Solar Car Telemetry System

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Joon Tae Kim

Spring, 2022.

Technical Project Team Members

Victor Pham

Vinay Bhaip

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Daniel Graham, Department of Computer Science

Solar Car Telemetry System

A dashboard developed for the Solar Car Team at UVA

Joon Kim
Computer Science
University of Virginia School of Engineering and Applied Sciences
Charlottesville, VA
jtk2rw@virginia.edu

ABSTRACT

The goal of the Solar Car Team at the University of Virginia (UVA) is to design, implement, and maintain efficient and effective solar-powered cars that can compete with other solar cars from across the nation in the annual Formula Sun Grand Prix. To keep the cars competitive, designers and engineers must constantly improve and maintain the car, but such changes must be supported by data showing the changes are warranted. Additionally, it is important for drivers to monitor vehicle performance so they are able to make informed judgements about potential dangers to the vehicle and/or driver. To address this issue, I worked with a team that created a web-based application that displays data collected from the Solar Car. We created the application using asynchronous websockets in the frontend and a Python-centered backend. We also collected and saved data for future viewing and analysis. The web application consists of a few pages, each dedicated to a different component of the car (e.g., solar panels, motors, battery), where various data are displayed and updated in real time. To assist the team with future design changes and improvements, the application gives members of the team a means of monitoring and judging the performance of various parts of the car as it is on the road. Currently, the system is still a work in progress.

1 INTRODUCTION

Creating a piece of technology or software that would allow for use without difficulty is only half the effort involved in creating a successful long-term product. To conduct maintenance and additional research needed to improve the product, performance must be monitored through data, where potential faults and shortcomings may be identified. In addition, the safety of the product may not be guaranteed without proper monitoring of components that may pose any danger to the user. The importance of tracking data for these purposes is amplified when put into the context of the goal of the Solar Car Team at UVA. The mechanical components of the car must be improved and reworked every year in order to stay competitive with other solar cars racing against it. In addition, while gas provides a stable reliable source of fuel to cars, solar power must be constantly monitored to make sure enough power is flowing through the car for the duration of the drive. This is what motivated our team to develop a telemetry system for the team's solar car. The solar car will measure and collect data through its various instruments, and send it out as radio waves through a radio module. Our web application that is hosted on an external laptop will receive this data through a receiving radio module before processing and displaying it through our web application.

Telemetry is defined as "the process of recording and transmitting the readings of an instrument" [1]. Our solar car telemetry system, in particular, handles the displaying of collected data through a web application. The solar car will measure and collect data through its various instruments and send it out as radio waves through a radio module. Our web application that is hosted on an external laptop will receive this data through a receiving radio module before processing and displaying it through our web application. The reception of asynchronous and continuous data is made possible using Socket.IO, a JavaScript library that allows for the establishment of websockets (bidirectional communication channel) between the web application and the computer that is receiving the data [2]. Each individual data value is assigned a key, and the JavaScript

uses this to display the data to their respective positions on the UI as the web socket detects new incoming data. The design of the dashboard UI was implemented using the frontend HTML and CSS framework called Bootstrap.

2 RELATED WORKS

Many similar past projects have existed in the past, each of which have attempted to create efficient telemetry communication systems. Hackystat, starting development in 2001, is a tool to provide "automated collection and analysis of software engineering process and product data", with an aim to simulate a "telemetry-based approach to software measurement trend definition and display" [3]. Initially developed with a focus on communication between client and server, interestingly it was later reworked completely to use a service-oriented architecture to process more types of data and stakeholders. The server-oriented architecture uses distributed computation, caching, and pre-fetching to preserve the quicker asynchronous performance that comes with client-server architecture [3].

Many similar telemetry systems for solar cars have been implemented by other teams with similar motives and functionality. The telemetry system developed by the Missouri S&T Solar Car Team is one such example. They similarly utilize a Controller Area Network (CAN) and broadcast data through a CAN to Ethernet bridge using the User Data Protocol (UDP) [4]. The team uses a chase vehicle that will stay within proximity to the solar car "running the telemetry software layer full-time logging data from the vehicle and reading data off to the team" [4]. The goals of our team may prove more difficult compared to this approach, as we hoped to be able to receive data with such a strong enough connection that the host computer running the software would be able to remain in a stationary position. However, this goal may be unrealistic considering the current budget of the team and vast size of the race track, which stretches up for 2 miles in length.

3 SYSTEM DESIGN

The system design for the solar car consisted of the architecture, the web application requirements, and key components that allowed the web application to function.

3.1 Review of System Architecture

The architecture of the solar car telemetry system can be divided into 3 different tiers: the vehicle layer, the server, and the client.

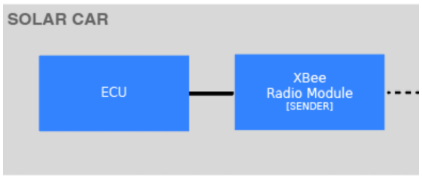

Figure 1. Vehicle Layer

The vehicle layer, as shown in Figure 1, exists inside of the solar car itself and holds the electronic control unit (ECU) which collects data and sends it to the sender radio module. The sender radio module in turn transmits the data to the server layer.

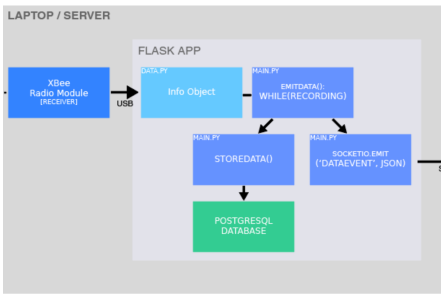

Figure 2. Server side architecture

The server layer, as shown in Figure 2, is connected to the vehicle layer via a radio module, which acts as the receiver for the data transmitted from the radio module on the solar car which acts as the sender. This data is then transferred to the server via a simple USB connection. The server side of the application contains a PostgreSQL database for the storing of incoming data, and creates the websockets for the transmission and communication with the client.

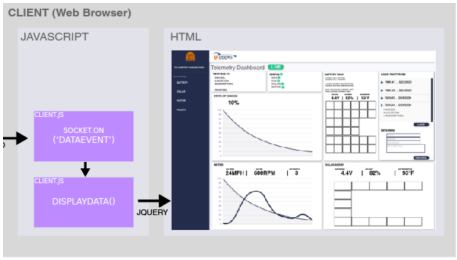

Figure 3. Client side architecture

Lastly, the client-side layer, as shown in Figure 3, receives and displays the processed data sent by the server through a websocket. This layer contains the front-end code for creating the dashboard UI by rendering HTML

templates. It also contains JavaScript code for the rendering and updating of the graphs on the dashboard.

3.2 Web Application Requirements

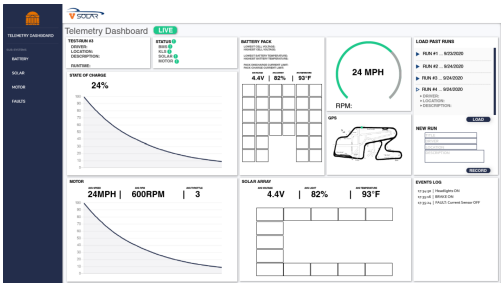

Figure 4. Finished dashboard concept

Ideally, the client side of the web application should be usable to all members of the team. Thus, it is important that the web application displays the appropriate data that each sub team may find useful for analysis. For these reasons the team has decided to divide the web application into different pages for each subteam (battery, solar, and motor). The battery page should display a diagram of the layout of all battery cells, along with their respective temperatures within each cell in the diagram. General statistics such as average battery temperature, lowest/highest cell voltage, pack amp hours, etc. will be displayed as well in a separate table.
Similarly, the solar page will also display a diagram showing the arrangement of solar panels on the car. Each cell in the diagram will display the respective solar panel's generated voltage, light level, and temperature. The averages of these statistics across solar panels will also be displayed. The motor page will display relevant statistics corresponding to each motor on the car such as revolutions per minute (RPM), temperature, throttle, min/max voltage, and min/max temperature.
The web application will also have a general "homepage" that will display a broad overview of all areas of the car. From this screen, users will also be able to initiate recordings of data as well as see the physical location of the car through GPS. A mockup of the ideal layout of the web application with all these design elements in consideration is shown in figure 4.

3.3 Key Components

3.3.1 Socket.IO

The real time display of data as it is transmitted from the solar car is made possible with websockets. We have specifically used a JavaScript framework called Socket.IO for our websocket implementation. Socket.IO enables the establishment of a "full-duplex communication channel" that allows for "real-time, streaming updates between a web server and a browser client" [5]. This communication between server and client can be represented through "emit events" and "listener" methods from either side. Mainly, the "socket.send method will send the message on the socket," classified by an event name, while socket.on listener methods calls corresponding event handlers and "is triggered when a message sent with socket.send is received." [5]

Figure 5. Emitting and listening for Socket.IO events

Our web application uses the socketio.emit method instead, which is identical to socket.send but used to trigger custom events. When the server receives data from the solar car, it immediately emits it as a event named "dataevent" with the data formatted using JavaScript object notation (JSON). On the client side, the socket.on method listens for this "dataevent" from the server. As soon as it receives this event, it executes the JavaScript to display the JSON data onto the dashboard. This interaction between the socketio.emit method from the client and the socket.on listener method is shown in figure 5.

3.3.2 Database and Runs System

A database is essential for any task that requires data to be stored and retrieved for future use. A database is "an organized collection of structured information, or data, typically stored electronically in a computer system" [6]. For our web application, we used a free and open-source data management service named PostgreSQL to manage our tables in which we would be storing data. Data was stored and retrieved primarily through our "runs" system. A user can initiate a run on the main dashboard, which will set a boolean variable to true. While this variable is true, data that is emitted from the socket.emit method will be formatted as JSON before being sent to a method named storeData. The storeData method itself will then unpack the JSON and store each attributed key value pair into their respective columns in the database.

Each entry in the table will also be associated with the ID of the current run that is determined by user input.

Queries are made to the database as the user requests runs to be loaded onto the web application. When the user clicks a button to load the run, an event is emitted with the corresponding run id that is listened for by the server. When the server catches this event, it initiates the query and returns all data associated with the run id, and emits this back to the client as another event. Lasty, the client catches this method, loops through the data entries, and displays the data accordingly.

3.4 Challenges

Our team encountered many challenges beginning with development. One challenge that still stands today is the need to optimize the performance of the dashboard. The application must handle and process a large amount of incoming data sent at a very quick rate, and must constantly update graphs and values on the dashboard in order to display the most up-to-date statistics about the components of the car. We considered a few solutions to address this. Socket.IO comes with a "sleep" method that will pause the channel for a set amount of time. We have placed an instance of this method inside of our method that will emit data, in order to slow down the rate at which data is being sent to the client. We currently have the channel to sleep in 1 second intervals between emissions of data [7]. We also discovered that dividing the application into separate pages based on different areas of the car (see subsection 3.2) would help in that less data would need to be updated on the current page view.

Another challenge encountered earlier in development was related to being able to associate incoming data with correct corresponding parts of the dashboard. Data came into the server in a certain sequence but without any specific label that would help in identifying values apart from each other. With this, we were able to associate each value with a specific key in a Python dictionary using labels that we defined. In this way, the JavaScript could pull values using the labels of key names from the dictionary.

4 RESULTS

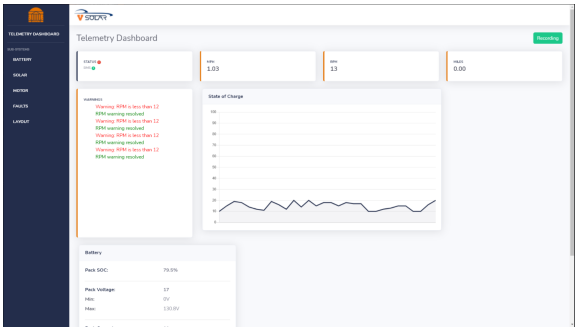

Figure 6. Current dashboard implementation

Although not yet in widespread use by the solar car team due to incomplete development, the web application was able to simulate the receiving, processing, and displaying of data with a class that would generate random values and send the data through a fake serial port. These random values were successfully displayed onto the dashboard in an asynchronous manner. Teammates were able to access the dashboard by connecting to the host computer's public IP address, which was running the application on a local server.

5 CONCLUSIONS

Overall, the telemetry web application would be a very useful tool for the team, as it would allow members of the team to make more informed design decisions based on observations in data displayed by the dashboard. The asynchronous nature of the web application will always keep the team up-to-date with the performance of the car, and will allow team members to help the driver make more informed driving decisions. The database and runs system will allow team members to look back on collected data to help identify strong and weak elements of the car. Although development has not yet been completed, the web application has much potential to provide more benefit to the team. It can be expanded to include many more features and applications in data analysis, as well as more types of charts that may be more insightful than simple digits. Just as the possibilities with computer science are limitless, so can this web application be limitless, given enough research and development.

6 FUTURE WORK

There are many ways in which our web application can be expanded to provide more use for the team. Currently, we don't utilize the database in any other way other than storing data associated with runs. We have

agreed that more meaningful graphs and statistics could be produced by associating data values together and creating new values through calculations between values, but our lack of knowledge of database management and querying languages such as SQL has prevented us from achieving any desirable results with this. There are many more optional features that may be added to the web application that may be useful for the team. We have started looking at developing a GPS system using MapBox's API but have only gone as far as to displaying/updating the current position of the car on the map. The team has yet to find a way to transmit GPS coordinates to the web application, but some possibilities have been explored such as a physical GPS module or a phone that will transmit the coordinates with the rest of the data.

7 COURSEWORK RELEVANCE

Although I conducted this project during my first year at the university, I found the knowledge acquired in CS2150 Program and Data Representation to be helpful in the context of this project. Particularly, acquired knowledge about hash maps proved essential at optimizing the speed of the application. As hash maps provide for constant time access for values [8], our web application was able to access and display values passed in by the server at a desirable pace. The convenience of being able to access values by their associated keys eliminated the need to display values on the dashboard in a specific sequence according to value positions in the data, as well as the need for searches in the data.

8 ACKNOWLEDGMENTS

I would like to express my sincerest thanks to Victor Pham, for working with me on this project since the beginning, and for being able to co-lead the team with me. I would also like to thank Vinay Bhaip, for being able to quickly pick up our work and our position as sub-team leader soon after joining.

I also would like to thank Sandesh Banskota, for onboarding me onto the team. His leadership and assistance during the first year of the was essential to the success of the project and the team.

Lastly, I would like to thank the UVA Solar Car team for giving me the opportunity to work for them in their mission for pioneering greener future.

REFERENCES

[1] Lexico. Telemetry English definition and meaning. Retrieved October 2021 from https://www.lexico.com/en/definition/telemetry

[2] Socket.IO. Introduction: What Socket.IO is. Retrieved 24, 2021 from https://socket.io/docs/v4/

[3] Johnson, Philip & Zhang, Shaoxuan & Senin, Pavel. (2009). Experiences with hackystat as a service-oriented architecture.

[4] Eric Walter, Nicholas Glover, Jesse Cureton, and Kurt Kosbar. Telemetry System Architecture for a Solar Car. *International Telemetering Conference Proceedings*.

[5] Rohit Rai. 2013. *Socket.io real-time web application development build modern real-time web applications powered by Socket.io*, Birmingham: Packt Pub.

[6] Oracle, "What is a database?" Retrieved October 2021 from https://www.oracle.com/database/what-is-database/

[7] Socket.io. API. Retrieved October 2021 from https://python-socketio.readthedocs.io/en/latest/api.html

[8] Oracle, "HashMap" Retrieved October, 2021 from https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html