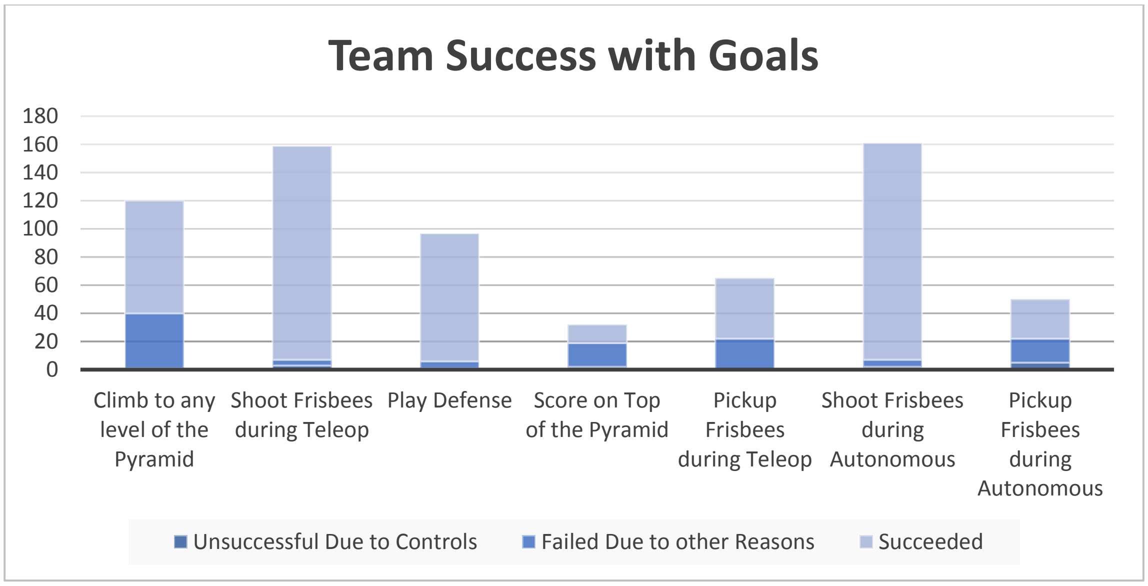


Figure 12: Teams success at goals that they chose. Height indicates the number of team that chose each goal (Question 7).

4.3.3.3: Goals Avoided

Most of the goals had a relatively small number of people who avoided them due to controls. The biggest exception was the goal of picking up Frisbees during autonomous. Figure 13 shows that 18 respondents' teams avoided it due to controls. This is relatively large compared to the 50 respondents' teams that chose it as a goal. Scoring Frisbees on top of the pyramid also had a relatively high number of people who avoided it due to controls. Eleven avoided it, while 32 had it as a goal. This data shows that for the tasks that most teams chose, the control system did not cause them to fail. However, for the tasks that less teams chose as goals, more teams had issues with the control system that caused them to fail, so the goals may have just been more challenging. For more information on issues that teams had, see the results in Appendix D: Final Survey Results. The issues themselves where captured in questions 9-29, but do not have enough statistical significance to analyze given how few teams have them.

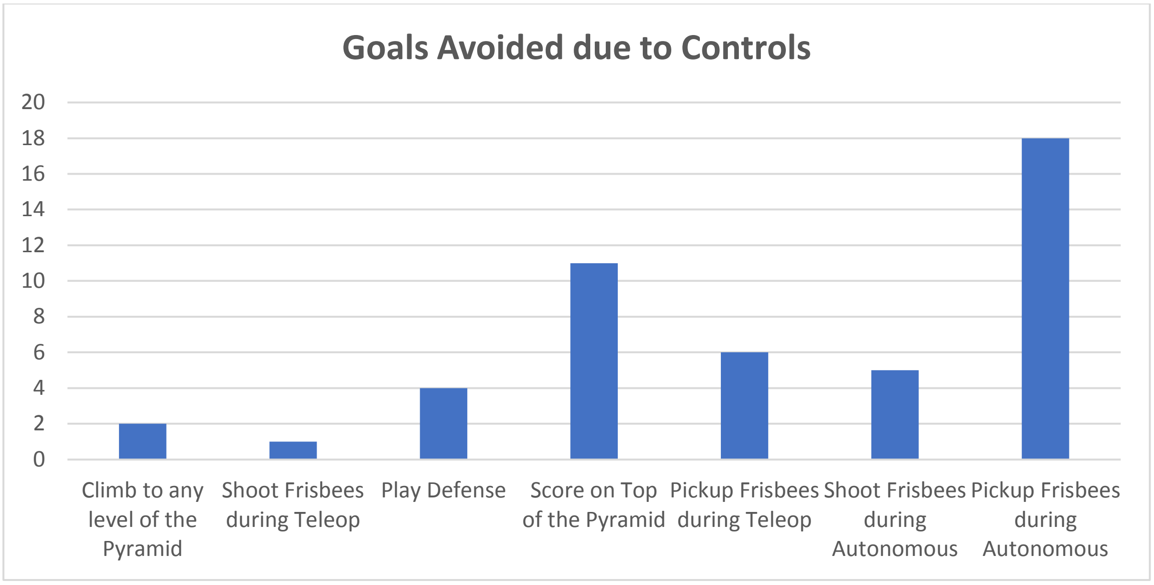


Figure 13: Number of responses that avoided each goal due to controls (Question 6).

4.3.4: Knowledge of Tools and Documentation

Most of the tools provided to FRC teams are well known. However, many respondents did not know about a few tools, shown in Figure 14. These exceptions were RobotBuilder and RoboRealm. Both had approximately a third of respondents who did not know about the tool. Very few respondents that attempted to use any of the tools were unsuccessful.

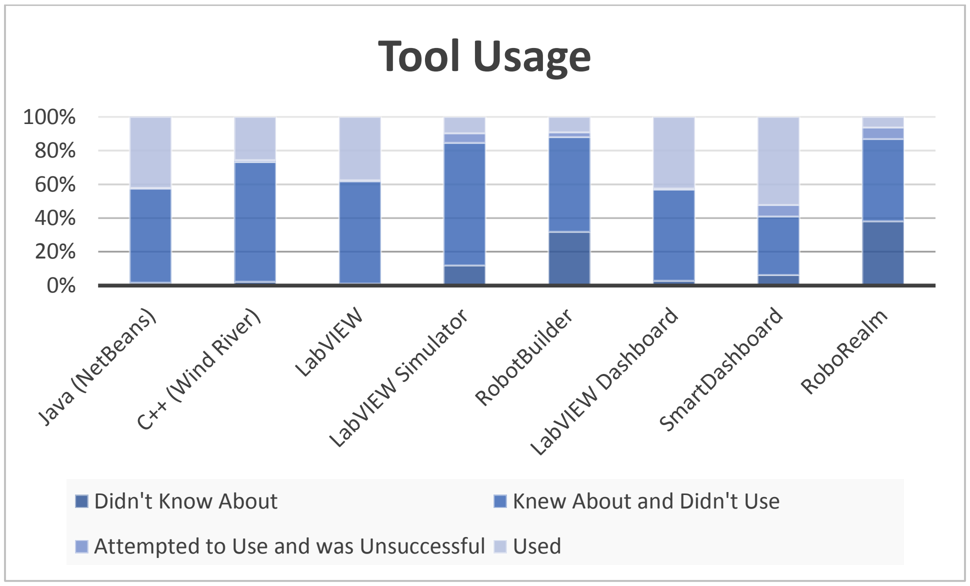


Figure 14: This graph shows the respondents' knowledge and usage of each tool (Question 30).

There are many sources of help. Surprisingly, ScreenStepsLive, the official WPILib documentation source, was unknown to almost half of respondents. While there is a bias towards C++ and Java in the documentation on ScreenStepLive, there is still key documentation for all teams on it. While more LabVIEW users did not know about it than C++ and Java users, a number of text-based language users did not know about it. This is a significant gap in knowledge. For a possible reason, see the recommendation in “Section 5.5:”. The single most used source of help was ChiefDelphi, an unofficial forum for FRC discussions. Its near universal usage can be seen in Figure 15. Other sources of help had good usage and the people who did not use them at least knew that they were available. It is important to note that due to the method of distribution, this is likely a biased result, as more than half of respondents found the survey through ChiefDelphi.

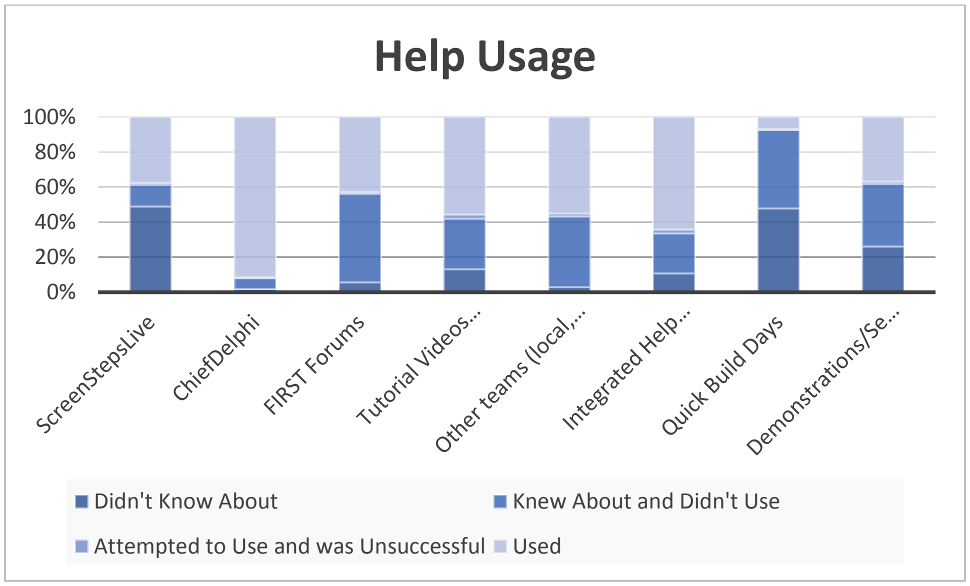


Figure 15: This graph shows the respondents' knowledge and usage of each source of help (Question 31).

4.3.5: Summary

The 176 survey responses provide enough samples to be within our targeted 10% margin of error. The sample includes a variety of respondents; they are geographically diverse, include an equal number of students and mentors, from teams of many different ages and possessing differing levels of experience in FRC. The respondents were involved with their team’s use of the control system and have the relevant experience. The majority discovered the survey through their use of ChiefDelphi. Overall, the respondents had some bias, as can be seen in the programming languages they used, but they were still a diverse group. These respondents felt that their teams were successful at most the goal they set or failed for reasons other than the control system.

Section 4.4: Conclusion

By designing the survey in an iterative manner, we were able to create a survey that captured the data we wanted. To do this, we integrated the results of interviews into the survey and broadened our focus from tools to include the sources of help. We then sent out the improved survey to a test audience, found some confusing questions and broken display logic in the survey. We used this knowledge to polish the survey, in addition to making other edits to fix issues we found in other review sessions. Once we were happy, we sent out the survey and got enough responses to be within our margin of error. The response themselves were pretty diverse and allowed us to look into what teams were having trouble with and what was working. We use this data in the next section as evidence backing our suggestions.

CHAPTER 5: RECOMMENDATIONS

As FIRST developers continually strive to improve the FRC Control System, the results of our survey show several possible improvements that they can make to the control system to help teams. Although these recommendations come from a small sample size, all results occur with a 10% confidence level, unless otherwise stated. We feel that as a part of the on-going effort, these improvements would greatly help to improve the success of teams, and the overall FIRST experience.

We have listed our recommendations below. Each recommendation has a section that goes into more detail.

- Re-Structuring of the FRC Technical Resources Page
- Better Promotion of ScreenStepsLive
- Increase Task-Based Documentation
- FRC Simulator Improvements
- Maintain Current Status of RobotBuilder
- Continue Providing Dashboard Capabilities
- Language Feature Parity

Section 5.1: Re-Structuring of FRC Technical Resources Page

Motivation: The FRC Technical Resources page is too confusing, and too difficult to reach.

Suggestion: Re-structure the page, and provide a shorter URL for easier use and access.

Reason: The Technical Resources page is where teams expect to find information, but they are finding that information is too spread out.

In the survey, respondents answered questions about where they expected to find documentation. As seen in Figure 16, 58% of teams felt that they should go to the Technical Resources (TR) page on usfirst.org (the official FIRST homepage) for help, while 50% feel that there should be a common link for documentation. In the past, FIRST has aided teams by creating easily remembered documentation websites such as “www.frcmastery.com”. These short and easily memorized webpages allow teams to easily navigate to a webpage, and have easier access to documentation.

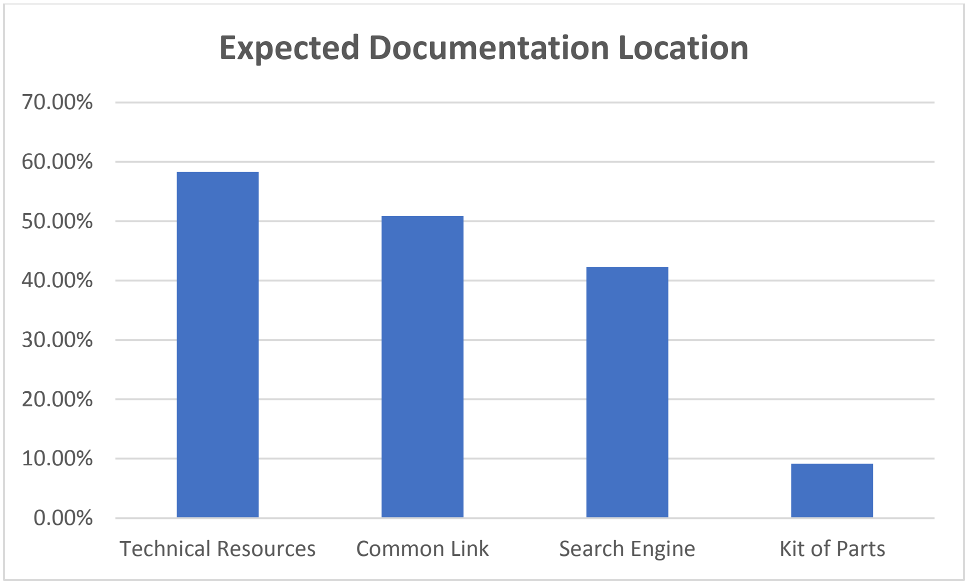


Figure 16: Where teams expect to find documentation (Question 56)

Additionally, the fact that 42% of respondents feel that they need to use a search engine to find help hints that documentation is not centralized or organized well enough. Although not statistically significant, there were an additional four respondents who wrote-in expressing that they felt that information is too spread out and that the development team should spend more effort centralizing it. A centralization of documentation would help teams that are looking for help on a specific topic and aid in the promotion of other tools and resources as well.

5.1.1: Making Resources more Accessible

Currently, the process for accessing the Technical Resources page on usfirst.org requires a team to first navigate to usfirst.org, select the FRC page, then select Resources heading from the sidebar, and finally click on the link for the Technical Resources page in the dropdown menu. Alternatively, they may navigate directly to the page by typing “www.usfirst.org/roboticsprograms/frc/Technical-Resources” into their browser. Although this link is logical and somewhat memorable, it is not short and easy to type in or relay to other team members. It may assist team if there was a more easily memorized and typed page such as “www.usfirst.org/frcresources”. Additionally, there could be a link to the current control system documentation at “www.usfirst.org/frccontrolsystem”. This link currently exists, but currently redirects to the 2010 technical resources page.

5.1.2: Better Organization of Technical Resources

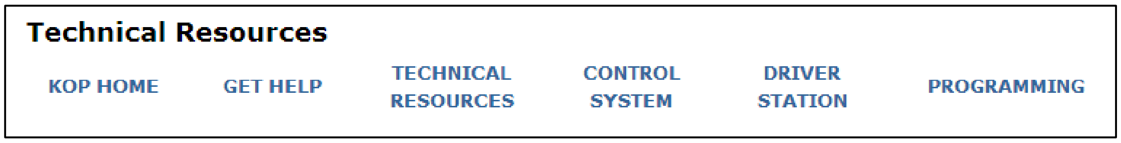


Figure 17: Screen Shot of the 2013 Technical Resources Page Header

The TR page from the 2013 FRC Season has six main categories: KOP (Kit of Parts) Home, Get Help, Technical Resources, Control System, Driver Station, and Programming (Figure 17). These six categories often have overlapping content, and lack definition. In the case of the Technical Resources category, it simply link to the current page and provides no additional detail. In addition to these categories, the TR page also contains several sections with links to various types of documentation, tools, and software tools. These sections sometimes seem to relate to the category links, and sometimes they do not. For example, one of the sections on the page is "Control System". This heading name is identical to the category name "Control System", and there is some overlap between the two pages, but there are also differences. Both pages have a linked titled "Getting Started With the 2013 Control System". The link on the TR page links to the ScreenStepsLive documentation, whereas the link on the Control System category page links to a pdf document. The Control System category page has links to several technical documents that are not contained on the TR page, such as the FMS Whitepaper. There are also several links on the Control System category page that are under a different heading on the TR page, such as the Axis camera product sheets. It would be beneficial to divide the material into three core disciplines of robotics: Electrical, Mechanical, and Programming.

Section	Information Contained
Mechanical	KoP Information, Available CAD Software, KitBot Assembly Instructions
Electrical	Connectivity Diagram, Power Distribution Diagram, Motor Controller Datasheets, Control System Component Datasheets, Camera Information
Programming	WPILib Documentation, Driver Station Information, Programming tools, Tools Updates, cRIO Utilities, Camera Utilities, Getting Started Guide

Table 5: Proposed Technical Resources Distribution

As seen in Table 5, all of the information that is currently scattered around the TR page and subpages can be centralized based around these three disciplines. With this organization, a team can easily decide where they need to go for information.

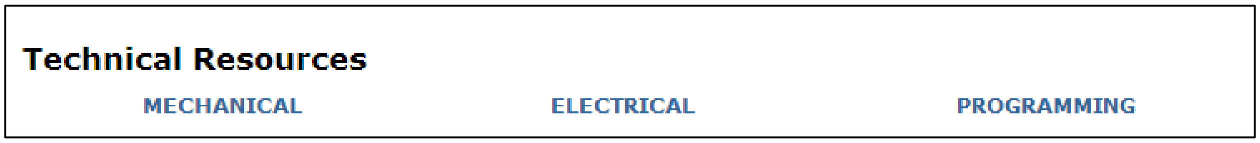


Figure 18: Proposed TR Page Headings

As developers add resources, there may be conflicts where multiple disciplines can benefit from the resource, so it may be acceptable to have multiple sections reference that resource. In this case, to provide better clarity to a team, the links that developers provide should have consistent naming, although descriptions of the links may change to better relate to the current documentation section. An example could be the Axis Camera documentation. The Programming page would describe the software resources available, such as picture configuration options, while the Mechanical page would describe the physical dimensions and mounting options.

Section 5.2: Better Promotion of ScreenStepsLive

Motivation:	Only 51% of respondents know what ScreenStepsLive is.
Suggestion:	Increase promotion of ScreenStepsLive via the FRC Technical Resources Pages and other web pages.
Reason:	Although not many people know about it, it is the most liked officially supported resource.

Newly added in 2013, ScreenStepsLive is an documentation source for teams. The survey asked respondents whether they knew if a certain documentation source existed. As shown by Figure 19, ScreenStepsLive is the documentation source with the highest percentage of respondents who did not know about its existence. Second to it, but within our margin of error are local Quick-Build days, another new source of help.

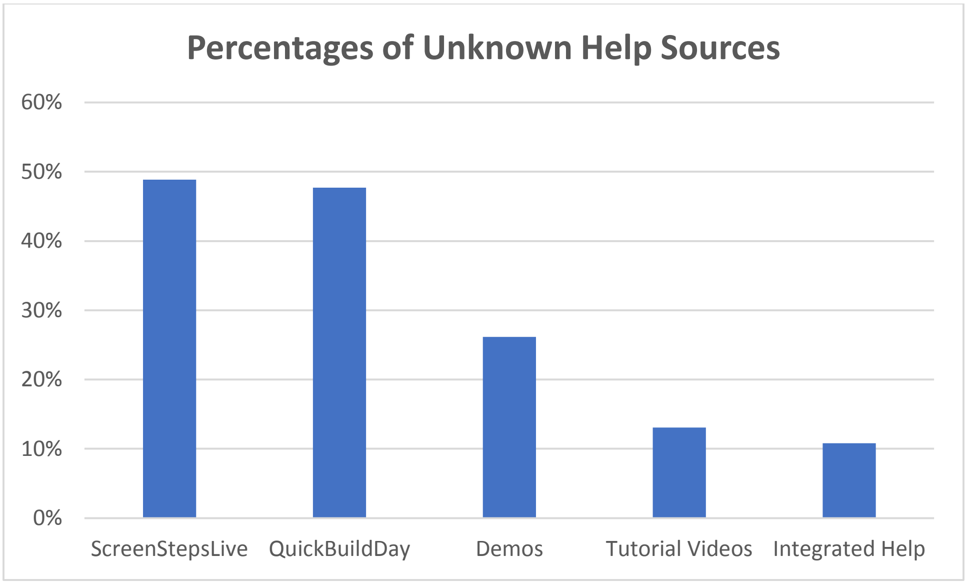


Figure 19: Unknown Help Sources (Question 31).

Almost half of the respondents do not know of the existence of ScreenStepsLive. However, those who used ScreenStepsLive considered it one of the most useful resources, second only to ChiefDelphi. Since ChiefDelphi is unaffiliated with FIRST, FIRST does not have the opportunity to improve that resource. Figure 20 shows how although not many people know about ScreenStepsLive, it has the second highest return rate of any source. This means that of the people that know about ScreenStepsLive, most of them are using it and finding it helpful.

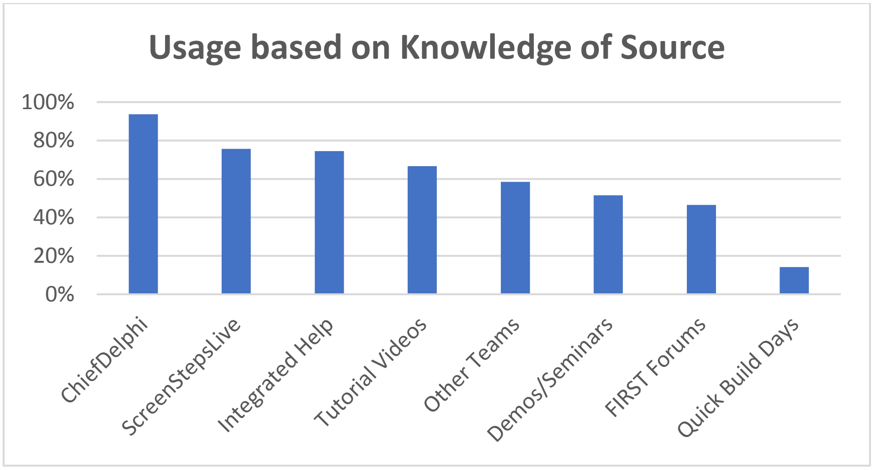


Figure 20: Use of known sources (Question 58).

5.2.1: Increase References to ScreenStepsLive

Currently there are several references to ScreenStepsLive on the FRC Technical Resources Page. These links are links to specific pages, and not to ScreenStepsLive as a whole. This may make it difficult for a team to associate ScreenStepsLive as a source of general help, rather than specific help topics. This is because teams may not even realize that they are on ScreenStepsLive when they go to a specific link off FIRST's page. As developers add information, they should move links that currently reference old documentation to ScreenStepsLive to increase its use. Additionally there is no significant information that indicates ScreenStepsLive is a valid source for LabVIEW documentation.

Promotion on third party websites should be increased as well. The use of social networking has the opportunity to increase visibility and hit rate of ScreenStepsLive. This would not only help to promote the site, but it would also help to being the help closer to the teams by advertising where many students spend their time.

5.2.2: ScreenStepsLive Popularity

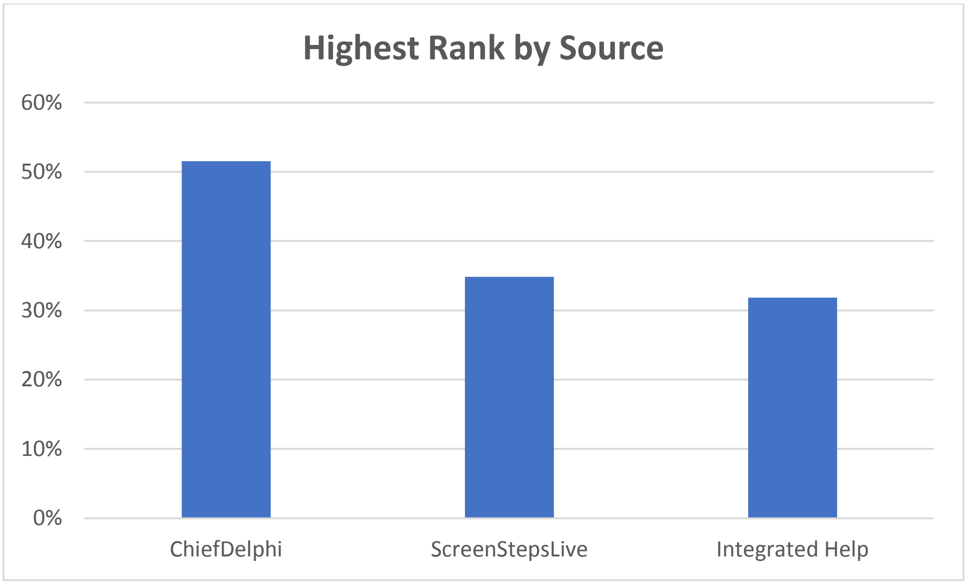


Figure 21: Percentage of #1 rank of known source (Question 53).

Despite being released in 2013, ScreenStepsLive has quickly become the most popular supported source of documentation among users who are aware of its existence. Above, you can see how, using a subset of respondents who knew about each source, 35% ranked ScreenStepsLive as their #1 source for documentation (Figure 21). This shows how more attention should be given to this source, rather than attempting to bring up less used sources.

5.2.3: Renaming ScreenStepsLive

Having a name of “ScreenStepsLive” does not properly promote the tool as a source of documentation. It may be beneficial to rename ScreenStepsLive into a more suitable name such as “WPILib Documentation” or “FRC Control System Users Guide”. In this manner, teams may be more likely to use the tool, as it may seem to fit their needs. Although the actual website name cannot be changed from “ScreenStepsLive”, another name could be used when referring to it. Although not statically significant, one respondent made a comment related to this while being asked about issues with ScreenStepsLive:

- *“It's fine. The big issue is that it is hard to find the one piece of info you need at the right time. All of the technical resources are too scattered around and frankly, clever [names] like ScreenStepsLive do nothing to make it easier.” (Question 60)*

Section 5.3: Increase the Task-Based Documentation on ScreenStepsLive

Motivation: Many tasks are not properly represented on ScreenStepsLive.

Suggestion: Add more task-based documentation.

Reason: Task-based documentation help teams to achieve the sub-goals of their robot programming.

As more and more teams start using ScreenStepsLive, the amount of information present on it needs development. At many points, there are help topics that are written for one language, but not another. For example, information on using a compressor exists only in C++ help, but operating a solenoid only exists in Java. By providing language-specific task-based documentation, it is possible to increase the usability of the source.

Although no statistically significant empirical data was gathered, there are a number of write-in and responses showing that the amount of documentation needs to be improved. Some of the write-ins are seen below:

- *“Often seemed biased to a single language.”*
- *“It's fine. The big issue is that it is hard to find the one piece of info you need at the right time. All of the technical resources are too scattered around and frankly, clever [names] like ScreenStepsLive do nothing to make it easier.”*
- *“The examples are mostly in Java and not in C++”*

For a full list of responses, see Appendix D: Final Survey Results. In addition to the write-in responses, 21 out of 66 (32%) of respondents felt that ScreenStepsLive was missing information that they needed. One way to solve this is by merging a lot of the already existing information into ScreenStepsLive format, in an attempt to unify sources. It may also be possible to provide an option for teams to write their own task-based documentation and submit it to ScreenStepsLive. In this manner, the time and resources required for developers can be reduced, as the work could be passed to teams. If a team felt passionate about helping and contributing to other teams, they could be given an outlet to do so. Many teams are already doing this in their own ways, such as through ChiefDelphi, and person team websites such as Team 1114. Providing an outlet for these teams to help on a more official level would both aide FIRST in the creation of documentation, but help promote healthy team growth as well.

Section 5.4: FRC Simulator

After reviewing the data, we have several suggestions relating to the FRC Simulator. The 17 respondents who used the simulator and the 128 respondents who knew about the simulator, but did not use it are the primary source for these suggestions. The full distribution of usage statistics can be seen in Figure 22.

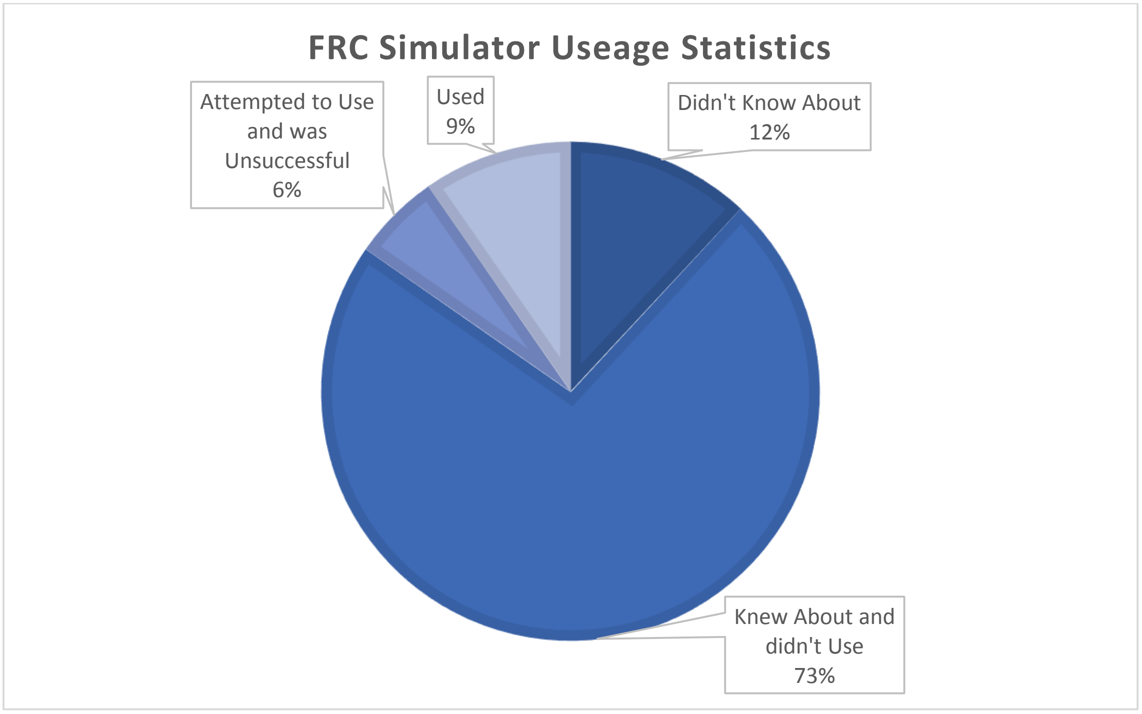


Figure 22: Usage of the FRC LabVIEW Simulator (Question 30).

5.4.1: Continue Providing the FRC Simulator

Motivation: Of the small set of teams using the simulator, it has a high retention rate.

Suggestion: Continue to provide the simulator and expand to other languages.

Reason: Teams using the simulator like it and find it helpful.

Although our data does not show many teams using the simulator, the simulator has a high retention rate. As seen in Figure 23, of the 16 respondents who answered the question of if they would use the simulator again, 14 of them responded positively that they would.

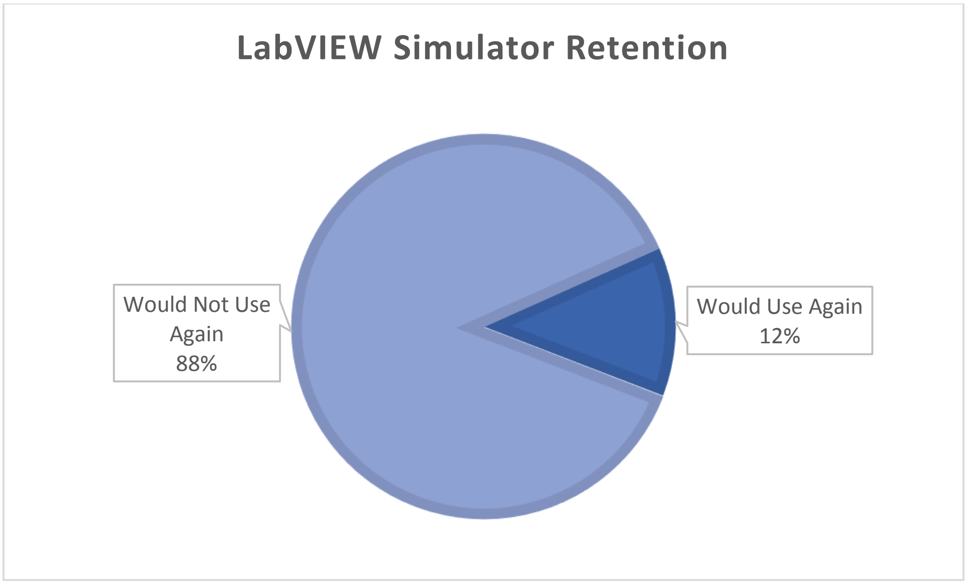


Figure 23: Retention rate of LabVIEW Simulator (Question 41).

Some respondents commented on how they felt about the simulator with respect to using it next year:

- *"I really liked the LabVIEW simulator. It was very helpful for testing our drive code before the robot was ready to go..."*
- *"We were able to test the code with the LABVIEW simulator, making sure the controls worked and could move the robot…"*
- *"It was a very useful tool (in my opinion) for getting younger student's feet wet in programming and then allowing them to test simple things where they could see the results in real time and enjoy it…"*

These responses do not provide any statistically significant evidence but serve to provide insight as to what teams are feeling about the LabVIEW Simulator. This paper shows the full set of responses in Appendix D: Final Survey Results.

In addition to the above data, we provided respondents who did not use the tool with a brief explanation of what it does. Next, we asked them if they would use the tool. Teams responded positively saying that 86% of teams would either use, or consider using this tool in the future. Given the 127 respondents who did not use or were unable to use the simulator, only 25% of them had no interest in using one.

5.4.2: Expand to other Languages and Support other Robots

Motivation: The current Simulator only supports example robots in LabVIEW

Suggestion: Expand the Simulator to Java and C++ and allow for custom robots.

Reason: Simulation can be largely beneficial and more teams are able to use simulation if it is more widely available.

Currently, LabVIEW is the only language supported by the simulator, leaving teams using Java or C++ without the ability to simulate. Below, Figure 24 shows that 29% of respondents who did not use the simulator would use it if they had the ability to import custom robots. Similarly, 46% of respondents showed an interest in using a simulator for their robots, meaning they would consider using a simulation.

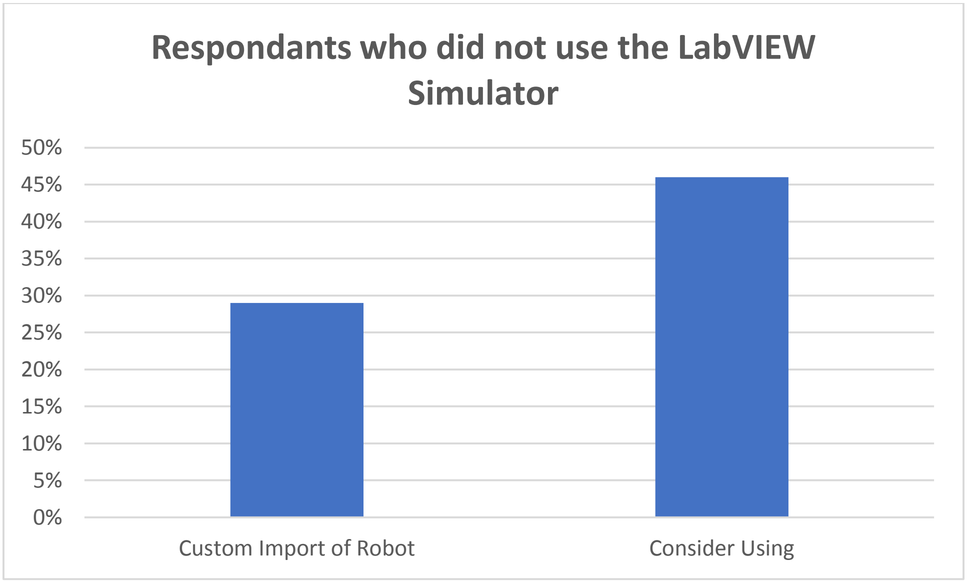


Figure 24: Percentages of respondents who did not use the Simulator (Question 38).

By providing the simulator to teams across all languages, every team would have the ability to simulate their robot. Furthermore, importing custom robots allows team to simulate their real robots, rather than just basic ones. As seen above, the ability to simulate a robot is largely beneficial to a team's ability to develop their robots. Although not statistically significant, there is some feedback from teams in the comments section. See Appendix D: Final Survey Results, question 40.

Section 5.5: Maintain Current Status of RobotBuilder

Motivation: Few teams appear to use RobotBuilder, but those that do really like it.

Suggestion: Maintain current program status and make new features a low priority.

Reason: Teams using RobotBuilder find it useful and helpful, but the majority of teams do not see a need for it.

Although few teams appear to be using RobotBuilder, those who use it enjoy it, and find its feature useful. Due to its small following, developers should give new features a lower priority, and focus more on important improvement elsewhere. This however, should not mean that support for RobotBuilder should be dropped, as interest in it may rise.

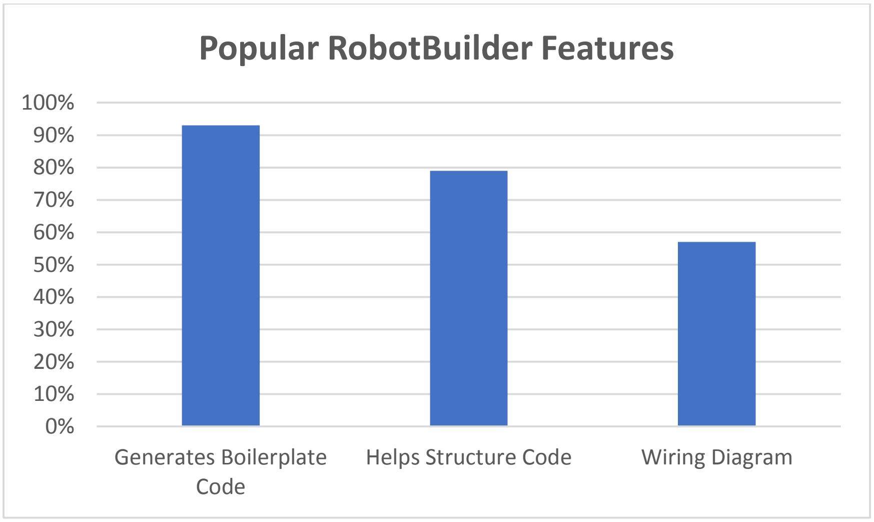


Figure 25: Most popular RobotBuilder Features (Question 44).

Above, Figure 25 shows the most popular RobotBuilder features. The program's main goal of providing boilerplate code is the most enjoyed feature at 93%. Next, the ability for RobotBuilder to generate a program structure is rated at 79%.

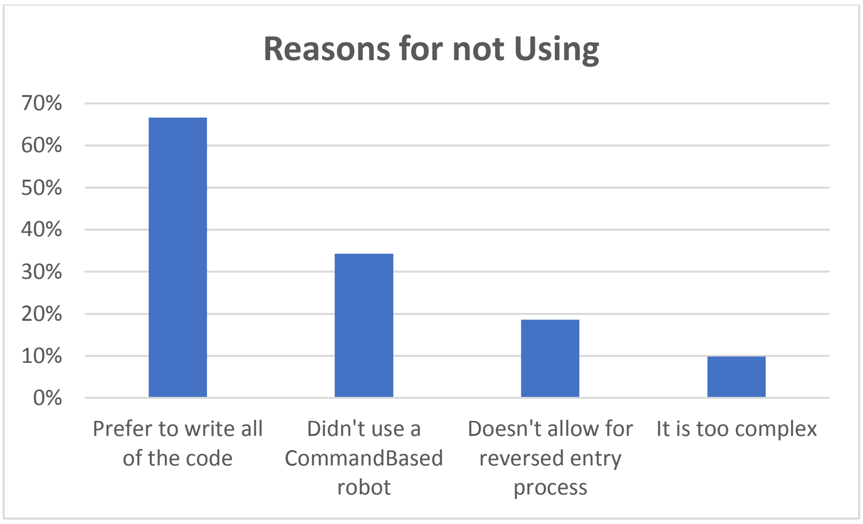


Figure 26: Responses for not using RobotBuilder (Question 43).

When asked why respondents did not use RobotBuilder, Figure 26 shows their responses of the given response set. Some respondents provided anecdotal responses about RobotBuilder. These responses are not statistically significant, but offer insight about RobotBuilder. A subset of their responses is shown below:

- *"It didn't have all the features we wanted"*
- *"It simplifies things that don't need to be."*
- *"It makes us uncomfortable to be that far away from the code."*
- *"User interface was a bit clumsy"*
- *"When we ran into problems program would crash on cRIO with little [explanation] on how to fix it. Had a lot of trouble interfacing multiple systems of a complex robot."*

It is important to maintain support for newer tools, as removing them hurts consistency for teams year-to-year. In the case of RobotBuilder, it may be better to hold off on adding improvements or new features.

Although both the LabVIEW Simulator and RobotBuilder have very few teams using the software, RobotBuilder does not show nearly as much future use as with simulations. Similar to simulations, we provided respondents who did not know about RobotBuilder with a brief explanation. When we asked if they would use RobotBuilder in the future, 27% responded yes, while 53% responded maybe. These numbers may be very close to the simulator; however, these values come from a sample of only 55 respondents. This means that more teams know about RobotBuilder and are capable of using it, but are choosing not to.

Section 5.6: Continue Providing Dashboards

Motivation: Over 75% of teams are using dashboards.

Suggestion: Maintain current velocity with dashboards

Reason: Dashboards provide teams with feedback and data that would otherwise be difficult to obtain.

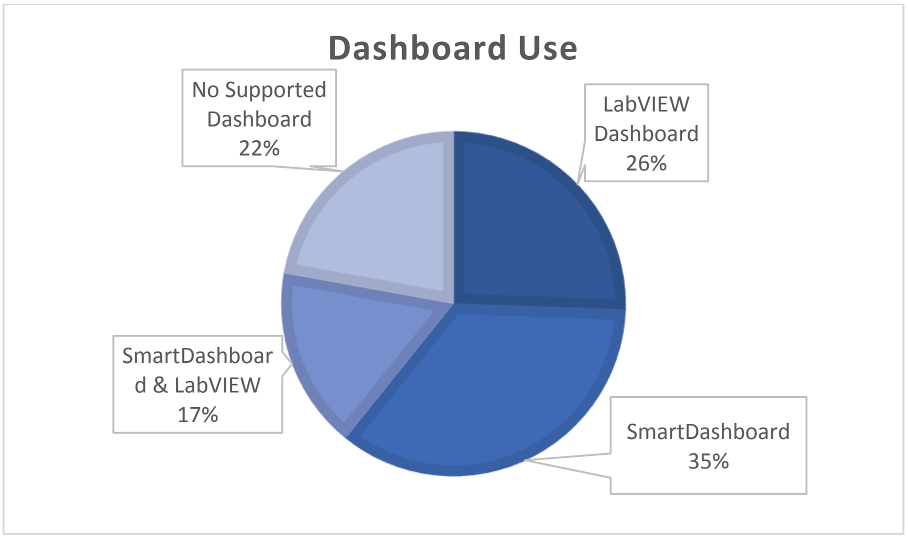
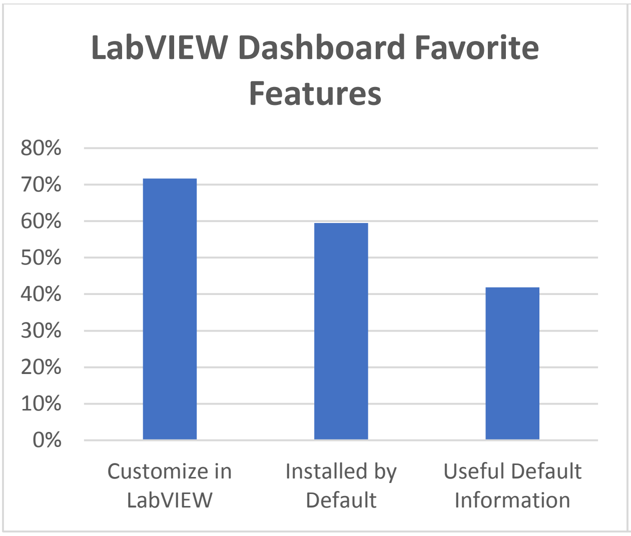


Figure 27: Dashboard Use among Teams (Question 30).

As shown in Figure 27, over 75% of responded teams are using an FRC Supported Dashboard. These dashboards (LabVIEW and SmartDashboard) are supported by FIRST Developers and have similar functionality. Although it is possible that a team is using a dashboard that is not supported by FIRST, these dashboards were outside the scope of this project.

Many features of the supported dashboard provide users with information about how their robot is reacting to the world and other internal systems. The survey asked respondents what they liked the most about each of the dashboards they used.



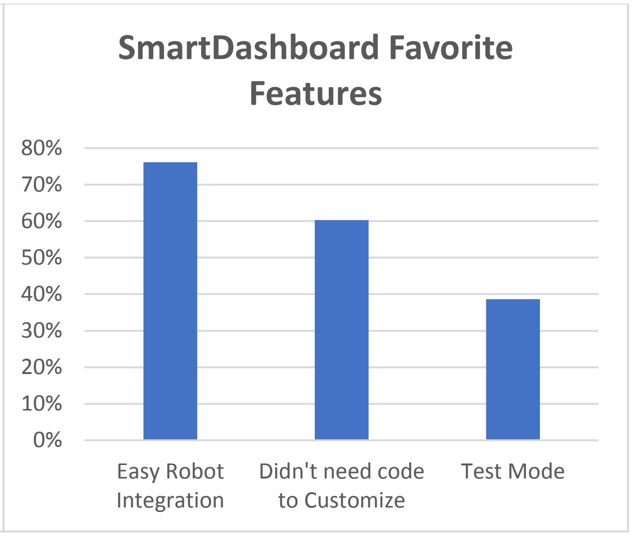


Figure 28: Favorite Dashboard Features (Question 48, Question 47).

Shown above in Figure 28, Integration, Customization, and Installation are the three most popular features of dashboards. As so, these features should be maintained as to maintain current popularity. Less popular features such as Test Mode in SmartDashboard, need to be maintained, as this feature may be on the rise. Although no conclusive evidence on the effectiveness of Test Mode can be gathered from this research, further studies may show how teams benefit from its tools.

Much of the popularity of a dashboard may be caused by the many uses of it. Below, you can see how the main use of a dashboard is to provide information to drivers. As show, 89% of respondents use a dashboard for this capability. Often, teams may be using this information to provide a more educated decision for the human-controlled robot actions. Next, Figure 29 shows that 81% of users use their dashboard for debugging in robot testing. Finally, only 58% are using it for tuning PID’s and determining setpoints.

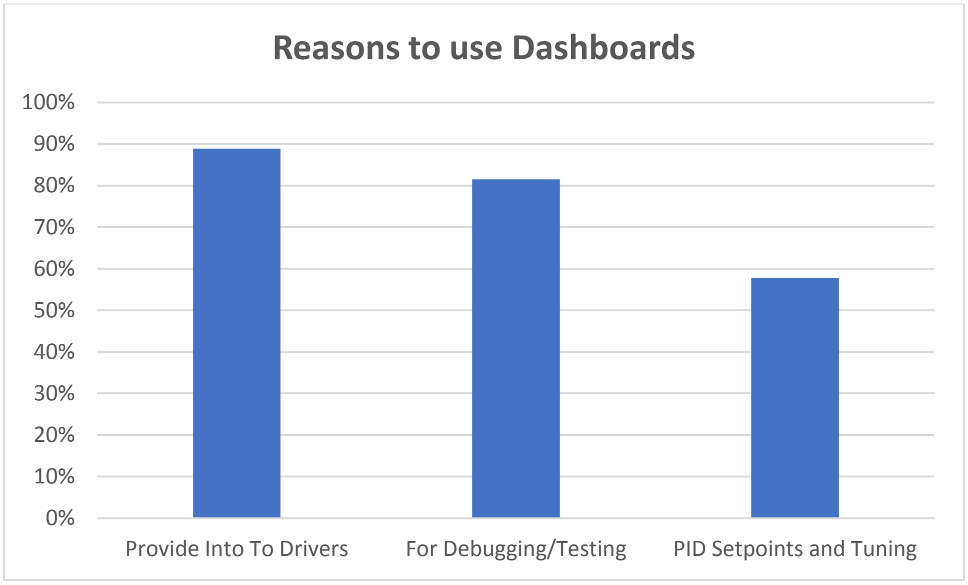


Figure 29: Why Teams use Dashboards (Question 49).

Many of the features of a dashboard are invaluable to a team and should be kept. In this manner, we recommend maintaining the currently status with FIRST Supported Dashboards, and continue to improve them for teams. Developers should proceed to add functionality as needed, but not remove any of the current features.

Section 5.7: Language Feature Parity

Motivation: There are three languages supported in FRC each with slightly different features.

Suggestion: Maintain feature parity across languages.

Reason: Teams are selecting language based on their local help and personal abilities rather than language-specific features.

As the three main languages continue to grow, and develop, it is important to ensure that one language does not have a significant advantage over another. Our survey sought to ensure that teams do not feel compelled to use one language over another solely because of the library features that it has.

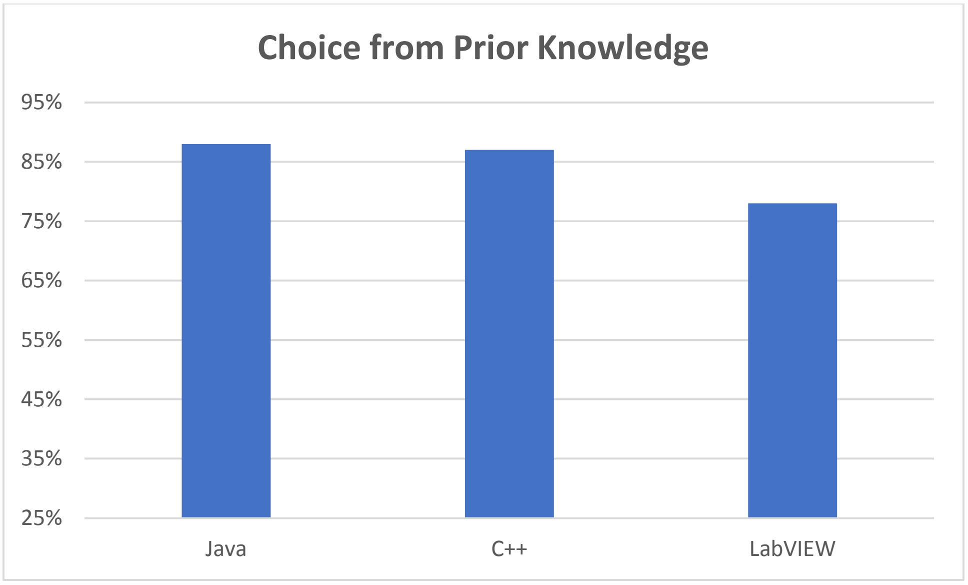


Figure 30: Top Reason for Language Choice (Question 33, Question 34, Question 35).

Shown above in Figure 30, the primary reason for using a language is prior knowledge of student and mentors. This implies that teams are not choosing one language because of its unique features, and that developers are not biasing one language. Teams in FRC should not be given a disadvantage simply because of their language choice. In addition to the choice due to prior knowledge, many teams choose their language due to the help available locally to them. Seen below, Figure 31 shows how 30% of teams using Java feel that their decision to use Java was influenced by the help available locally to them. Similarly, the values for C++ and LabVIEW are 10% and 43% respectively.

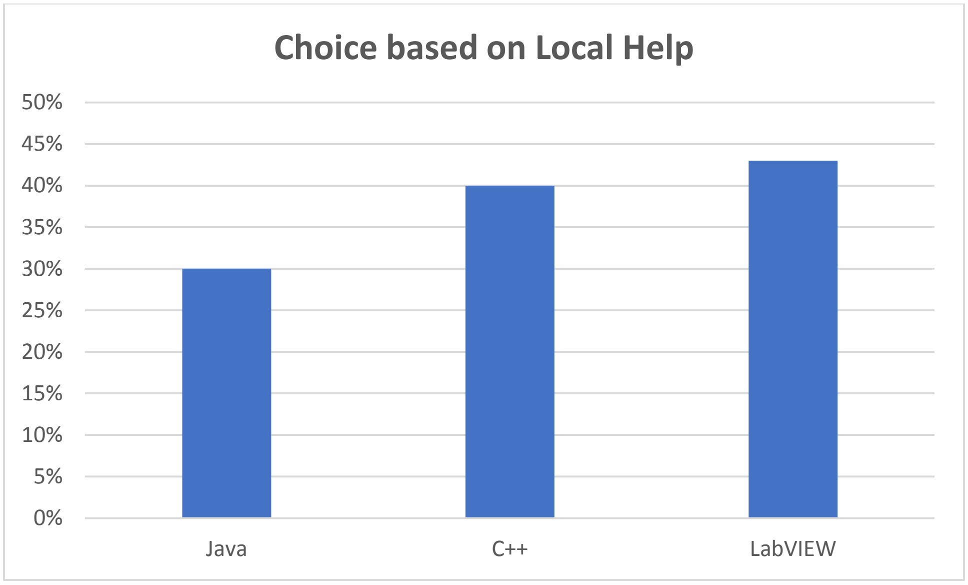


Figure 31: Language Choice based on Local Help (Question 33, Question 34, and Question 35).

Using this data, we can imply that the language parity across the three languages is acceptable, and that teams are making their decisions based on their own skillsets. Teams are not feeling that they need to use a particular language because of its features, and not because of their skillsets. Although there are a few write-ins implying that the NI Vision Libraries in C++ and Java are lacking, this is not a driving factor. The parity across the languages should be maintained and features of all three languages should continue to be developed.

CHAPTER 6: CONCLUSION

In this paper, we looked at how the FRC control system enables teams to succeed at their goals for the 2013 game and what the control system team can do improve the system in the upcoming years. Our goal was to generate a list of recommendations for the control system features that enable teams' success. This list included current features that are well done and should be kept and features that should to be improved. These features covered several aspects of the control system, including both the tools, such as the programming environments for LabVIEW, C++, and Java, as well as the documentation, such as online forums, ScreenStepsLive, and Integrated Help.

In order to determine the success of these tools, we conducted interviews with team members at an offseason event in order to help gauge the type of things that might give teams issues. We then took this feedback and updated our survey. We tested it on a small email alias at WPI, in order to verify the functionality of the survey logic and to make sure that we did not miss any obvious questions. After verifying the survey, we distributed it through a variety of social media sources, including Facebook, ChiefDelphi, Twitter, and with the help of mentors on other teams.

Our survey collected data on three main sections: the goals that teams had during the 2013 season, and whether or not teams succeeded at these goals; the control system tools that teams used, and what they liked or had issues with; and the documentation sources that teams used, and whether these sources were able to answer all of their questions. From this data, we were able to create our list of control system features that help teams to succeed, and some suggestions for what the control-system development team can do to improve some of these features.

Our recommendations encompass all sections of our survey. We suggested updating the Technical Resources page on usfirst.org, as the current layout is confusing and content inconsistencies. We also showed that current simulation tools are very useful to the subset of users that can use them, and that the control system team should maintain and improve them to support more languages in the coming years. Our next recommendation is that ScreenStepsLive needs to be better promoted, as it has one of the best respondent satisfaction ratings, but is the least known about documentation source. It also needs more content, as it misses some key pieces of information, or has language inconsistencies. We also showed that the RobotBuilder tool is useful and liked among teams, and should be maintained. The Dashboards proved very popular among all teams, with most teams using one of the two available programs. In particular, we showed that teams like the customizability of the dashboards, and that they are very popular for debugging purposes. Finally, we looked at the video documentation available to teams. We recommend that the content be updated and promoted, since respondents felt that what was available was useful, but some key topics were missing.

We can summarize our project with a few sentences. The control system tools and documentation currently available to teams is high quality and well liked. However, there is often too little content, and it is spread out and hard to find. It is our recommendation that the control system team focus on collecting all of the information and tools available and making them easily accessible and well documented.

REFERENCES

Azevedo, R., Cromley, J. G., & Seibert, D. 2004. "Does adaptive scaffolding facilitate students' ability to regulate their learning with hypermedia?" *Contemporary Educational Psychology* 29 (3): 344-370.

Bolderston, Amanda. 2012. "Conducting a Research Interview." *Journal of Medical Imaging and Radiation Sciences* 43 (1): 66-76.

FIRST. 2013. "FIRST Internal Report on the 2013 Kit of Parts Survey." Compiled by Kate Pilotte. Manchester, MA: FIRST Headquarters. Accessed September 3, 2013.

FIRST Robotics. 2012. "Einstein Investigation Report." *usfirst.org.* Accessed November 14, 2012. http://www.usfirst.org/sites/default/files/uploadedFiles/Robotics_Programs/FRC/Game_and_Season__Info/2012_Assets/Einstein%20Investigation%20Report.pdf.

Fischer, G. B. 1999. "Developing Students' Adaptive Learning Skills." *College Teaching* 47 (3): 96-100.

Gilder, G. 1992. "Data Flow Visual Languages." *IEEE Potentials* 1: 30-33.

Gomes, L., & Lourenço, J. 2010. "Rapid Prototyping of Graphical User Interfaces for Petri-Net-Based Controllers." *Industrial Electronics, IEEE Transactions on* 57 (5): 1806-1813.

Henning, J. 2009. "Survey software success." *vovici.com.* Accessed 2013. http://blog.vovici.com/blog/bid/18188/Survey-Software-Success-Free-EBook.

Hsiao, I. H., Sosnovsky, S., & Brusilovsky, P. 2010. "Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming." *Journal of Computer Assisted Learning* 26 (4): 270-283.

Johnson, J. 2003. "Children, robotics, and Education." *Artificial Life Robotics* 16-21.

Kelleher, C., & Pausch, R. 2005. "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers." *ACM Computing Surveys (CSUR)* 37 (2): 83-137.

Kölling, M. Quig, B., Patterson, A., and Rosenberg, J. 2003. "The BlueJ system and its pedagogy." *J. Comput. Sci. Educ.* 13: 4.

Kölling, M. 2010. "The greenfoot programming environment." *ACM Transactions on Computing Education (TOCE)* 10 (4): 14.

Kotrlik, Joe W, and Chadwick C Higgins. 2001. "Organizational Research: Determining appropriate sample size in survey research." *Information Technology, Learning, and Performance Journal* 43-45.

Malmsten, P. 2011. "Usability Of WPILib 2012."

Melchior, A., Cohen, F., Cutter, T. & Leavitt, T. 2005. "More than robots: An evaluation of the first robotics competition participant and institutional impacts." *Heller School for Social Policy and Management, Brandeis University.*

Petruccelli, J. D., M. Chen, and B. Nandram. 1999. *Applied Statistics for Engineers and Scientists.* New Jersey: Prentice-Hall.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. 2009. "Scratch: programming for all." *Communications of the ACM* 52 (11): 60-67.

Rowley, Jennifer. 2012. "Conducting research interviews." *Management Research Review* 35 (3/4): 260-271. doi:http://dx.doi.org/10.1108/01409171211210154.

The University of Texas at Austin. 2007. "Survey Question Types." Accessed 2013. http://www.utexas.edu/academic/ctl/assessment/iar/teaching/plan/method/survey/survey_tables_questiontypes.pdf.

USFIRST. n.d. *Vision and Mission | usfirst.org.* Accessed 2013. http://www.usfirst.org/aboutus/vision.

Welch, Anita G. 2010. "Using the TOSRA to Assess High School Students' Attitudes toward Science after Competing in the FIRST Robotics Competition: An Exploratory Study." *Eurasia journal of mathematics, science and technology education (1305-8223)* 6 (3): 187.

Wentling, R., Camacho, C. 2008. "Women Engineers: Factors and Obstacles Related to the Pursuit of a Degree in Engineering." *Journal of Women and Minorities in Science and Engineering* (1): 83-118.

APPENDIX A: SCRIPT FOR INTERVIEWS

Introduction

This is the beginning of the script for a session with the participant. You should start it after the participant meets you in the specified room. Before you start, make sure to receive oral consent from the participant after reading the following script. Anything indented and italicized is intended to be read aloud to the participant. When a participant first walks into the room, read the following script:

Welcome. My name is ____, I'll be conducting the interview today. (If there are any assistants in in the room, introduce them: This is ____ he/she will be assisting with the interview today.) I'd like to start by thanking you for participating in this interview. As a result of your participation, we will gather information on how to improve team's experience with the FRC Control System. The result of this will help inform and shape the 2015 control system. This interview will be recorded so that we refer back to it later if necessary. You will not be identified by any data analysis or publication, but the survey team, the WPI FIRST Robotics Research Group, and, under certain circumstances, the WPI Institutional Review Board will have access to the data recorded during this interview, which includes the identifying information of your team number and whether you are a mentor or student on the team. Remember, if at any point during this study you no longer wish to participate, you may ask to leave. Please, take a seat and we will get started.

Interview

This section contains a number of questions to ask the participant. Each question should take approximately 5 minutes to answer, some question will be longer and others will be shorter. If participants are providing very short information, the interviewer can ask probing questions. The following list has some generic probing questions that can be asked:

- "Please say more about that."
- "Can you give me more details?"
- "What is it about ...?"
- "Tell me more about …"
- "What is your experience with ...?"
- "Describe …"
- "Imagine …"
- "What caused you to ...?"
- "What features of (the topic) do you particularly like/dislike ...?"

- “How...When....Where....How often do you...?”

Each question may have more topical leading questions if relevant.

Question 1

The question:

What is your team number? Are you a mentor or a student on the team (to each participant)?

Question 2

The question:

What were your goals when using the FRC Control System this past season? Did you accomplish these goals with the current control system?

Probing questions include:

- Was your goal to climb or shoot?
- Did you have a goal involving the autonomous period?

Question 3

The question:

What problems, if any, did you had with the FRC control system? Did you solve them?

Question 4

The question:

What would you like to see improved with the FRC Control System?

Don’t ask probing questions relating to anything specific, these will be addressed in the final question if time permits. Probing questions include:

- Is there anything that is too slow and should be sped up?
- Did you use anything that felt awkward and unintuitive?

Question 5

The question:

What do you feel the FRC Control System lacks the most?

Probing questions include:

- Do any of the tools you use lack sufficient documentation?
- Do any of the tools you use lack sufficient supporting material?

Question 6

The question:

Is there anything else you would like me to know?

Bonus Question

This question is only intended to be asked if the interview wraps up very early and the participants are not being forthcoming. The purpose is to try to get some useful information out of them since we already have them in the interview. This question explores ideas we've already developed, but would like team's feedback on.

The question:

Would you find a unified documentation site useful?

How did you get help when you ran into problems?

Would you use more higher level libraries like the command-based ones?

Conclusion

Once the questions are done, all that is left is to thank the participants and end the interview.

Thank you for participating. Your feedback, it will be very helpful in shaping the 2015 control system.

Open the door for them and let them leave.

APPENDIX B: PRE-SURVEY MATERIALS

Message to WPI Students

Hi Everyone,

We're part of the FIRST Research Group, and we're doing our IQP on the FRC Control System. We are doing a survey of FIRST teams to see what their experiences with the control system were in 2013, and we could use your help. It would be a huge help if you could take our survey in order to test that it addresses concerns that teams will have. Here's the link:

http://wpi.qualtrics.com/SE/?SID=SV_eUSUv8zbM7uH6wB

Again, this is for testing for WPI students only. Please do not distribute it outside of the WPI community. Your responses will not be included in the final analysis, but will be very useful for validating our survey. If you have any feedback on the survey, such as missing questions or answers, please let us know in the final question of the survey. You can also give us feedback by emailing us at frcstudy-team@wpi.edu.

Thanks,

Fredric, Alex, and Brendan"

Pre-Survey Questions

Are you a WPI Student?

- ❍ Yes (1)
- ❍ No (2)

Thank you for your interest but this survey is only open to current WPI Students at this time. There will soon be a survey distributed to all FRC Students and Mentors. Be on the lookout!

Were you a Mentor or Student in the 2013 FRC Season?

- ❍ Mentor (1)
- ❍ Student (2)

What was your team number in the 2013 FRC Season? (Number only)

FRC Team: (1)

How many years have you participated in FRC? (Number only)

How successful do you feel your team was using the control system in the 2013 season?

- ❍ Highly Unsuccessful (1)
- ❍ (2)
- ❍ (3)
- ❍ (4)
- ❍ Highly Successful (5)

What were your goals for your robot for the 2013 competition?

	Not a Goal Due to Controls (1)	Not a Goal for other Reasons (2)	Was a Goal (3)
Climb the Pyramid (1)	❍	❍	❍
Shoot Frisbees during Teleop (2)	❍	❍	❍
Play Defense (3)	❍	❍	❍
Score on Top of the Pyramid (4)	❍	❍	❍
Pickup Frisbees during Teleop (5)	❍	❍	❍
Shoot Frisbees during Autonomous (6)	❍	❍	❍
Pickup Frisbees during Autonomous (7)	❍	❍	❍

Did you accomplish these goals by the end of the regular season?

	Failed Due to Controls (1)	Failed Due to other Reasons (2)	Succeeded (3)
Climb the Pyramid (x1)	❍	❍	❍
Shoot Frisbees during Teleop (x2)	❍	❍	❍
Play Defense (x3)	❍	❍	❍
Score on Top of the Pyramid (x4)	❍	❍	❍
Pickup Frisbees during Teleop (x5)	❍	❍	❍
Shoot Frisbees during Autonomous (x6)	❍	❍	❍
Pickup Frisbees during Autonomous (x7)	❍	❍	❍

If you had any other goals, please list them below and include whether or not you succeeded.

Why did controls prevent you from choosing to climb the pyramid? (Select all that apply)

- ❑ We couldn't think of an intuitive control scheme (1)
- ❑ We were afraid that climbing process would not be robust enough and our robot would break (2)
- ❑ We weren't sure we could get the right feedback from the sensors (3)
- ❑ Other (4) ____________________

How did controls prevent you from climbing the pyramid? (Select all that apply)

- ❑ The climbing process was not repeatable (1)
- ❑ Our drivers couldn't figure out how to climb effectively (2)
- ❑ We couldn't get the right feedback from the sensors (3)
- ❑ Other (4) ____________________

Did you have any controls problems climbing the Pyramid? (Select all that apply)

- ❑ We couldn't get the full climbing sequence correct (1)
- ❑ Our sensors were not giving us the feedback we expected (2)
- ❑ The climbing sequence is not repeatable and only works sometimes (3)
- ❑ We did not have any problems (4)
- ❑ Other (5) ____________________

Why did controls prevent you from choosing to shoot Frisbees? (Select all that apply)

- ❑ We weren't sure we had the time and resources to get vision processing to work (1)
- ❑ We couldn't figure out how to control the Frisbee's path. (2)
- ❑ We weren't sure we could get the right feedback from the sensors (3)
- ❑ Other (4) ____________________

How did controls prevent you from shooting Frisbees? (Select all that apply)

- ❑ We couldn't get the Frisbee to launch consistently (1)
- ❑ We couldn't get the right feedback from the sensors (2)
- ❑ Vision processing didn't determine the target accurately enough (3)
- ❑ We could not accurately control our shooting mechanism (4)
- ❑ Other (5) ____________________

Did you have any controls problems shooting Frisbees in teleop? (Select all that apply)

- ❑ We couldn't get the Frisbee to launch consistently (1)
- ❑ We couldn't get the right feedback from the sensors (2)
- ❑ Vision processing didn't determine the target accurately enough (3)
- ❑ We could not accurately control our shooting mechanism (4)
- ❑ We did not have any problems (5)
- ❑ Other (6) ____________________

Why did controls prevent you from choosing to play defense? (Select all that apply)
- ❑ We weren't sure we could block Frisbees quickly enough (1)
- ❑ We weren't confident we could control our choice of drivetrain (2)
- ❑ Other (3) ____________________

How did controls prevent you from playing defense? (Select all that apply)
- ❑ The drivetrain was difficult to maneuver (1)
- ❑ We couldn't detect Frisbees to block effectively (2)
- ❑ We couldn't maneuver quickly enough to perform effectively (3)
- ❑ Other (4) ____________________

Did you have any controls problems playing defense? (Select all that apply)
- ❑ The drivetrain was difficult to maneuver (1)
- ❑ We couldn't detect Frisbees to block effectively (2)
- ❑ We couldn't maneuver quickly enough to perform effectively (3)
- ❑ We did not have any problems (4)
- ❑ Other (5) ____________________

Why did controls prevent you from choosing to score on top of the pyramid? (Select all that apply)
- ❑ We didn't think it would be possible to aim accurately enough (1)
- ❑ We didn't think we could distinguish colored Frisbees from white Frisbees (2)
- ❑ We weren't sure we had the time and resources to get vision processing to work (3)
- ❑ We weren't sure we could get the right feedback from the sensors (4)
- ❑ Other (5) ____________________

How did controls prevent you from scoring on top of the pyramid? (Select all that apply)
- ❑ Our aiming wasn't accurate enough (1)
- ❑ We couldn't distinguish the color of Frisbees (2)
- ❑ We couldn't get vision processing to work accurately enough (3)
- ❑ We had issues climbing to the top to dump Frisbees (4)
- ❑ We did not get the sensor feedback we expected (5)
- ❑ Other (6) ____________________

Did you have any controls problems scoring on top of the pyramid? (Select all that apply)
- ❑ Our aiming wasn't accurate enough (1)
- ❑ We couldn't distinguish the color of Frisbees (2)
- ❑ We couldn't get vision processing to work accurately enough (3)
- ❑ We had issues climbing to the top to dump Frisbees (4)
- ❑ We did not get the sensor feedback we expected (5)
- ❑ We did not have any problems (6)
- ❑ Other (7) ____________________

Why did controls prevent you from choosing to pick up Frisbees? (Select all that apply)
- ❑ We weren't confident we could control how many Frisbees we had (1)
- ❑ We weren't sure we could sense the orientation of the Frisbees (2)
- ❑ We weren't sure we could detect Frisbees getting stuck in the robot (3)
- ❑ Other (4) ____________________

How did controls prevent you from picking up Frisbees? (Select all that apply)
- ❑ We couldn't control how many Frisbees we had (1)
- ❑ We couldn't reliably sense the orientation of the Frisbees (2)
- ❑ We couldn't deal with Frisbees getting stuck (3)
- ❑ We couldn't get the right feedback from sensors (4)
- ❑ Other (5) ____________________

Did you have any controls problems picking up Frisbees? (Select all that apply)
- ❑ We couldn't control how many Frisbees we had (1)
- ❑ We couldn't reliably sense the orientation of the Frisbees (2)
- ❑ We couldn't deal with Frisbees getting stuck (3)
- ❑ We couldn't get the right feedback from sensors (4)
- ❑ We did not have any problems (5)
- ❑ Other (6) ____________________

Why did controls prevent you from choosing to shoot Frisbees during autonomous? (Select all that apply)
- ❑ We weren't sure we could get the right feedback from the sensors (1)
- ❑ We weren't sure we could figure out how to control the Frisbee's path (2)
- ❑ We weren't sure we could accurately aim Frisbees autonomously (3)
- ❑ Other (4) ____________________

How did controls prevent you from shooting Frisbees during autonomous? (Select all that apply)
- ❑ We couldn't get the Frisbee to launch consistently (1)
- ❑ We couldn't get the right feedback from the sensors (2)
- ❑ Vision processing didn't determine the target accurately enough (3)
- ❑ Other (4) ____________________

Did you have any controls problems shooting Frisbees during autonomous? (Select all that apply)
- ❑ We couldn't get the Frisbee to launch consistently (1)
- ❑ We couldn't get the right feedback from the sensors (2)
- ❑ Vision processing didn't determine the target accurately enough (3)
- ❑ We did not have any problems (4)
- ❑ Other (5) ____________________

Why did controls prevent you from choosing to pick up Frisbees during autonomous? (Select all that apply)

- ❑ We weren't confident we could control how many Frisbees we had (1)
- ❑ We weren't sure we could sense the orientation of the Frisbees (2)
- ❑ We weren't sure we could detect Frisbees getting stuck in the robot (3)
- ❑ Other (4) ____________________

How did controls prevent you from picking up Frisbees during autonomous?

- ❑ We couldn't control how many Frisbees we had (1)
- ❑ We couldn't reliably sense the orientation of the Frisbees (2)
- ❑ We couldn't deal with Frisbees getting stuck (3)
- ❑ We couldn't get the right feedback from sensors (4)
- ❑ Other (5) ____________________

Did you have any controls problems picking up Frisbees during autonomous? (Select all that apply)

- ❑ We couldn't control how many Frisbees we had (1)
- ❑ We couldn't reliably sense the orientation of the Frisbees (2)
- ❑ We couldn't deal with Frisbees getting stuck (3)
- ❑ We couldn't get the right feedback from sensors (4)
- ❑ We did not have any problems (5)
- ❑ Other (6) ____________________

For each tool, please select the most appropriated answer.

	Didn't Know About (1)	Knew About and Didn't Use (2)	Attempted to Use and was Unsuccessful (3)	Used (4)
Java (NetBeans) (1)	❍	❍	❍	❍
C++ (Wind River) (2)	❍	❍	❍	❍
LabVIEW (3)	❍	❍	❍	❍
LabVIEW Simulator (4)	❍	❍	❍	❍
RobotBuilder (5)	❍	❍	❍	❍
LabVIEW Dashboard (6)	❍	❍	❍	❍
SmartDashboard (7)	❍	❍	❍	❍
RoboRealm (8)	❍	❍	❍	❍

For each source of documentation or help, please select the most appropriated answer.

	Didn't Know About (1)	Knew About and Didn't Use (2)	Attempted to Use and was Unsuccessful (3)	Used (4)
ScreenStepsLive (1)	❍	❍	❍	❍
ChiefDelphi (2)	❍	❍	❍	❍
FIRST Forums (3)	❍	❍	❍	❍
Tutorial Videos (FRC Mastery, YouTube) (4)	❍	❍	❍	❍
Other teams (local, at events) (5)	❍	❍	❍	❍
Integrated Help (popup help in Wind River, NetBeans, LabVIEW) (6)	❍	❍	❍	❍
Quick Build Days (7)	❍	❍	❍	❍
Demonstrations/Seminars (8)	❍	❍	❍	❍

Are there any other sources of documentation or help that you found useful? If so please describe them below.

Why did you choose to use Java? (Select all that apply)
- ❑ We already knew Java (1)
- ❑ We develop on a Mac or Linux (2)
- ❑ We don't like working with a dataflow language (3)
- ❑ We don't want to deal with the complexities of C++ (4)
- ❑ It seemed like the easiest language (5)
- ❑ Other (6) ____________________

Why did you choose to use C++? (Select all that apply)
- ❑ We already knew C++ (1)
- ❑ We don't like working with a dataflow language (2)
- ❑ We need the performance gains and/or C++ language features (3)
- ❑ It seemed like the easiest language (4)
- ❑ Other (5) ____________________

Why did you choose to use LabVIEW? (Select all that apply)
- ❑ We already knew LabVIEW (1)
- ❑ We like working with a dataflow language (2)
- ❑ It's NI's language and they make the cRIO (3)
- ❑ It seemed like the easiest language (4)
- ❑ Other (5) ____________________

Which programming language would you use if you had to choose again?
- ❍ Java (1)
- ❍ C++ (2)
- ❍ LabVIEW (3)
- ❍ Other (4) ____________________

The LabVIEW simulator provides 3 prebuilt robots that are run in a simulated environment. Students can write LabVIEW programs using WPILib, just as they would for a real robot. They can then run these programs on the simulated robots to practice and learn robot programming. Do you think you would you use this tool?
- ❍ Yes (1)
- ❍ Yes, if it supported other programming languages (2)
- ❍ Maybe (3)
- ❍ No (4)

If the LabVIEW simulator was extended to support import your own robot, do you think you would use it?
- ❍ Yes (1)
- ❍ Maybe (2)
- ❍ No (3)

What issues did you have with the LabVIEW simulator? (Select all that apply)
- ❑ Limited choices of robot models (1)
- ❑ Only supports LabVIEW (2)
- ❑ Too much effort to use (3)
- ❑ Don't have a computer that can handle simulation (4)
- ❑ Other (5) ____________________

How was your experience with the LabVIEW simulator? What would you like to see with the simulator moving forward?

Would you use the LabVIEW simulator again?
- ❍ Yes (1)
- ❍ No (2)

RobotBuilder provides a graphical interface for describing your robot in terms of its subsystems, sensors and actuators. Once the robot has been described, RobotBuilder can export a custom CommandBased template for your robot. You can fill in the generated code with custom functionality. It also has support for generating wiring tables and allowing non-programmers to change the operator interface. Do you think you would you use this tool?

- ❍ Yes (1)
- ❍ Maybe (2)
- ❍ No (3)

Why didn't you use RobotBuilder? (Select all that apply)

- ❑ We didn't use a CommandBased robot template (1)
- ❑ It is too complex (2)
- ❑ We prefer to write all of the code myself (3)
- ❑ It doesn't allow for updating the model based on code we have added (4)
- ❑ Other (5) ____________________

What did you enjoy about RobotBuilder? (Select all that apply)

- ❑ It generated the boring boilerplate code for us (1)
- ❑ It helped structure the program for us (2)
- ❑ It allowed non-programmers to change the operator interface (3)
- ❑ It generated a nice table of how to wire the robot to match the code (4)
- ❑ Other (5) ____________________

Would you use RobotBuilder again?

- ❍ Yes (1)
- ❍ No (2)

Why didn't you use an FRC supported dashboard? (Select all that apply)

- ❑ We didn't need one (1)
- ❑ We used an alternative dashboard (2)
- ❑ We didn't have the time or resources to figure out a dashboard (3)
- ❑ Other (4) ____________________

What did you like about the LabVIEW dashboard? (Select all that apply)

- ❑ It was installed by default (1)
- ❑ It showed me what we wanted by default (2)
- ❑ We could customize it with LabVIEW (3)
- ❑ Other (4) ____________________

What did you like about SmartDashboard? (Select all that apply)
❑ We didn't have to write code to customize it (1)
❑ Test mode is very useful (2)
❑ It's easy to integrate with the robot program (3)
❑ Other (4) ____________________

Why do you use a dashboard? (Select all that apply)
❑ To debug/test (1)
❑ To provide information to drivers (2)
❑ To help determine setpoints and/or tune PID controllers (3)
❑ Other (4) ____________________

RoboRealm is a tool that that assists in vision tracking. It allows you to design an image processing algorithm graphically with continuous feedback. It shows what the image looks like at each step of processing, to help make design and debugging easier. The result can run on the DriverStation and send feedback to the robot over a variety of network protocols. Do you think you would you use this tool?
❍ Yes (1)
❍ Maybe (2)
❍ No (3)

Why didn't you use RoboRealm? (Select all that apply)
❑ Didn't have time/resources to figure out vision processing (1)
❑ Didn't need vision processing (2)
❑ Was afraid of networking issues (3)
❑ Other (4) ____________________

Would you use RoboRealm again?
❍ Yes (1)
❍ No (2)

Please rank the tools from most useful to least useful. (1 being most useful)
______ ScreenStepsLive (1)
______ ChiefDelphi (2)
______ FIRST Forums (3)
______ Tutorial Videos (FRC Mastery, YouTube) (4)
______ Other teams (local, at events) (5)
______ Integrated Help (popup help in Wind River, NetBeans, LabVIEW) (6)
______ Quick Build Days (7)
______ Demonstrations/Seminars (8)

What did you feel was the largest restricting factor in your team's software development?

- ❍ Our ability to understand the documentation (1)
- ❍ Our ability to find documentation (2)
- ❍ The time we had to develop software (3)
- ❍ Our ability to comprehend software (4)
- ❍ Other (5) ____________________

Were you able to answer most (or all) of your questions from ${q://QID31/ChoiceGroup/ChoiceWithLowestValue}?

- ❍ Yes, we found that ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} had answers to most of my questions (1)
- ❍ No, we continually had to use other sources as well (2)

What sorts of things was ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} missing? (Select all that apply)

- ❑ Information about the tools that we can use (1)
- ❑ Solutions to common problems that we have (2)
- ❑ Examples of code for certain tools (3)
- ❑ Examples of code for common items (such as a drive train, a PID controller, or vision tracking) (4)
- ❑ Other (5) ____________________

Where did you expect to find most information about tools and libraries available for FRC? (Select all that apply)

- ❑ From a commonly available link on USFIRST.org (Something that can be memorized and easily typed in) (1)
- ❑ Through the Technical Resources page on USFIRST.org (2)
- ❑ An insert in our Kit of Parts (3)
- ❑ A 3rd party website dedicated to FRC Programming resources (4)
- ❑ We expected to have to use a search engine such as Google (5)
- ❑ Other (6) ____________________

Overall how useful did your team find the content on the documentation tools?

	Difficult to understand; useless (1)	Not easy to understand, some useful information (2)	Somewhat easy to understand, relatively useful (3)	Easy to understand; useful (4)
ScreenStepsLive (1)	❍	❍	❍	❍
ChiefDelphi (2)	❍	❍	❍	❍
FIRST Forums (3)	❍	❍	❍	❍
Videos (4)	❍	❍	❍	❍
Integrated Help (5)	❍	❍	❍	❍
Quick Build Days (6)	❍	❍	❍	❍
Demos/Seminars (7)	❍	❍	❍	❍

What issues did you have while using ScreenStepsLive? (Select all that apply)

❑ It was missing information that we needed (1)
❑ It was hard to understand (2)
❑ It was hard to navigate (3)
❑ We had no issues (4)
❑ Other (5) ____________________

What issues did you have with the FIRST Forums? (Select all that apply)

❑ Our questions wasn't answered by the community (1)
❑ The answers we got were not helpful (2)
❑ We could not find a section for our question (3)
❑ We had no issues (4)
❑ Other (5) ____________________

What issues did you have with the Videos (FRC Mastery, YouTube)? (Select all that apply)

❑ There were no videos covering topics that we needed (1)
❑ The videos lacked information (2)
❑ The videos were too confusing (3)
❑ We had no issues (4)
❑ Other (5) ____________________

What issues did you have with the Integrated Help (popup help in Wind River, NetBeans, LabVIEW)? (Select all that apply)

- ❑ The documentation wasn't easily accessible (1)
- ❑ The documentation was incomplete (2)
- ❑ The documentation was incorrect (3)
- ❑ We had no issues (4)
- ❑ Other (5) ____________________

What issues did you have with the Quick Build Days? (Select all that apply)

- ❑ There were no Quick Build Days in my area (1)
- ❑ There were conflicts on the day in my area, and we could not attend (2)
- ❑ The language used by the demonstrators is not the language our team uses (3)
- ❑ We had no issues (4)
- ❑ Other (5) ____________________

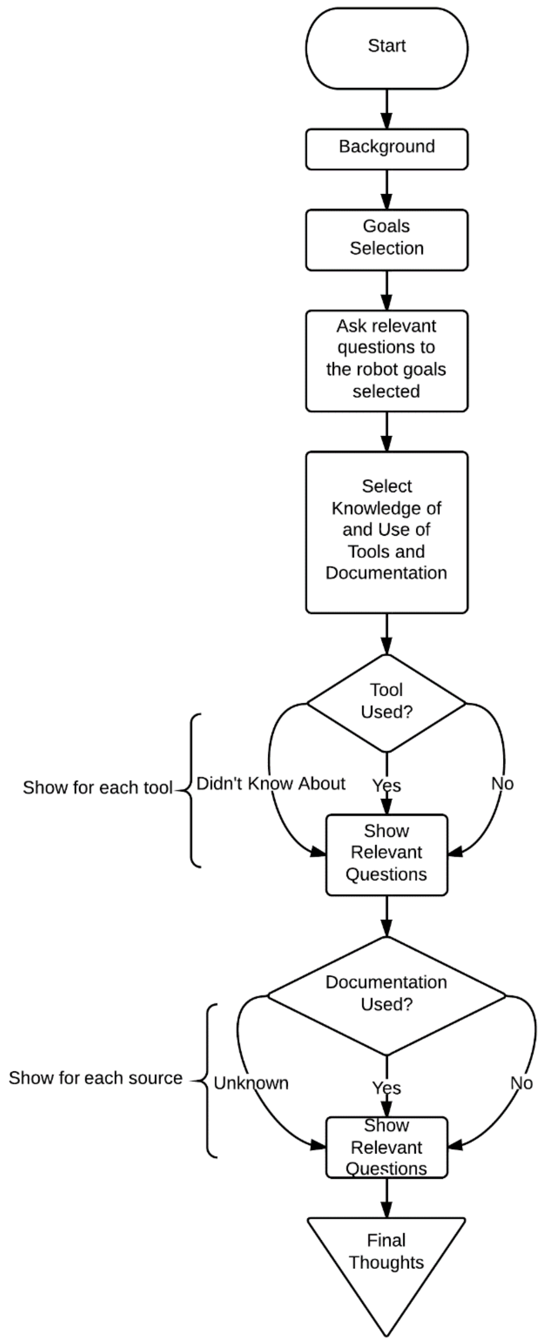
What issues did you have with the Demonstrations/Seminars? (Select all that apply)

- ❑ There were no demonstrations in our area (1)
- ❑ The demonstrations did not cover tools our team could use (2)
- ❑ There were no demonstrations for our team's language (3)
- ❑ We had no issues (4)
- ❑ Other (5) ____________________

Is there anything else not included in this survey that you wish to express to us?

APPENDIX C: FINAL SURVEY MATERIALS

Survey Flowchart



Final Survey

Q1.1 Were you a Mentor or Student in the 2013 FRC Season?

❍ Mentor
❍ Student

Q1.2 What was your team number in the 2013 FRC Season? (Number only)

FRC Team:

Q1.3 How many years have you participated in FRC? (Number only)

Q1.4 How involved were you with the control system during the 2013 season? (Controls encompasses software and tools for interacting with the CRIO, speed controllers, sensors, etc.)

❍ Not Involved
❍
❍
❍
❍ Highly Involved

Q2.1 How successful do you feel your team was using the control system in the 2013 season? (Controls encompasses software and tools for interacting with the CRIO, speed controllers, sensors, etc.)

❍ Highly Unsuccessful
❍
❍
❍
❍ Highly Successful

Q2.2 What were your goals for your robot for the 2013 competition?

	Not a Goal Due to Controls	Not a Goal for other Reasons	Was a Goal
Climb to any level of the Pyramid	❍	❍	❍
Shoot Frisbees during Teleop	❍	❍	❍
Play Defense	❍	❍	❍
Score on Top of the Pyramid	❍	❍	❍
Pickup Frisbees during Teleop	❍	❍	❍
Shoot Frisbees during Autonomous	❍	❍	❍
Pickup Frisbees during Autonomous	❍	❍	❍

Q2.3 Did you accomplish these goals by the end of the regular season?

	Unsuccessful Due to Controls	Failed Due to other Reasons	Succeeded
Climb to any level of the Pyramid	❍	❍	❍
Shoot Frisbees during Teleop	❍	❍	❍
Play Defense	❍	❍	❍
Score on Top of the Pyramid	❍	❍	❍
Pickup Frisbees during Teleop	❍	❍	❍
Shoot Frisbees during Autonomous	❍	❍	❍
Pickup Frisbees during Autonomous	❍	❍	❍

Q2.4 If you had any other goals, please list them below and include whether or not you succeeded.

Q3.1 Why did controls prevent you from choosing to climb the pyramid? (Select all that apply)
- ❑ We couldn't think of an intuitive control scheme
- ❑ We were afraid that climbing process would not be robust enough and our robot would break
- ❑ We weren't sure we could get the right feedback from the sensors
- ❑ Other ____________________

Q3.2 How did controls prevent you from climbing the pyramid? (Select all that apply)
- ❑ The climbing process was not repeatable
- ❑ Our drivers couldn't figure out how to climb effectively
- ❑ We couldn't get the right feedback from the sensors
- ❑ Other ____________________

Q3.3 Did you have any controls problems climbing the Pyramid? (Select all that apply)
- ❑ We couldn't get the full climbing sequence correct
- ❑ Our sensors were not giving us the feedback we expected
- ❑ The climbing sequence is not repeatable and only works sometimes
- ❑ We did not have any problems
- ❑ Other ____________________

Q3.4 Why did controls prevent you from choosing to shoot Frisbees? (Select all that apply)
- ❑ We weren't sure we had the time and resources to get vision processing to work
- ❑ We couldn't figure out how to control the Frisbee's path.
- ❑ We weren't sure we could get the right feedback from the sensors
- ❑ Other ____________________

Q3.5 How did controls prevent you from shooting Frisbees? (Select all that apply)
- ❑ We couldn't get the Frisbee to launch consistently
- ❑ We couldn't get the right feedback from the sensors
- ❑ Vision processing didn't determine the target accurately enough
- ❑ We could not accurately control our shooting mechanism
- ❑ Other ____________________

Q3.6 Did you have any controls problems shooting Frisbees in teleop? (Select all that apply)
- ❑ We couldn't get the Frisbee to launch consistently
- ❑ We couldn't get the right feedback from the sensors
- ❑ Vision processing didn't determine the target accurately enough
- ❑ We could not accurately control our shooting mechanism
- ❑ We did not have any problems
- ❑ Other ____________________

Q3.7 Why did controls prevent you from choosing to play defense? (Select all that apply)
- ❑ We weren't sure we could block Frisbees quickly enough
- ❑ We weren't confident we could control our choice of drivetrain
- ❑ Other ____________________

Q3.8 How did controls prevent you from playing defense? (Select all that apply)
- ❑ The drivetrain was difficult to maneuver
- ❑ We couldn't detect Frisbees to block effectively
- ❑ We couldn't maneuver quickly enough to perform effectively
- ❑ Other ____________________

Q3.9 Did you have any controls problems playing defense? (Select all that apply)
- ❑ The drivetrain was difficult to maneuver
- ❑ We couldn't detect Frisbees to block effectively
- ❑ We couldn't maneuver quickly enough to perform effectively
- ❑ We did not have any problems
- ❑ Other ____________________

Q3.10 Why did controls prevent you from choosing to score on top of the pyramid? (Select all that apply)
- ❑ We didn't think it would be possible to aim accurately enough
- ❑ We didn't think we could distinguish colored Frisbees from white Frisbees
- ❑ We weren't sure we had the time and resources to get vision processing to work
- ❑ We weren't sure we could get the right feedback from the sensors
- ❑ Other ____________________

Q3.11 How did controls prevent you from scoring on top of the pyramid? (Select all that apply)
- ❑ Our aiming wasn't accurate enough
- ❑ We couldn't distinguish the color of Frisbees
- ❑ We couldn't get vision processing to work accurately enough
- ❑ We had issues climbing to the top to dump Frisbees
- ❑ We did not get the sensor feedback we expected
- ❑ Other ____________________

Q3.12 Did you have any controls problems scoring on top of the pyramid? (Select all that apply)
- ❑ Our aiming wasn't accurate enough
- ❑ We couldn't distinguish the color of Frisbees
- ❑ We couldn't get vision processing to work accurately enough
- ❑ We had issues climbing to the top to dump Frisbees
- ❑ We did not get the sensor feedback we expected
- ❑ We did not have any problems
- ❑ Other ____________________

Q3.13 Why did controls prevent you from choosing to pick up Frisbees? (Select all that apply)
- ❑ We weren't confident we could control how many Frisbees we had
- ❑ We weren't sure we could sense the orientation of the Frisbees
- ❑ We weren't sure we could detect Frisbees getting stuck in the robot
- ❑ Other ____________________

Q3.14 How did controls prevent you from picking up Frisbees? (Select all that apply)
- ❑ We couldn't control how many Frisbees we had
- ❑ We couldn't reliably sense the orientation of the Frisbees
- ❑ We couldn't deal with Frisbees getting stuck
- ❑ We couldn't get the right feedback from sensors
- ❑ Other ____________________

Q3.15 Did you have any controls problems picking up Frisbees? (Select all that apply)

- ☐ We couldn't control how many Frisbees we had
- ☐ We couldn't reliably sense the orientation of the Frisbees
- ☐ We couldn't deal with Frisbees getting stuck
- ☐ We couldn't get the right feedback from sensors
- ☐ We did not have any problems
- ☐ Other ____________________

Q3.16 Why did controls prevent you from choosing to shoot Frisbees during autonomous? (Select all that apply)

- ☐ We weren't sure we could get the right feedback from the sensors
- ☐ We weren't sure we could figure out how to control the Frisbee's path
- ☐ We weren't sure we could accurately aim Frisbees autonomously
- ☐ Other ____________________

Q3.17 How did controls prevent you from shooting Frisbees during autonomous? (Select all that apply)

- ☐ We couldn't get the Frisbee to launch consistently
- ☐ We couldn't get the right feedback from the sensors
- ☐ Vision processing didn't determine the target accurately enough
- ☐ Other ____________________

Q3.18 Did you have any controls problems shooting Frisbees during autonomous? (Select all that apply)

- ☐ We couldn't get the Frisbee to launch consistently
- ☐ We couldn't get the right feedback from the sensors
- ☐ Vision processing didn't determine the target accurately enough
- ☐ We did not have any problems
- ☐ Other ____________________

Q3.19 Why did controls prevent you from choosing to pick up Frisbees during autonomous? (Select all that apply)

- ☐ We weren't confident we could control how many Frisbees we had
- ☐ We weren't sure we could sense the orientation of the Frisbees
- ☐ We weren't sure we could detect Frisbees getting stuck in the robot
- ☐ Other ____________________

Q3.20 How did controls prevent you from picking up Frisbees during autonomous?

- ☐ We couldn't control how many Frisbees we had
- ☐ We couldn't reliably sense the orientation of the Frisbees
- ☐ We couldn't deal with Frisbees getting stuck
- ☐ We couldn't get the right feedback from sensors
- ☐ Other ____________________

Q3.21 Did you have any controls problems picking up Frisbees during autonomous? (Select all that apply)
- ❑ We couldn't control how many Frisbees we had
- ❑ We couldn't reliably sense the orientation of the Frisbees
- ❑ We couldn't deal with Frisbees getting stuck
- ❑ We couldn't get the right feedback from sensors
- ❑ We did not have any problems
- ❑ Other ____________________

Q4.1 For each tool, please select the most appropriated answer.

	Didn't Know About	Knew About and Didn't Use	Attempted to Use and was Unsuccessful	Used
Java (NetBeans)	❍	❍	❍	❍
C++ (Wind River)	❍	❍	❍	❍
LabVIEW	❍	❍	❍	❍
LabVIEW Simulator	❍	❍	❍	❍
RobotBuilder	❍	❍	❍	❍
LabVIEW Dashboard	❍	❍	❍	❍
SmartDashboard	❍	❍	❍	❍
RoboRealm	❍	❍	❍	❍

Q4.2 For each source of documentation or help, please select the most appropriated answer.

	Didn't Know About	Knew About and Didn't Use	Attempted to Use and was Unsuccessful	Used
ScreenStepsLive	❍	❍	❍	❍
ChiefDelphi	❍	❍	❍	❍
FIRST Forums	❍	❍	❍	❍
Tutorial Videos (FRC Mastery, YouTube)	❍	❍	❍	❍
Other teams (local, at events)	❍	❍	❍	❍
Integrated Help (popup help in Wind River, NetBeans, LabVIEW)	❍	❍	❍	❍
Quick Build Days	❍	❍	❍	❍
Demonstrations/Seminars	❍	❍	❍	❍

Q4.3 Are there any other sources of documentation or help that you found useful? If so please describe them below.

Q5.1 Why did you choose to use Java? (Select all that apply)
- ❑ We already knew Java
- ❑ We develop on a Mac or Linux
- ❑ We don't like working with a dataflow language
- ❑ We don't want to deal with the complexities of C++
- ❑ It seemed like the easiest language
- ❑ We had access to local help
- ❑ Other ____________________

Q5.2 Why did you choose to use C++? (Select all that apply)
- ❑ We already knew C++
- ❑ We don't like working with a dataflow language
- ❑ We need the performance gains and/or C++ language features
- ❑ It seemed like the easiest language
- ❑ We had access to local help
- ❑ Other ____________________

Q5.3 Why did you choose to use LabVIEW? (Select all that apply)
- ❑ We already knew LabVIEW
- ❑ We like working with a dataflow language
- ❑ It's NI's language and they make the cRIO
- ❑ We had access to local help
- ❑ Other ____________________

Q5.4 Which programming language would you use if you had to choose again?
- ❍ Java
- ❍ C++
- ❍ LabVIEW
- ❍ Other ____________________

Q5.5 The LabVIEW simulator provides 3 prebuilt robots that are run in a simulated environment. Students can write LabVIEW programs using WPILib, just as they would for a real robot. They can then run these programs on the simulated robots to practice and learn robot programming. Do you think you would you use this tool?
- ❍ Yes
- ❍ Yes, if it supported other programming languages
- ❍ Maybe
- ❍ No

Q5.6 If the LabVIEW simulator was extended to support import your own robot, do you think you would use it?

- ❍ Yes
- ❍ Maybe
- ❍ No

Q5.7 What issues did you have with the LabVIEW simulator? (Select all that apply)

- ❑ Limited choices of robot models
- ❑ Only supports LabVIEW
- ❑ Too much effort to use
- ❑ Don't have a computer that can handle simulation
- ❑ Other ____________________

Q5.8 How was your experience with the LabVIEW simulator? What would you like to see with the simulator moving forward?

Q5.9 Would you use the LabVIEW simulator again?

- ❍ Yes
- ❍ No

Q5.10 RobotBuilder provides a graphical interface for describing your robot in terms of its subsystems, sensors and actuators. Once the robot has been described, RobotBuilder can export a custom CommandBased template for your robot. You can fill in the generated code with custom functionality. It also has support for generating wiring tables and allowing non-programmers to change the operator interface. Do you think you would you use this tool?

- ❍ Yes
- ❍ Maybe
- ❍ No

Q5.11 Why didn't you use RobotBuilder? (Select all that apply)

- ❑ We didn't use a CommandBased robot template
- ❑ It is too complex
- ❑ We prefer to write all of the code ourselves
- ❑ It doesn't allow for updating the model based on code we have added
- ❑ Other ____________________

Q5.12 What did you enjoy about RobotBuilder? (Select all that apply)

- ❑ It generated the boring boilerplate code for us
- ❑ It helped structure the program for us
- ❑ It allowed non-programmers to change the operator interface
- ❑ It generated a nice table of how to wire the robot to match the code
- ❑ Other ____________________

Q5.13 Would you use RobotBuilder again?
- ❍ Yes
- ❍ No

Q5.14 Why didn't you use an FRC supported dashboard? (Select all that apply)
- ❑ We didn't need one
- ❑ We used an alternative dashboard
- ❑ We didn't have the time or resources to figure out a dashboard
- ❑ Other ____________________

Q5.15 What did you like about the LabVIEW dashboard? (Select all that apply)
- ❑ It was installed by default
- ❑ It showed me what we wanted by default
- ❑ We could customize it with LabVIEW
- ❑ Other ____________________

Q5.16 What did you like about SmartDashboard? (Select all that apply)
- ❑ We didn't have to write code to customize it
- ❑ Test mode is very useful
- ❑ It's easy to integrate with the robot program
- ❑ Other ____________________

Q5.17 Why do you use a dashboard? (Select all that apply)
- ❑ To debug/test
- ❑ To provide information to drivers
- ❑ To help determine setpoints and/or tune PID controllers
- ❑ Other ____________________

Q5.18 RoboRealm is a tool that that assists in vision tracking. It allows you to design an image processing algorithm graphically with continuous feedback. It shows what the image looks like at each step of processing, to help make design and debugging easier. The result can run on the DriverStation and send feedback to the robot over a variety of network protocols. Do you think you would you use this tool?
- ❍ Yes
- ❍ Maybe
- ❍ No

Q5.19 Why didn't you use RoboRealm? (Select all that apply)
- ❑ Didn't have time/resources to figure out vision processing
- ❑ Didn't need vision processing
- ❑ Was afraid of networking issues
- ❑ Other ____________________

Q5.20 Would you use RoboRealm again?
- ❍ Yes
- ❍ No

Q6.1 Please rank the tools from most useful to least useful. (1 being most useful)
______ ScreenStepsLive
______ ChiefDelphi
______ FIRST Forums
______ Tutorial Videos (FRC Mastery, YouTube)
______ Other teams (local, at events)
______ Integrated Help (popup help in Wind River, NetBeans, LabVIEW)
______ Quick Build Days
______ Demonstrations/Seminars

Q6.2 What did you feel was the largest restricting factor in your team's software development?
- ❍ Our ability to understand the documentation
- ❍ Our ability to find documentation
- ❍ The time we had to develop software
- ❍ Our ability to understand software tools
- ❍ Our ability to understand the libraries
- ❍ Other ____________________

Q6.3 Were you able to answer most (or all) of your questions from ${q://QID31/ChoiceGroup/ChoiceWithLowestValue}?
- ❍ Yes, we found that ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} had answers to most of my questions
- ❍ No, we continually had to use other sources as well

Q6.4 What sorts of things was ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} missing? (Select all that apply)
- ❑ Information about the tools that we can use
- ❑ Solutions to common problems that we have
- ❑ Examples of code for certain tools
- ❑ Examples of code for common items (such as a drive train, a PID controller, or vision tracking)
- ❑ Other ____________________

Q6.5 Where did you expect to find most information about tools and libraries available for FRC? (Select all that apply)
- ❑ From a commonly available link on USFIRST.org (Something that can be memorized and easily typed in)
- ❑ Through the Technical Resources page on USFIRST.org
- ❑ An insert in our Kit of Parts
- ❑ A 3rd party website dedicated to FRC Programming resources
- ❑ We expected to have to use a search engine such as Google
- ❑ Other ____________________

Q6.6 Overall how useful did your team find the content on the documentation tools?

	Difficult to understand				Easy to understad
ScreenStepsLive	❍	❍	❍	❍	❍
ChiefDelphi	❍	❍	❍	❍	❍
FIRST Forums	❍	❍	❍	❍	❍
Videos	❍	❍	❍	❍	❍
Integrated Help	❍	❍	❍	❍	❍
Quick Build Days	❍	❍	❍	❍	❍
Demos/Seminars	❍	❍	❍	❍	❍

Q6.7 Overall how useful did your team find the content on the documentation tools?

	Useless				Useful
ScreenStepsLive	❍	❍	❍	❍	❍
ChiefDelphi	❍	❍	❍	❍	❍
FIRST Forums	❍	❍	❍	❍	❍
Videos	❍	❍	❍	❍	❍
Integrated Help	❍	❍	❍	❍	❍
Quick Build Days	❍	❍	❍	❍	❍
Demos/Seminars	❍	❍	❍	❍	❍

Q6.8 What issues did you have while using ScreenStepsLive? (Select all that apply)
- ❑ It was missing information that we needed
- ❑ It was hard to understand
- ❑ It was hard to navigate
- ❑ We had no issues
- ❑ Other ____________________

Q6.9 What issues did you have with the FIRST Forums? (Select all that apply)
- ❑ Our questions weren't answered
- ❑ The answers we got were not helpful
- ❑ We could not find a section for our question
- ❑ We had no issues
- ❑ Other ____________________

Q6.10 What issues did you have with the Videos (FRC Mastery, YouTube)? (Select all that apply)
- ❑ There were no videos covering topics that we needed
- ❑ The videos lacked information
- ❑ The videos were too confusing
- ❑ We had no issues
- ❑ Other ____________________

Q6.11 What issues did you have with the Integrated Help (popup help in Wind River, NetBeans, LabVIEW)? (Select all that apply)
- ❑ The documentation wasn't easily accessible
- ❑ The documentation was incomplete
- ❑ The documentation was incorrect
- ❑ We had no issues
- ❑ Other ____________________

Q6.12 What issues did you have with or why didn't you attend any Quick Build Days? (Select all that apply)

- ❏ There were no Quick Build Days in my area
- ❏ There were conflicts on the day in my area, and we could not attend
- ❏ The language used by the demonstrators is not the language our team uses
- ❏ We had no issues
- ❏ Other ____________________

Q6.13 What issues did you have with or why didn't you attend any Demonstrations/Seminars? (Select all that apply)

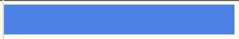
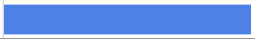
- ❏ There were no demonstrations in our area
- ❏ The demonstrations did not cover tools our team could use
- ❏ There were no demonstrations for our team's language
- ❏ We had no issues
- ❏ Other ____________________

Q7.1 Is there anything else that you wanted us to know about? Areas we are interested include, but are not limited to: Documentation, Tools, Sensors, Libraries, or anything else controls related that you think would have helped you be more successful this past season.

APPENDIX D: FINAL SURVEY RESULTS

This is the raw, tabulated data we collected from our survey. Note that we have omitted question 2, “What is your team number?” from the results to comply with our privacy policy. If you are interested in using this data in other research, please contact Brad Miller at brad@wpi.edu.

1. Were you a Mentor or Student in the 2013 FRC Season?

#	Answer		Response	%
1	Mentor		85	48%
2	Student		91	52%
	Total		176	100%

Statistic	Value
Min Value	1
Max Value	2
Mean	1.52
Variance	0.25
Standard Deviation	0.50
Total Responses	176

3. How many years have you participated in FRC? (Number only)

Text Response
5
10
6
7
3
4
6
6
2
11
4
12
4
7
2
11
8
4
6
2
7
1
3
10
15
3
4
8
17
4
4
5
3
2
3
5
3
3
3
3
6
8
5
3
5
3
4

3
2
18
5
6
3
12
2
5
5
7
2
12
3
2
2
5
9
1
4
8
2
5
1
4
3
12
3
2
2
2
2
1
7
3
11
4
11
4
1
2
5
2
2.5
9
15
2
2

3
3
2
8
3
2

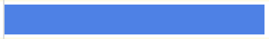
Statistic	Value
Total Responses	176

4. How involved were you with the control system during the 2013 season? (Controls encompasses software and tools for interacting with the CRIO, speed controllers, sensors, etc.)

#	Answer		Response	%
1	Not Involved		1	1%
2			7	4%
3			11	6%
4			26	15%
5	Highly Involved		131	74%
	Total		176	100%

Statistic	Value
Min Value	1
Max Value	5
Mean	4.59
Variance	0.68
Standard Deviation	0.82
Total Responses	176

5. How successful do you feel your team was using the control system in the 2013 season? (Controls encompasses software and tools for interacting with the CRIO, speed controllers, sensors, etc.)

#	Answer		Response	%
1	Highly Unsuccessful		1	1%
2			12	7%
3			31	18%
4			96	55%
5	Highly Successful		36	20%
	Total		176	100%

Statistic	Value
Min Value	1
Max Value	5
Mean	3.88
Variance	0.69
Standard Deviation	0.83
Total Responses	176

6. What were your goals for your robot for the 2013 competition?

#	Question	Not a Goal Due to Controls	Not a Goal for other Reasons	Was a Goal	Total Responses	Mean
1	Climb to any level of the Pyramid	2	54	120	176	2.67
2	Shoot Frisbees during Teleop	1	16	159	176	2.90
3	Play Defense	4	75	97	176	2.53
4	Score on Top of the Pyramid	11	133	32	176	2.12
5	Pickup Frisbees during Teleop	6	105	65	176	2.34
6	Shoot Frisbees during Autonomous	5	10	161	176	2.89
7	Pickup Frisbees during Autonomous	18	108	50	176	2.18

Statistic	Climb to any level of the Pyramid	Shoot Frisbees during Teleop	Play Defense	Score on Top of the Pyramid	Pickup Frisbees during Teleop	Shoot Frisbees during Autonomous	Pickup Frisbees during Autonomous
Min Value	1	1	1	1	1	1	1
Max Value	3	3	3	3	3	3	3
Mean	2.67	2.90	2.53	2.12	2.34	2.89	2.18
Variance	0.25	0.10	0.30	0.23	0.29	0.16	0.36
Standard Deviation	0.50	0.32	0.54	0.48	0.54	0.40	0.60
Total Responses	176	176	176	176	176	176	176

7. Did you accomplish these goals by the end of the regular season?

#	Question	Unsuccessful Due to Controls	Failed Due to other Reasons	Succeeded	Total Responses	Mean
x1	Climb to any level of the Pyramid	1	39	80	120	2.66
x2	Shoot Frisbees during Teleop	3	4	152	159	2.94
x3	Play Defense	1	5	91	97	2.93
x4	Score on Top of the Pyramid	2	17	13	32	2.34
x5	Pickup Frisbees during Teleop	0	22	43	65	2.66
x6	Shoot Frisbees during Autonomous	2	5	154	161	2.94
x7	Pickup Frisbees during Autonomous	5	17	28	50	2.46

Statistic	Climb to any level of the Pyramid	Shoot Frisbees during Teleop	Play Defense	Score on Top of the Pyramid	Pickup Frisbees during Teleop	Shoot Frisbees during Autonomous	Pickup Frisbees during Autonomous
Min Value	1	1	1	1	2	1	1
Max Value	3	3	3	3	3	3	3
Mean	2.66	2.94	2.93	2.34	2.66	2.94	2.46
Variance	0.24	0.10	0.09	0.36	0.23	0.08	0.46
Standard Deviation	0.49	0.31	0.30	0.60	0.48	0.28	0.68
Total Responses	120	159	97	32	65	161	50

8. If you had any other goals, please list them below and include whether or not you succeeded.

Text Response
Climb level 1 of pyramid, get frisbees from feeder station
Create an automatic vision targeting system, failed due to a multitude of problems and misunderstanding of how controls worked.
At least move every match. We failed due to controls.
Goal: shoot full court. Succeeded.
Active real-time vision to the driver station, failed.
One big goal was to have a vision tracking system for both autonomous and teleop, but unfortunately we did not succeed due to time constraints.
Goal: climb to level one. Success Goal: take frisbees from feeder station. Success
Drive Fast - we did this. Closed loop speed control in the cRIO for the drive motors. Previous years we did closed loop speed control with the Jaguars. Both worked really good. We did a very precise position control for the angle of the shooter. This worked well.
High level of scoring accuracy - goal was reached
LV for the season - We wanted to shoot in autonomous but for some weird reason with NetworkTables and the FMS it just didn't work.   We switched to Java for the offseason and it went great :)
Dumping frisbees in the low goal during autonomous- successful
Vision for finding goals and changing shooter speed and angle. We succeded but found it unpractical due to effort required to offload vision to make it usable and the game design allowed us to shoot from a protected and known position on the field.
We attempted to create a file based autonomous command system. Although we did create it and it was stable, we decided we would not use it enough to justify the increased maintenence time. In short, we got away with a number of set autonomi thatfulfilled the same task. We are still debating if that was the right course of action.
Another goal was auto-targeting, but that failed to succeed due to time constraints.
climb 10 points - succeeded move back to middle line during autonomous - succeeded
Secondary goal to floor load. Never did it as it was not planned into the design early enough. Succeeded in the goal of being able to shoot auton from several starting positions.
We tried creating an auto aim for the goal but it never really worked.
Auto-targeting with ip camera. We were successful.
Camera tracking in autonomous/teleop; this failed, as it has the last several years because even though we can find the targets, we never had time to implement the connection between location and power/angle of the shooter. This would have required empirical testing, for which our robot was not ready in time.
Drive in Autonomous mode - not successful because we could not get the sensor to work properly.
Attempted to automatically face Frisbee launcher at targets using camera. Did not succeed due to inadequate time to complete and tune implementation. Controls not responsible for failure.
Run reliable code on the cRio. We failed abysmally on that one, and managed to salvage our season by moving all our sensors and the rest of our code off the cRIO. We had multiple matches at Davis where WPILib would either crash reboot our crio continually or where the network task went down. We were eliminated because our robot was continually rezeroing the wrist and rebooting for one of our elimination matches. When we tried to get more information from FIRST to fix this by SVR, even with Jim Beck's help, we were ignored.
To climb for ten points - succeeded.
Autonomous movement (move to midline in auto) - succeeded Closed loop control of shooter wheel and turret - succeeded

We tried to run vision processing on the CRIO, but found the NI vision libraries hard to use. We ended up using an off board processor
Our whole robot was designed around the blasted ascension. If it weren't for the impromptu mentor-takeover, our robot would have actually functioned. It could shoot, and that was about the scope of its abilities.   Its drive train was terrible due to the weight and lack of power. Moreover, to aggregate enough space for the ascension system, wheels were placed away from parallel.
We planned to integrate vision assisted targeting, but were unable to complete that goal due to time constraints.
We attempted to write a 3 disc autonomous routine which entailed driving forward from the back of the pyramid. We weren't able to get it working because our robot used Mecanum wheels, and it was difficult to determine how far the robot moved due to the vector mathematics involved. We also used the KOP gyro for field oriented driving, but it drifted by about 1 degree per second, at least, due to the voltage midpoint being incorrectly determined during calibration. Perhaps the temperature sensor could have helped with this or there was just noise on the analog channel. The gyro often stopped working during many of our matches later in the competition weekends and we were unable to diagnose the failure. Replacing the gyro yielded the same results.
Move to C++ from LabView, more use of pneumatics (aiming, firing, transmission shifter, hanging on first level).
succeeded in 2-point dump in autonomous
Automate parts of the climbing process. Unsuccessful.
This really does not cover the control the control influenced the requirements of the mechanicals and even though the mechanics did not work out the the control system could be modified to allow for less mechanical considerations
Full court shooting - successful
to have fun....
Using image processing to find the frisbees' goals and auto aim - succeeded
Autonomous goal was to use vision processing to score. Failed not due to control systems, but other mechanical issues
No serious injuries (succeeded). Students learn new skills that are actually useful in industry (succeeded). To someday tune a PID loop (failed).
We designed out robot as a pure climber (including starting to climb in autonomous). We actually had no drivetrain on the robot. During our regional we were 100% successfuk to climbing to the third level.

Statistic	Value
Total Responses	38

9. Why did controls prevent you from choosing to climb the pyramid? (Select all that apply)

#	Answer		Response	%
1	We couldn't think of an intuitive control scheme		1	50%
2	We were afraid that climbing process would not be robust enough and our robot would break		1	50%
3	We weren't sure we could get the right feedback from the sensors		0	0%
4	Other		1	50%

Other

Statistic	Value
Min Value	1
Max Value	4
Total Responses	2

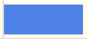
10. How did controls prevent you from climbing the pyramid? (Select all that apply)

#	Answer		Response	%
1	The climbing process was not repeatable		0	0%
2	Our drivers couldn't figure out how to climb effectively	▬	1	100%
3	We couldn't get the right feedback from the sensors		0	0%
4	Other		0	0%

Other

Statistic	Value
Min Value	2
Max Value	2
Total Responses	1

11. Did you have any controls problems climbing the Pyramid? (Select all that apply)

#	Answer		Response	%
1	We couldn't get the full climbing sequence correct		6	8%
2	Our sensors were not giving us the feedback we expected		5	6%
3	The climbing sequence is not repeatable and only works sometimes		6	8%
4	We did not have any problems		61	77%
5	Other		13	16%

Other
We did not get to the top level
insufficient sensors for fully automated climb, so it was implemented as partially operator controlled
We only chose to perform a 10 point hang
Mechanical problems, or improperly installed sensors
Drivers not lining up correctly
Our climbing mechanism (a simple wedge-shape) would cease working occasionally for unknown reasons.
Hard to judge robot climber position relative to pyramid. Some digital feedback would have been a good idea.
Did not climb past L1
Our climber used no software.
weak radio signals!
We only climbed one level.
Only 10 point pneumatic hang so we had no problems here.
Manual climb always successful, automated climb required additional development time.

Statistic	Value
Min Value	1
Max Value	5
Total Responses	79

12. Why did controls prevent you from choosing to shoot Frisbees? (Select all that apply)

#	Answer		Response	%
1	We weren't sure we had the time and resources to get vision processing to work	▬	1	100%
2	We couldn't figure out how to control the Frisbee's path.	▬	1	100%
3	We weren't sure we could get the right feedback from the sensors	▬	1	100%
4	Other	▬	1	100%

Other
We can barely program the robot to respond to joysticks, forget about controlling a flying object.

Statistic	Value
Min Value	1
Max Value	4
Total Responses	1

13. How did controls prevent you from shooting Frisbees? (Select all that apply)

#	Answer		Response	%
1	We couldn't get the Frisbee to launch consistently		1	33%
2	We couldn't get the right feedback from the sensors		0	0%
3	Vision processing didn't determine the target accurately enough		0	0%
4	We could not accurately control our shooting mechanism		2	67%
5	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	4
Total Responses	3

14. Did you have any controls problems shooting Frisbees in teleop? (Select all that apply)

#	Answer		Response	%
1	We couldn't get the Frisbee to launch consistently		36	25%
2	We couldn't get the right feedback from the sensors		22	15%
3	Vision processing didn't determine the target accurately enough		20	14%
4	We could not accurately control our shooting mechanism		23	16%
5	We did not have any problems		64	44%
6	Other		25	17%

Other
Wheel Slippage and Wear Issues
Slight jamming, no problems do to control systems
We had to build our own high-speed encoder, and tuning the PID was somewhat challenging.
Connection problems & unreliability
We had mechanical issues that led to the Frisbees jamming
We shot using manual alignment with the pyramid
Could not fine tune or control drive train precisely enough to aim well all the time
We had encoders fail from use a few times.
Joysticks kept randomly switching axes
lag with camera
Couldn't do automatic wheel speed control
Minor problems with shooter tilt; largely successful
Camera was giving spotty images due to bandwidth constraints
To clarify, we chose bad sensors, and often missed narrow, bad flags passing light gates between robot code cycles. Had a solution in the works, never put it in due to lack of testing time...
PID control had to be hacked to work for a nonzero setpoint
It was very hard to schedule a precise 100 hz task. This was contributing to noise in the control loops.
needed another sensor to detect jammed frisbees
The Hall's Effect sensor we used to determine RPM had a lot of noise
Bug in Victor and/or WPI Lib caused shooter Victors to occasionally stop working
PID was our biggest issue
Jamming on critical matches...
No vision system. Aiming was done by positioning on the field.
vison processing took to much bandwidth
Sensors not installed by mechnical
We had mechanical problems with the storage of our frisbees.

Statistic	Value
Min Value	1
Max Value	6
Total Responses	146

15. Why did controls prevent you from choosing to play defense? (Select all that apply)

#	Answer		Response	%
1	We weren't sure we could block Frisbees quickly enough	▬▬▬▬	2	50%
2	We weren't confident we could control our choice of drivetrain	▬▬▬▬	2	50%
3	Other	▬▬	1	25%

Other
we were able to shoot into the goals

Statistic	Value
Min Value	1
Max Value	3
Total Responses	4

16. How did controls prevent you from playing defense? (Select all that apply)

#	Answer		Response	%
1	The drivetrain was difficult to maneuver		0	0%
2	We couldn't detect Frisbees to block effectively		0	0%
3	We couldn't maneuver quickly enough to perform effectively		0	0%
4	Other		0	0%

Other

Statistic	Value
Min Value	-
Max Value	-
Total Responses	0

17. Did you have any controls problems playing defense? (Select all that apply)

#	Answer		Response	%
1	The drivetrain was difficult to maneuver		11	13%
2	We couldn't detect Frisbees to block effectively		0	0%
3	We couldn't maneuver quickly enough to perform effectively		15	17%
4	We did not have any problems		61	70%
5	Other		7	8%

Other
The drivetrain worked, but I thought along with the drivers that there was a lot of room for improvement, we think it was more of a mechanical problem than a software problem.
drivetrain locked up for an unknown reason. only happened on playing field and could never be replicated
meccanum
DB37 for sidecar came loose.
Very sensitive steering
Overcurrent on the 2 drive CIMS in pushing matches; fixed that with a current limiter. For tippiness, we set coast mode, which kind of helped.
Our drive train drew enough current to trip our main breaker when playing defense

Statistic	Value
Min Value	1
Max Value	5
Total Responses	87

18. Why did controls prevent you from choosing to score on top of the pyramid? (Select all that apply)

#	Answer		Response	%
1	We didn't think it would be possible to aim accurately enough		6	55%
2	We didn't think we could distinguish colored Frisbees from white Frisbees		0	0%
3	We weren't sure we had the time and resources to get vision processing to work		6	55%
4	We weren't sure we could get the right feedback from the sensors		4	36%
5	Other		1	9%

Other
The mechanical challenge was too great. We only use vision as a LAST RESORT after all other mechanical solutions have been ruled out.

Statistic	Value
Min Value	1
Max Value	5
Total Responses	11

19. How did controls prevent you from scoring on top of the pyramid? (Select all that apply)

#	Answer		Response	%
1	Our aiming wasn't accurate enough		1	100%
2	We couldn't distinguish the color of Frisbees		0	0%
3	We couldn't get vision processing to work accurately enough		0	0%
4	We had issues climbing to the top to dump Frisbees		0	0%
5	We did not get the sensor feedback we expected		0	0%
6	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	1
Total Responses	1

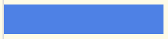
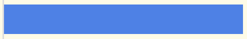
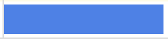
20. Did you have any controls problems scoring on top of the pyramid? (Select all that apply)

#	Answer		Response	%
1	Our aiming wasn't accurate enough		2	15%
2	We couldn't distinguish the color of Frisbees		0	0%
3	We couldn't get vision processing to work accurately enough		0	0%
4	We had issues climbing to the top to dump Frisbees		2	15%
5	We did not get the sensor feedback we expected		1	8%
6	We did not have any problems		9	69%
7	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	6
Total Responses	13

21. Why did controls prevent you from choosing to pick up Frisbees? (Select all that apply)

#	Answer		Response	%
1	We weren't confident we could control how many Frisbees we had		2	33%
2	We weren't sure we could sense the orientation of the Frisbees		3	50%
3	We weren't sure we could detect Frisbees getting stuck in the robot		3	50%
4	Other		2	33%

Other
we didn't have the time to test our autonomous to do this
We couldn't shoot, so it would be pointless anyway.

Statistic	Value
Min Value	1
Max Value	4
Total Responses	6

22. How did controls prevent you from picking up Frisbees? (Select all that apply)

#	Answer		Response	%
1	We couldn't control how many Frisbees we had		0	0%
2	We couldn't reliably sense the orientation of the Frisbees		0	0%
3	We couldn't deal with Frisbees getting stuck		0	0%
4	We couldn't get the right feedback from sensors		0	0%
5	Other		0	0%

Other

Statistic	Value
Min Value	-
Max Value	-
Total Responses	0

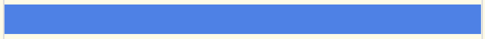
23. Did you have any controls problems picking up Frisbees? (Select all that apply)

#	Answer		Response	%
1	We couldn't control how many Frisbees we had		9	22%
2	We couldn't reliably sense the orientation of the Frisbees		7	17%
3	We couldn't deal with Frisbees getting stuck		16	39%
4	We couldn't get the right feedback from sensors		2	5%
5	We did not have any problems		15	37%
6	Other		8	20%

Other
Took too long to pick up Frisbees
Occasional Frisbee jam
Problems tracking Frisbees in the robot
our pickup (plunger) was not robust enough
The design of our robot and the sensors we put on it ment we did not know how many frisbees we possesed at any time. We could have done it, we just didn't because we were working on more important things.
Our wrist hall effect would sometimes reboot the cRIO on the first edge that it saw, putting us in a crash loop for the rest of the match.
Our pickup arm lost counts on the encoder due to low voltage on the main battery. The DI pins seem to "brown out" when the main battery hits 5.5V. This caused our calibration to be off and arm to smash in to the ground.
lots of interference

Statistic	Value
Min Value	1
Max Value	6
Total Responses	41

24. Why did controls prevent you from choosing to shoot Frisbees during autonomous? (Select all that apply)

#	Answer		Response	%
1	We weren't sure we could get the right feedback from the sensors		2	40%
2	We weren't sure we could figure out how to control the Frisbee's path		2	40%
3	We weren't sure we could accurately aim Frisbees autonomously		5	100%
4	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	3
Total Responses	5

25. How did controls prevent you from shooting Frisbees during autonomous? (Select all that apply)

#	Answer		Response	%
1	We couldn't get the Frisbee to launch consistently		1	100%
2	We couldn't get the right feedback from the sensors		0	0%
3	Vision processing didn't determine the target accurately enough		0	0%
4	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	1
Total Responses	1

26. Did you have any controls problems shooting Frisbees during autonomous? (Select all that apply)

#	Answer		Response	%
1	We couldn't get the Frisbee to launch consistently		43	29%
2	We couldn't get the right feedback from the sensors		16	11%
3	Vision processing didn't determine the target accurately enough		17	11%
4	We did not have any problems		78	52%
5	Other		17	11%

Other
Wheel Slippage and Wear Issues
Rangefinder sensor inaccurate
Not always even able to move
We shot using alignment
We did not get feedback completed and working until after the season
We aimed approximately during math setup; there were no sensors involved, just shooting.
Used a manual tuned setting instead of vision; would have preferred vision
PID control had to be hacked to work for a nonzero setpoint
We just sucked at autonomous
Difficulty to accurately control shooting mechanism
Shooter seldom jammed
It's incredibly hard to get consistent loop timing in the JVM. This killed our control loops at points.
Positioning after picking up
We had no vision system, so aiming was flakey
Autonomous sometimes didn't run on field (always worked in pit)

Statistic	Value
Min Value	1
Max Value	5
Total Responses	149

27. Why did controls prevent you from choosing to pick up Frisbees during autonomous? (Select all that apply)

#	Answer		Response	%
1	We weren't confident we could control how many Frisbees we had		7	41%
2	We weren't sure we could sense the orientation of the Frisbees		9	53%
3	We weren't sure we could detect Frisbees getting stuck in the robot		8	47%
4	Other		5	29%

Other
Our programmers are idiots
We ran out of time to focus on this task
we didn't have the time to test our autonomous to do this
We did not have a frisbee pickup method, and our method of loading fribees was impossible to accurately do during autonomous mode.
We could not drive in autonomous

Statistic	Value
Min Value	1
Max Value	4
Total Responses	17

28. How did controls prevent you from picking up Frisbees during autonomous?

#	Answer		Response	%
1	We couldn't control how many Frisbees we had		0	0%
2	We couldn't reliably sense the orientation of the Frisbees		1	25%
3	We couldn't deal with Frisbees getting stuck		2	50%
4	We couldn't get the right feedback from sensors		1	25%
5	Other		3	75%

Other
software would occasionlly crash during autonomous while picking up frisbees. Tried for several weeks to get java debugger working to track down issue.
Reliable autonomous driving is hard...
crash robot connection

Statistic	Value
Min Value	2
Max Value	5
Total Responses	4

29. Did you have any controls problems picking up Frisbees during autonomous? (Select all that apply)

#	Answer		Response	%
1	We couldn't control how many Frisbees we had		8	29%
2	We couldn't reliably sense the orientation of the Frisbees		4	14%
3	We couldn't deal with Frisbees getting stuck		11	39%
4	We couldn't get the right feedback from sensors		3	11%
5	We did not have any problems		11	39%
6	Other		7	25%

Other
The whole vision "toolchain" was prone to error. We used a webcam->roborealm (running on the driver station) -> robot, all using network tables. Many, many times one or more of these components would fail to work properly (such as network tables not updating, or saturating, or roborealm crashing, or hanging...)
Occasional Frisbee jam
Driving and twisting with swerve in auto is not an easy chore.
ground orientation limitations and vision processing speed
We had some difficulty picking up frisbees, but that was a mechanical issue. One autonomous issue (although it is present in teleop as well) is that the gyro has float and drift to it even if the robot is powered on an the gyro is initialized in the same fashion. Repeated testing showed the float and drift were not dependent on any particular factor of power up or initialization. Other than the gyro, the issues we encountered were issues in our design.
Unreliable collector mechanism, also running over missed frisbees would cause bad positioning
We didn't have any sensors for determining the state of the frisbee cache.

Statistic	Value
Min Value	1
Max Value	6
Total Responses	28

30. For each tool, please select the most appropriated answer.

#	Question	Didn't Know About	Knew About and Didn't Use	Attempted to Use and was Unsuccessful	Used	Total Responses	Mean
1	Java (NetBeans)	3	98	1	74	176	2.83
2	C++ (Wind River)	4	125	2	45	176	2.50
3	LabVIEW	2	107	1	66	176	2.74
4	LabVIEW Simulator	21	128	10	17	176	2.13
5	RobotBuilder	56	99	5	16	176	1.89
6	LabVIEW Dashboard	5	95	1	75	176	2.83
7	SmartDashboard	11	61	12	92	176	3.05
8	RoboRealm	67	86	12	11	176	1.81

Statistic	Java (NetBeans)	C++ (Wind River)	LabVIEW	LabVIEW Simulator	RobotBuilder	LabVIEW Dashboard	SmartDashboard	RoboRealm
Min Value	1	1	1	1	1	1	1	1
Max Value	4	4	4	4	4	4	4	4
Mean	2.83	2.50	2.74	2.13	1.89	2.83	3.05	1.81
Variance	1.02	0.81	0.97	0.55	0.70	1.06	1.12	0.67
Standard Deviation	1.01	0.90	0.98	0.74	0.84	1.03	1.06	0.82
Total Responses	176	176	176	176	176	176	176	176

31. For each source of documentation or help, please select the most appropriated answer.

#	Question	Didn't Know About	Knew About and Didn't Use	Attempted to Use and was Unsuccessful	Used	Total Responses	Mean
1	ScreenStepsLive	86	22	2	66	176	2.27
2	ChiefDelphi	3	11	1	161	176	3.82
3	FIRST Forums	10	89	2	75	176	2.81
4	Tutorial Videos (FRC Mastery, YouTube)	23	51	4	98	176	3.01
5	Other teams (local, at events)	5	71	3	97	176	3.09
6	Integrated Help (popup help in Wind River, NetBeans, LabVIEW)	19	40	4	113	176	3.20
7	Quick Build Days	84	79	1	12	176	1.66
8	Demonstrations/Seminars	46	63	2	65	176	2.49

Statistic	ScreenSteps Live	ChiefDelphi	FIRST Forums	Tutorial Videos (FRC Mastery, YouTube)	Other teams (local, at events)	Integrated Help (popup help in Wind River, NetBeans, LabVIEW)	Quick Build Days	Demonstrations/Seminars
Min Value	1	1	1	1	1	1	1	1
Max Value	4	4	4	4	4	4	4	4
Mean	2.27	3.82	2.81	3.01	3.09	3.20	1.66	2.49
Variance	1.94	0.38	1.13	1.38	1.07	1.27	0.65	1.52
Standard Deviation	1.39	0.61	1.06	1.17	1.03	1.13	0.80	1.23
Total Responses	176	176	176	176	176	176	176	176

32. Are there any other sources of documentation or help that you found useful? If so please describe them below.

Text Response
Source Code Documentation (Doxygen)
Javadocs for WPILib (not sure if that counts as integrated help)
The integrated LabVIEW help and examples are invaluable for troubleshooting and learning new features.
I would have liked the SmartDashboard javadocs to be more readily avilable. It would be nice if the plugins could attach the javadocs to Netbeans as well
ScreenStepsLive was helpful, but didn't fully contain all documentation that used to exist. It would be nice to leave comments about articles.
Websites of other teams, I believe it was 359 that had a great website for all sorts of coding help
Help from Brad Miller was extremely helpful.
We can always call NI and get great help if it so comes to that.
It would be nice to have an extremely high resolution poster of the entire control system which would describe the proper configuration of both current and previous year's setups, such as the module configuration for the 8 and 4 slot cRIO, a list of steps to configure the router, and other useful stuff. We saw one at the competition a few years ago and now have to have one (we're trying to make it ourselves)
The willingness of the volunteers to help with practically anything was amazing, and usually resulted in fixing the problem, if that specific volunteer didn't have an answer they would find someone who did.
I think very step by step video tutorials for rookies who have no idea how to complete any of these tasks. For example, I would really like to see a very step by step video to help me understand how everything works and how it integrates with all of the other systems.
I went to the CSA training at NI in December, that was really helpful.
We use Java. The Javadocs are and should be very helpful. They could certainly use some attention though. There are a number of classes which don't accurately/adequately describe how they should be used (which methods need to be called for configration) or the parameter comments do not ellaborate sufficiently (not units provided, etc.). Examples: the Encoder class has a bunch of getDistance methods, return types don't define units. It may be helpful to state that the units are defined by the parameter provided by the setDistancePerPulse methd. The Command Based Robot documentation doesn't make it clear how parallel/series commands in a command group are handled by the scheduler. IT would be nice to have a complex multi-level example of a parallel/sequential command group.   Often I'll have to resort to reading the source, which isn't terrible, but shouldn't be required.   An example project for how to successfully use the KOP accelerometer over I2C would be nice too.
code and documentation from previous years
Brad Miller youtube series of RobotBuilder & Java (NetBeans) was really, really helpful!

This isn't really for the WPI people (who have truthful documentation), but hopefully this can get passed along to the right people. The 2012 Einstein report claimed "improved documentation" on threading and error reporting/logging. When switching over to the 2013 development tools, there was no documentation on threading, and if you had multiple threads and used livewindowmode in C++, the robot crashed. The most frustrating problem was that the "improved" logs created by the 2013 driver station could not be opened with the driver station log viewer until several weeks after kickoff, and the scale for the CPU graph was wrong (only showed up to 10 percent). Besides for this, the screenstepslive was a really great resource for newer members. The last few years, the documentation has come a long way, and it's really a helpful resource for our team. The examples are well written and documented too. The documentation for java is hard to find. Searching the internet for "FRC Java" or "wpilib java" returns the 2012 FIRSTForge page that is not updated as the first link. The second link is for the 2011 updates (http://first.wpi.edu/FRC/frcjava.html). Why does this website need to be there?
Our team has generally been the local team resource for Java programming.
Documentation on FIRST's website for the control system and driver's station
NI Decibel. We got quite a bit of info from there.
Alumni
WPILibJ API.
Used netbeans' 'goto to declaration' feature. Very helpful when it loads the WPILibJ source.
Java API documentation
For the students, I made javadoc changes at http://web.joshua-colp.com/travis/
Stackoverflow for c++ syntax oddities, WPILib source I read through partially to understand things like the sampling rate for the analog channels, the way PWM worked, and for curiosity networktables details and the available interrupts and multithreading tools.
No one piece of documentation covered everything we need - most of documentation is looking around to find other documentation!
The help documentation in the windriver folder (under WindRiver\docs\extensions\FRC) was probably the most used documentation, especially the WPILib cookbook and the WPILib C++ Reference help file.
We read a lot of VXWorks header files and source code to learn how the code actually works. The documentation is very lacking and incomplete. The WPILib documentation is usable, but once you learn that WPILib isn't designed to do what you want to do, you are completely on your own.
Java API
A hosted javadoc/doxygen of the current WPIlib version would be nice
The WPILib source was incredibly helpful in determining what was causing our C++ programs to misbehave. There was the setpoint bug in WPILib's PID controller class corrected later, and the LiveWindow code prevented us from reloading our code's kernel module since it didn't clean up the socket it used properly upon exit.
The automatically generated documentation can be very useful as a reference for the names of methods and classes. Occasionally it's useful to just look through the WPILib source code.
We mostly just grep'd the WPILib source code.
Previous year's code!
spending time in the offseason
StackOverflow
the internet is a good place to get help at, not only in FRC related places...
pdf explaining CommandBasedRobotTemplate, pdf explaining vision
The wpilib class list.

ni.com has a lot of useful documentation in addition to product integrated help. Spec sheets for the sensors that made it onto the robot.
LabVIEW forums General research on control approaches via internet
I found the national instruments documentations to be useful in understanding how LabVIEW works and what could we use with it.

Statistic	Value
Total Responses	42

33. Why did you choose to use Java? (Select all that apply)

#	Answer		Response	%
1	We already knew Java		65	88%
2	We develop on a Mac or Linux		17	23%
3	We don't like working with a dataflow language		21	28%
4	We don't want to deal with the complexities of C++		28	38%
5	It seemed like the easiest language		17	23%
6	We had access to local help		22	30%
7	Other		25	34%

Other
LabVIEW, especially with the template code, grew difficult for the students to maintain when the program got large when we used it in previous years
Taught in school. Elegant language easier to use than C++
The students use Java in AP Classes
We started the season with C++, but switched to Java in the offseason since it is easier to teach to new programmers.
There is a Java class at the high school
AP Compsci Curriculum
It is a widely used language in both CS college courses and in industry.
Bad Experience with LabView
Mandated by the Teacher at the school
Taught in Computer Programming courses
Has the power of C++, easier to teach than C++, free to download development tools, quick deploy times
I think it's important to give the students an opportunity to learn and use languages (Java and C++) that they can then use in their careers
SmartDashboard
Our sole programmer knew Java and an aspiring programmer wished to learn, so robot building seemed like a good tutorial.
We used to use C++ but switched to Java for sustainability reasons and to pull programmers from the school's AP Computer Science program
We like the integrated NetConsole, and automatic FTP->Reboot. Last time we used Windriver, that wasn't implemented.
It's what our team has always used
Taught at their school
Why not.
Computing requirements didn't warrant using C++
Kids take AP Comp sci with Java, don't want to teach them a language.
Taught at high school
Students are taught Java in school, it makes to keep things consistent in their early days of programming.
Used to develop plugin to Smartdashboard for targetting
APCS Language

Statistic	Value
Min Value	1
Max Value	7
Total Responses	74

34. Why did you choose to use C++? (Select all that apply)

#	Answer		Response	%
1	We already knew C++		39	87%
2	We don't like working with a dataflow language		18	40%
3	We need the performance gains and/or C++ language features		19	42%
4	It seemed like the easiest language		5	11%
5	We had access to local help		18	40%
6	Other		13	29%

Other
We had prototyped a C++ in the offseason. We had used LabVIEW before, but LabVIEW's integration with source control systems leaves much to be desired.
Educational value
It's fast to rebuild and redeploy code
To teach students a well used language
As a mentor I think it is the most useful experience for the students
Used for 5 years in a row, team was familiar. Moving to java this year though.
It's awesome.
We found project organization to be easier with C++, and it can easily be used with a VCS
Java isn't realtime and we have spent weeks fighting realtime problems with C++ and new better than to try Java.
We used it last year, but now we are switching to labview for simplicity sake
For Wide Knowledge
We wanted the freedom of working with an open language which will be useful outside of the FRC.

Statistic	Value
Min Value	1
Max Value	6
Total Responses	45

35. Why did you choose to use LabVIEW? (Select all that apply)

#	Answer		Response	%
1	We already knew LabVIEW		51	78%
2	We like working with a dataflow language		24	37%
3	It's NI's language and they make the cRIO		20	31%
4	We had access to local help		28	43%
5	Other		15	23%

Other
We only had one programmer and it was the easiest way for him to do all the programming himself.
Good for rookie programmers
At the first regional we attended all but 3 teams used lab view
To program the driver station
It was said to be most supported in Israel
Only because the vision libraries that are "supported" by java aren't anywhere near complete, have memory leaks, and don't really match with the rest of the structure (the only code that requires you to call image.free())
We didn't know anything else.
We don't know any other language
Colored Spaghetti!
more appropriate for our team setup
new experience
Just like it
We tried CPP and decided that Labview is better in our opinion
a mentor in the team is NI worker
Learning curve seems shorter for students lacking any programming experience

Statistic	Value
Min Value	1
Max Value	5
Total Responses	65

36. Which programming language would you use if you had to choose again?

#	Answer		Response	%
1	Java		74	42%
2	C++		41	23%
3	LabVIEW		52	30%
4	Other		8	5%
	Total		175	100%

Other
We will likely continue to use Java and/or C++
python
Not sure
C++ on Linux
I would use all three, extremely useful. however due to the nature of first and resources available labview is better choice
Doing both labview and C++ this year
Same mixture, use the right tool for the job. Like to debug with Labview, application development on driverstation in Java and C++ on robot

Statistic	Value
Min Value	1
Max Value	4
Mean	1.97
Variance	0.91
Standard Deviation	0.95
Total Responses	175

37. The LabVIEW simulator provides 3 prebuilt robots that are run in a simulated environment. Students can write LabVIEW programs using WPILib, just as they would for a real robot. They can then run these programs on the simulated robots to practice and learn robot programming. Do you think you would you use this tool?

#	Answer		Response	%
1	Yes		5	24%
2	Yes, if it supported other programming languages		8	38%
3	Maybe		5	24%
4	No		3	14%
	Total		21	100%

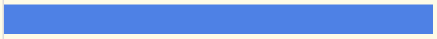
Statistic	Value
Min Value	1
Max Value	4
Mean	2.29
Variance	1.01
Standard Deviation	1.01
Total Responses	21

38. If the LabVIEW simulator was extended to support import your own robot, do you think you would use it?

#	Answer		Response	%
1	Yes		37	29%
2	Maybe		58	46%
3	No		32	25%
	Total		127	100%

Statistic	Value
Min Value	1
Max Value	3
Mean	1.96
Variance	0.55
Standard Deviation	0.74
Total Responses	127

39. What issues did you have with the LabVIEW simulator? (Select all that apply)

#	Answer		Response	%
1	Limited choices of robot models		6	60%
2	Only supports LabVIEW		2	20%
3	Too much effort to use		9	90%
4	Don't have a computer that can handle simulation		2	20%
5	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	4
Total Responses	10

40. How was your experience with the LabVIEW simulator? What would you like to see with the simulator moving forward?

Text Response
It was adequate but it is meant for lab settings, not a robotics competition.
We used it very rarely mainly just to play with some code. Would like to see robot in simulator more customize able.
Yes. We beta tested the Simulator extensively. I feel comfortable saying we beta tested it the most of any FRC team. It was a very useful tool (in my opinion) for getting younger student's feet wet in programming and then allowing them to test simple things where they could see the results in real time and enjoy it. I can provide a few cute stories about people enjoying it if that is desired.
I haven't had excessive experience with the simulator. I think it would be nice to be able to change the robot in the simulator using an Autodesk or SolidWORKS file to more closely resemble the teams own robot.
I didn't really like it, way too much work to be done to get it going, and it just doesn't seem like it was worth it.
The simulator was not useful at all because the robot models were extremely basic and were not customizable. There was no way to test the vast majority of our code.
We did not use the simulator much for the simple fact that we found it easier to test any electrical component in physical form, rather than a glitchy virtual environment.
The ability to model team robots in the simulator. Field appropriate for the current game. More sensors in the simulator.
I really liked the LabVIEW simulator. It was very helpful for testing our drive code before the robot was ready to go. I would love it if the simulator ran faster, although some of that is probably the computer we use. A function were we can input the mass and CoG of our robot would be nice for traction and stability testing.
More customization to the robot bases. Ex: swerve drive
Simulator was resource intensive and didn't always run well.
It was neat but not that useful. Would love to be able to have your actual robot in the simulator instead of a kitbot, and on an actual field.
Simulation seemed sluggish in response. Was useful in checking functionality of some code constructs but not so much in visualizing operating mechanism (all it had was an arm with a gripper).
We were able to test the code with the LABVIEW simulator, making sure the controls worked and could move the robot. I would like to see a simulator for Java and C++.
It was a neat tool, but we didn't use it a lot because we had access to another crio to test on from the beginning of the season.
The extra effort and time required to simulate in LabVIEW, especially for smaller, incremental changes meant is was more worthwhile to simply run on our target. However, for things like Vision Processing (one of our attempted goals) it would be useful to work in simulation. It would be nice to see the FRC field in the simulation software.
The Labview Simulator constantly lagged even with vision off.
great, but would like to customize robot/field
State machine configuration has been a big problem for us
we did not use it a lot, but it was fun
Didn't make time to explore its usefullness
The simulator has been a mixed bag for the team. The main issues are that it is not realistic enough, there needs to be more custombility with the robot given in the simulator, there has been times where it is not reliable to test with and recently, one of our new programmers came across a bug where when running atuonomous the robot will fail around the field. The robot ended up upside down.

Statistic	Value
Total Responses	22

41. Would you use the LabVIEW simulator again?

#	Answer		Response	%
1	Yes		14	88%
2	No		2	13%
	Total		16	100%

Statistic	Value
Min Value	1
Max Value	2
Mean	1.13
Variance	0.12
Standard Deviation	0.34
Total Responses	16

42. RobotBuilder provides a graphical interface for describing your robot in terms of its subsystems, sensors and actuators. Once the robot has been described, RobotBuilder can export a custom CommandBased template for your robot. You can fill in the generated code with custom functionality. It also has support for generating wiring tables and allowing non-programmers to change the operator interface. Do you think you would you use this tool?

#	Answer		Response	%
1	Yes		15	27%
2	Maybe		29	53%
3	No		11	20%
	Total		55	100%

Statistic	Value
Min Value	1
Max Value	3
Mean	1.93
Variance	0.48
Standard Deviation	0.69
Total Responses	55

43. Why didn't you use RobotBuilder? (Select all that apply)

#	Answer		Response	%
1	We didn't use a CommandBased robot template		35	34%
2	It is too complex		10	10%
3	We prefer to write all of the code ourselves		68	67%
4	It doesn't allow for updating the model based on code we have added		19	19%
5	Other		29	28%

Other
Labview doesnt use RobotBuilder. Plus we have a system already for using subsystems.
User interface was a bit clumsy
Seemed like overkill
We used LabVIEW (and our own RobotBuilder-like scheme)
I felt as if my kids would learn a lot less. We had ready and willing students who wanted to learn, and I felt that the tool was more for teams that struggle running "Hello World"
It makes us uncomfortable to be that far away from the code..
never tried it
We already had a base that we liked from the previous year using the command & subsystem structure
Our development process is highly itterative and dependant upon mechanical changes, having the students familiar with the real code allows for more flexibility in the design IMO.
We disliked where it placed the sensor & actuator variables (we prefer inside subsystems)
IDK
We didn't really need it, and even though we use the command based template, we don't always use it the way it's intended to be used
Students get more experience by doing more themselves (although it's good for them to also experience tools like RobotBuilder).
It doesn't support LabVIEW
We didnt need it.
followed the cookbook and wrote code ourselves
Already comfortable just programming in LabVIEW and didn't want to take the time to investigate it.
I think the students tried and it had a problem with adding things later, so they just added things manually.
Unaware
We had our own typical structure set up from the last few years, which was more transparent.
Used LABVIEW for most of the season.
(used LV)
When we ran into problems program would crash on cRIO with little explainion on how to fix it. Had a lot of trouble interfacing multiple systems of a complex robot.
Student decision - we do plan to use it in 2014
It simplifies things that don't need to be
It didn't have all the features we wanted
This decision was made by students before I was involved.
Mentioned to students but no interest in using the tool

Statistic	Value
Min Value	1
Max Value	5
Total Responses	102

44. What did you enjoy about RobotBuilder? (Select all that apply)

#	Answer		Response	%
1	It generated the boring boilerplate code for us		13	93%
2	It helped structure the program for us		11	79%
3	It allowed non-programmers to change the operator interface		4	29%
4	It generated a nice table of how to wire the robot to match the code		8	57%
5	Other		0	0%

Other

Statistic	Value
Min Value	1
Max Value	4
Total Responses	14

45. Would you use RobotBuilder again?

#	Answer		Response	%
1	Yes		15	94%
2	No		1	6%
	Total		16	100%

Statistic	Value
Min Value	1
Max Value	2
Mean	1.06
Variance	0.06
Standard Deviation	0.25
Total Responses	16

46. Why didn't you use an FRC supported dashboard? (Select all that apply)

#	Answer		Response	%
1	We didn't need one		11	50%
2	We used an alternative dashboard		5	23%
3	We didn't have the time or resources to figure out a dashboard		9	41%
4	Other		6	27%

Other
lack knowledge to customize
Worried about network traffic and slowing down our DS
We read the source and it isn't implemented in a way that is confidence inspiring and worth the effort.
We wrote our own since we didn't know how to use the SmartDashboard, and honestly didn't try to. The one we made worked really well with widget-like support and MJPEG streaming.
Text output on the driver-station was good enough

Statistic	Value
Min Value	1
Max Value	4
Total Responses	22

47. What did you like about the LabVIEW dashboard? (Select all that apply)

#	Answer		Response	%
1	It was installed by default		45	60%
2	It showed me what we wanted by default		31	41%
3	We could customize it with LabVIEW		53	71%
4	Other		7	9%

Other
had a camera view
It didn't crash, and it supports network tables.
We prefer SmartDashboard, but the NetworkTables issues make it unreliable in competition (at least currently)
It looks nice
It's much more flexible and customizable than SmartDashboard for Java and C++
the video
Low bandwidth, easy library

Statistic	Value
Min Value	1
Max Value	4
Total Responses	75

48. What did you like about SmartDashboard? (Select all that apply)

#	Answer		Response	%
1	We didn't have to write code to customize it		53	60%
2	Test mode is very useful		34	39%
3	It's easy to integrate with the robot program		67	76%
4	Other		14	16%

Other
Outrageously easy to use. Reasonably easy to create custom plugins (would like a couple more examples)
Easy to create flashy and useful displays
It made adding things to the dashboard so much easier. So much.
We didn't like it. Too buggy, but had no choice
CameraExtension widgets crash after a while
Works great for debugging/PID tuning, but had issues with automatically adding a selectableChooser during competition.
Our head programmer had a irrational hated of all things LabVIEW and forced the switch away from the LabVIEW dashboard. We were sucessfull.
Sending data, video
same language as robot code
Plugins
It gave very customiz-able feedback
Very good tool for debuging

Statistic	Value
Min Value	1
Max Value	4
Total Responses	88

49. Why do you use a dashboard? (Select all that apply)

#	Answer		Response	%
1	To debug/test		110	81%
2	To provide information to drivers		120	89%
3	To help determine setpoints and/or tune PID controllers		78	58%
4	Other		12	9%

Other
To give additional robot controls
vision processing
Log data
Vision processing
Set the mode of autonomous we want to run.
view camera image
vision processing ran on it. see http://www.chiefdelphi.com/media/papers/2854
Image Processing
to choose between drive modes that are more "comfi" to diffrent drivers
to have extra inputs to the robot code.
Drivers have very little attention tion to give to Dashboard during match play.

Statistic	Value
Min Value	1
Max Value	4
Total Responses	135

50. RoboRealm is a tool that that assists in vision tracking. It allows you to design an image processing algorithm graphically with continuous feedback. It shows what the image looks like at each step of processing, to help make design and debugging easier. The result can run on the DriverStation and send feedback to the robot over a

variety of network protocols. Do you think you would you use this tool?

#	Answer		Response	%
1	Yes		27	40%
2	Maybe		33	49%
3	No		7	10%
	Total		67	100%

Statistic	Value
Min Value	1
Max Value	3
Mean	1.70
Variance	0.42
Standard Deviation	0.65
Total Responses	67

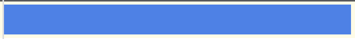
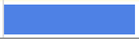
51. Why didn't you use RoboRealm? (Select all that apply)

#	Answer		Response	%
1	Didn't have time/resources to figure out vision processing		0	0%
2	Didn't need vision processing		0	0%
3	Was afraid of networking issues		0	0%
4	Other		0	0%

Other

Statistic	Value
Min Value	-
Max Value	-
Total Responses	0

52. Would you use RoboRealm again?

#	Answer		Response	%
1	Yes		8	73%
2	No		3	27%
	Total		11	100%

Statistic	Value
Min Value	1
Max Value	2
Mean	1.27
Variance	0.22
Standard Deviation	0.47
Total Responses	11

53. Please rank the tools from most useful to least useful. (1 being most useful)

#	Answer	1	2	3	4	5	6	7	8	Total Responses
1	ScreenStepsLive	23	27	6	5	3	1	0	1	66
2	ChiefDelphi	83	43	20	12	3	0	0	0	161
3	FIRST Forums	2	10	19	10	18	11	5	0	75
4	Tutorial Videos (FRC Mastery, YouTube)	12	21	24	25	11	3	2	0	98
5	Other teams (local, at events)	11	31	22	11	15	5	2	0	97
6	Integrated Help (popup help in Wind River, NetBeans, LabVIEW)	36	23	27	12	6	7	2	0	113
7	Quick Build Days	0	1	1	2	2	3	2	1	12
8	Demonstrations/Seminars	6	5	16	23	10	3	1	1	65
	Total	173	161	135	100	68	33	14	3	-

Statistic	ScreenSteps Live	ChiefDelphi	FIRST Forums	Tutorial Videos (FRC Mastery, YouTube)	Other teams (local, at events)	Integrated Help (popup help in Wind River, NetBeans, LabVIEW)	Quick Build Days	Demonstrations/Seminars
Min Value	1	1	1	1	1	1	2	1
Max Value	8	5	7	7	7	7	8	8
Mean	2.18	1.81	4.13	3.19	3.11	2.63	5.25	3.68
Variance	1.94	1.08	2.47	1.99	2.31	2.52	3.11	2.03
Standard Deviation	1.39	1.04	1.57	1.41	1.52	1.59	1.76	1.43
Total Responses	66	161	75	98	97	113	12	65

54. What did you feel was the largest restricting factor in your team's software development?

#	Answer		Response	%
1	Our ability to understand the documentation		6	3%
2	Our ability to find documentation		16	9%
3	The time we had to develop software		82	47%
4	Our ability to understand software tools		13	7%
5	Our ability to understand the libraries		10	6%
6	Other		48	27%
	Total		175	100%

Other
Time
Lack of prior student knowledge, and lack of tutorials that actually teach the programming language (Java) - not just list the FRC/WPI APIs - that the students can work through outside of team meetings
Finding VxWorks documentation
debugging C++ problems, vision toolchain complexity.
Training new members
Library quality
Not enough knowledgable programmers.
Usage of sensors.
Mechanical team. If we had more time prior to ship we could have fixed more issues and had a ton more features. Not your guys fault.
The fact that our programming lead was almost hospitalized for depression
We could be more complex development in offseason and translate that knowledge to build season. We just don't as much as we could.
The programmers' understanding of programming
Not having a full sized field to test with
Time to educate students who are new to programming so that they have the confidence to develop code independently.
Multiple Reasons... Outdated & Incorrect Documentation, Inconsistent API function between languages
We didn't really have too many limitations (we developed everything we wanted to).
Lack of openness in the "lower levels" of the software stack (below WPILib, e.g. FRCNetworkCommunication)
Bugs in networktables, lack of a good logging solution
the shortag eof members
We had numerous technical issues with labview
The time we had to test software on the robot
Matriculating students capable and determined enough to program an FRC robot
Limited understanding of algorithms and controls design.
really slow reboot times
We only had a single qualified programmer
It took so long to download code to the cRIO that debugging was impossible sometimes.
we were new to programming.
The lack of support from the hardware team
Terrible compile/load times in LabVIEW, and bad wireless boot times.
our knowledge of certain features
The time we had to test software on the robot we had.
Labviw's quirky GUI elements
The amount of programmers on our team.
Our limited number of trained programmers
We ran into reliability issues with the cRIO code and had to engineer around it. Try capturing encoder values using an interrupt when a disc enters your indexer or when your wrist zeros, and you'll find all sorts of nasty bugs.
LV not suited for multiple programmers
The obscure bugs we found in WPILib took more of our development time to work out than we would have liked. Nevertheless, our most limiting factor was understanding advanced control algorithms we wanted to use, like Trapezoidal Motion profiles and Kalman filters.

Not aware about software development
Not having a modern IDE supported (Intellij?) and the very old JDK (my phone runs Java 1.7)
C-rio issues
Access to robot for debugging
teams stability
Bugs in the SmartDashboard causing us to spend extra time circumventing the bugs
the limtisons you have from first
The time it takes to run the code for testings.
Poor documentation
Student require training in software development, and doing training at the same time as developing code for a robot is hard
experience mitigates the other factors

Statistic	Value
Min Value	1
Max Value	6
Mean	3.85
Variance	2.29
Standard Deviation	1.51
Total Responses	175

55. Were you able to answer most (or all) of your questions from ${q://QID31/ChoiceGroup/ChoiceWithLowestValue}?

#	Answer		Response	%
1	Yes, we found that ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} had answers to most of my questions		133	76%
2	No, we continually had to use other sources as well		41	24%
	Total		174	100%

Statistic	Value
Min Value	1
Max Value	2
Mean	1.24
Variance	0.18
Standard Deviation	0.43
Total Responses	174

56. Where did you expect to find most information about tools and libraries available for FRC? (Select all that apply)

#	Answer		Response	%
1	From a commonly available link on USFIRST.org (Something that can be memorized and easily typed in)		89	51%
2	Through the Technical Resources page on USFIRST.org		102	58%
3	An insert in our Kit of Parts		16	9%
4	A 3rd party website dedicated to FRC Programming resources		66	38%
5	We expected to have to use a search engine such as Google		74	42%
6	Other		23	13%

Other
Perusing the WPI Library in Labview
One of the key issues with controls documentation is how spread-out the information is. Looking for known bugs with the current system? That's one location. Looking for startup info? That's under the kit of parts on the FIRST website. Looking to get a copy of LabVIEW because your DVD was lost? You won't find it. Trying to keep up with new file releases? Check every day or you won't know when they're posted. In fact, just finding the integrated FRC LabVIEW examples through the LabVIEW front menu isn't intuitive. That's the benefit of integrated help - it's always there and easy to find.
I feel like the C++/Java resources are scattered all over the place. Control system stuff is on FIRST, software stuff is on WPI, you need to go to NI to get updates. One centralized locations, even if it was just a better splash page then tech resources would be nice
Documentation distributed through WPILib
Built in documentation
source code
Common link in Workbench
I'm a CSA and inspector, I read more CD than pretty much anyone, I know where most stuff is. I wouldn't think to look at the kop papers.
WPI Documentation
Packaged with the software.
Chief Delphi
Online. There is a lot of info in the web.
Mentors
Mentor with the team since the first FIRST season.
I tended to use the WPILIb documentation, Chief Delphi for cRIO details
FIRSTForge
Some of our problems involved referencing WPILib's implementation, which was most easily accessible from the .zip file that comes with each WPILib release.
chiefdelphi
Chief Delphi
knowledge passed on through generations
This is terrible. It is all spread out all over the internet.
Chief Delphi

Statistic	Value
Min Value	1
Max Value	6
Total Responses	175

57. Overall how useful did your team find the content on the documentation tools?

#	Question	Difficult to understand				Easy to understad	Total Responses	Mean
1	ScreenStepsLive	0	0	8	10	18	36	4.28
2	ChiefDelphi	2	6	12	48	49	117	4.16
3	FIRST Forums	4	16	46	29	14	109	3.30
4	Videos	0	7	32	29	29	97	3.82
5	Integrated Help	3	6	34	30	26	99	3.71
6	Quick Build Days	2	2	16	4	9	33	3.48
7	Demos/Seminars	1	2	34	20	16	73	3.66

Statistic	ScreenStepsLive	ChiefDelphi	FIRST Forums	Videos	Integrated Help	Quick Build Days	Demos/Seminars
Min Value	3	1	1	2	1	1	1
Max Value	5	5	5	5	5	5	5
Mean	4.28	4.16	3.30	3.82	3.71	3.48	3.66
Variance	0.66	0.86	0.99	0.90	1.05	1.32	0.81
Standard Deviation	0.81	0.93	1.00	0.95	1.02	1.15	0.90
Total Responses	36	117	109	97	99	33	73

58. Overall how useful did your team find the content on the documentation tools?

#	Question	Useless				Useful	Total Responses	Mean
1	ScreenStepsLive	0	1	10	8	17	36	4.14
2	ChiefDelphi	1	8	8	29	71	117	4.38
3	FIRST Forums	10	15	47	21	16	109	3.17
4	Videos	2	4	37	25	29	97	3.77
5	Integrated Help	5	12	29	18	36	100	3.68
6	Quick Build Days	7	5	16	0	6	34	2.79
7	Demos/Seminars	4	8	36	12	12	72	3.28

Statistic	ScreenStepsLive	ChiefDelphi	FIRST Forums	Videos	Integrated Help	Quick Build Days	Demos/Seminars
Min Value	2	1	1	1	1	1	1
Max Value	5	5	5	5	5	5	5
Mean	4.14	4.38	3.17	3.77	3.68	2.79	3.28
Variance	0.87	0.89	1.27	0.99	1.49	1.68	1.10
Standard Deviation	0.93	0.94	1.13	0.99	1.22	1.30	1.05
Total Responses	36	117	109	97	100	34	72

59. What sorts of things was ${q://QID31/ChoiceGroup/ChoiceWithLowestValue} missing? (Select all that apply)

#	Answer		Response	%
1	Information about the tools that we can use		7	19%
2	Solutions to common problems that we have		15	41%
3	Examples of code for certain tools		17	46%
4	Examples of code for common items (such as a drive train, a PID controller, or vision tracking)		20	54%
5	Other		8	22%

Other
Instruction on the programming languages themselves that teach the skills to be able to extend the code beyond the Getting Started example code. It doesn't help to just give example code if teams can't understand it
Known bug list
Typically pointed us to the correct place to find needed information
example for how to set up c++ debugging would be nice
SunSpot-specific configuration/deployment/debugging
The manual.
Information on the specific sensors and VIs we were using
A general procedure for pinpointing code inefficiencies, like using the profiler, CPU utilization vi, & Greg Mckaskle's duration vi.

Statistic	Value
Min Value	1
Max Value	5
Total Responses	37

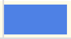
60. What issues did you have while using ScreenStepsLive? (Select all that apply)

#	Answer		Response	%
1	It was missing information that we needed		21	32%
2	It was hard to understand		3	5%
3	It was hard to navigate		15	23%
4	We had no issues		31	47%
5	Other		10	15%

Other
Wasn't in depth
Often seemed biased to a single language
I would prefer PDF so I can have them offline. We don't always have internet access when working on robot. PDF files on ScreenStepsLive didn't seem to always be in sync with the html pages.
sometimes we missed updated info
Didnt use it
It's fine. The big issue is that it is hard to find the one piece of info you need at the right time. All of the technical resources are too scattered around and frankly, clever nemaes like ScreenStepsLive do nothing to make it easier.
We didn't find it detailed enough. However, a new team could find it useful
search is poor
The examples are mostly in Java and not in C++
Could not download answers

Statistic	Value
Min Value	1
Max Value	5
Total Responses	66

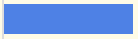
61. What issues did you have with the FIRST Forums? (Select all that apply)

#	Answer		Response	%
1	Our questions weren't answered		12	16%
2	The answers we got were not helpful		15	20%
3	We could not find a section for our question		16	21%
4	We had no issues		40	53%
5	Other		10	13%

Other
We didn't realy use the official FIRST Forums much this past season
Helpful answers were not quick.
couldn't create account
I like it better that CD, but it does not seem to be as popular.
Blast the Trolls.
I didn't know about them.
Not very active community
Did not use.
maybe we found co tent less useful since we already have a handle on that content

Statistic	Value
Min Value	1
Max Value	5
Total Responses	76

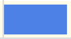
62. What issues did you have with the Videos (FRC Mastery, YouTube)? (Select all that apply)

#	Answer		Response	%
1	There were no videos covering topics that we needed		27	27%
2	The videos lacked information		21	21%
3	The videos were too confusing		8	8%
4	We had no issues		49	49%
5	Other		6	6%

Other
Didn't know about them
The Mastery stuff is great but isn't updated.
Youtube blocked at school
out of date info
We had to watch the video - I'd much rather quickly read through a document with pictures.
we knew about LV Mastery, but not FRC Mastery.

Statistic	Value
Min Value	1
Max Value	5
Total Responses	99

63. What issues did you have with the Integrated Help (popup help in Wind River, NetBeans, LabVIEW)? (Select all that apply)

#	Answer		Response	%
1	The documentation wasn't easily accessible		13	11%
2	The documentation was incomplete		28	25%
3	The documentation was incorrect		5	4%
4	We had no issues		67	59%
5	Other		15	13%

Other
Found it to be slow
The paths for the Javadoc and WPILib source had to be configured manually each time
Had to go out of the way to include sources/javadocs
Sometimes you have to know what you are looking at the fully understand the help.
Insufficient JavaDoc
It was slow/laggy on our computer
This really isn't a big deal, but a few of the javadocs for some of the more obscure classes/methods are missing/incomplete or have some formatting typos.
The WatchDog documentation is an extended metaphor. A good and humorous metaphor, but not really documentation.
I find library documentation convoluted. Though it might be me since I am fairly new with C++
documentation written at a higher level of expertise than the team members needed
when examples were absent, at times difficult to understand operation without experimentation

Statistic	Value
Min Value	1
Max Value	5
Total Responses	114

64. What issues did you have with or why didn't you attend any Quick Build Days? (Select all that apply)

#	Answer		Response	%
1	There were no Quick Build Days in my area		24	27%
2	There were conflicts on the day in my area, and we could not attend		9	10%
3	The language used by the demonstrators is not the language our team uses		6	7%
4	We had no issues		39	43%
5	Other		23	26%

Other
We hosted the quick build day in our area
We have enough team knowledge already to be able to do without - the quick build days only cover very basic skills
Had experienced members to teach
Not applicable to our team
Didn't need it
Our team kick-off and development cycle didn't work well with the Quick Build paradigm
No benefit from attending
Didn't know about it
Our team is experienced enough to not find any benefit from quick build days
We chose not to attend because we saw little value in it.
We don't need to go to a Quick Build Day because we don't use the kitbot and we know what we are doing
We've been around long enough that we think we are better off not touching anything for a couple of days and just working on strategy.
We hosted one, so we didn't use one.
Too early in season - did not want any build activity occurring before strategy analysis.
No time and we understand the system better than anyone in the area.
We just didn't go to one.
Never heard of them
Didn't need to
Team has plenty of experience
The build days covered simple topics
We are trying to start designing before building.
Baseline Programming

Statistic	Value
Min Value	1
Max Value	5
Total Responses	90

65. What issues did you have with or why didn't you attend any Demonstrations/Seminars? (Select all that apply)

#	Answer		Response	%
1	There were no demonstrations in our area		18	15%
2	The demonstrations did not cover tools our team could use		20	16%
3	There were no demonstrations for our team's language		6	5%
4	We had no issues		72	59%
5	Other		17	14%

Other
We have enough team knowledge already to be able to do without - the seminars only cover very basic skills
Had experienced members that could teach
Demonstrations are nice, but there is never enough time to cover the topics that everyone wants to cover. That makes them fairly ineffective in helping out individual teams.
Not applicable to our team
We were the ones giving the demos. :)
We could not attend due to timing
We didn't find any of them useful or necessary for our needs
Didn't know about it
Our team was the one presenting at demonstrations and seminars for Java
The demonstrations were rather elementary; for example, the Java demonstration simply showed how to install the NetBeans IME, which in itself is not even a tool every team will want to use, nor is it a difficult enough task to merit attending a seminar.
We feel we can do it better ourselves.
Stuff we already knew
Personal lack of time; we beta tested, and knew most of our information. We hosted a pre-season seminar-thing.
We were unaware of the existance of demos/seminars
We just didn't go to any.
Attended Seminars were less focused and more broad.
Technically, I did attend one, but I was the instructor.

Statistic	Value
Min Value	1
Max Value	5
Total Responses	122

66. Is there anything else that you wanted us to know about? Areas we are interested include, but are not limited to: Documentation, Tools, Sensors, Libraries, or anything else controls related that you think would have helped you be more successful this past season.

Text Response
We would use the LabVIEW robot simulator if it supported Java
Keep written documentation (located in single pdf for example) up to date. Polish to remove ambiguity. Provide working examples.
It's very hard to test code without a CRIO, creating a process bottleneck.
Would love to see a robot simulator for Java.   It's too costly and difficult to train kids with the actual hardware. Love the command based framework.
A missing feature is much better than a buggy feature.   Also, it would be great if the development and control system could be installed in a reasonable amount of time.
The gyro was not helpful as is. We needed to use a kalman filter with the accelerometer as well. a Kalman filter class would have been nice.
One area that was a bit lacking as my team tried to develop more advanced software was documentation of the underlying system: VxWorks and the Squawk JVM. I was not sure at what level it was safe to use things such as multithreading, networking, C++ dynamic memory management, etc. When we were working with C++ in the beginning of the season, we had trouble with not being able to depend on some of WPILib's newer features such as SmartDashboard, Network Tables, and Command Based Robot, and thus were not able to use them. After switching to Java we have not had as many issues. For the most part, though, my experience with WPILib and the FRC Control System/Programming Tools has been a pleasant one.
I would like to see two things to help make competition easier for teams. The first is a single web-page broken up by section with each pertinent link. FIRST's web page is a start, but it's usually structured poorly and you end up having to click through links to get to the information. Essentially, there needs to be a 'one stop shop' in terms of controls. That includes hardware.   With regards to controls hardware, many teams are school based and have teachers leading them. These teachers rarely have expertise in manufacturing equipment and design. That means they rarely know trade names, brands, and types of sensors. In addition, it can be a real bear to source those sensors at reasonable prices. Through-beam photoeyes, energetic or retroreflective photoeyes and ferrous proximity sensors are just a few items that highly successful teams regularly make use of. I think the 'next step' for FIRST is to build a library of components with explanations of how they can be used. Consider it a FIRST controls 'cookbook'. In addition to explaining how a through-beam photoeye can be useful in detecting game pieces, include suppliers of those particular items. We are STILL using the adjustable Rockwell photo-eyes that were supplied for line following in 2011, and those parts introduced a lot of teams to the idea of using sensors. Even if one sensor of each type were supplied to teams so they had a part number to reference, that would be a step forward. The gyro included is a key example of a high-tech part that's become ubiquitous through inclusion in the kit-of-parts.
More online examples of advanced programming on ScreenstepsLive: Vision Targeting, custom control algorithms, etc.
Anything related to little tips and tricks are always useful and more documentation on some of the more obscure things about electrical/programming are welcome.
- a one place to get started at WPI website that has a rundown of the different robot products WPI offers with pointers to documentation. - wpi is doing great stuff, it's hard to hear about it.
Vision processing on the cRIO is very difficult. The cRIO has a very difficult time working with the Axis camera and doing anything with the images. Our team resorted to writing our own vision libraries and running it on a raspberry pi coupled to the network in order to get the vision results we wanted. Hopefully this will be improved with the athena controller.
It felt at points like the documentation and libraries were disconnected in terms of where to find them, compared to the easiness of getting LabVIEW information and libraries. The documentation would also have levels of incompleteness, depending on where we managed to find the information.

We wish that more methods in the libraries were public so that we could build off of them. We wanted to modify some of the WPILib source code and we had difficulty because some important functions were private.
You all are fabulous.
I think an MJPEG Library for Java and C++ would vastly improve team's ability to utilize cameras other than the AXIS (such as Kinect or Asus Xtion)
Last year there was a cookbook document, which wasn't quite complete, but was very helpful. This year, I didn't see similar documentation. The JavaDoc could be better.
I really like the concept of the documentation coming in with the updates. It seems there are too many places to get documents and they are not all synchronized so I never really get the feeling that I found all there is. I know his is a huge task and generally I think you are doing a good job. I don't really like ScreenStepsLive, but I am slowly getting use to it. Biggest issue is that we are not always connected when we need it. A local copy of it would be nice. Have it in a SVN repository so we can just update it. Maybe you could roll out an email message to tell us to update. That would really be cool! This could also teach teams about SVN.
Easier understanding of the library, and how to do vision processing.
We have never had success with integrating vision processing successfully enough to rely upon it. I found it harder to find the documentation this year than in past years due to the many different sources of information and them not being linked to through a single point (ie. the Technical Resources on FRC page) as previously. I noted several points where old documentation co-exists with newer documentation which caused some confusion as to which was correct. Some parts were updated and others were not. This was particularly true during the installation of WindRiver and LabView. Note that we did not initially like the SmartDashboard but came to like it a lot as we learned to use it. However, we never got the test mode to work (at least not as we thought it should).
RobotBuilder - Better integeration of all the hardware and sensors to it. WPILibJ - Abillity to easily use the GearTooth sensor's rate function in a PID Controller, and to customize what is shown in TestMode in the SmartDashboard, for each "section"(what it the PID input? in the GearTooth it automatically shows the counts, and we used the rate, so it was counter-intutive). Overall - Really great job! Definitely waiting for more great stuff :)
1) The 3 languages vary greatly in the depth of their support of various feature of the hardware. example is encoder averaging, not exposed in C++ or JAVA. CAN Periodic Status only available in Labview. A better job needs to be done making all 3 languages support the same things. 2) The documentation for ALL products is incomplete with regards to FRC functions. Better examples tied to the documentation, and keeping the documentation updated. Example... JAVA/C++ Iterative Robot Template radically different than the documentation.
The serial library for java is tricky to use compared to the rest of the library.
The lower levels of the robot software stack (FRCNetworkCommunication, etc), along with the FMS, should be open-sourced (with the exception of safety functions) to allow full diagnosis and trouble-shooting of field communication issues
NI Sucks
We would have a random Crio complete failure at the start of some matches. The problem was never determined and it caused us to be disabled in a regional quarter final match.
Our controls problems were not due to WPILib. Basically, reliably automating hardware control is hard, particularly given limited time with a functional robot.
A lot of the low-level stuff in the LabVIEW WPILib is very poorly documented, especially compared to how well-documented it is in Java and C++.
Pneumatics in labview were confusing to use

It seemed as if the SmartDashboard complicated the implementations of giving and collecting data from the dashboard compared to previous years.
Most teams (including ours) operate by building and wiring for most of the build season, giving the programming team only a few days to test their code on the robot. There really needs to be an easy to use test interface that teams can run their code against in order to get it working correctly quickly. It doesn't have to simulate a whole robot - give the teams a graphical interface that can let them drag and drop basic components on it to represent wheels on a drive train, or a motor that controls a rotating part hitting a limit switch or providing encoder/potentiometer feedback. In essence, it's an abstract form of whatever robot the team is building. Make the system easily extensible so the community can develop modules that provide additional types of sensors or actuators as they need.
Where we found the documentation incomplete in Labview were the some of the blocks in the vision example where there were many inputs and outputs to the block but it was not another subVI. With sensors, we had issues controlling a low inertia system to where you would give it a speed setpoint and once the wheel reaches that speed, the output command would go to zero. We had a few alternate methods but ran out of time to test them. We ended up shooting the discs whenever the wheel reached a certain speed and waited to shoot the next disc until the wheel returned to that speed. Some of the code in the libraries can be written very inefficiently. There is much discussion on ChiefDelphi by Team #33 about this. More documentation on the advanced features of Labview (tasks, scheduling...) would be helpful.
Faster reboot times would be awesome :)
There seemed to be quite a lack of support and tools for Linux and Mac OS, such as the control dashboard (the name escapes me). Windows dependence is something that I personally (not necessarily the rest of the team) would like to eliminate, even if certain members continue to choose Windows.
More documentation and examples of integrating different sensors would be helpful. Also, more info on options for drive base control, something beyond standard Tank and Arcade drives. We learned from another team of a hybrid/split arcade drive where each stick on an Xbox controller is used for one axis in Arcade drive. That is helpful.
We would have liked to do more in the quick build session than just install the LabVIEW updates.
Other than the float (the tendency of the gyro value to change even when the robot is still) and drift (the tendency of the gyro to continuously change at a steady rate, regardless off robot movement (i.e. 5 degrees/minute)), there were no serious issues that did not arise from user error and stupidity.
Please make vision processing easier to use.
More info/examples on how to use sensors. Thank you!
The technical resources section of the FRC website lacks organization. One shouldn't have to try a lot of links to find what the need. Naming should be more descriptive.
IT would be nice to have more complete documentation on ScreenStepsLive
Our biggest issue was getting sensors to communicate through the digital sidecar. There was very little documentation on what we were trying to do and as a result found very little help anywhere.
I like the demo programs in LabView. Actual wiring diagrams are shown on the program forms. I often use Labview just to make sure everthing is wired correctly before going to C++. Would like this sort of documentation in C++
Javadoc (and the others, no doubt) benefit from more code comments with example code, units, figures and pictures. There is no authoritative source for all information, so the students (especially new ones) have difficulty piecing together an understanding.

Please, please use a different method of connecting to the field. WiFi can still be used but use a better class of bridges at least for the competitions. The current D-Link ones are flakey and have problems that even, I as a CCNP still had issues throughout the competition seasons. The 2012 season really showed that we need hardened bridges. Please fix this in 2015.
Programming is not my focus, but I am reasonably competent. I have found, however, the Labview, Java, or C++ programming to be extremely difficult to begin. I would love to begin developing some younger team members to become team programmers, but have not been able to master the skill sufficiently myself. The information on Chief Delphi is so advanced that I am embarrassed to ask questions, and the documentation I have been exposed to has been extremely difficult to follow.
Eclipse / Windriver is intimidating to new students, since most of its functionality is almost never used. It might be interesting to create a default mode (like Advanced Device Development, Git Repository Exploring) optimized for the stuff teams use - making a new robot project from a sample, running debug, installing svn/git/bazaar/darcs/mercurial plugins. Several times I have looked into the WPILib source to find information on what methods do, i.e. if disabling a PIDController writes 0 to the output controlled, or whether the PIDController's constants are time-invariant, or if the 7-bit I2C address occupies the highest or lowest 7 bits of the uint8_t device address. There are probably a few more such details that the code documentation is not clear about. At least the library code is easy to read and understand.
WPILib documentation could benefit from more detail and more thorough accuracy checking prior to release. WPILib's PID system, if memory serves, did not work with raw PWM. Camera processing software (i.e. RoboRealm) could be given more intuitive design.
I had some difficulty in properly initializing the Labview WPI counter open vi. Creating a constant in for the up source terminal does not properly describe the sensor input parameters. Only found out the correct way by referring to example code plus digging deep into the help files. The pop-up help could be a bit more detailed in how this terminal should be used.
Explaining complex parts of the documentation more. For example, something like a parameter being "direction" with the description being "Degrees the robot should travel" doesn't fully explain it. It doesn't allow the user to know if it means degrees relative to the robot, field, joysticks, or something else. However, very little was this a problem.
WPILib, while it provides a lot of the functionality we need, does not seem to have a simple design and has difficult-to-read source code that seems more optomized for Java than C++. Unfortunately, reading the source code seems to be necessary, as the library reference documentation is not complete. Additionally, the amount of boilerplate needed to accomplish simple tasks in the Command-Based robot model makes on-the-fly changes to the code during competition difficult. The documentation also does not address how to reset the scheduling and PID controllers when entering disabled mode, which resulted in our team rebooting our cRIO every time the robot was disabled, because it would resume in an unexpected state. The network tables documentation for C++ seems spotty at best, and we abandoned our attempts to use networktables and test mode to tune PID because of this. Finally, as I mentioned in a previous section, the PID controller class had a bug/misdesign that caused it to be nonfunctional for setpoints other than zero, because it would only set the output to the change, rather than the current added to the change. We had to write a wrapper class for our motor controllers to enable this behavior.

Please break WPILib up into 2 pieces.   1) Low level sensor interfaces. Like really low level. Don't mix network tables code in there, and put them in a namespace or name them "AnalogRaw" or things like that. 2) Higher level code like CommandBasedRobot, and all the sensors that have network tables features.   I like that you guys are trying to build systems for teams to make programming easier, but the lowest level objects should be light weight and as simple as possible.   Fix the Notifier class so that it can actually be used for controls. It needs to be X hz, +- 1% for it to be worth my time. It has gotten to the point where when I go to help another team with controls problems, we always end up fighting the timing jitter. I have run into this problem with both C++ and Java.   I keep running into the problem where I need to capture an encoder value when a hall effect sensor goes high. I have run into this on multiple teams. Make this easy and test it very carefully.
The deploy time for LABVIEW code to get on our robot hindered our team's ability during competitions when we needed to fix something quick.
The camera/OpenCV memory leaks in SmartDashboard need to be fixed
We wish the WPI library was better designed. I ended up writing my own networking utilities, fileio, thread utilities, command templates, PID controllers, subsystem and joystick classes. The network tables are hard to use, the notifier class does not work as described (period is not constant), the joystick class lacks obvious features like toggle-able buttons, the PID object uses a very simple algorithm, the command object is too complicated and so on.   Furthermore, we wish there was better c++11 integration. I found it frustrating to debug a class on my desktop only to find it used prohibited c++11 features. We are debating switching to a linux development environment for this reason.
Mentor takeovers are NOT for the team's advantage.   It restricts both learning, and success.
No.
We had a lot of trouble figuring out proper structure for autonomous that communicated well with FMS. Methods used in previous years hung up or consumed too many resources. Only after communicating with multiple different teams were we able to work something out 100% at our 2nd event. Whether this was directly because of our robot or not, we don't know, but to eliminate the risk, more documentation regarding FMS and program flow changes would help dramatically.
A document on how to program certain sensors without using the example sensor code from labview in the documentation
I would like a state machine for all languages. Especially labview because it can get very messy very fast. I also would like the iteration and feedback time to be faster. The Killer Bees had a 7ms loop/feedback time, but they had to use a state machine and a bunch of techniques that most teams dont know how to use.
Our team took the summer and reworked our 2013 Robot in Java in order to evaluate whether or not to switch. Team has decided to switch; for the summer rework, we used RobotBuilder and Command Based. Answers to some of the questions were different for the summer session because of different tool set.
An easily accessible API reference for VxWorks 6.3 would have been extremely useful as well. We uncovered some interesting quirks in VxWorks' networking besides such as it only supporting buffering 3 packets at one time. If this is limit is passed on 8 slot cRIOs, it crashes the network stack and the robot loses communications. Other than that, more resources on control and filtering algorithms would have been nice, but I understand that this would be a small subset used by few teams, and thus would be inappropriate for inclusion in WPILib.
RobotBuilder is FANTASTIC for getting the team started in programming with no prior experience. The PID controller needs better documentation on how to use it from WPILib. It needs an example to explain the input and output UNITS and how to use it for rate based PID operation. Reverse engineering the code is too difficult unless a software engineer is involved.

How to better interface command based programming with many complex subsystems.
Documents for onboard processors integration
The command-based parts of the libraries seem overly verbose and unsuited to use in a language like C++ or Java. When we tried writing some sample code using commands, much of it became spread across multiple files and hard to find, and inside each file there was boilerplate associated with the command libraries and not our code. It also seems more suited to use in a functional programming language, like OCaml, Haskell, or Lisp. The IO type in Haskell essentially implements the command-style of programming for standard I/O actions like file I/O and standard input/output/error. The monadic and applicative nature of the IO type make working with and combining several actions easy, while the command libraries require a lot of boilerplate to do similar things. For example, sequencing several actions is easy to read in Haskell (a >> b >> c) but takes up several lines using commands in WPILib.
Please upgrade the control system to support Java 1.6 or 1.7, and use IntelliJ as the default supported editor. The current tools are antiquated and miss the many benefits of Java (static type-safety to prevent common bugs from even compiling, in-place class reloading to avoid long boot-up times, not to mention reflection and annotations). Aside from the strictly technical issues, the disrepair of these tools hurts the image of FRC as a program which brings the cutting edge of technology to our students.
We had several problems with having to re-image the c-rio. Long download times. Compressor problem with provided code. Our developement and maintenece of our climbing state machine was problematic. A state machine builder app?
A bunch of classes and methods in WPILibJ are declared with private access. We had to copy and paste a bunch of code in to different packages to override methods or change some basic functionality. EVERYTHING should be public.
Documentation was noticeably better in 2013. Screensteps was generally good but it was missing an overall organization for documents.
Examples for threads would be helpful. An IterativeRobot Example would also be quite nice, especially if it included a state machine example; I feel the SimpleRobot template teaches a lot of bad practices (ie Wait()). Some sample code added to the examples, with different joystick button handling methods would also be helpful for beginners (active while pressed, toggle on press, if/elseif/else handling, etc), to give beginner programmers ideas for different HMI patterns. I plan on trying out RobotBuilder this year after the great presentation at Champs.
I really did not like the bugs in the Smart Dashboard, we found very strange issues, not all of which I can recall. But there were problems in which the layout would disappear if I pressed "save layout," and issues in which the dashboard would sometimes crash randomly, and once during a match. This was critical as our drivers would select autonomous modes from the smart dashboard and receive real-time video from the camera. The Smartdasboard is a wonderful tool, but it definitely leaves much to be desired.
I'd like you guys to concentrate more on the libraries, and worry less about the extras (robot builder, simulation, smart dashboard, etc.). I find that the Java libraries aren't designed very well in the OOP sense, and are difficult to fit into our programs. I've built a library just to make this easier for us. (github.com/team4334/atalibj) I'd also be happy to contribute to library development, especially with the "front-end" parts (ie. things programmers will worry about). Thanks for everything you do!
The vision processing is quite left aside in tutorials, and we have to learn it mostly by looking at non-FRC sites. Also, we would like to know better the new changes of the 2014 Control System that we could be ready for the season. Finally, if you could provide also some information about how to minimize lags on fileds etc. it would be wonderful.
If there could be a site that has examples to each and every class in the WPI lib it could help a lot.

The common Chief Delphi users and FIRST Forum users don't bring the language down to an understandable level. We commonly have to research many topics to understand one response to a question asked on either site. And they seem to be unforgiving when we admit that we don't know what they're talking about. The control systems work great for basic uses but help for larger applications becomes a little blurry.
It would be interesting to see more information on how to handle sensors that aren't in the standard libraries (for example there are quite a few different ultrasonic sensors using different trigger methods, and I can't figure out which one the one in the Labview library was for or how to handle several of the others.)
Our main problem we had with programming this year related to the complexity involved in originally setting up the system at its current state with updating and other factors. When it came to actual programming we only ran into problems when the mechanical mounts our actual subsystems were not properly developed to support the more complex closed loop systems we wanted to incorporate.
Algorithms for using sensor data can be important
We REALLY need a simulator that supports Java and not just Labview. It still seems that Java is a second class language for FRC. NI donates an incredible amount of resources to FRC, but there should be a way to get more resources to help WPI to support the other platforms.

Statistic	Value
Total Responses	80

67. Source

Value	Total
493	23
848	7
904	9
824	2
624	113
331	4
509	4
80	10
698	4

68. Referrer

Value	Total
https://www.facebook.com/	33
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=493	6
http://t.co/WaksxhdZhp	2
http://m.facebook.com/l.php?u=http%3A%2F%2Fwpi.qualtrics.com%2FSE%2F%3FSID%3DSV_8vHdoVx6YY1mHHL%26Source%3D824&h=WAQESQMKt&s=1	1
http://www.chiefdelphi.com/forums/showthread.php?threadid=120445	68
http://www.chiefdelphi.com/forums/showthread.php?p=1296590	4
http://www.facebook.com/	1
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=904	2
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=624	7
http://www.chiefdelphi.com/forums/showthread.php?s=089c49971902413e52e92f7ec8fd6923&threadid=120445	1
http://m.facebook.com/l.php?u=http%3A%2F%2Fgoo.gl%2FJ1O2Br&h=3AQEHvBKF&s=1	1
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=331	1
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=824	1
http://www.chiefdelphi.com/forums/showthread.php?t=120445	16
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=848	3
http://cloud.feedly.com/	1
http://www.chiefdelphi.com/forums/printthread.php?t=120445	1
http://wpi.qualtrics.com/notify-Survey_Notify?aHR0cDovL3dwaS5xdWFsdHJpY3MuY29tL1NFLz9TSUQ9U1ZfOHZIZG9WeDZZWTFtSEhMJlNvdXJjZT02MjQ=	1
http://www.chiefdelphi.com/forums/showthread.php?s=12ad0ab02736a1ac1b5d11cf509f5185&threadid=120445	1
http://www.facebook.com/l.php?u=http%3A%2F%2Fgoo.gl%2F7vZDAz&h=XAQE8N8Bd	1
http://www.chiefdelphi.com/forums/showthread.php?p=1296897	1
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=509	4
http://www.chiefdelphi.com/forums/showthread.php?s=2634e9412c71c67a45c60db6979314a7&threadid=120445	1
http://t.co/8KEeI6Yjbw	5
http://www.facebook.com/l.php?u=http%3A%2F%2Fgoo.gl%2F2iWzJO&h=fAQGOuFAyAQHG0IWuwGLvoramxFqC3dXkjwefzZWkTcdirQ&enc=AZPOWWoZXF_PDQ55v4IUfpYeeRkDn83Bznj7Dei8k3Jw65mDGnldTOB56ZOB4Hd8Sr8hPF5DWXSm6O4ukdTAAUnzTWK3X4CP5VO9j48aX1vP9n0_EepwIk22mS6KuKCzAKP9E9mTWb9Xh8h-R02YvXZJO6ASJaQiIQuiUMQk9Z5_Lg&s=1	1
http://www.facebook.com/l/qAQH-dW68AQGNupiIZMpldfumTWEgEZYpJlmUQr8gqvZsfg/goo.gl/2iWzJO	1
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=698	4
http://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=80	1
https://wpi.qualtrics.com/SE/?SID=SV_8vHdoVx6YY1mHHL&Source=624	1
http://goo.gl/7vZDAz	1

http://www.chiefdelphi.com/forums/showthread.php?s=e013a6c4399faae5e69dcb4664264c78&threadid=120445	1
http://m.facebook.com/l.php?u=http%3A%2F%2Fwpi.qualtrics.com%2FSE%2F%3FSID%3DSV_8vHdoVx6YY1mHHL%26Source%3D624&h=5AQFFlVz7&s=1	1
http://www.facebook.com/groups/100569823370583/	1
http://www.chiefdelphi.com/forums/showthread.php?t=120445&highlight=version	1