CS 307: LogBotics | Project Backlog

By Team #24:

Team Members: James Gilliam | Roger Braun | HyunShu Kim | Jenna Rigdon

Problem Statement

First Robotics Competition (FRC) Robotics teams face challenges in diagnosing performance issues and optimizing their robots due to the lack of comprehensive data logging and visualization tools. Current solutions do not capture critical metrics such as motor activity, sensor feedback, and control system operations, making it difficult to troubleshoot or improve robot performance during competitions. Teams also lack a way to visualize their robot's movement throughout a match to better analyze strategy. This project aims to develop a software solution that logs, visualizes, and analyzes robot data to help teams improve performance and make data-driven decisions.

Background Information

Audience:

- Primary Users: FRC Robotics teams (students) who will utilize the software for robot performance analysis and troubleshooting.
- Secondary Users: FRC coaches and mentors, who will guide teams based on the data and insights.
- Developers and Technicians: Future developers who will contribute to extending and improving the platform.

Similar Platforms:

FRC Driver Station:

- Contains basic logging functions
- Files are not exportable
- Unclear/hard to read logs
- Very little visibility of output

SmartDashboard:

- Requires user programming to log data

- Difficult to navigate

Limitations of Current Platforms

- Existing platforms only provide basic data logging and lack comprehensive visualizations and detailed metrics. This makes diagnosing performance issues or identifying patterns during matches unnecessarily sophisticated.
- In addition, they do not offer robot path tracking or provide actionable troubleshooting insights. FRC teams lack time or resources to develop custom solutions, and the existing tools are often difficult to set up and use effectively in high-pressure competition settings.

Functional Requirements

Data Logging:

1. As a user, I want the system to automatically log motor activity during matches so that I can diagnose issues
2. As a user, I want the system to automatically log sensor feedback during matches so that I can diagnose issues
3. As a user, I want the system to automatically log control system data during matches so that I can diagnose issues
4. As a user, I want to be able to view each motor's activity separately so that I can see issues with specific motors
5. As a user, I want the data to be automatically backed up such that I can restore the data in case I delete them by mistake.
6. As a developer, I want a framework that can receive, store, and send logged data to the app so that data logging and display can be implemented in the app
7. As a developer, I want there to be a motor object that tracks data, errors, and states of different motors so that visualizing motor activity can be implemented
8. As a developer, I want there to be a robust library of errors with the robot that can be detected with the logged data so that error messages can be implemented
9. As a developer, I want to have a stored database of information on potential errors that could be associated with the robot's performance so that I can warn the user of potential problems the robot is facing
10. As a user, I want to be able to view error messages from the CAN bus on the control system so that they are clear and easy to see
11. As a developer, I want to implement an API for retrieving real-time data from the robot's control system so that it can be streamed directly into the logging system without delay.
12. As a developer, I want to create unit tests for the data logging module to ensure that it accurately captures and logs motor activity, sensor feedback, and control system data.

Data Visualization:

1. As a user, I would like to be able to see graphs of the motor, sensor, and control system activity so that I can better understand the current state and trajectory of the robot.

And, if time allows, as a user, I would like to see tables of the motor, sensor, and control system raw data so that I can catch specific errors or anomalies with the robot system.

2. As a user, I would like to be able to view the robot's path during the autonomous part of a match as a map so that I can record/visualize the robot's resulting motion in a match
3. As a user, I would like to be able to view the robot's path during the teleoperated part of a match as a map so that I can record/visualize how the robot is responding to commands overall
4. As a developer, I would like there to be a framework to back-calculate the expected path of the robot based on motor output data so that the motion visualization can be implemented
5. As a developer, I would like there to be a GUI to prompt the user for wheel dimensions and placement on the robot as well as the number of wheels (allowing between 3 and 6) so that the path can calculated

Data Export:

1. As a user, I want to be able to export individual plots as visual files (jpeg, pdf, etc) so that they are easy to open and use elsewhere
2. As a user, I want to be able to export individual plots as .csv files so that they are easy to open and use universally
3. As a user, I want to be able to export all data at one time or select which data plots to export so that such an export process is simple and/or storage/organization/time isn't wasted on unused data
4. As a developer, I want to implement robust error handling for the export feature to ensure that if the export process fails due to file permission issues, clear error messages are shown to the user, so that they know how to troubleshoot and correct the issue.
5. As a developer, I want to implement data compression for exported files, so that large datasets can be exported efficiently without overwhelming storage space or upload/download time.

System Configuration:

1. As a user, I want to be able to set my team number and team name so that data won't be confused with other teams
2. As a user, I want to be able to create an account on the app so that set up and save my personal settings/customizations
3. As a user, I want to be able to access my account on any device that has the app and log in and out of it so that the software is easier to access and multiple accounts on the same machine are possible
4. As a user, I want to be able to customize the color settings of the app to increase/decrease contrast so that the app is more ergonomic and tuned to preferences
5. As a user, I want to be able to select different visualizations of data logging (various plots, various colors, etc.) so that the data is easier to see and customizable to preferences
6. As a user, I want to be able to add or delete data visualization plots from the screen so that the most important data to see is always displayed/prioritized

7. As a user, I want to be able to open multiple tabs for viewing data to that large numbers of data visualization plots are easier to work with
8. As a user, I want to be able to label these tabs so that data is more organized
9. As a user, I want to be able to move plots from tab to tab so that data plots can be grouped according to needs, priority, type, ect.
10. As a user, I want to be able to freely label plots so that plots are easier to keep track of according to my specific needs
11. As a user, I want to be able to view the source code of my robot
12. (If time allows) As a user, I want to be able to develop my robot code in an integrated development environment within the application
13. As a developer, I want there to be a server-client framework (fat-client) so that accounts can be changed and accessed on other devices
14. As a developer, I want there to be a plot object so that different visualizations, data types, windows, and tiles can be implemented
15. As a developer, I want there to be (or to find) a Windows tab framework and tile objects so that multiple tabs with different data can be implemented

Non-Functional Requirements

a. Performance:

   The data collecting tool will be designed to allow for transmission of data, in real time, to the user's computer during the match, where the user can then instantly access and display the data within the desktop application. There should be little to no time where the user has to wait for transmission of data from the robot to the user's host computer.

b. Scalability:

   We intend for the data logging tool to be scalable in that it should be able to be downloaded and used as a package/application on any number of Windows computers. The tool should readily be accessible to all the participants of FRC should they desire to use it. In addition to this, since this tool will eventually be open source, we intend to make it maintainable and developer-friendly, allowing other developers to add their own features if they would like.

c. Usability:

   One of the most significant components of the desktop app is the ease-of-use. Users should be able to open their desktop application and easily be able to interact with their data in a variety of different, intuitive ways via graphs, tables, and other data formats. In addition to this, users should be able to perform file exports so that they can compare data from previous sessions with their robots. These aspects of the tool are paramount to the user's experience.

d. Security:

Since the application is a locally hosted tool that will run on the user's computer with a connection simply to the user's robot, there is a lesser need for a security system to protect data. However, we will ensure that the database that hosts the user's data is both secure and robust in that their data will not become corrupted in any way while they are using the application.

e. Portability:

   The current intention for the tool is to make it deployable on all computers hosting a Windows operating system.

f. Development:

   For ease of development, all scripts, objects, and functions will be limited to 200 lines. This ensures the modularization of code which helps greatly in debugging, understanding each other's code, and updates. Ideally, functions won't exceed 100 lines. For the readability of code, which is essential for debugging and working with other people, the code standard: https://www.cs.purdue.edu/homes/cs240/code.html will be followed as applicable to ensure comments are formatted the same way in a coherent manner so they can always be understood.