Formula One Telemetry Analysis

Andrew Struthers[1]

Central Washington University

Introduction

This presentation focuses on the decoding Formula One telemetry data in the form of encoded UDP packets sent by the simulator F1 2021 by Codemasters. Telemetry gathered by the simulator includes the player's car speed, steering, throttle, tyre temperatures, lap times, and more. This project will use a combination of C and Python to receive and display this data. Data will be sent out from the simulator in encoded User Datagram Protocol (UDP) packets and received by a PIC24 16-bit microprocessor running C code. Selected data will be sent out via Controller Area Network (CAN) communication protocol to a different microprocessor running a Python script. The selected data will be displayed in real-time updating graphs. This project serves as a proof of concept regarding data analysis and transfer on microprocessors using CAN communication.

Purpose

The purpose of this project is to prototype and simulate the real world data acquisitions and analysis performed on actual Formula One cars. While there are many more factors and pieces of information gathered in real life, taking the simulator provided data and extrapolating meaningful results will prove the concept of data analysis using a microprocessor on a Formula One car.

Workflow

Throughout this project, I will be using the force-feedback Thrustmaster TMX Pro steering wheel and pedal controller as my input device to the simulator. This provides a more realistic car driving experience in a simulator compared to a standard XBox or PS4 controller.

1. Thrustmaster controller sends input data to F1 2021
2. F1 2021 simulates a Formula One car throughout a race
3. F1 2021 sends data packets including "Race Events" and "Car Telemetry" data as UDP packets
4. PIC 24 microprocessor receives UDP packets
5. Microprocessor decodes packets and builds pre-defined data structures defined in the F1 2021 Documentation
6. Microprocessor gathers the wanted data from pre-defined structures, does data analysis, and builds a custom data structure containing the appropriate information
7. Microprocessor sends out custom data structure via CAN protocol to other processor
8. Second processor receives new packets and displays on real-time updating graphs

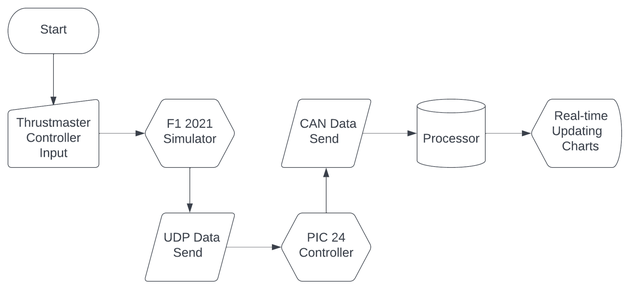

Figure 1. Project Workflow

Data Formats

Outside of the microprocessors, the transmitted data looks quite unruly, and at first nonsensical. In order to be sent and received consistently, standards for data transmission were created. Regardless of the method, however, the data ends up looking something like this:

$$\xe5\x07\x01\x12\x01\x03!]\xa4;t\xfa\x0b\x16\x8a\x96\x9eA\x91\x01\x00\x00\x13\xff$$

Decoded and processed, this data actually represents:

$$[2021, 1, 18, 1, 3, 1588638670702009633, 19.823505401611328, 401, 19, 255]$$

and makes up the header of a random packet sent by F1 2021.

User Datagram Protocol (UDP)

The first network communication method used in this project is UDP. UDP is a wireless, connectionless model that implements a minimum amount of protocol mechanics. Relative to the rest of the Internet protocol suite, UDP is relatively lightweight. UDP is commonly used on real-time systems or other time-sensitive applications because the sender doesn't wait for a confirmation that their data was received before moving on to the next packet. UDP doesn't care whether the receiving end actually got any data, so the network protocol takes up very little operating time compared to other methods.

Controller Area Network (CAN)

CAN protocol is a standard network implementation mainly used on vehicles to transmit data between microcontrollers rapidly. One of the benefits of using CAN is this method can operate without having a dedicated "host" computer. Instead, microcontrollers can upload and download data from the CAN bus. CAN has a maximum data transfer amount of 8 bytes per frame, so it is really only practical for small amounts of data that need to be transmitted rapidly. The custom packet I am creating is only 36 bytes, so this could be fully sent with 5 CAN frames. To put this in perspective, F1 2021 sends out UDP packets at around 2000 bytes per packet, 20 times a second. This is far too much information to be transmitted over CAN, but gathering a small subsection of relevant data for this project and sending it over CAN is entirely doable.

Results

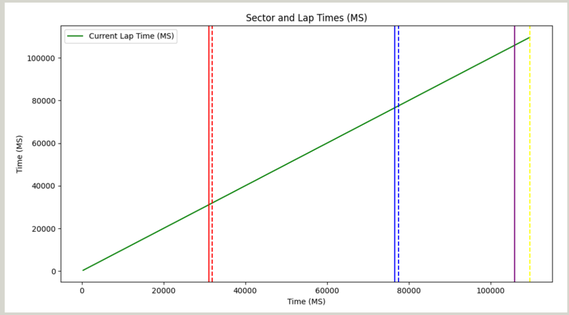

Figure 2. Lap Time Graph

Figure 2 displays my lap time in milliseconds and compares each sector time to my race best sector times. The dashed lines represent the current sector time for this lap. The purple line is my theoretical fastest achievable lap time.

Results Continued

Figure 3. Speed Graph

Figure 3 displays my speed throughout the lap in blue, my top speed throughout the entire race in red, and my average lap speed in green.

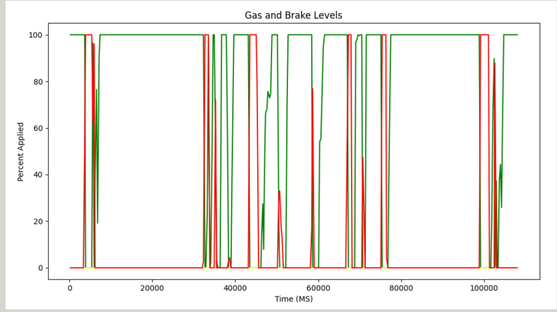

Figure 4. Input Levels

Figure 4 is harder to read, but it is comprised of my inputs to the car throughout the lap, where the green line represents throttle and the red line represents brake. I also attempted to graph steering percent between $y = -100$ representing full left and $y = 100$ representing full right, but because there are so many minor steering changes throughout the lap, the graph is really messy.

Conclusion and Future Work

I would like to take the tyre temperature, direction of the circuit, and acceleration of the car and attempt to mathematically model tyre wear throughout the race. This would be an interesting extension of data analytics, mathematical modeling, and data communication. The simulator also provides kilobytes of more information than what I used for this project, so working with more data to create more charts seems like a natural progression. Finally, the simulator provides information on every car in the race, not just mine. I could extend this project and analyze my cornering speeds with the speed of every other car, to see if I am relatively faster or slower on a corner by corner basis.

Source Code