ECE 3220 Final Project

Max Houck and Marshall Lindsay

FSAE Tire Temperature Visualization

**Abstract**

**Introduction**

Formula SAE is a collegiate engineering design project in which students design and manufacture a quarter scale formula style racecar to compete in both static and dynamic events. A very important part of the design is using data to validate engineering decisions. The car has already been equipped with tire temperature sensors, using three infrared sensors per wheel to measure the temperature gradient across different points of the tire. This data is extremely useful, although it can be difficult to make sense of when viewed as raw numbers or in line graphs. The goal of this project is to create a base for a simple, extendable interface that can be used to process and visualize this data in order to make it easier to understand.

**Background**

Data acquisition is a very important part of engineering. It allows us to validate our previous designs and also improve new designs. Tire temperature data in particular is important to the suspension design of the vehicle, as it shows how efficiently the tread of the tire is being used. Temperatures should be even across the entire tread, and not too hot or cold on one particular edge of the tire. An existing system has already been designed using AVR microcontrollers and infrared temperature sensors to measure the temperature of 3 points across the inside, middle, and outside of the tire tread. These sensors are fitted to the car and then the data is recorded using the cars datalogging system. Once the data has been acquired, analysis must be performed in able to interpret it and make changes based upon the results.

Although there are a couple high end racecar data analysis software packages already in existence, these are extremely expensive often not very customizable. By developing our own platform for visualizing tire temperature data, it will allow us to meet the needs of the team and expand upon our team in the future, if we want to add more data channels to visualize along with the tire temperatures. The program will also be extendable, so if for example a more advanced temperature sensor that reads more than 3 points on the tire is used, the program can be easily modified to support this. This would give an advantage over purchasing a software package in that we can customize this to fit our changing needs.

**Implementation**

The goals of the project were to create a simple visualization package, filter and smooth incoming data, and start developing a real-time telemetry viewing system. The focus was on elements of the project that were within the scope of this course in C++. We started by creating a UML diagram of the code, which is shown below. This UML diagram changed and grew as the code was written and the project was restructured, but the diagram helped to keep us organized and keep the bigger picture in mind.

*Uml diagram*

A simple command line graphics library, windows.h was utilized to create the graphics for the program. Not a lot of time was spent on this, as it was not part of the scope of this course, but it was intended to be a base foundation on which the program may be expanded in the future. The data processing and analysis algorithms may be reused with a more extensive graphics application, perhaps featuring an in-car video or other data plots. The color gradients for the graphics were calculated using linear formulas to modify the RGB values of the colors displayed. By sweeping the colors up and down, full values of red, green, blue, and all colors in between are achieved and displayed nicely to mesh together. A simple linear interpolation between the 3 data points on each tire was used to calculate the gradient to be displayed in between them. This logic was used to view both the logged data and the real time input.

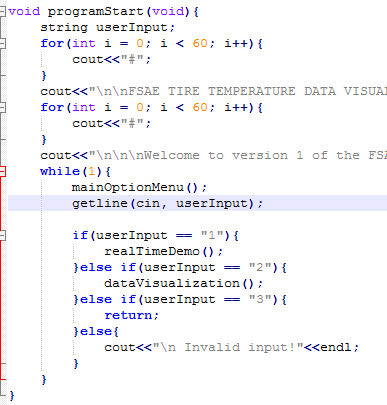
*Color gradient screenshot*

The real time interface was constructed as a basis for a more advanced system that may be implemented in the future. This system will eventually need to consist of a full wireless system to interact with the car as it drives, but for now we focused on taking the data, filtering it, and displaying it. It was very simple to set up an Arduino using existing serial libraries and I2C code to simulate this data for our purposes. Algorithms were developed to filter out ‘bad’ data.

*Picture of Arduino and sensor setup*

**Experiments and Results**

We ran experiments with many different error cases and data inputs. Exception handling was implemented heavily throughout the program to catch all different error cases and handle them accordingly. Erroneous user inputs were caught and corrected using very simple try catch blocks throughout the program, as seen in the example below. This made the program more user friendly and robust. All possible different user inputs were tested, and were handled without flaw by the exception handling blocks.



*Handling errors in user input with try-catch blocks*

In addition to user input, we also ran many different trials with the input data. Error handling was implemented for these files and many different cases with invalid formats, strange numbers, and different filename issues. These cases were handled at the appropriate point in the code and the program responded correctly.

**Discussions and Conclusions**

**Appendices**