# Patriot Racing Design Project

## Telemetry Requirements

### Project Name: Realtime car telemetry data

### Author: Coach K

### Date: 7/18/2023

### Due Dates

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Due Date | Assigned | Status |
| Update .h struct definition | July 18-2023 | Coach K | Complete, tested, working |
| Update datalogger | July 18-2023 | Coach K | Complete, tested, working |
| Update Wifi Server | July 18-2023 | Coach K | Complete, tested, working |
| Update Receiver | Aug 11, 2023 | Conrado |  |
| Update Repeaters | Sept 1, 2023 | Conrado |  |
|  |  |  |  |

### Problem statement

The purpose of this project is to speed up data rates that are being sent from the car’s data logger to the receiver and the WiFi server, and simplify the datalogger code. Key data that is being computed on the datalogger and sent are driver averages, maximums, and minimums. Examples are shortest lap, shortest distance per lap. Unless these are plotted, knowing just the shortest lap is not helpful. Few team data analysts look at such data. Approximately 15% of the transmitted data are these driver statistics. Also, since the pit TV does not have enough room to show all data for 3 cars, some data is not needed during a race. The time required by the datalogger to perform such calculations is insignificant, however the transmission times can suffer. The WiFi server has been updated to listen to 3 cars simultaneously and sending less data will help the large TV display data in real-time.

Data rates on the datalogger can be increased from the current 9600 baud, to has high as 19K, but transmit distance is sacrificed.

### Proposed Solution

Since air transmission data rates can’t be increased, less data must be sent. Lap max/min and averages can still be computed and displayed on the receiver and WiFi server, but those calculation can be done on those devices.

A list of changes is a follows

1. Transmit struct (DONE)
   1. Remove unnecessary lap average, max, min data from the transmission struct
   2. Remove “source ID”
2. Datalogger (DONE)
   1. Update the SendData() function to leverage the new struct
   2. Remove ShortestDistance, ShortestLap, and a few other calculations
   3. Modify the usage screen to remove relevant driver statistics
   4. Remove writing source id
3. Pit Display
   1. Update the loop code to leverage the new struct
   2. Add a lap checker to alert when laps have changed
   3. Add code to remember previous distance, time, and other variables to perform lap max/min/averages
   4. Remove reading source id
4. Repeaters
   1. Remove source id code
   2. Recompile with the new struct definition
5. Wifi server (DONE, but generally)
   1. Change all parsers to use the new struct
   2. Change the XML writer to remove irrelevant average/max/min
   3. Modify the loop to read 3 independent transceivers
   4. Modify the HTML/CSS/Java script to read data from 3 cars, and plot

### Anticipated problems

The code changes are not complicated, but since every device must get updated, issues will be prevalent. Fortunately, code will not compile if we have a mismatch between the code encryptor and the .h encryptor definition. Issues will be handling averages and accounting for divide by zero.

### Code

The code for devices will not be covered, but the struct definition is as follows. All downstream devices need to adapt to use the new struct definition. Note each item in the struct is a uint16\_t and commented with what data and how many bits that data will be assigned. Decisions have been made on what data ranges and precisions are needed. Limiting data to relevant range and precision will allow less data to be sent. For example, speed is computed as a float (4 bytes or 32 bits) but we don’t need to account for 8865.876235 MPH. Reducing the data to allow a valid range and precision (Speed needs to be 0 to 40.0 mph) will let us send less. If we realize speed will never exceed 51.0 MPH and we only need a single decimal of precision, then we can squeeze speed into 9 bits. We simply 10x speed and shift 9 bits, leaving 6 bits from something else.

The struct is complete.

### struct Transceiver {

### uint16\_t ID\_RPM; // CARID(2) RPM(12)

### uint16\_t WARNINGS\_TEMPF; // WARNINGS(8) TEMPF(8)

### uint16\_t VOLTS\_LAPS; // VOLTS(9) LAPS(7)

### uint16\_t SPEED\_EREM; // SPEED(9) EREM (7)

### uint16\_t DISTANCE\_TREM; // DISTANCE(9) TREM(7)

### uint16\_t AMPS\_D0ID; // AMPS(11) D0ID(5)

### uint16\_t ENERGY\_D0LAPS; // ENERGY(10) D0LAPS(6)

### uint16\_t D1LAPS\_D1ID\_D2ID; // D1LAPS(6) D1ID(5) D2ID(5)

### uint16\_t D2LAPS\_LAP2AMPS; // D2LAPS(6) LAP2AMPS(9)

### uint16\_t RACETIME; // RACETIME(16)

### uint16\_t D0TIME; // D0TIME(16)

### uint16\_t D1TIME; // D1TIME(16)

### uint16\_t D2TIME; // D2TIME(16)

### uint16\_t LAPTIME; // LAPTIME(16)

### uint16\_t GFORCE\_DRIVERNO; // GFORCE(9) DRIVER NUMBER(2)

### uint16\_t LAPENERGY\_LAPAMPS; // LAPENERGY(7) LAPAMPS(9)

### } ;

### Acceptance Testing

Library struct

1. All data packed w/o wasted bits, noting shifting used to pack bytes into uint16\_t segments

Datalogger

1. execute SendData() w/o compile errors
2. Code should not compute DriverLapEfficiency, DriverFastestLap, DriverShortestLap amongst other things
3. The driver should not see any differences in what data is presented

Receiver

1. read Struct w/o loss of data
2. Compute DriverLapEfficiency, DriverFastestLap, DriverShortestLap for each lap
3. Display compute data in the driver screen, basically the Pit Display user sees no difference

WiFi Server

1. read Struct w/o loss of data
2. Possibly compute DriverLapEfficiency, DriverFastestLap, DriverShortestLap for each lap
3. Possibly display compute data in the driver screen if we have the room
4. Other update defined in a separate project

# End of Requirements